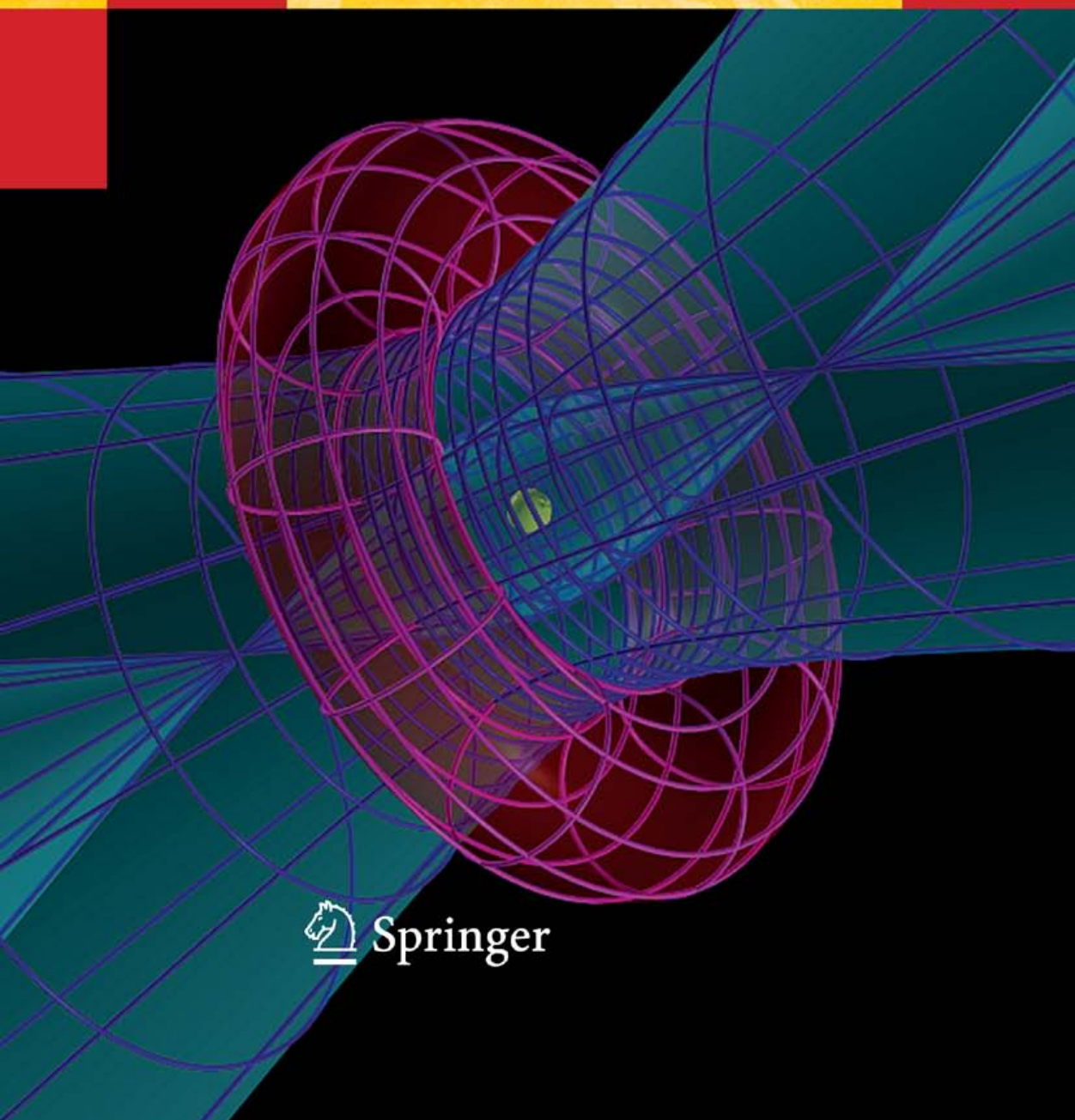
Leila De Floriani
Michela Spagnuolo Editors

Shape Analysis and Structuring

Mathematics + Visualization



Springer

Mathematics and Visualization

Leila De Floriani
Michela Spagnuolo

Editors

Shape Analysis and Structuring

With 156 Figures, 29 in Color and 7 Tables

Springer

Leila De Floriani
Department of Computer
and Information Sciences (DISI)
University of Genova
Via Dodecaneso, 35
16146 Genova, Italy
deflo@disi.unige.it

and

Department of Computer Science
University of Maryland
College Park, MD 20740, USA

Michela Spagnuolo
Istituto per la Matematica Applicata
e le Tecnologie Informatiche
Consiglio Nazionale delle Ricerche
Via De Marini, 6
16149 Genova, Italy
spagnuolo@ge.imati.cnr.it

ISBN 978-3-540-33264-0 e-ISBN 978-3-540-33265-7

Springer Series in Mathematics and Visualization ISSN 1612-3786

Library of Congress Control Number: 2007935447

Mathematics Subject Classification Numbers (2000): 68U05, 68U07, 65D18, 65D17

Cover image: Gershon Elber, TECHNION, Israel

Cover design: WMX Design GmbH. Heidelberg

Printed on acid-free paper.

9 8 7 6 5 4 3 2 1

springer.com

1 Introduction

The idea of composing this book arose from the desire to enrich and systematise the extensive state-of-the-art studies that were carried out within the framework of AIM@SHAPE, a Network of Excellence funded by European Commission under the FP6-IST [1]. The main goal of the network is to develop new methodologies for modelling and processing knowledge embedded in digital *shapes*.

Current approaches to modelling are focused on the geometry of shapes, while their semantics, e.g., meaning or functionality in a given context, is still overlooked. This is partly due to the lack of methods for the automatic extraction of the semantic content from digital shapes, known as the process of *semantic annotation* in research areas related to the development of the Semantic Web, and partly to the evolution of research on shape modelling which in the past years was highly focused on the geometric aspects of shapes. The shift from a purely geometric to a semantic-aware level of representation of digital shapes is the ultimate scientific objective of AIM@SHAPE.

In this scenario, a crucial role is played by geometry processing methods that are aimed to *preserve and enhance shape information* as well as to effectively *capture the structure of a shape* by identifying relevant shape components and their mutual relationships. Each chapter of the book provides a detailed state-of-the-art review on a specific topic, which is crucial for shape analysis and structuring, contains a classification of the techniques developed in the area, and discusses open problems.

Structural analysis play a fundamental role in the automatic extraction of semantic information. While shapes are fully characterized by a specific geometry, shape information is treated differently by the human brain with respect to several other forms of information. At a geometric level, a digital shape is a computational structure which defines a geometric representation. Different types of geometric models can be used to describe the same object. Examples are polygons, surface models (e.g., splines, NURBS surfaces), or solid models (e.g., triangle or tetrahedral meshes, boundary representations, constructive volumetric representations). The structural level in the representation is reached by organizing geometric information to reflect, or make explicit, the association between the various components of the shape. A structural description is the basis for developing semantic-based shape representations, since it abstracts from the low-level, detailed, description provided by a geometric model.

Several techniques have been developed in the literature for processing different aspects of the geometry of shapes, in particular shape interrogation and re-meshing techniques enhance a shape description with information which can be effectively used to attach semantics to the shape. *Shape interrogation* is the process of extracting information from a geometric model. Geometric models need to be analyzed with respect to different aspects, such as visual pleasantness, technical smoothing, geometric constraints, or surface intrinsic properties. The various methods devel-

[1] AIM@SHAPE, http://www.aimatshape.net

oped in the literature are used to detect surface imperfections, to analyze shapes, or to visualize different forms. Such methods are reviewed in *Chapter one*.

Re-meshing is often used for efficient shape approximation, and it consists of repartitioning a set of primitives so that they best fit the original shape. Re-meshing preserves the shape, in the sense that it still approximates the shape after re-meshing, and it can be designed to enhance the shape. Every shape feature is locally fit with a primitive that minimally characterizes the shape. In addition, some recent re-meshing techniques operate through a careful analysis based on multi-scale discrete differential geometry so as to estimate the main (and detailed) axis of symmetry of the shape. The shape is, thus, locally classified as spherical, parabolic, elliptic, and hyperbolic in order to drive the re-meshing process. Such classification may be also used for shape enhancement. *Chapter two* presents a survey on re-meshing techniques.

A first way of structuring shape information is provided by those techniques that organize a geometric shape description defined by a function, by a mesh, or by a set of points into a representation of the shape at different levels of resolutions, from which concise and adaptive shape descriptions can be extracted. This topic has received considerable attention in recent years in many fields of computer graphics, geometric modelling and visualization, and numerous research efforts have been devoted to it. This book contains two chapters on focused on multi-resolution analysis, and the other on subdivision surfaces.

Multi-resolution analysis provides a powerful tool for efficiently representing functions at multiple levels of detail. Herein, a complex function is decomposed into a coarser low-resolution part, together with a collection of detail coefficients, necessary to recover the original function. Multi-resolution analysis has many inherent advantages, including compression, progressive transmission, visualization and editing at different levels of detail. An overview of methods for multi-resolution analysis is presented in *Chapter three*.

Subdivision surfaces define the basis for generating a smooth surface from a coarse mesh, and, thus, they have bees extensively used in geometric modeling for creating, editing and transmitting a shape. The surface is defined by the initial coarse mesh plus a subdivision scheme to progressively subdivide the mesh by inserting new vertices and connecting them to the edges and faces until a smooth surface is obtained in the limit. *Chapter four* contains a review of surface subdivision schemes and their application in geometric modeling.

The third part of the book is devoted to structural shape representations. In the above framework, many research efforts have been devoted to study concise, structural representations of a shape based on *skeletal structures*, such as the medial axis, or the Reeb graph. Skeletal structures provide an abstract shape representation by idealized lines that retain the connectivity of the original shape. In advanced fields, such as virtual human modeling, available modeling tools to represent structured geometry focus on adding a skeleton to the 3D geometry in order to animate it and provide different degrees of realism. A survey of different skeletal structures is presented in *Chapter five*.

Another class of structural representations is provided by *morphology-based descriptions for scalar and vector fields*. There has been a considerable amount of work

in the literature on extracting critical features (point, integral lines, etc.) from two-dimensional scalar fields describing grey-level images and terrains, and, more recently, some work has been done on volume data on extracting critical features and for representing the topological structure of the field iso-surfaces. A survey of morphological representations for two-dimensional and three-dimensional scalar fields is presented in *Chapter six*.

Topological methods based on features, like critical points or separatrix lines, have also been applied for the analysis of vector fields. The basic idea is to use such features for segmenting the flow into areas of different flow behaviour, and use this as a tool for understanding complex phenomena described by the vector fields. After introducing topological features for two-dimensional and three-dimensional vector fields, *Chapter seven* presents a survey of methods for extracting topological features from vector fields and using them as visualization tools for complex flow phenomena, represented both as static and dynamic fields. Applications of topological methods for compressing, simplifying, comparing, and constructing vector fields are also discussed.

The last part of the book, namely *Chapter eight*, provides a review on the use of structural data for modelling shapes with a high semantic characterization, e.g., *virtual humans*. In this case, the structural model, called the *control skeleton*, has by itself a specific role in the evaluation of the many different shapes associated with all the possible postures that the body model can reach. The first part of the Chapter discusses the control articulated skeleton structure and different approaches to build skeletons and bind it to the shape geometry. The second part addresses the generation of level-of-detail models for virtual humans, in terms of the geometry and of the articulated skeleton.

We would like to acknowledge the work of all the authors of the various chapters in this book, that contributed with their expertise and energy to assemble a substantial part of state-of-the-art reports on a variety of interesting topics. Our special thanks go to Bianca Falcidieno, the coordinator of the AIM@SHAPE Network of Excellence, for her enthusiasm and support in this work and for having inspired much of the motivations of this collection, and to all partners in AIM@SHAPE. Many people contributed to the preparation of the book, and we would like to thank specifically Emanuele Danovaro for his invaluable help in editing and preparation.

Finally, we would like to acknowledge the support of the European Network of Excellence AIM@SHAPE, contract number 506766.

Leila De Floriani *Michela Spagnuolo*

Contents

1 Introduction V

Shape Interrogation
Stefanie Hahmann, Alexander Belyaev, Laurent Busé, Gershon Elber, Bernard Mourrain, Christian Rössl 1
1 Introduction 1
2 Differential Geometry of curves and surfaces 2
2.1 Curves 2
2.2 Surfaces 3
3 Interrogation of discrete shapes 5
3.1 Surface Normal Estimation 5
3.2 Curvature Tensor Estimation 6
3.3 Applications to Discrete Shape Analysis 8
4 First-Order Shape Analysis 9
4.1 Reflection lines 9
4.2 Highlight lines 10
4.3 Isophotes 11
4.4 Detection of inflections 12
4.5 Geodesic paths on surfaces and meshes 13
5 Second-order shape analysis 13
5.1 Local shape analysis with Gaussian curvature 14
5.2 Focal Surface and Corresponding Surface Features 15
5.3 Hedgehog diagrams and curvature plots 17
5.4 Generalized focal surfaces 18
5.5 Color mappings 19
6 Characteristic lines 20
6.1 Isolines 21
6.2 Lines of curvature, umbilics 22
6.3 Curvature Extrema for Shape Interrogation 22
6.4 Special Surface Points 24
7 Robust Symbolic based Shape Interrogation and Analysis 25

7.1 Curvature Analysis 26
7.2 Silhouette, Isoclines/Isophotes and Reflection lines 28
7.3 Surface Recognition 28
8 Interrogation of algebraic curves and surfaces 30
8.1 Subdivision methods 32
Univariate polynomials 32
Multivariate polynomials 33
Univariate Root Solver 34
Multivariate root finding 34
8.2 Algebraic methods 35
Resultant-based methods 35
Normal forms 39
9 Conclusion 41
References 42

Recent Advances in Remeshing of Surfaces
Pierre Alliez, Giuliana Ucelli, Craig Gotsman, Marco Attene 53
1 Introduction 53
1.1 Remeshing 54
1.2 Applications 54
1.3 Main Issues 54
2 State of the Art 56
2.1 Structured Remeshing 57
Semi-Regular 57
Completely Regular 61
Highly Regular 62
2.2 Compatible Remeshing 63
2.3 High Quality Remeshing 66
2.4 Feature Remeshing 71
Feature-preserving 72
Feature-enhancing 73
2.5 Error-driven Remeshing 74
References 78

Multiresolution Analysis
Georges-Pierre Bonneau, Gershon Elber, Stefanie Hahmann, Basile Sauvage . . 83
1 Introduction 83
2 Hierarchical Freeform Representations 84
3 Multiresolution Methods for Freeform Representations 85
3.1 Wavelet Decomposition of B-spline Functions 86
3.2 Direct Freeform Curve and Surface Manipulation 88
3.3 Linear Constraints 91
3.4 Bi-Linear and Non-Linear Constraints 92
3.5 Intrinsic Multiresolution Decomposition of Freeform Shapes 96
3.6 Multiresolution Morphing 98

3.7 Variational Multiresolution Methods for Freeform Surface 100
Variational Shape Design 101
Variational Multiresolution Modeling 102
4 Multiresolution Analysis for Irregular Mesh-based Representations 102
4.1 Irregular Triangulations 103
4.2 Surface Meshes 108
5 Conclusions and Open Issues 109
References 110

Subdivision Surfaces and Applications
Chiara Eva Catalano, Ioannis Ivrissimtzis, Ahmad Nasri 115
1 Introduction 115
2 Subdivision basics 116
2.1 Subdivision schemes 116
2.2 Subdivision analysis 118
Laurent polynomials 118
Spectral analysis of the subdivision matrix 119
The continuity degree of subdivision surfaces 120
3 Artifacts in subdivision surfaces 120
3.1 First order artifacts 121
3.2 Second order artifacts 123
3.3 Higher level artifacts 126
4 Implementation and usability issues 128
4.1 Direct evaluation of subdivision surfaces 128
4.2 Visual quality and shape predictability 129
4.3 Multi-resolution subdivision surfaces 131
5 Constraint-based subdivision surfaces 131
5.1 Point interpolation 132
5.2 Interpolation with normal constraints 132
5.3 Interpolation of isolated curves 132
5.4 Interpolation of isolated curves with cross derivative 135
5.5 Lofted subdivision surfaces 135
5.6 Interpolation of a net of curves 137
5.7 Non-uniform subdivision surfaces 138
6 Conclusions 138
References 140

Skeletal Structures
Silvia Biasotti, Dominique Attali, Jean-Daniel Boissonnat, Herbert Edelsbrunner, Gershon Elber, Michela Mortara, Gabriella Sanniti di Baja, Michela Spagnuolo, Mirela Tanase, Remco Veltkamp 145
1 Introduction 145
1.1 Overview 146
2 Definitions of geometric medial structures 148
3 Exact representation of medial structures 150

3.1 Bisectors for freeform shapes 150
3.2 Exact computation of the medial axis 154
4 Approximation of the medial axis 156
4.1 Skeletons from Voronoi Diagrams 157
Instability and semi-continuity 157
Approximation paradigm for the medial axis 158
Punctured Euclidean spaces 159
Pruning the Voronoi graph 160
4.2 Skeleton trough the simulation of the grassfire 161
Straight skeleton 161
The Linear Axis 163
4.3 Skeletons based on topological thinning 164
4.4 Skeletons from distance maps 166
5 Skeletons from topological structures 168
5.1 Methods based on wavefront propagation 169
5.2 Methods based on the Reeb graph 172
6 Conclusions and future developments 175
References 177

Morphological Representations of Scalar Fields
Silvia Biasotti, Leila De Floriani, Bianca Falcidieno, Laura Papaleo 185
1 Introduction 185
2 Background Notions 186
2.1 Cell and Simplicial Complexes 186
2.2 Digital Models of a Scalar Field 188
2.3 Morse Theory 188
2.4 Morse Complexes and Morse-Smale Complexes 189
2.5 Contour Trees 191
3 Extracting Critical Points 193
3.1 Extracting Critical Points from a Piecewise Linear Field 193
3.2 Extracting Critical Points from a Regular Grid 195
3.3 Extracting Critical points from Contours 196
4 Extracting Approximations of a Morse-Smale Complex 197
4.1 Boundary-based Algorithms on a Simplicial Model 197
4.2 Boundary-based Algorithms on a Regular Model 198
4.3 Region-based Algorithms for Approximating Morse complexes 199
4.4 Generalization of Morse-Smale Complexes 200
5 Algorithms for Extracting a Contour Tree 202
6 Concluding Remarks 207
References 208

Topological Representations of Vector Fields
Holger Theisel, Christian Rössl, Tino Weinkauf 215
1 Introduction 215
2 Topological features of 2D vector fields 216
2.1 Concepts 216
Classification of critical points 216
Boundary switch points 218
Separatrices 218
2.2 Visualizing 2D topology 219
3 Topological Features of 3D Vector Fields 220
3.1 Concepts 220
Critical points 220
Boundary switch curves 222
Separatrices 223
Saddle- and boundary switch connectors 224
3.2 Visualizing 3D topology 224
Example: 226
4 Topological features of time-dependent vector fields 226
4.1 Stream line oriented topology 227
Tracking critical points 228
Saddle connections 229
Tracking closed stream lines 230
4.2 Path line oriented topology 231
4.3 An Example: 231
5 Further applications of topological features 231
5.1 Compressing vector fields 232
5.2 Topological simplification of vector fields 232
5.3 Topological comparison of vector fields 234
5.4 Constructing vector fields 236
6 Conclusions 237
References 238

Control Structure and Multi-Resolution Techniques for Virtual Human Representation
Thomas Di Giacomo, HyungSeok Kim, Laurent Moccozet, Nadia Magnenat-Thalmann 241
1 Definitions and Background 241
1.1 Control skeleton definition 242
1.2 Historical background 242
2 Skeleton control methods 243
2.1 Control Techniques 244
2.2 Space-time Constraints and Controllers 246
3 Skeleton skinning and skin mapping 247
3.1 Skeleton skinning 247
3.2 Skin mapping 248

4 Skeleton-driven deformation 249
5 Generation of Control Skeleton 253
5.1 Medial axis-based methods 253
5.2 Template-based methods 255
5.3 Mesh decomposition based methods 258
6 Discussion on skeleton for Virtual Humans 259
7 Multi-Resolution Techniques 261
7.1 Simplification of Shape and Control Structure 261
Simplification of textured body surfaces 261
Preservation of Features in Simplification 263
7.2 Multi-Resolution Modeling for Key Parameters 265
7.3 Discussion on LoD for Virtual Humans 268
References 268

Colour Plates: Shape Interrogation 275

Colour Plates: Recent Advances in Remeshing of Surfaces 277

Colour Plates of the Chapter: Multiresolution Analysis 279

Colour Plates: Subdivision Surfaces and Applications 281

Colour Plates: Skeletal Structures 285

Colour Plates: Morphological Representations of Scalar Fields 287

Colour Plates: Topological Representations of Vector Fields 289

Index 293

List of Contributors

Pierre Alliez
INRIA Sophia-Antipolis, France
pierre.alliez@sophia.inria.fr

Dominique Attali
LIS-CNRS, Saint Martin d'Hères, France.
dominique.attali@lis.inpg.fr

Marco Attene
IMATI-GE, CNR, Italy
jaiko@ge.imati.cnr.it

Alexander Belyaev
MPII, Max Planck Institut für Informatik, Saarbrücken, Germany
belyaev@mpi-sb.mpg.de

Silvia Biasotti
IMATI-GE, CNR, Italy
silvia@ge.imati.cnr.it

Jean-Daniel Boissonnat
INRIA Sophia-Antipolis, France
jean-daniel.boissonnat@inria.fr

Georges-Pierre Bonneau
Université Joseph Fourier, Grenoble, France,
georges-pierre.bonneau@imag.fr

Laurent Busé
INRIA Sophia-Antipolis, France
lbuse@sophia.inria.fr

Chiara Eva Catalano
IMATI-GE, CNR, Italy
chiara.catalano@ge.imati.cnr.it

Herbert Edelsbrunner
Duke University, Dept. of Computer Science and Raindrop Geomagic, USA.
edels@cs.duke.edu

Gershon Elber
Technion - Israel Institute of Technology, Haifa, Israel
gershon@cs.technion.ac.il

Bianca Falcidieno
IMATI-GE, CNR, Italy
bianca@ge.imati.cnr.it

Leila De Floriani
University of Genova, Dept. of Computer Science, Genova, Italy
deflo@disi.unige.it

Thomas Di Giacomo
MIRAlab, University of Geneva, Switzerland
giacomo@miralab.unige.ch

Craig Gotsman
Technion - Israel Institute of Technology, Haifa, Israel
gotsman@cs.technion.ac.il

Stefanie Hahmann
Laboratoire Jean Kuntzmann, Institut National Polytechnique de Grenoble, France
stefanie.hahmann@imag.fr

HyungSeok Kim
MIRAlab, University of Geneva,
hyung.kim@acm.org

Ioannis Ivrissimtzis
Durham University, Dept. of Computer Science, UK
ioannis.ivrissimtzis@durham.ac.uk

Nadia Magnenat-Thalmann
MIRAlab, University of Geneva,
thalmann@miralab.unige.ch

Laurent Moccozet
MIRAlab, University of Geneva,
moccozet@miralab.unige.ch

Michela Mortara
IMATI-GE, CNR, Italy
michela@ge.imati.cnr.it

Bernard Mourrain
INRIA Sophia-Antipolis, France
bernard.mourrain@sophia.inria.fr

Ahmad Nasri
American University of Beirut, Dept. of Computer Science, Lebanon
anasri@aub.edu.lb

Laura Papaleo
University of Genova, Dept. of Computer Science, Genova, Italy
papaleo@disi.unige.it

Christian Rössl
INRIA Sophia-Antipolis, France
christian.roessl@sophia.inria.fr

Gabriella Sanniti di Baja
CNR - Ist. di Cibernetica "E. Caianello", Pozzuoli, Napoli, Italy.
gsdb@imagm.cib.na.cnr.it

Basile Sauvage
Laboratoire Jean Kuntzmann, Institut National Polytechnique de Grenoble, France
basile.sauvage@imag.fr

Michela Spagnuolo
IMATI-GE, CNR, Italy
michi@ge.imati.cnr.it

Mirela Tanase
Universiteit Utrecht (UU), The Netherlands.

Holger Theisel
Bielefeld University, Germany
theisel@techfak.uni-bielefeld.de

Giuliana Ucelli
IGD / GraphiTech, Italy
giuliana.ucelli@infotn.it

Remco Veltkamp
Universiteit Utrecht (UU), The Netherlands.
remco.veltkamp@cs.uu.nl

Tino Weinkauf
Zuse Institute Berlin (ZIB), Germany
weinkauf@zib.de

Shape Interrogation

Stefanie Hahmann[1], Alexander Belyaev[2], Laurent Busé[3], Gershon Elber[4], Bernard Mourrain[3], and Christian Rössl[3]

[1] Laboratoire Jean Kuntzmann, Institut National Polytechnique de Grenoble, France
Stefanie.Hahmann@imag.fr
[2] MPII, Max Planck Institut für Informatik, Saarbrücken, Germany
belyaev@mpi-sb.mpg.de
[3] INRIA Sophia-Antipolis, France
lbuse@sophia.inria.fr, Bernard.Mourrain@sophia.inria.fr,
Christian.Roessl@sophia.inria.fr
[4] Technion - Israel Institute of Technology, Haifa 32000, Israel
gershon@cs.technion.ac.il

Summary. Shape interrogation methods are of increasing interest in geometric modeling as well as in computer graphics. Originating 20 years ago from CAD/CAM applications where "class A" surfaces are required and no surface imperfections are allowed, shape interrogation has become recently an important tool for various other types of surface representations such as triangulated or polygonal surfaces, subdivision surface, and algebraic surfaces. In this paper we present the state-of-the-art of shape interrogation methods including methods for detecting surface imperfections, surface analysis tools and methods for visualizing intrinsic surface properties. Furthermore we focus on stable numerical and symbolic solving of algebraic systems of equations, a problem that arises in most shape interrogation methods.

1 Introduction

Shape interrogation is the process of extraction of information from a geometric model. Surface interrogation is of central importance in modern Computer Graphics and Computer Aided Design (CAD) systems. Wherever geometrical models are used, they often need to be analyzed with respect to different aspects like, for example, visual pleasantness, technical smoothness, geometric constraints or surface intrinsic properties. The various methods, which are presented in this survey can be used to detect surface imperfections, to analyze shapes or to visualize different forms. We not only restrict the shapes to be investigated to free-form surfaces, but include polygonal meshes as well as algebraic surfaces. Artefacts of subdivision surfaces are subject of Chapter 4 of this book [23]. Particular attention is paid to stable numerical and symbolic solving of algebraic systems of equations, a problem that arises in most shape interrogation methods.

In Section 2, fundamental notions of differential geometry are briefly recalled. Interrogation methods for polygonal meshes are discussed in Section 3. First and

second order shape interrogation and visualization techniques are discussed in Sections 4, 5, focusing mainly on free-form curves and surfaces. The computation and visualization of characteristic curves on surfaces is subject of Section 6. Section 7 discusses the use of robust symbolic computation methods for shape interrogation. Interrogation of algebraic curves and surfaces is finally discussed in Section 8, in particular the transversal problem of solving of algebraic systems of equations is described.

2 Differential Geometry of curves and surfaces

Fundamental notions of differential geometry of curves and surfaces that are needed in the following of the paper will briefly be reviewed in this section. For a complete bibliography on differential geometry the reader is referred to standard literature [39, 102, 79, 176].

2.1 Curves

A **parametric curve** is a mapping x from $I = [a, b] \subset \boldsymbol{R}$ into $\boldsymbol{R}^n$ of class C^r $(r \geq 1)$. x is called **regular**, if $\frac{dx}{dt}(t) \neq 0$ for all $t \in I$. If L is the length of $x([a, b])$, there exists a unique parameter transformation s from I into $[0, L]$ such that for all $t_0, t_1 \in [0, L]$ the length of the arc $x([t_0, t_1])$ is equal to $s(t_1) - s(t_0)$. For all $t \in [a, b]$ $s(t) = \int_a^t \| \frac{dx}{dt} \| dt$. s is called the **arc length parameterization**. It is a geometric invariant of a curve and is therefore also called **natural parameterization**.
Let $x : [0, L] \rightarrow \boldsymbol{R}^3$, $s \mapsto x(s)$ be a regular and naturally parameterized curve of class C^3, such that $\|x''(s)\| \neq 0$ for all $s \in]0, L[$, then

- $v_1(s) := x'(s)$ is called **tangent vector** of x in s.
- $v_2(s) := \frac{x''}{\|x''\|}$ is called **unit normal vector** of x in s.
- $v_3(s) := v_1(s) \times v_2(s)$ is called **binormal vector** of x in s,

where $\times$ denotes the vector product (cross product) in $\boldsymbol{R}^3$. $\{v_1(s), v_2(s), v_3(s)\}$ form an orthonormal basis of $\boldsymbol{R}^3$ called the **Frenet frame** of x in s.
The following holds: v_1, v_2, v_3 are mappings of class C^1, and

$$\begin{aligned} v_1' &= \kappa_1 v_2 \\ v_2' &= -\kappa_1 v_1 + \kappa_2 v_3 \\ v_3' &= -\kappa_2 v_2 \end{aligned}$$

where

$$\kappa(s) = \|x''\| \ , \qquad \tau(s) = \frac{|x', x'', x'''|}{\|x''\|}$$

are mappings of class C^1 and C^0 respectively. $|\cdot, \cdot, \cdot|$ denotes the determinant of the matrix formed by the three vector arguments. κ and τ are called **curvature** and **torsion** of the curve x. The curvature measures the deviation of a curve from a straight line, and the torsion measures the deviation of a curve from being planar.

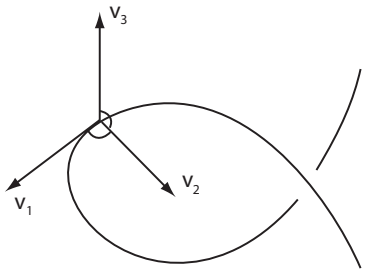


Fig. 1. Frenet frame.

2.2 Surfaces

A **parametric surface** is a mapping X from $\Omega \subset \boldsymbol{R}^2$ into $\boldsymbol{R}^3$ of class C^r ($r \geq 1$). X is called **regular** if for all $\mathbf{u} = (u, v) \in \Omega$, $dX_{\mathbf{u}}$ is an invertible linear mapping. The two partial derivatives of X in $\mathbf{u}$ are denoted by $X_u(\mathbf{u})$ and $X_v(\mathbf{u})$. The affine subspace $T_{\mathbf{u}}X := \{X(\mathbf{u}) + \lambda X_u(\mathbf{u}) + \mu X_v(\mathbf{u}) \mid (\lambda, \mu) \in \boldsymbol{R}^2\}$ is called **tangent plane** to X in $\mathbf{u}$.

The **unit normal vector field** N is given by

$$N := \frac{X_u \times X_v}{\|X_u \times X_v\|}.$$

The moving frame $\{X_u, X_v, N\}$ is the **Gauss frame**. The Gauss frame is in general not an orthogonal frame.

The bilinear form on $T_{\mathbf{u}}X$ induced by the inner product of $\boldsymbol{R}^3$ is called the first fundamental form of the surface. The matrix representation of the first fundamental form $I_{\mathbf{u}}$ with respect to the basis $\{X_u, X_v\}$ of $T_{\mathbf{u}}X$ is given by $G = (g_{ij})$ with $i, j = 1, 2$:

$$\begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{pmatrix} = \begin{pmatrix} \langle X_u, X_u \rangle & \langle X_u, X_v \rangle \\ \langle X_v, X_u \rangle & \langle X_v, X_v \rangle \end{pmatrix}$$

where $<, >$ denotes the scalar product. The first fundamental form $I_{\mathbf{u}}$ is symmetric, positive definite and geometrically invariant. The first fundamental form allows measurements on the surface (length of curves, angles of tangent vectors, areas of regions) without referring back to the space $\boldsymbol{R}^3$, in which the surface lies.

The linear mapping $L_{\mathbf{u}}$

$$\begin{aligned} L_{\mathbf{u}} \;:\; & T_{\mathbf{u}}X \to T_{\mathbf{u}}X \\ & x \mapsto dN_{\mathbf{u}} \circ dX_{\mathbf{u}}^{-1}(x) \end{aligned}$$

is called the **Weingarten map**.

The bilinear symmetric form $II_{\mathbf{u}}$ defined on $T_{\mathbf{u}}X$ by

$$II_{\mathbf{u}}(x, y) = \langle L_{\mathbf{u}}(x), y \rangle$$

is called the **second fundamental form** of the surface X.

Its matrix in the basis $\{X_u, X_u\}$ of $T_{\mathbf{u}}X$ is denoted $H = (h_{ij})$ with $i, j = 1, 2$:

$$\begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix} = \begin{pmatrix} \langle N, X_{uu} \rangle & \langle N, X_{uv} \rangle \\ \langle N, X_{vu} \rangle & \langle N, X_{vv} \rangle \end{pmatrix}.$$

The matrix HG^{-1} of the Weingarten map $L_{\mathbf{u}}$ is symmetric and real and therefore it has two real eigenvalues κ_1, κ_2 with corresponding orthogonal eigenvectors. κ_1, κ_2 are called **principle curvatures** of the surface X, also labeled as κ_{max}, κ_{min}. The product of the principle curvatures $K = \kappa_1 \cdot \kappa_2 = det(L_{\mathbf{u}}) = \frac{det(H)}{det(G)}$ is called the **Gaussian curvature** and its mean $M = \frac{1}{2}(\kappa_1 + \kappa_2) = trace(L_{\mathbf{u}})$ is called the **mean curvature**.

Another approach for the principle curvatures is the following: Let $A := \Delta u \cdot X_u + \Delta v \cdot X_v$ be a tangent vector with $\|A\| = 1$. If we intersect the surface with the plane given by N and A, we get an intersection curve y with the following properties:

$$\dot{y}(s) = A \quad \text{and} \quad e_2 = \pm N$$

where e_2 is the principal normal vector of the space curve y. The implicit function theorem implies the existence of this normal section curve. To calculate the extreme values of the curvature of a normal section curve (the normal section curvature) we can use the method of Lagrange multipliers because we are looking for the extreme values of the normal section curvature κ_N with the condition $\|\dot{y}(s)\| = 1$.

As a result of these considerations we obtain the following. Unless the normal section curvature is the same for all directions there are two perpendicular directions A_1 and A_2 in which κ_N attains its absolute maximum and its absolute minimum values. These directions are the *principal directions* with the corresponding normal section curvatures κ_1 and κ_2.

For $A = A_1 \cos\varphi + A_2 \sin\varphi$ we get Euler's formula:

$$\kappa_N = \kappa_1 \cos^2\varphi + \kappa_2 \sin^2\varphi,$$

If the principal directions are taken as coordinate axes, Euler's formula implies the so-called **Dupin indicatrix:**

$$\kappa_1(u)^2 + \kappa_2(u)^2 = \pm 1. \tag{1}$$

We use the Dupin indicatrices as a tool to visualize curvature situations on surfaces. The Dupin indicatrices at elliptic points $(K > 0)$ are ellipses, at hyperbolic points $(K < 0)$ pairs of hyperbolas, and at parabolic points $(K = 0)$ pairs of parallel lines. **Flat points** $(\kappa_1 = \kappa_2 = 0)$ are degenerated parabolic cases. Points with $\kappa_1 = \kappa_2$ are called **umbilical points**.

3 Interrogation of discrete shapes

Polygonal meshes constitute the primary tool for 3D surface representation and are frequently used in a wide range of scientific applications, including computer graphics, visualization, and numerical simulations. Two fundamental questions of surface approximation by polygonal meshes concern approximation quality (accuracy) [60] and the relation between the accuracy and size of the approximation [61]. Recently both of these questions were also addressed in [29] where a variational approach for surface approximation by polygonal meshes was developed. Shape approximation with polygonal meshes is discussed in more detail in Chapter 2 of this book [1].

Accurate estimation of geometric properties of a surface from its discrete approximation is important for many applications. Nevertheless there is no consensus on how to achieve accurate estimations of simple surface attributes such as the normal vector and curvatures [122]. An accurate polygonal approximation of surface geometry in a least-squares sense [60, 29] does not guarantee accurate approximations of surface normals and curvatures by their discrete counterparts [121, 132, 119, 14]. Thus, deriving accurate, consistent, and numerically robust estimates for the surface normal vector and curvature tensor remains an area of active and creative research today.

3.1 Surface Normal Estimation

Given a smooth surface approximated by a dense triangle mesh, an accurate and robust estimation of vertex normals is important for a number of tasks including smooth shading [66, 156], curvature estimation (see, e.g., [180]), and feature extraction (see, e.g., [87]).

Usually the normal vector at a vertex of a triangle mesh is estimated as the normalized weighted sum of normals of the incident facets (triangles). A survey of various methods to estimate the normal vector can be found in [174]. Uniform (equal) weights are justified in [63] via finite difference approximations. In [180] the weights are chosen to be equal to the areas of the incident triangles. Weighting by the inverse areas was considered in [174, 87], and weights equal to the facet angles at the vertex are proposed in [185]. A weighting scheme assuming that the mesh locally approximates a sphere was developed in [120]. The vertex normal vector can be also obtained from the mean curvature vector and, therefore, mean curvature vector estimates proposed in [37, 122] lead to approximations of the vertex normals. A standard approach for testing and comparing various methods to estimate surface normals and curvatures consists of tessellating known (analytical) surfaces and comparing the estimates from the resulting mesh and from the original surface [73, 180, 104, 32, 122]. An interesting statistical approach was recently proposed in [125, 126]. First steps towards a rigorous mathematical analysis and comprehensive comparison of various weighting schemes are made in [106].

3.2 Curvature Tensor Estimation

Estimates of the curvature tensor on polygonal meshes are applied in a variety of applications ranging from the detection of surface defects to the detection of features. Many techniques have been proposed (see, e.g., [153] for a recent survey), in this section we provide an overview of different approaches.

In order to estimate the curvature tensor at a vertex a certain neighborhood of this vertex is considered, typically its 1-ring. A common approach is to first discretize the normal curvature along edges. Given is an edge (i, j), vertex positions X_i, X_j, and the normal N_i, then

$$\kappa_{ij} \;=\; 2\frac{\langle (X_j - X_i), N_i \rangle}{\|X_j - X_i\|^2} \tag{2}$$

provides an approximation of the normal curvature at X_i in the tangent direction which results from projecting X_i and X_j into the tangent plane defined by N_i. This expression can be interpreted geometrically as fitting the osculating circle interpolating X_i and X_j with normal N_i at X_i (cf. [130]). Alternatively, the equation can be derived from discretizing the curvature of a smooth planar curve (cf. [180]). With estimates κ_{ij} of the normal curvature for all edges incident to vertex i, Euler's formula can be applied to relate the κ_{ij} to the unknown principal curvatures (and principal directions). Then approximates to the principal curvatures can be obtained either directly as functions of the eigenvalues of a symmetric matrix ([180, 147]) or from solving a least-squares problem ([130, 122]). Alternatively, the trapezoid rule is applied in [188] to get a discrete approximation of the mean curvature M expressed as the integral over the normal curvatures κ_N, the Gaussian curvature K is obtained from a similar integral over κ_N^2, then M and K define the principal curvatures. Exact quadrature formulas for curvature estimation are provided in [107].

Another class of techniques for curvature tensor estimation locally fits a smooth parametric surface patch and then derives the differential quantities from that. This leaves the choice for the surface – typically polynomials of low degree – the geometric quantities to interpolate or approximate – e.g., the vertex positions in a 1-ring neighborhood – and a projection operator to obtain a parameterization – in general the projection into the tangent plane. A straightforward choice is to consider the quadratic height surface

$$z(x, y) \;=\; \tfrac{1}{2}a_{20}x^2 + a_{11}xy + \tfrac{1}{2}a_{02}y^2 \;,$$

for a local coordinate system spanned by the normal N_i (in z-direction) and two orthogonal tangent vectors (in x- and y-direction) and with origin $X_i = 0$ [64]. Then the parameters a_{20}, a_{11}, and a_{02} obtained as a least-squares solution are the elements of the symmetric matrix defining the Weingarten map. This can be interpreted as estimating the normal curvature from parabolas rather than circles (as with (2)) and then solving a least-squares system like in [122].

In [189] a quadratic Taylor polynomial of different form is applied, namely

$$X(u, v) \;=\; X_u u + X_v v + \tfrac{1}{2}u^2 X_{uu} + X_{uv}uv + \tfrac{1}{2}v^2 X_{vv} \;.$$

The coefficients of the local least-squares approximating polynomial are the first and second order partials and hence define the fundamental forms. For robustness reasons, an exponential map is used as projection operator rather than a simple projection to the tangent plane.

The use of a cubic approximation scheme which takes into account vertex normals in the 1-ring is proposed in [64]. As the normals themselves are local estimates, this effectively enlarges the neighborhood. Again, a least-squares problem is solved to find the coefficients of a cubic height surface, where the Weingarten matrix is obtained entirely from the quadratic terms in the same way as before.

In general, least-squares methods may suffer from degenerate cases – even for reasonable geometric configurations – which lead to ill-conditioned system matrices. In [189] the polynomial basis is successively reduced in such cases. An alternative is to provide more samples e.g. from linear interpolation. In [24] the patch fitting approach is discussed from an approximation theory point of view including robustness and numerical issues. For high-quality and consistent estimation of curvatures and their derivatives, [145] applies a (rather expensive) global fitting of an implicit surface to the surface mesh.

In contrast to the previously mentioned techniques, tensor averaging methods estimate the curvature tensor as an average over a certain region of a polyhedral mesh. In [30] the curvature tensor is derived building upon the theory of normal cycles. This work includes a proof of convergence under certain sampling conditions based on geometric measure theory. The curvature tensor is defined at each point along an edge, and all contributions are integrated over a small region, see also [2]. A similar discrete curvature measure is applied in [80].

Alternative approaches locally consider a triangle with given vertex normals. In [167], the directional derivatives of the normal are expressed as finite differences for every edge of a triangle. The resulting system of six equations is set up from the vertex positions (in parameter space) and normals and then solved for the three unknowns of the Weingarten matrix in least-squares sense. The tensors which are obtained per triangle are transformed to a common coordinate system to get a per-vertex average over the 1-ring. The algorithm can be applied with only slight modifications to compute curvature derivatives from the prior result.

In [181] the curvature tensor is estimated as smooth function (rather than a constant value) per triangle. This technique is inspired by Phong shading [156], where the vertex normals are linearly interpolated over the triangle. These interpolated normals are used to define the first and second order partials of the unit normal. This yields a piecewise smooth function defining the curvature tensor and elegant expressions for the Gaussian and mean curvature. Although this function is in general not continuous over edges of the triangulation, the approximation error is comparable to other approaches. For the estimation at vertices, the error is reduced by taking averages from all incident triangles.

3.3 Applications to Discrete Shape Analysis

The techniques reviewed in the previous section enable the estimation of curvature on discrete shapes: curvature estimates such as principal curvatures, Gaussian curvature and mean curvature are available at every vertex. These values can then be linearly interpolated in triangles. This is illustrated in Figure 2(a) and (b) where M and K are color coded. For efficient visualization (scaled) curvature values are used as 1D texture coordinates such that linear interpolation is done by the graphics hardware. Principal curvature directions define a vector field on the surface. Figure 2(c) and (d)

(a) (b) (c) (d)

Fig. 2. Visualization of mean curvature M (a) and Gaussian curvature K (b) estimated on the Feline triangle mesh. Here, red, green and blue denote positive, zero and negative values, respectively, and lighting is enabled. (c) and (d) show the maximum curvature with lines of curvature on the Mannequin mesh.

shows lines of curvature obtained from stream line integration (see Figure CP-1 in Appendix A).

In addition to these examples, many surface interrogation methods which were initially developed for smooth surfaces can be adapted easily to work in the discrete setting. This applies to first order analysis (Section 4) using estimates of the surface normal: reflection lines can be simulated by environment mapping techniques, highlight lines and isophotes can be emulated similarly. With curvature estimates being available, second order analysis (Section 5) can be applied. For the computation of discrete characteristic lines (Section 6), curvature derivatives are approximated by appropriate differences.

The following sections discuss shape analysis of smooth surfaces. Interrogation of discrete shapes follows the general ideas closely and applies estimates of surface normals and curvature.

4 First-Order Shape Analysis

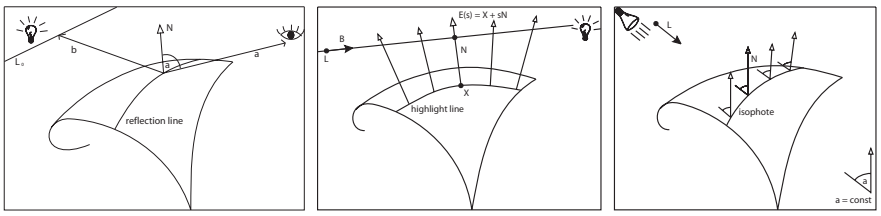
First-order surface interrogation methods make generally use of the surface normal vector by simulation of particular light reflecting behavior of the surface. The light reflection methods all simulate the special reflection behavior of light sources or light lines on the surface. Due to the intuitive understanding that everybody has when he observes light reflections, these methods are very effective in detecting surface irregularities. They are therefore very well suitable for testing the fairness of surfaces. Because the surface normals are involved in the computation of these lines, they also can be used to visualize first order discontinuities, like tangent discontinuities.

4.1 Reflection lines

The reflection line method determines unwanted dents by emphasizing irregularities in the reflection line pattern of parallel light lines. Let $X(u, v)$ be a representation of the surface to investigate, and let $N(u, v)$ be the unit normal vector of the surface. Furthermore a light line L is given in parameter form:

$$L(t) = L_0 + t \cdot \mathbf{s}$$

where L_0 is a point on L, s is a vector defining the direction of L, $t \in \boldsymbol{R}$. The reflection line is the projection of the line L on the surface X, which can be seen from the fixed eye point A, if the light line L is reflected on the surface, see Figure 3(a). From geometric dependencies the following reflection condition is derived:



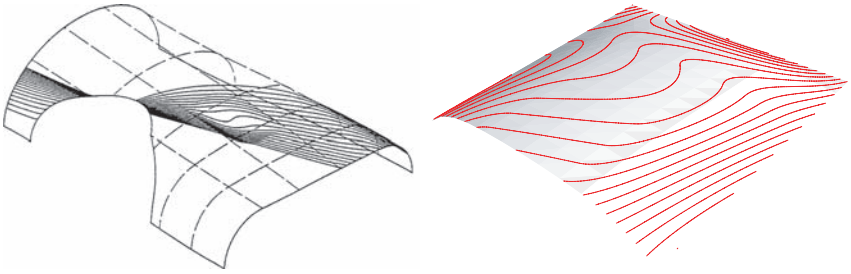
(a) Reflectlion line (b) Highlight line (c) Isophote

Fig. 3. First order shape analysis by simulating light reflection.

$$\mathbf{b} + \lambda \mathbf{a} = 2 \Big(N(u, v) \cdot \mathbf{b} \Big) N(u, v) \quad \text{with} \quad \lambda := \frac{\|\mathbf{b}\|}{\|\mathbf{a}\|}, \tag{3}$$

where $\mathbf{a} = P - A$, $\mathbf{b} = L - P$. Equation (3) has to be solved for the unknown parameters u and v of the reflection point P. These three non-linear equations can be reduced to two equations by eliminating λ; they can then be solved by numerical methods, but the existence and uniqueness of solutions has to be ensured by an appropriate choice of the eye point A [94, 98]. To analyze visually the surface one uses

a set of parallel reflection lines with direction $\mathbf{s}$, a fixed eye point A, and one steps along each curve of the set. Figure 4(a) shows a reflection line pattern on a part of a hair dryer and visualizes some surface irregularities.



(a) Reflectlion lines (b) Isophotes

Fig. 4. Pattern of computed reflection lines and isophotes on NURBS surfaces.

4.2 Highlight lines

A highlight line is defined as the loci of all points on the surface where the distance between the surface normal and the light line is zero. The linear light source idealized by a straight line with an infinite extension

$$L(t) = L_0 + Bt$$

(L_0 is a point on L, B is a vector defining the direction of L, $t \in \boldsymbol{R}$), is positioned above the surface under consideration, see Figure 3(b). The highlight line method also detects surface irregularities and tangent discontinuities by visualizing special light reflections on the surface. In comparison with the reflection line method, the highlight lines are calculated independently from any observers view point. For a given surface point $X(u, v)$ let $N(u, v)$ be the unit normal vector. The surface point $X(u, v)$ belongs to the highlight line if both lines, $L(t)$ and the extended surface normal

$$E(s) = X(u, v) + s \cdot N(u, v)\,, \quad s \in \boldsymbol{R}$$

intersect, i.e. if the perpendicular distance

$$d = \frac{\|[B \times N] \cdot [L_0 - X]\|}{\|[B \times N]\|}$$

between these lines is zero, see Figure 3(b). This method can be extended to highlight bands, lines where $d \leq r$ (r fixed) is verified. For details on the algorithms to compute highlight lines see [7].

4.3 Isophotes

Isophotes are lines of equal light intensity. If $X(u, v)$ is a parameterization of the surface and L the direction of a parallel lighting, then the isophote condition is given by:

$$N(u, v) \cdot L = c\,,$$

where $c \in \boldsymbol{R}$ is fixed, see Figure 3(c). Note that silhouettes are special isophotes ($c = 0$) with respect to the light source. **Isoclines** are lines of equal normal inclination with respect to some direction V. If $X(u, v)$ is a parameterization of the surface, then the isocline condition is given by:

$$N(u, v) \cdot V = c$$

where $N(u, v)$ is the unit normal field of X and $c \in \boldsymbol{R}$ is fixed. In other words, isophotes are isoclines with respect to the light source direction. Similar to reflection lines and highlight lines, the isophotes provide a powerful tool to visualize small surface irregularities, which can not be seen with a simple wire-frame or a shaded surface image. In Figure 4(b) we use 20 different values for c in order to get an isophote pattern on a NURBS test surface.

Now, as stated out in the introduction of this section, the light reflection methods can be used to visualize first and second order discontinuities, because the surface normal vector is always involved in the line definitions. In fact, *if the surface is C^r-continuous, then the isophotes are C^{r-1}-continuous curves* (see [157] for more details). A curvature discontinuity can be recognized, where the isophotes possess tangent discontinuities (breaks). One should nevertheless be careful by using isophotes for this purpose, because sometimes the break points of the isophotes at curvature discontinuities may not be clearly recognized, because of an ill-conditioned light direction. This special case occurs if the orthogonal projection of the light direction L in the tangent plane at a boundary point $X(u, v)$ is parallel to the tangent of the isophote at this point.

Isophotes for curvature discontinuity:

There is another isophote method, which on one hand is an automatic method (independent of a special light direction), but which on the other hand only visualizes curvature discontinuities across the boundaries of a patch work. It makes use of the fact that along a common boundary curve y between two surface patches that join only with tangent plane continuity the Dupin indicatrices i_1 and i_2 on both sides are different. In general there are two conjugate diameters of the Dupin indicatrix. This relation degenerates at parabolic points, because the asymptotic direction (the direction in which the normal section curvature vanishes) is the conjugate to itself, but also conjugate to all other directions. At planar points, we have this degeneration for each (tangent) direction. Since both patches have a common boundary curve, and the tangent planes along that curve are unique, the Dupin indicatrices i_1, i_2 have a common diameter, but differ in the other.

We now consider an isophote c passing through P. The tangent t_i of c at P with respect to X_i is conjugate to the orthogonal projection f of the light ray onto the

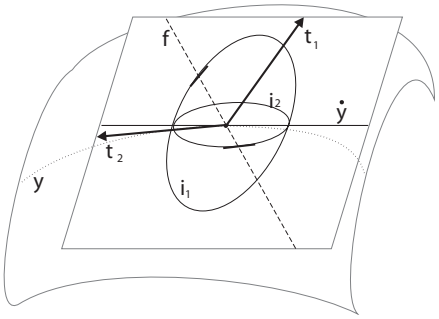


Fig. 5. Isophotes for curvature discontinuity.

tangent plane ($i = 1, 2$), see Figure 5. In general the isophote c shows a tangent discontinuity at P if the Dupin indicatrices of X_1 and X_2 are not equal, but we have to avoid the situations $f = \dot{y} = t$ and $f = t'$. More details can be found in [161].

4.4 Detection of inflections

Orthotomics and the polarity method are both interactive interrogation tools capable to detect only one particular type of surface "imperfection": the change of the sign in the Gaussian curvature. For example, surface with only convex iso-parameter lines are not necessarily convex, i.e. their Gaussian curvature is not required to be positive at all surface points. Such surface imperfections are difficult to detect visually in this case and therefore a curvature based surface analysis is needed like color maps or generalized focal surfaces, see Section 5. The following methods in contrast can visualize a change of sign in the Gaussian curvature without computing second order derivatives of the surface.

Orthotomics

In [85] it has been shown that for a regular surface $X(u, v)$ and for a point P that does not lie on the surface or on any tangential plane of the surface the k-orthotomic surface $Y_k(u, v)$ with respect to P defined by

$$Y_k(u, v) = P + k\Big((X(u, v) - P) \cdot N(u, v)\Big)\, N(u, v)\,,$$

where $N(u, v)$ is the unit normal vector of the surface has a singularity in (u_0, w_0), if and only if the Gaussian curvature of X vanishes, or changes its sign at this point. To illustrate this method we consider a Bézier surface with completely convex parameter lines, see in Figure 6(left). But this surface is not convex: as shown in Figure 6(right), the orthotomic analysis emphasizes the change of sign of the Gaussian curvature in the corner region.

Polarity method

The polarity method is a further method able to detect unwanted changes in the sign

Fig. 6. Bicubic surface patch with line of vanishing Gaussian curvature (left). Orthotomic analysis (right).

of the Gaussian curvature without computing second order derivatives of the surface. It works for curves as well. It uses the polar image of a curve or surface, where the singularities (cusps, edge of regression) of this image indicate the existence of points with vanishing Gaussian curvature. The polar surface looks similar to the orthotomic surface, because the center of polarity is chosen to be equal to the projection point of the orthotomic analysis. For more information about the polarity method and on how removing the inflections see [86].

4.5 Geodesic paths on surfaces and meshes

Geodesic paths, or simply geodesics, on a surface are surface curves which connect two surface points with minimum path length. A thorough study of geodesics and their role of in surface interrogation requires much more attention than the present overview can provide. So below we give only a brief literature survey.

Geodesics deliver rich information about surface geometry and, therefore, have various theoretical and practical applications. In particular, detecting geodesic paths on surfaces approximated by triangles meshes is a common operation for many graphics and modeling tasks such as mesh parameterization [103], mesh segmentation [93], skinning [175], mesh watermarking [162], and mesh editing [100].

A rigorous mathematical treatment of geodesics can be found in [102, 42]. Some numerical aspects are presented in [56]. An algorithm for approximate computation of geodesic paths on smooth parametric surfaces has been explored in [155, 154]. Various algorithms exit for computing geodesic paths and distances. The so-called MMP algorithm [124] computes an exact solution for the "single point, all distances" shortest path problem by partitioning each mesh edge into a set of intervals over which the exact distance can be computed. In [179] an accelerated implementation of this algorithm is presented. An algorithm to solve the "single source, single distance" geodesic problem is given in [91]. See also [123] for a broad survey of algorithms for computing shortest paths on graphs.

5 Second-order shape analysis

Surface curvature is of central importance for surface design. Often the result is required to be mathematically smooth (continuous in the 2nd derivative) and aesthetically pleasing, i.e. have smooth flowing highlights and shadows. To obtain an

aesthetically pleasing shape, the designer works with the curvature. A color map (see Section 5.5) can be used to visualize curvature (Gaussian, principal curvatures) over the surface. The problem is the good choice of the color scale, which depends on the curvature function and therefore on the underlying surface.

The surface interrogation methods presented in this section are therefore **curvature analysis** tools which are able to detect all surface imperfections related to curvature, like bumps, curvature discontinuity, convexity, and so on.

5.1 Local shape analysis with Gaussian curvature

Let us look at a smooth surface in a neighborhood of one of its point.

The simplest classification of local surface shapes is given by the the sign of the Gaussian curvature $K = \kappa_1 \cdot \kappa_2$.

- $K > 0$. The normal curvatures $\kappa_N(\varphi)$ has the same sign in all directions, so the tangent plane touches the surface at one point. The usual convex or concave regions corresponding to this, as demonstrated by the left image of Figure,7 and the left images of Figure 8.
- $K < 0$. The normal curvature becomes zero twice during the half rotation of the normal plane around the normal. The tangent plane intersects with the surface in these directions of zero curvature. The surface is locally saddle-shaped, as seen in middle images of Figure 7 and Figure 8.
- $K = 0$. At least one principal curvature is zero. It produces a parabolic point. See the right image of Figure 7 and the middle-right image of Figure 8. A set parabolic points may form a parabolic region shown in the right image of Figure 8.

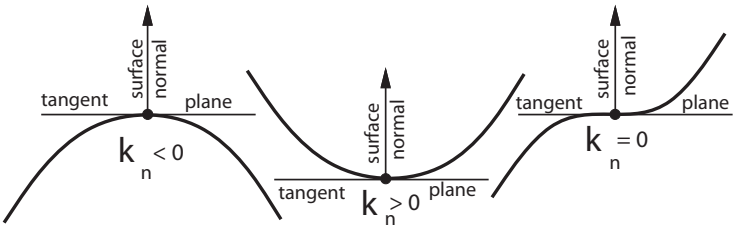


Fig. 7. Local shape of normal section curve is defined by curvature.

The Gaussian curvature of a surface can be expressed through the coefficients of the first fundamental form. Thus we arrive at the following famous result called Gauss's Theorema Egregium: the Gaussian curvature of a surface is a bending invariant.

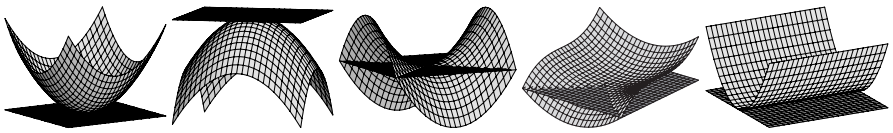


Fig. 8. Gaussian curvature determines local shape of surface. Left images: convex and concave regions ($K > 0$). Middle: saddle-shaped region ($K < 0$). Middle-right: a parabolic point ($K = 0$) Right: a region consisting of parabolic points.

Now let us consider a simple geometrical interpretation of the Gaussian curvature, by means of which Gauss originally introduced it.

Consider a two-sided surface in three-dimensional space. Let us transport the positive unit normal vector from each point of the surface to the origin. The ends of these vectors lie on the unit sphere. We obtain the mapping of the surface into the unit sphere, see Figure 9. It is called the **Gauss map**.

The Gauss mapping takes areas on surfaces to areas on the unit sphere. Consider the unit surface normals at the surface points within the area ΔS on the surface. Let us denote the area on the unit sphere (solid angle) corresponding to ΔS by ΔA. It turns out that the Gaussian curvature at the point is the limit of the ratio of these areas:

$$K = \lim_{\Delta S \to 0} \frac{\Delta A}{\Delta S}.$$

This remarkable formula resembles the definition of the curvature of the plane curves: $\kappa = d\varphi/ds$.

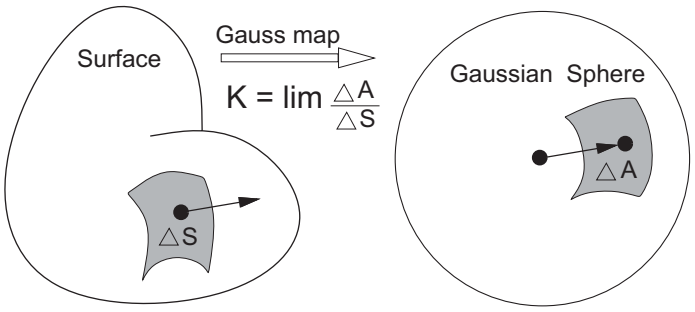


Fig. 9. Gauss map and geometric meaning of Gaussian curvature.

The Gauss map can be used for detecting spherical, cylindrical, and conical regions on a surface [12].

5.2 Focal Surface and Corresponding Surface Features

For a smooth surface $X = X(u, v)$ its **focal surface** is given by

$$X_F(u,v) = X(u,v) + \frac{N(u,v)}{\kappa(u,v)}, \qquad \kappa = \kappa_1,\, \kappa_2,$$

where $N(u,v)$ is the oriented normal. The focal surface is formed by the principal centers of curvature and consists of two sheets corresponding to the maximal and minimal principal curvatures κ_1 and κ_2. One can show that the focal surface is the envelope of the surface normals. In geometrical optics [77], a **caustic** generated by a family of rays is defined as the envelope of the family. Thus the focal surface is the caustic of the family of surface normals. The focal surface can be also defined as a surface swept by the singularities of the offset surfaces $O_d(u,v) = X(u,v) + d\,N(u,v)$.

The focal surface is the 3D analogue of the evolute of a planar curve and has singularities. The singularities of the focal surface consist of space curves called **focal ribs**.

Ridges, the surface curves corresponding to the focal ribs are natural generalization of the curve vertices for surfaces. The ridges can be defined as sets of surface points where the principal curvatures have extremes along their associated principal directions and points where the principal curvatures are equal (**umbilics**). A thorough study of the ridges and their properties is conducted by Porteous [158]. See also [72] where a detail classification of the ridges is presented. Below we briefly discuss the ridges from a singularity theory point of view.

Near a point on a focal rib the focal surface can be locally represented in the parametric form $(c_1 t^3, c_2 t^2, s)$, where $c_1 \neq 0$ and $c_2 \neq 0$, in well chosen coordinates (s,t). The focal ribs themselves have singularities at points corresponding to the umbilics and those ridge points where one of the principal curvatures has an inflection along its corresponding curvature line. Generic (typical) singularities of the focal surface are shown in Figure 10.

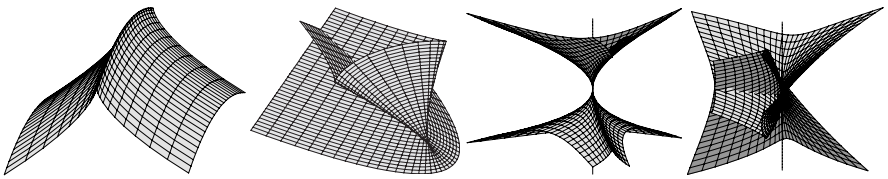


Fig. 10. Typical singularities of the focal surface. From left to right: cuspidal edge (rib), swallowtail, pyramid, purse. At the swallowtail singularity the rib has a cusp. The pyramid and purse correspond to the umbilical point on the surface. The vertical lines at the bottom images are the surface normals at the corresponding umbilics.

The umbilics and ridge points can be also characterized as surface points where the osculating spheres (spheres of curvature) have high-order contacts with the surface. Therefore the umbilics and ridges are invariant under inversion of the surface with respect to any sphere.

The focal surface points can be also described in terms of degenerate singular points of distance functions. Given a surface and a point in 3D, let us consider the

distance function from the point and restrict the function onto the surface. This gives a three-dimensional family of distance functions defined on the surface and parameterized by points in 3D. Now the focal surface is generated by those point-parameters for which the distance function has degenerate critical points. A typical degenerate critical point has on of the following two forms $\pm s^2 + t^3$ in proper coordinates s and t. If the point-parameter is a typical point on a focal rib, the distance function has a critical point in one of the following four forms: $\pm s^2 \pm t^4$. More degenerate critical points occur when the point-parameter is located either at a swallowtail singularity of the focal surface or at an umbilical points. It is interesting that the **cut locus** of the surface [190] (skeleton or medial axis of a figure bounded by the surface) consists of those point-parameters which define the distance functions with two equal global minima. Thus, as illustrated in Figure 11, the edges of the skeleton are located at focal ribs.

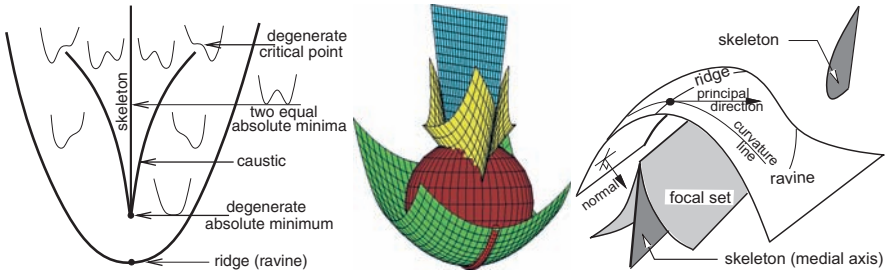


Fig. 11. Left: zoo of distance functions; thin lines are used to sketch typical profiles of the surface functions defined by the distance from a given point to the surface points. Center: the skeleton (blue), caustic (yellow), ridge (red) and an osculating sphere (brown) at a ridge point of the elliptic paraboloid. Right: schematic illustration of relationships between the cut locus, focal surface and ridges.

The focal surface possesses many interesting properties. For example, for each line of curvature on a surface there is a corresponding line on the corresponding sheet of the focal surface. It can be shown that those raised lines of curvature are geodesics on the focal surface [159, 131].

In [118, 99] umbilics are used for shape interrogation and shape matching purposes. Statistics of various types of umbilics on random surfaces computed and analyzed in [11] may have have many potential applications for for inspecting and interrogating surface properties.

5.3 Hedgehog diagrams and curvature plots

The hedgehog diagrams and curvature plots are well known interrogation tools for planar curves [6, 54]. A hedgehog diagram for planar curves visualizes the curve normals proportional to the curvature value at some curve points. A new curve is obtained by $\tilde{X}_{hedgehog}(t) = X(t) + \kappa N(t)$ thus visualizing curvature distribution and

discontinuity. The inspection of surfaces with these methods can be done by applying them to planar curves on the surface (intersections of the surface with planes). [97] shows an example of application. Hedgehog diagrams for entire surfaces are nevertheless difficult to interpret and are therefore not to be recommended.

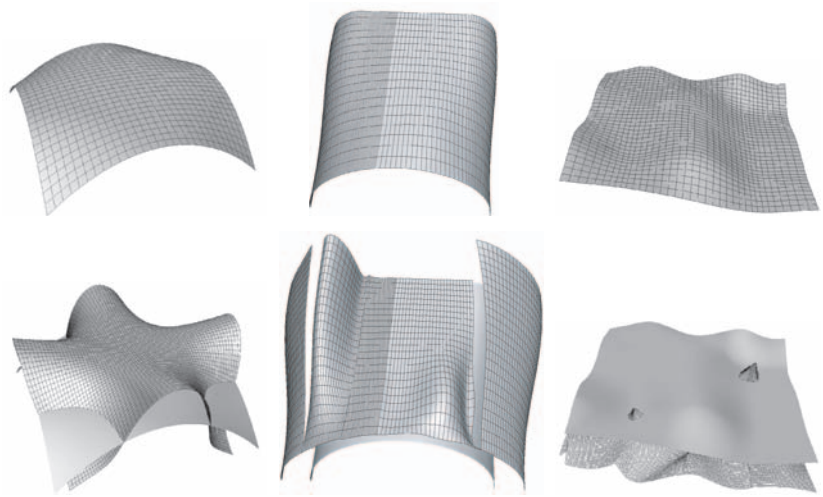
5.4 Generalized focal surfaces

Although the idea of generalized focal surfaces is quite similar to hedgehog diagrams, their application area is much larger. Instead of drawing surface normals proportional to a function value, only the point on the surface normal proportional to the function is drawn. The loci of all these points is the **generalized focal surface**. This method was introduced by [71], and is based on the concept of focal surfaces which are known from line geometry, introduced in Section 5.2. The generalization of this classical concept leads to the generalized focal surfaces:

$$F(u,v) = X(u,v) + s \cdot f(\kappa_1, \kappa_2) \cdot N(u,v)\,, \quad \text{with } s \in \boldsymbol{R}$$

where N is the unit normal vector of the surface X. f is a real valued function of the parameter values (u, v).
The variable offset function f can be any arbitrary scalar function, but in the context of surface interrogation it is quite natural to take f as a function depending on the principal curvatures κ_1, κ_2 of X, f.ex. $f = \kappa_1\kappa_2$ Gaussian curvature, $f = \frac{1}{2}(\kappa_1 + \kappa_2)$ mean curvature, $f = (\kappa_1^2 + \kappa_2^2)$ energy functional, $f = |\kappa_1| + |\kappa_2|$ absolute curvature, $f = \kappa_i$ principal curvatures, $f = \frac{1}{\kappa_i}$ focal points, $f = const$ offset surfaces. This not only enables to visualize a particular curvature behavior, but it can interrogate and visualize surfaces with respect to various criteria: A **convexity test** can be performed using the Gaussian curvature offset $f = \kappa_1 \cdot \kappa_2 = K$. A surface is locally convex at $X(u, v)$, if the Gaussian curvature is positive at this point. Often a surface is called non-convex, if there is a change in the sign of the Gaussian curvature. the two surfaces $X(u, v)$ and $F(u, v)$ intersect at the parabolic points, see Figure 12(a). The generalized focal surface therefore pin points directly on the area where the sign of K changes in contrast to orthotomics (Section 4) which are also used to test the convexity. **Flat points** which are special umbilic points with $\kappa_1 = \kappa_2 = 0$ can be detected using $f = |\kappa_1| + |\kappa_2|$ as well as $f = \kappa_1^2 + \kappa_2^2$. Flat points are undesired surface points because they make the surface bumpy. **Curvature discontinuity** can be visualized through gaps in the surface F with $f = \kappa_1^2 + \kappa_2^2$ since it is a second order surface analysis tool, see Figure 12(b). **Visualizing surface irregularities:** Surfaces are aesthetically pleasing if they have "nice" light reflections. Thus similar to reflection lines the generalized focal surfaces are also a tool for visualizing such surface imperfections because they are very sensitive to small irregularities in the shape. In Figure 12(b) part of a hair dryer is shown. It consists of biquintic C^1-continuous patches. The iso-parametric lines do not reflect the bump in the surface, which is however emphasized by the focal analysis. Another aspect of surface analysis is the visualization of **technical aspects**. A surface which should be treated by a spherical cutter is not allowed to have a curvature radius smaller than the

radius of the cutter R_{cutter}. The generalized focal surfaces are able to detect such undesired regions by intersection with the surface X. The offset function to choose in this special case, is $f = \frac{1}{R_{cutter}} - \kappa_{max}$. Figure 12(c) shows such a surface which is not allowed to be cut. Generalized focal surfaces not only visualize surface imperfections, they also give the user a 3D impression of the relative amount of the offset function over the surface, what color maps can't do.



(d) Convexity test (e) Imperfections and curvature discontinuity (f) Milling test

Fig. 12. Second order surface analysis with generalized focal surfaces.

5.5 Color mappings

Color is used to emphasize features on the surface. Texturing can emphasize the spatial perception of an 2D image of the surface. A **color-coded map** is an application, which associates to a scalar function value a specific color. The color scale presents an even gradation of color corresponding to the range of function values. Colors are principally used to visualize either continuously or discontinuously any scalar function over a surface [38, 5, 4, 59], like pressure, temperature, or curvature, see Figure 13 (see Figure CP-2 in Appendix A). Colors are used as a fourth dimension and show the user immediately and quantitatively how the function varies over the surface.

An even gradation of the linear or cyclic color coding is important to visualize the rapid curvature variation by the presence of color fringes. Beck et al. [5] propose to use the HSI (hue, saturation, intensity) model and to perform transformations between this space and the three primary colors RGB. See [58] for more details on color spaces and transformations. An example of discrete color-coding of the interval [0,1] is the following one:

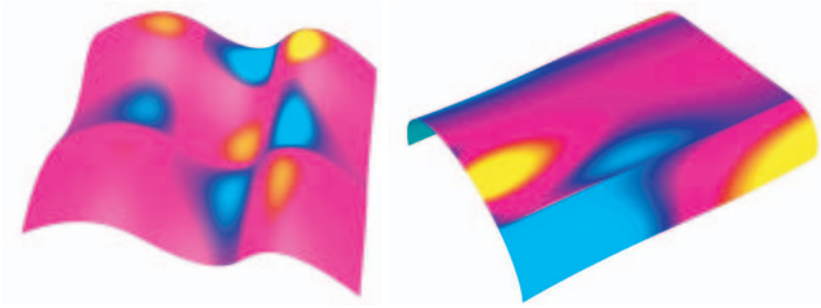


Fig. 13. Color codings of Gaussian curvature.

Interval	Red	Green	Blue	Color
0.0 - 0.2	1	0	0	red
0.2 - 0.4	1	1	0	yellow
0.4 - 0.6	0	1	0	green
0.6 - 0.8	0	1	1	turquoise
0.8 - 1.0	0	0	1	blue

The main difficult of this simple interrogation method is the choice of a convenient color scale, which obviously depends on the function values to be visualized.

Pseudo texture
The use of colors for displaying a surface helps to emphasize the 3D understanding of an 2D image by simulating shadows, perspective and depth of the object. An artificial texturing is an aid for visualizing rendered surfaces. Isoparametric lines are commonly used, but they are in some situations ambiguous. Schweitzer [170] projects equally spaced dots of equal size over the surface in order to increase the visual perception of the form.

6 Characteristic lines

Drawing lines on surfaces is a powerful and widely used tool for analysis and visualization of surface features. The techniques of **isolines**, **lines of curvature**, **geodesic paths** and **ridges** are presented. Numerous graphical examples are illustrated in [159, 56]. In the last three cases a set of lines on the surface can be created, and should be interpreted with the knowledge of differential geometry. They are the most sophisticated tools from the mathematician's point of view. The user should interpret the lines of curvature or the geodesic paths.

6.1 Isolines

Isolines are lines of a constant characteristic value on the surface. They provide an interrogation tool with a wide variety of applications. They help analyzing surface characteristics, and they are used to visualize the distribution of scalar quantities over the surface. The visualization of a certain number of isolines, with respect to an even distribution of the characteristic values allows to study the behavior of these values.
Contour lines are planar lines on the surface which are all parallel to a fixed reference plane. Closed contour lines indicate maxima and minima of the surface with respect to the direction given by the plane's normal vector [76, 5]. Saddle points appear as "passes". The contour lines only cross in the exceptional case of a contour at the precise level of a saddle point. [141] describes systematically the distribution of other critical points on a surface. A disadvantage of contour lines is the fact that they are costly to compute. Several surface contouring methods exist, which are sometimes depending of the specific surface formulation [152, 169, 108]. Hartwig and Nowacki [76] propose to subdivide the surface into sufficient small pieces which are then approximated by bilinear surfaces. Then the contour lines can easily be computed.
Iso-contouring is the technique of extracting constant valued curves and surfaces from 2D and 3D scalar fields. Interactive display and quantitative interrogation helps understanding the overall structure of a scalar field and its evolution over time. Traditional iso-contouring techniques examine each cell of a mesh to test for intersection with the iso-contour of interest. For an overview see [168]. Extraction of isosurfaces from 3D scalar field is generally be done by the Marching Cubes algorithm and its variants [111, 143, 27]

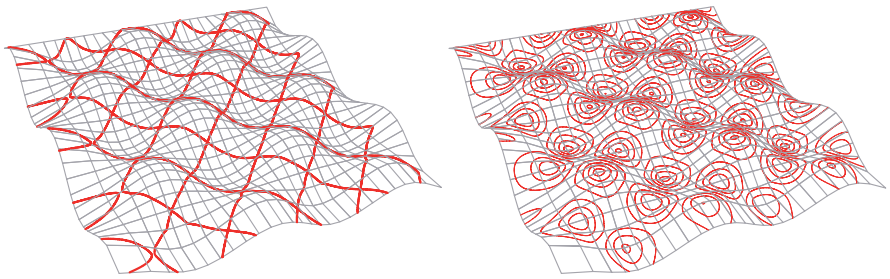


Fig. 14. Gaussian curvature isoline. Left: parabolic lines. Right: isolines corresponding to different constant Gaussian curvature values.

Parabolic lines are isolines of zero Gaussian curvature on the surface. They are of particular interest for intrinsic surface interrogation, since they divide the surface into elliptic and hyperbolic regions and they reflect therefore the local curvature behavior of a surface. Parabolic lines are special Gaussian curvature lines, see Figure 14. In [79] a more complex example with the statue Apollo Belvedere is drawn.

6.2 Lines of curvature, umbilics

Lines of curvature are curves whose tangent directions coincide with those of the principal directions, which are orthogonal. They form therefore an orthogonal net on the surface.

The net of lines of curvature becomes singular at an umbilical point where κ_1 and κ_2 are identical and the principal directions are indeterminate. Some numerical integration method is used to calculate the lines of curvature. But the integration process becomes unstable near an umbilic. Unfortunately umbilics appear frequently on free-form surfaces. A recent work about umbilics [117], destined for use in CAGD (Computer Aided Geometric Design), presents a procedure to compute the lines of curvature near an umbilic. And in [116] a computational method to locate all isolated umbilics on parametric polynomial surfaces is described. The discrete field of principle curvature directions computed on a surface mesh has been used for remeshing [2]. More details about umbilics and lines of curvature figures are found in classical differential geometry literature [35], or in a more recent book [159].

6.3 Curvature Extrema for Shape Interrogation

Surface features invariant under rotations, translations, and scaling are important for studying shapes of 3D objects. The ridge curves discussed briefly in Section 5.2 are among the most important view- and scale-invariant features of a smooth surface.

The ridges are defined as the extremes of the principal curvatures along their corresponding curvature lines and constitute powerful surface descriptors. They have been intensively studied in connection with research on the accommodation of the eye lens [69], structural geology [163] and geomorphology [109], human perception [83], image analysis [191, 129, 127, 40, 110], quality control of free-form surfaces [84], reverse engineering [87], analysis and registration of anatomical structures [68, 67, 151], face recognition [72], and non-photorealistic surface rendering [89, 114, 36]. (See also references therein.)

An explanation of why some ridges are good for sketching complex 3D shapes can be found in [191]: given a grey-scale image of an illuminated 3D object, under general illumination and reflection conditions, the zero-crossings of the second directional derivative of the image intensity along the direction of the image intensity gradient occur near the extremes of the principal curvature along their principal directions. Thus the projections of ridges onto the image plane are usually located near edges, the most salient image features.

Some subsets of ridges play an important role in perceptual shape organization. Human perception experiments suggest the so-called minima rule [83] which sets region boundaries along lines divides shapes into parts at negative minima of the principal curvatures along their lines of curvature. The minima rule was employed in [146] for mesh segmentation purposes.

The ridges on a surface have interesting relations with the skeleton (medial axis) of a figure bounded by the surface and can be described via high-order contacts

between the surface and its osculating spheres. See [158, 101, 192, 159, 8], [72, Chapter 6], and recent reviews in [26, 25] for rigorous mathematical treatments revealing beautiful properties of these curvature features. Surface landmarks associated with the ridges were considered in [101, 131, 160]. Bifurcations of the ridges on dynamic shapes were studied in [159, 15, 16, 160, 112].

Recently the so-called **crest lines**, a subset of the ridges consisting of the extremes of the principal curvature maximal in absolute value along its corresponding curvature line, draw much attention because of their ability to represent surface creases [184, 127, 151, 177, 145, 81]. See also references therein. One motivation for describing surface creases as the crest lines is based upon the following analogy with edges of grey-scale images [145]. Consider a surface and its Gauss map which associates with every point $\mathbf{p}$ of the surface the oriented normal vector $\mathbf{n}(\mathbf{p})$. The derivative $\nabla\mathbf{n}(\mathbf{p})$ (Jacobian matrix) of the Gauss map measures the variation of the normal vector near $\mathbf{p}$, i.e., how the surface bends near $\mathbf{p}$. It is easy to see that the eigenvalues and eigenvectors of $\nabla\mathbf{n}(\mathbf{p})$ are the principal curvatures and principal directions of the surface at $\mathbf{p}$, respectively. Thus the maximal variation the surface normal is achieved in the principal direction of the principal curvature maximal in absolute value. So it is natural to define surface creases as loci of points where the positive (negative) variation of the surface normal in the direction of its maximal change attains a local maximum (minimum). Figure 16 shows the crest lines detected on various models represented by dense triangle meshes (see Figure CP-3 in Appendix A).

Practical detection of the ridges and their subsets is a difficult computational task since it involves estimating of high-order surface derivatives. Various techniques were proposed for detecting the ridge lines and their subsets on

- surfaces in implicit form and isosurfaces of 3D images [158, 129, 128, 184, 182, 10, 13];
- surfaces approximated by polygonal meshes [113, 9, 188, 82, 177, 26, 25, 145, 81];
- height data [65, 96, 95, 109];
- surface given in parametric form [84, 75].

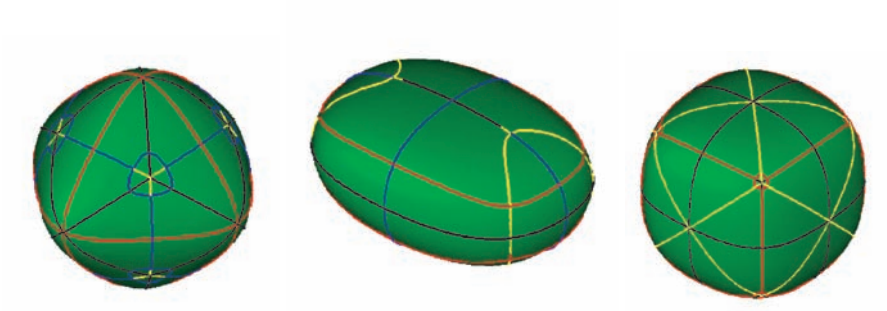


Fig. 15. Various types of ridges detected on smooth surfaces. The images are taken from [13].

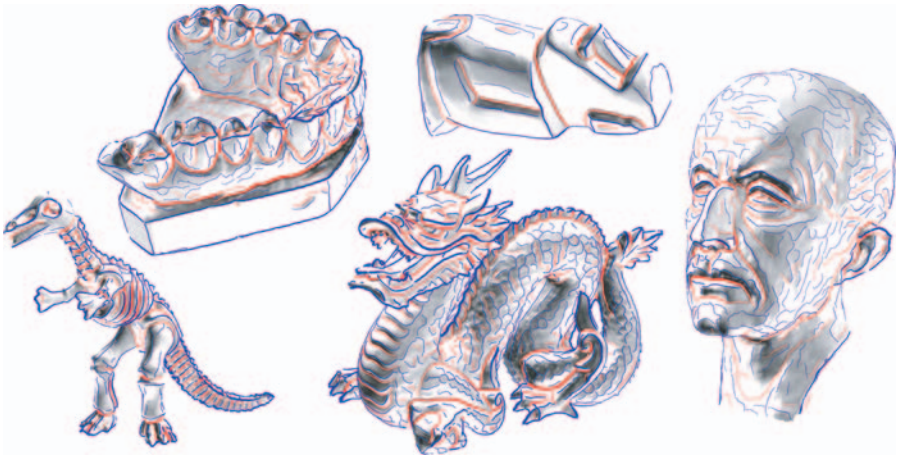


Fig. 16. The crest lines detected on various surfaces approximated by dense triangle meshes.

For shape interrogation purposes (shape quality control and analysis of aesthetic free-form surfaces), the ridges were used in [84, 78]. Moreton and Sequin [130] used the sum of the squared derivatives of the principal curvatures along their corresponding curvature lines as a measure of surface fairness.

Often, instead of the ridges and their subsets defined via extremes of the principal curvatures, simpler surface features are detected. In geometric modeling, there has been considerable effort to develop robust methods for detecting surface creases, curves on a surface where the surface bends sharply. Interesting methods for crease detection on dense triangle meshes and point-sampled surfaces were proposed in [87, 166, 88, 70, 178, 148, 150]

Whereas the ridges were first studied one hundred years ago [69] and have rich history [159], the so-called **sub-parabolic lines**, the loci of points where one of the principal curvatures has an extreme value when moving along the curvature line corresponding to another principal curvature. The sub-parabolic lines were introduced in [17] and studied in [159, 131, 160]. They possess many remarkable properties: the sub-parabolic lines correspond to the parabolic lines on the focal surface, hence the name, and consist of geodesic inflections of the lines of curvature [131]. The sub-parabolic lines can be also detected by examining the profiles of surfaces [131].

6.4 Special Surface Points

In this section, following [131] we consider special surface points which lie on the ridges and sub-parabolic lines. We adapt the color scheme proposed by Porteous [158, 159]. Let us give the principal curvatures and corresponding principal directions, parabolic lines, and sheets of the focal surface a color (red or blue) in order to distinguish between them. The red (blue) sub-parabolic line consists of the extremes of the red (blue) principal curvature along the blue (red) curvature line. The following surface landmarks are useful for surface interrogation purposes:

- **Umbilic points.** See [118, 149] for application of umbilics in surface matching and shape interrogation.
- **A ridge and sub-parabolic line of the same color cross.** The principal curvature of the same color takes an extreme value there (maximum, minimum, or saddle).
- **A ridge is tangent to the line of curvature of the same color.** These surface landmarks corresponds to the swallowtail singularities of the focal surface.
- **A ridge crosses a ridge of other color.** In [183] it was suggested to use these landmarks for 3D image registration.
- **A ridge crosses the parabolic line of the same color.** The Gauss map has the the so-called pleat singularity at such a point [101].

Koenderink [101] introduced two curvature-based measures of surface curvature: the **curvedness**

$$C = \frac{2}{\pi} \ln \left(\kappa_1^2 + \kappa_2^2 \right)$$

and the **shape index**

$$S = -\frac{2}{\pi} \arctan \frac{\kappa_1 + \kappa_2}{\kappa_1 - \kappa_2}.$$

These measures are often more convenient for practical purposes then the standard curvature descriptors $\{\kappa_1, \kappa_2\}$ and $\{M, K\}$, where K and M are the Gaussian and mean curvatures, respectively. In [142] it was suggested to use local maxima of the curvedness to define surface corner points.

7 Robust Symbolic based Shape Interrogation and Analysis

Interrogation of polynomial and rational surfaces could be made with the aid of symbolic processing. The advantage of the symbolic approach over sampling of properties, like curvature, at a discrete set of point stems from the ability to analyze the properties globally and provide global (error) bounds. Many properties of free-form geometry are differential and can be derived after executing a few basic operations over the polynomial or rational representation of the original interrogated curve $C(t)$ or surface $S(u, v)$, namely: differentiations, summations and products. We also assume the availability of a zero set finding tool, an operation that is equivalent to intersecting a polynomial or a rational function with a line in R^2 (a plane in R^3). As a simple example, consider the curvature field of a planar regular curve $C(t) = (x(t), y(t))$ that is equal to:

$$\kappa(t) = \frac{x'(t)y''(t) - y'(t)x''(t)}{(x'^2(t) + y'^2(t))^{2/3}}.$$

$\kappa(t)$ is not rational due to the fractional power in the denominator, in the normalization factor. Nonetheless, if one only seeks the *inflection* points of $C(t)$, only the numerator of κ needs to be considered. Then, the solution of the constraint of

$$x'(t)y''(t) - y'(t)x''(t) = 0 \tag{4}$$

finds all the inflection points in the regular planar curve $C(t)$, if any. In Equation (4), the problem of finding all the inflection points of a planar regular curve was reduced to that of finding a zero set. Differentiation and products were used to compute the inflection points' constraints.

Differentiation of piecewise polynomials and rationals is well known [28, 53]. Similarly, the addition of two (piecewise) polynomials that share a function space (same order and knot sequence) is realized by simply adding the corresponding coefficients. Two polynomials could be elevated to the same function space via knot insertion and degree elevations; see [28, 53] for more details. Products are the last operator we seek, an operation also required because of the quotient rules over addition and differentiation of rationals. Products are more complex to compute (see [43, 53]) but, clearly, products of piecewise polynomials and/or rationals are piecewise polynomials and/or rationals as well.

In summary, the ability to form a closure and compute a differential property in the piecewise polynomial and/or rational domains, makes it far simpler and robust to analyze that property. While κ is not rational, its numerator is and so inflection points could be detected as a zero set of $x'(t)y''(t) - y'(t)x''(t)$. For similar reasons, the unit normal $N(t)$ of $C(t)$ is not rational but both $\kappa(t)N(t)$ and $N(t)/\kappa(t)$ are rational. Hence, x-extreme points and y-extreme points on $C(t)$ can be identified as

$$\langle \kappa(t)N(t), (0,1) \rangle = 0, \quad \text{and} \quad \langle \kappa(t)N(t), (1,0) \rangle = 0,$$

and the local maximum curvature locations in $C(t)$ are detectable [45] as the zeros of

$$\frac{d \langle \kappa(t)N(t), \kappa(t)N(t) \rangle}{dt},$$

yet another rational function.

In [45], points of extreme curvature, or alternatively, inflection points are detected using these schemes. In addition, a scheme to approximate an arc-length reparametrizations for piecewise polynomial and/or rational curves is presented.

In the next section, Section 7.1, we will demonstrate the power of symbolic based interrogation in geometric design, for curvature analysis. In Section 7.2, silhouette curves, isoclines and isophotes curves, and reflection curves are all shown to be reducible to zero set finding. Then, in Section 7.3, we consider the problem of symbolic recognition of simple primitive surface shapes.

7.1 Curvature Analysis

Reexamining the second order differential analysis of parametric surfaces (recall Section 2), it turns out that given a rational surface $S(u, v)$, the Gaussian curvature K is rational whereas the mean curvature M is not (while M^2 is). In [47], a rational form of (the numerator of) K is symbolically computed and its zeros are used to robustly extract the parabolic lines of the surface. Figure 17 presents one example of computing the parabolic curves for a bicubic surface patch as the zeros of K (see CP-4 in Appendix A).

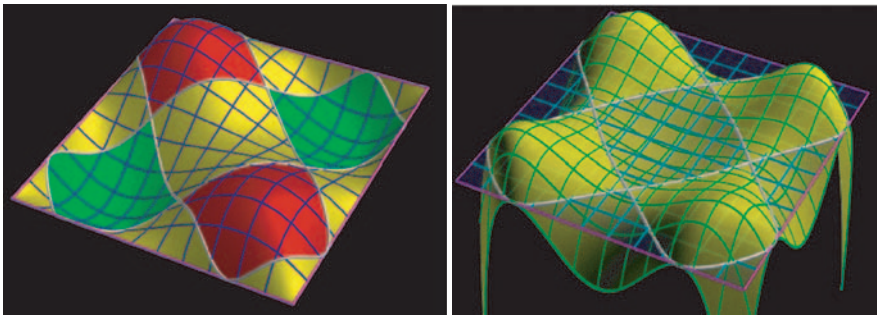


Fig. 17. Left: a free-form B-spline surface is presented, after being subdivided into convex (red), concave (green), and hyperbolic regions (yellow). The parabolic lines (white) separate the regions. Right: presents the function of $K(u, v)$ (in yellow) and its zero set (the parabolic lines).

While M is not rational, one can compute M^2 as a rational form. Similarly, the form of $\kappa_1^2 + \kappa_2^2$, where $\kappa_i,\ i = 1, 2$, are the two principle curvatures, is rational and can capture regions that are highly curved. By subdividing the original surface into regions that prescribe different values of $\kappa_1^2 + \kappa_2^2$, one can separate the surface into regions that could be NC-machined more efficiently with different sizes of ball- and flat-end cutters [44]. Let $\mathcal{K}_0 = \kappa_1^2 + \kappa_2^2$ at $S_0 = S(u_0, v_0)$. Then, the normal curvature at S_0 is bounded from above by $\sqrt{\mathcal{K}}$ or an NC ball end cutter of radius $1/\sqrt{\mathcal{K}}$ could be locally fitted to S_0 without (local) gouging. Figure 18 shows one such example where a surface is divided into regions of different values of extreme curvature, $\mathcal{K} = \kappa_1^2 + \kappa_2^2$. See also Equation (1) and CP-5 in Appendix A).

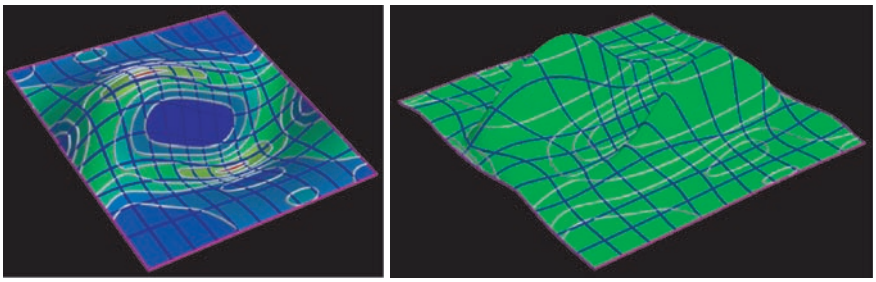


Fig. 18. Left: a free-form B-spline surface is presented, after being subdivided into regions of different levels of $\kappa_1^2 + \kappa_2^2$. Right: presents the rational surface $\kappa_1^2 + \kappa_2^2$ and its contouring (in white) at the different levels.

7.2 Silhouette, Isoclines/Isophotes and Reflection lines

The extraction of silhouettes of a free-form surface could be easily reduced to a zero set finding problem. Looking at a rational surface $S(u,v)$ from direction vector V, the silhouettes of S are characterized as the rational constraints of

$$\langle N(u,v), V\rangle = 0,$$

where $N(u,v) = \frac{\partial S}{\partial u} \times \frac{\partial S}{\partial v}$. If the view is a perspective view through point P (the eye), the silhouettes could be derived as the rational form of

$$\langle N(u,v), S(u,v) - P\rangle = 0.$$

Interestingly enough, highlight lines [7] (see Section 4.2), isoclines and isophotes (see Section 4.3) could be similarly reduced to a zero set finding, using symbolic manipulation. Let the unit view direction vector for which isoclines are sought be V. Then, positions on surface $S(u,v)$ that present a normal with a constant inclination angle of α degrees could be characterized as

$$\left\langle \frac{N(u,v)}{\|N(u,v)\|}, V \right\rangle = \cos(\alpha),$$

which is not a rational but could be made into one by squaring both sides as,

$$\langle N(u,v), V\rangle^2 - \|N(u,v)\|^2 \cos^2(\alpha) = 0, \tag{5}$$

at the cost of extraction both the $+\cos(\alpha)$ and the $-\cos(\alpha)$ isoclines, simultaneously. Figure 19 shows an example of subdividing a free form surface into regions of steep slopes (more than 45 degrees) and shallow slopes, using isoclines' analysis. Such a dichotomy might be desired, for example, in layered manufacturing processing where support is to be added to the geometry only below a certain slope.

Reflection lines (see Section 4.1) can also be reduced to rational zero set constraints as follows. An incoming ray V that hits surface $S(u,v)$ will be reflected in direction $r(u,v)$,

$$r(u,v) = 2N(u,v) - V\frac{\langle N(u,v), N(u,v)\rangle}{\langle N(u,v), V\rangle}. \tag{6}$$

In practice, Equation (6) might be difficult to work with near silhouettes (where $\langle N(u,v), V\rangle$ vanish) and so, in [46], $2N(u,v)\langle N(u,v), V\rangle - V\langle N(u,v), N(u,v)\rangle$ was proposed as a better alternative. In [46], reflection ovals, or reflections of circular curves, were similarly reduced to zero set finding problems.

7.3 Surface Recognition

A fundamental question when analyzing free-form geometry is whether the given curve or surface is of a basic primitive nature. That is, a curve is tested if it is a

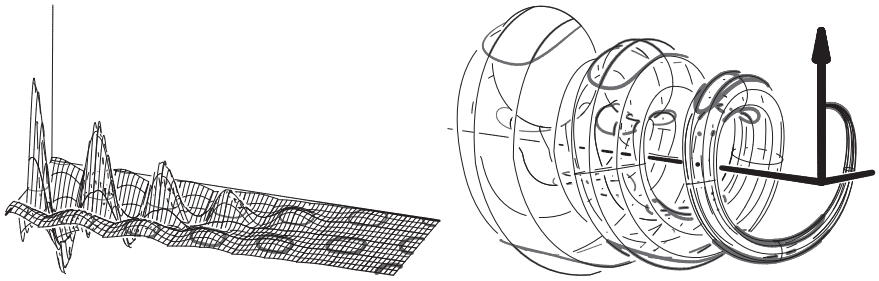


Fig. 19. Isoclines at 45 degrees from the vertical direction V. Left: the function whose zero set (see Equation (5)) prescribes the isoclines of the surface shown in the right figure is presented. Right: Isoclines also serve to split the surface into regions of slopes (normals) with more than 45 degrees (in thin lines) and regions of less than 45 degrees (in thick lines) with respect to the vertical direction V.

circle, or a surface is tested if it is a cylinder, or alternatively, a surface of revolution. In [48], these questions are answered using symbolic differential analysis. A rational curve is a circle if its rational squared curvature field, $\kappa^2(t) = \langle \kappa(t)N(t), \kappa(t)N(t) \rangle$, is constant. In other words, given a B-spline curve $C(t)$, all its coefficient of the B-spline representation of $\kappa^2(t)$ should be the same and in fact equal to the square of the reciprocal of the radius of the circle. Alternatively, the evolute curve,

$$E(t) = C(t) + N(t)/\kappa(t),$$

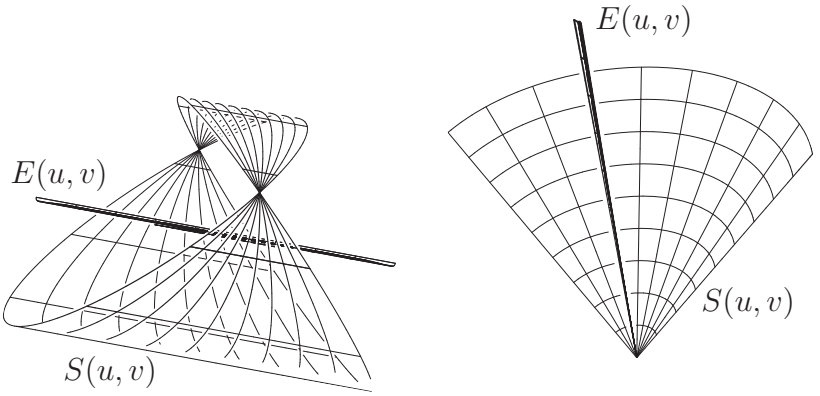
which is also rational, should vanish (along with all its control points) at the circle's center locations.

A surface called the *mean evolute* surface,

$$E(u,v) = S(u,v) + \frac{N(u,v)}{2M(u,v)},$$

where M is the mean curvature (see Section 2.2) is also defined in [48] and was shown to be rational for rational surface $S(u,v)$. If $S(u,v)$ is a circular cone, $E(u,v)$ is reduced to a line, the cone's center axis. Figure 20 presents two such examples. In [48], the connection is made between rational surfaces of revolution and rational pseudo-focal surfaces (see Section 5.4) $F_u(u,v) = S(u,v) + \frac{N(u,v)}{\kappa_u(u,v)}$, where κ_u is the normal curvature of $S(u,v)$ in the u iso-parametric direction. If the u iso-parametric directions are the latitude lines of the surface of revolution, then F_u reduces to the center axis line of the surface of revolution.

For more information, see the recent book on shape interrogation in geometric design and manufacturing [149] that discusses many of the above topics as well as intersection problems, distance queries, curvature properties, and geodesics and offsets curves and surfaces.



(a) (b)

Fig. 20. The mean evolute surface reduces to the center axis line of a circular cone or cylinder. In (a), a polynomial approximation of a cylinder surface $S(u, v)$ with unconventional parameterization is presented along with its mean evolute $E(u, v)$. (b) presents a similar view of a portion of a polynomial approximation of a region of a circular cone along with its mean evolute.

8 Interrogation of algebraic curves and surfaces

In this section we will focus on particular geometric models: the algebraic curves and surfaces. We will show how to solve in this context some important shape interrogation problems as singularity detection, intersection problems and offset computation. It turns out that all these problems require at one point to solve an algebraic system of equations, this step being the crucial one. We thus articulate this section mainly on methods that can be applied on these algebraic systems.

Most of the curves and surfaces used in CAGD are given by parametric equations, as defined in Section 2. Planar rational curves in CAGD are typically defined as

$$x(t) = \frac{a(t)}{c(t)}, \quad y(t) = \frac{b(t)}{c(t)}$$

where $a(t)$, $b(t)$ and $c(t)$ are polynomials in the Bernstein basis for rational Bézier curves or in the B-spline basis for NURBS. Note that the algebraic methods most commonly use polynomials in the power basis and polynomials can be converted from Bernstein basis to power basis. Parametric rational surfaces in CAGD are defined by

$$x(u, v) = \frac{a(u, v)}{d(u, v)}, \quad y(u, v) = \frac{b(u, v)}{d(u, v)}, \quad z(u, v) = \frac{c(u, v)}{d(u, v)}$$

where $a(u, v)$, $b(u, v)$, $c(u, v)$ and $d(u, v)$ are polynomials.

Most of the shape interrogation problems for algebraic curves and surfaces can be translated in terms of a system of polynomial equations, as this has been widely illustrated in the previous sections (see also the extensive work of Thomas Sederberg on this topic, e.g. [172]). Consequently, methods for solving polynomial systems are required. The aim of this section is to give a quick overview of such methods. In order to be as much concrete as possible we mention the following two typical problems of shape interrogation that can easily be reduced to polynomial system solving:

Singularity detection. An important problem in CAGD is the detection of singularities of a $3D$-surface. If an algebraic surface S is given implicitly by $P(x, y, z) = 0$ (that is $S = \{(x, y, z) \in \mathbb{R}^3; P(x, y, z) = 0\}$), a point (a, b, c) on S is singular if $\frac{\partial P}{\partial x}(a, b, c) = \frac{\partial P}{\partial y}(a, b, c) = \frac{\partial P}{\partial z}(a, b, c) = 0$. It is then clear that the singular points of S are the common roots of the polynomials $P, \frac{\partial P}{\partial x}, \frac{\partial P}{\partial y}, \frac{\partial P}{\partial z}$.

Computation of intersection points. Given two parameteric curves, one would like to compute their intersection points. By implicitizing one of the two curves this problem is reduced to finding the real roots of a univariate polynomial which is obtained by substituting the parameterization of a curve into the implicit equation of the second one. Similar approaches can be used to compute curve/surface intersection points and more generally to compute a parameterization of an intersection surface/surface curve. Though we are manipulating objects in dimension 3, the polynomial systems that we consider might involve more that 3 variables. For instance, the intersection of 2 parametric surfaces involve 4 variables. Therefore, we are not going to restrict the number of variables in the methods that we are going to describe. Hereafter, the variables will be denoted $x_1, \ldots, x_n$. However, since these systems come from real geometric modeling problems, we will consider only polynomials with real coefficients.

Since the problem of solving polynomial equations goes back to the ancient Greeks and Chinese, it is not surprising to see that a large number of methods exists to handle this task. Several families of solvers can however be identified:

- Analytic solvers exploit the value of a functional $f = (f_1, \ldots, f_m)$ and its derivatives in order to converge to a solution or all the solutions of $f = 0$. Typical examples are Newton like methods, minimization methods, etc.
- Subdivision methods use an exclusion criterion to remove a domain if it does not contain a root of $f = 0$ or refine the search in sub-domains otherwise. These solvers are often used to isolate the real roots in a given domain.
- Algebraic solvers exploit the known relations between the unknowns. They are based on polynomial manipulations and involve effective algebraic geometry tools.

We are going to focus essentially on the two last families, which yield information on all the roots (resp. in a fixed domain).

8.1 Subdivision methods

The methods that we describe in this section, exploit the properties of Bernstein's basis for representing univariate and multivariate polynomials. The Bernstein polynomials are ubiquitous in geometric modeling. The representation of a polynomial in the Bernstein basis is known to be numerically more stable than the monomial basis representation [57, 55]. It has a direct geometric meaning, in terms of control points and useful properties such that the convex hull and the variation diminishing property. These properties in conjunction with the subdivision nature of Bernstein's polynomials explain the large variety of algorithms proposed until today for solving univariate polynomials, starting with Lane and Riesenfeld [105], up to the *Bezier clipping* methods initiated by Nishita and al [144]. They combine a global control on the domain where the roots are searched with local and efficient refinements. The situation in the multivariate case has not been studied so extensively. Two main sub-families coexist: a first family which are based on subdivision techniques like [49, 171]; a second family of solvers which are based on reduction techniques as [173]. We briefly describe these approaches, starting with univariate polynomials. For more details, see [136].

Univariate polynomials

Any polynomial $f(x) \in \boldsymbol{R}[x]$, of degree d, can be represented as $f(x) = \sum_{i=0}^{d} b_i B_i^d(x)$ where $B_i^d(x) = \binom{d}{i}(1-x)^{d-i}x^i, i = 0, \ldots, d$ is the Bernstein basis associated to the interval $[0, 1]$. Similarly, we will say that a sequence $\boldsymbol{b}$ represents the polynomial f on the interval $[r, s]$ if:

$$f(x) = \sum_{i=0}^{d} b_i \binom{d}{i} \frac{1}{(s-r)^n} (x-r)^i (s-x)^{d-i}.$$

The polynomials $B_d^i(x; r, s) := \binom{d}{i} \frac{1}{(s-r)^n} (x-r)^i (s-x)^{n-i}$ form the Bernstein basis on $[r, s]$. Hereafter, we are going to consider the sequence of values $\boldsymbol{b}$ together with the corresponding interval $[r, s]$, as representing of our polynomial f.

A first property of this representation is that the derivative f' of f, is represented by the control coefficients: $d\boldsymbol{\Delta b} := d(b_{i+1} - b_i)_{0 \leqslant i \leqslant d-1}$. Another fundamental algorithm that we will use on such a representation is the de Casteljau algorithm [53]. It allows us to subdivide the representation of f into the two sub-representations on the intervals $[r, (1-x)r + xs]$ and $[(1-x)r + xs, s]$. It requires at most $2d(d+1)$ arithmetic operations. For a more detailed list of properties of this representation, we refer for instance to [53]. A simple but interesting property that we are going to use is the following:

Theorem 1 (Descartes rule). *The number of real roots of the polynomial* $f(x) = \sum b_i B_d^i(x; r, s)$ *in* $]r, s[$ *is bounded by the number* $V(\boldsymbol{b})$ *of sign changes of* $\boldsymbol{b} = (b_i)_{i=0..n}$*, and is equal modulo* 2*.*

As a consequence, if $V(\boldsymbol{b}) = 0$ there is no root in $]r, s[$ and if $V(\boldsymbol{b}) = 1$, there is one root in $]r, s[$. Another interesting property of this representation is the following (see e.g. [53], [164]):

Theorem 2 (Convex hull). *Let $\boldsymbol{b} = (b_i)_{i=0,\ldots,d}$ be the control coefficients of $f(x)$ on the interval $[r, s]$ and $\boldsymbol{c} = (c_i)_{i=0,\ldots,d}$ the corresponding control points. The graph $\{(x, f(x)); t \in [r, s]\}$ is in the convex hull of the control points $\boldsymbol{c}$.*

Multivariate polynomials

By a direct extension to the multivariate case, any polynomial $f(x_1, \ldots, x_n) \in \boldsymbol{R}[x_1, \ldots, x_n] = \boldsymbol{R}[\boldsymbol{x}]$ of degree d_i in the variable x_i, can be decomposed as:

$$f(x_1, \ldots, x_n) = \sum_{i_1=0}^{d_1} \cdots \sum_{i_n=0}^{d_n} b_{i_1,\ldots,i_n} B_{d_1}^{i_1}(x_1; r_1, s_1) \cdots B_{d_n}^{i_n} x(x_n; r_n, s_n)$$

where $(B_{d_1}^{i_1}(x_1; r_1, s_1) \cdots B_{d_n}^{i_n}(x_n; r_n, s_n))_{0 \le i_1 \le d_1, \ldots, 0 \le i_n \le d_n}$ is the tensor product Bernstein basis on the domain $\mathcal{D} := [a_1, b_1] \times \cdots \times [r_n, s_n] \subset \boldsymbol{R}^n$ and $\boldsymbol{b} = (b_{i_1,\ldots,i_n})_{0 \le i_1 \le d_1, \ldots, 0 \le i_n \le d_n}$ are the control coefficients of f on $\mathcal{D}$. The polynomial f is represented in this basis by the n^{th} order tensor of control coefficients $\boldsymbol{b} = (b_{i_1,\ldots,i_n})_{0 \le i \le d_1, 0 \le j \le d_2, 0 \le k \le d_3}$. The size of $\mathcal{D}$, denoted by $|\mathcal{D}|$, is $|\mathcal{D}| = \max\{|s_i - r_i|; i = 1, \ldots, n\}$.

De Casteljau's algorithm applies in each of the direction x_i, , $i = 1, \ldots, n$ so that we can split this representation in these directions. This algorithm can be used either to split the domain or to restrict the representation to a sub-domain. For a univariate polynomial of degree d, this costs $2\,(d + 1)d$ arithmetic operations. For a multivariate polynomial of degree d_i in x_i, we check that this restriction operation costs $2\sum_{i=1}^{n} d_i \prod_{i=1}^{n}(d_i + 1) = \mathcal{O}(d^{n+1})$, where $d = \max\{d_1, \ldots, d_n\}$. Thus, as the dimension and the degree increase, a good method to isolate the roots, should consider carefully when to apply this reduction operation, in order to save the computation time.

Notice that the univariate Bernstein representation also extends to the so-called triangular Bernstein basis representation. This representation can also be used in our approach, but we will concentrate on the tensor product one. For any $f \in \boldsymbol{R}[\boldsymbol{x}]$ and $j = 1, \ldots, n$, let

$$m_j(f; x_j) = \sum_{i_j=0}^{d_j} \min_{\{0 \le i_k \le d_k, k \ne j\}} b_{i_1,\ldots,i_n} B_{d_j}^{i_j}(x_j; r_j, s_j)$$

$$M_j(f; x_j) = \sum_{i_j=0}^{d_j} \max_{\{0 \le i_k \le d_k, k \ne j\}} b_{i_1,\ldots,i_n} B_{d_j}^{i_j}(x_j; r_j, s_j).$$

We have the following property: for any $\boldsymbol{u} = (u_1, \ldots, u_n) \in \mathcal{D}$, and any $j = 1, \ldots, n$, we have

$$m(f;u_j) \leq f(\boldsymbol{u}) \leq M(f;u_j).$$

As a direct consequence, for any root $\boldsymbol{u} = (u_1, \ldots, u_n)$ of the equation $f(\boldsymbol{x}) = 0$ in the domain $\mathcal{D}$, we have $\underline{\mu}_j \leq u_j \leq \overline{\mu}_j$ where

- $\underline{\mu}_j$ (resp. $\overline{\mu}_j$) is either a root of $m_j(f;x_j) = 0$ or $M_j(f;x_j) = 0$ in $[r_j, s_j]$ or r_j (resp. s_j) if $m_j(f;x_j) = 0$ (resp. $M_j(f;x_j) = 0$) has no root on $[r_j, s_j]$,
- $m_j(f;u) \leq 0 \leq M_j(f;u)$ for $u \in [\underline{\mu}_j, \overline{\mu}_j]$.

This transforms the problem of approximating the real roots of multivariate polynomials into problems on univariate polynomials.

Univariate Root Solver

Descartes rule (see theorem 1) yields a simple subdivision algorithm, which splits the domain when the number of sign variation of the control coefficients is bigger than 2. In the presence of a multiple root, the number of sign changes of a representation on any interval containing a multiple root is bigger than 2, and the algorithm splits the box until its size is smaller than a given ε. A detailled analysis of the behavior of the algorithm, has been carried out, using a partial inverse of Descartes rule given by the two circles theorem. See [140], [137], [51], [41].

This algorithm yields, in the presence of simple roots, an interval isolating the roots. But usually in practice, we are interested in approximating this root within a given precision ε. In order to approximate the isolated roots within ε, further steps of bisection may be required, either using de Casteljau's algorithm, Newton-like methods, or variants such as in [165].

Multivariate root finding

In this section, we consider a system of s equations in n variables

$$f_1(x_1, \ldots, x_n) = 0, \ldots, f_s(x_1, \ldots, x_n) = 0$$

with coefficients in $\boldsymbol{R}$, that we will also denote by $\boldsymbol{f}(\boldsymbol{x}) = 0$. We are looking for an approximation of the real roots of $\boldsymbol{f}(\boldsymbol{x}) = 0$ in the domain $\mathcal{D} = [r_1, s_1] \times \cdots \times [r_n, s_n]$, within a precision ϵ. The general framework of the families of algorithms that we will consider consists in (1) applying a preconditioning step on the equations, (2) in reducing the domain, and (3) if the reduction ratio is too small, in splitting the domain, until the size of the domain is smaller than a given epsilon. The solvers that we will consider are parameterized by the

- **Preconditioner**, that is, a transformation of the initial system $\boldsymbol{f} = 0$ into a system $M\,\boldsymbol{f} = 0$ (with an M invertible matrix), which has a better numerical behavior. We consider
  - **Global transformation** which aims at increasing the distance between the equations, considered as vectors of coefficients,

  - **Local straightening** which multiplies by the inverse of the Jacobian matrix at the center of the box, if it exists (it applies only for square systems).

- **Reduction strategy**, that is, the technique used to reduce the initial domain, for searching the roots of the system. We consider
  - **Convex hull reduction** as described in [173].
  - **Extreme root reduction**, which consists in computing the first (resp. last) root of the polynomial $m_j(f_k; u_j)$, (resp. $M_j(f_k; u_j)$), in the interval $[r_j, s_j]$. The improvement compared with the previous approach can be substantial (see Figure 8.1).

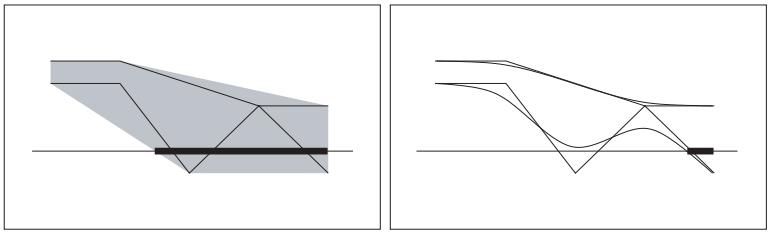


Fig. 21. Convex hull vs. extreme roots

- **Subdivision strategy**, that is, the technique used to subdivide the domain, in order to simplify the forthcoming steps, for searching of roots of the system. Here some simple rules that can be used to subdivide a domain, either in the parameter domain, or in the image.

This family of algorithms has been implemented in the C++ library SYNAPS[5] and compared on several benchmarks. It appears that the strategy of local preconditionning with an emphasis on reduction is significantly better than the other strategies. The performances of such a solver are very good.

8.2 Algebraic methods

We now turn to algebraic methods for solving polynomial systems. We will mainly discuss the ones based on resultant matrix constructions, but also mention a method based on normal form computations which generalizes the well-known concept of Gröbner basis.

Resultant-based methods

Resultant theory. The theory of resultant is devoted to the study of conditions on the coefficients of an over-determined system, to have a solution in a fixed variety. The

[5] `http://www-sop.inria.fr/galaad/software/synaps`

typical situation is the case of a system of $n+1$ equations in a space of dimension n of the form:

$$\boldsymbol{f_c} := \begin{cases} f_0(x) = \sum_{j=0}^{k_0} c_{0,j}\, \psi_{0,j}(x) \\ \quad \vdots \\ f_n(x) = \sum_{j=0}^{k_n} c_{n,j}\, \psi_{n,j}(x) \end{cases}$$

where $\boldsymbol{c} = (c_{i,j})$ are parameters, x is a point of a variety X of dimension n, and the vector $\mathcal{L}_i = (\psi_{i,j})_{j=0,\ldots,k_i}$ is a regular map [74] from X to the projective space $\mathbb{P}^{k_i}$ independent of $\boldsymbol{c}$. The elimination problem consists, in this case, in finding necessary (and sufficient) conditions on $\boldsymbol{c}$ such that the system $\boldsymbol{f_c} = 0$ has a solution in X. From a geometric point of view, we look for the values of parameters $\boldsymbol{c} = (c_{i,j})$ such that there exists $x \in X$ with $\sum_{j=0}^{k_i} c_{i,j}\psi_{i,j}(x) = 0$ for $i = 0, \ldots, n$. It turns out that these parameters are exactly the zero locus of a unique polynomial equation (defined up to the multiplication by a non-zero constant) in $\boldsymbol{c}$ which is called the resultant of $f_0, \ldots, f_n$ and is denoted by $\mathrm{Res}_X(\boldsymbol{f_c})$. It is a quite attractive object because one can compute it through some matrix constructions.

Construction of resultant matrices. In order to construct a non-trivial multiple of $\mathrm{Res}_X(\boldsymbol{f_c})$, we apply the following strategy. A vector $L(\boldsymbol{c}, \boldsymbol{x})$ of polynomials in $\boldsymbol{Z}[\boldsymbol{c}][\boldsymbol{x}]$, where $\boldsymbol{x}$ denotes a coordinate system of a projective space containing X, is constructed in such a way that

(1) the polynomial entries of $L(\boldsymbol{c}, \boldsymbol{x})$ are generically independent,
(2) the set $\boldsymbol{v}(\boldsymbol{x})$ of monomials (or polynomials) in $\boldsymbol{x}$ needed to decompose all the polynomials of $L(\boldsymbol{c}, \boldsymbol{x})$ has the same size than $L(\boldsymbol{c}, \boldsymbol{x})$, and
(3) the polynomials in $L(\boldsymbol{c}, \boldsymbol{x})$ vanish when the input system has a common root in the variety X.

The coefficient matrix of the polynomials entries of $L(\boldsymbol{c}, \boldsymbol{x})$ with respect to the set $\boldsymbol{v}(\boldsymbol{x})$ yields a matrix $\mathtt{S}(\boldsymbol{c})$ whose entries are in $\boldsymbol{Z}[\boldsymbol{c}]$. Its determinant is nonzero, according to the point (1).

Above and hereafter, we use the term *generic*, which means that the property that we are considering is true on an open subset of the coefficient space for $\boldsymbol{c}$. Our aim is to construct matrices $\mathtt{S}(\boldsymbol{c})$, which can be used for generic systems of a certain class of polynomial equations. In practice, the problem is not posed in these terms. We are given a system of equations and it may happen that the construction we are considering yields a degenerate matrix $\mathtt{S}(\boldsymbol{c})$. In this case, the system is not generic for the resultant formulation and we have to chose another class of systems for which we are in a generic position. This explains why a lot of different types of resultant formulations have been studied; we will give a list in a moment.

By construction, we have $\boldsymbol{v}(\boldsymbol{x})^t\, \mathtt{S}(\boldsymbol{c}) = L(\boldsymbol{c}, \boldsymbol{x})^t$. Thus, according to the point (2), if ζ is a common root of a specialized system $\boldsymbol{f_{c_0}} = \boldsymbol{0}$, we have $L(\boldsymbol{c}_0, \zeta) = \boldsymbol{0}$ and $\boldsymbol{v}(\zeta)^t\, \mathtt{S}(\boldsymbol{c}_0) = \boldsymbol{0}$. If (generically) $\boldsymbol{v}(\zeta)$ is not zero at a common root $\zeta \in X$ of $\boldsymbol{f_{c_0}} = \boldsymbol{0}$, we deduce that $\det\big(\mathtt{S}(\boldsymbol{c})\big)$ vanishes when the system has a common root in X. Therefore $\det\big(\mathtt{S}(\boldsymbol{c})\big)$ is a non-trivial multiple of its equation, that is of the resultant $\mathrm{Res}_X(\boldsymbol{f_c})$.

A usual way to construct these resultant matrices (which extends Sylvester's construction for the classical resultant of two univariate polynomial), consists in choosing for L, a list of monomial multiples of the polynomials f_i. In this case, the matrix $\mathtt{S}(\boldsymbol{c})$ is the matrix of a map of the form

$$\begin{aligned}\mathcal{S}: \langle \boldsymbol{x}^{E_0} \rangle \times \cdots \times \langle \boldsymbol{x}^{E_n} \rangle &\to \langle \boldsymbol{x}^{F} \rangle \\ (q_0, \ldots, q_n) &\mapsto g = \sum_{i=0}^{n} q_i f_i\end{aligned}$$

where $\langle \boldsymbol{x}^{E_i} \rangle$ is the vector subspace generated by a specific set of monomials $\boldsymbol{x}^{E_i}$. The entries of $\mathtt{S}(\boldsymbol{c})$ are filled as follows: every column of $\mathtt{S}$ is indexed by an element of some $\{E_i\}_{i=0,\ldots,n}$ and every row by an element of F; equivalently, the columns and rows are indexed by the monomials of q_i and the monomials of g, respectively. The coefficient in the row corresponding to $\beta \in F$ and in the column corresponding to $\alpha \in E_j$ is the coefficient of $\boldsymbol{x}^{\beta}$ in $\boldsymbol{x}^{\alpha} f_j$. The coefficients of monomials which do not explicitly appear in $\boldsymbol{x}^{\alpha} f_j$ have a zero entry. Thus the matrix $\mathtt{S}(\boldsymbol{c})$ is divided into blocks $(\mathtt{S}_0, \ldots, \mathtt{S}_n)$, where each block $\mathtt{S}_i$ depends linearly on the coefficients of f_i. Notice that such resultant matrices have a quasi-Toeplitz structure which can be exploited to accelerate many computations with them, by almost one order of magnitude in terms of the matrix dimension. This relies on FFT for fast multivariate polynomial and dense structured matrix arithmetic; a smaller acceleration is achieved by applying Karatsuba's divide-and-conquer arithmetic [135, 52].

Different resultant formulations. We recall briefly different resultant formulations which can be used in geometric problems. A detailed description is beyond the scope of this book. The formulation will be chosen according to the geometric properties of the problem to solve.

• **multivariate resultants:** They correspond to the classical case studied in [115, 187]. Here X is the projective space $\mathbb{P}^n$, $\mathcal{L}_i$ is the vector of all monomials of a fixed degree d_i, and the function f_i is a generic homogeneous polynomial of degree d_i. The most used resultant matrix in this context is the Macaulay's construction [115] which can be seen as an extension of the Sylvester's method to the multivariate case. However, some other multivariate resultant matrices have been developed (see e.g. [90, 34]).

• **toric (or sparse) resultants:** They have been introduced in [92], then developed in [62]. It takes into account the monomial support of the input polynomials of a system, and not only their respective degree. Thus it is possible to work with polynomials having negative exponents, that is *Laurent polynomials*. Methods for constructing toric resultant matrices can be found in [22, 50, 33].

• **Residual resultants:** In many situations coming from practical problems, the equations have common zeroes which are independent of the parameters of the problems, and which we are not interested in. The residual resultant constructions has been designed to take into account these degenerate cases. It is described, as well as matrix construction, in [20, 18, 21]. A more general construction has been developed in [19]

whose associated matrix construction is the so-called called Bezoutian matrix.

Solving polynomial systems via eigenvalues computations. Let $f_0(x)$, $f_1(x)$, ..., $f_n(x)$ be polynomials in n variables $x = (x_1, \ldots, x_n)$. By choosing an adapted resultant formulation one can construct a resultant matrix $\mathcal{S}$ associated to this system. It turns out that this matrix can be divided into four blocs $\mathcal{S} = \left(\begin{array}{c|c} \mathcal{S}_{00} & \mathcal{S}_{01} \\ \hline \mathcal{S}_{10} & \mathcal{S}_{11} \end{array}\right)$ and that the Schur complement $\mathcal{S}_{00} - \mathcal{S}_{01}{\mathcal{S}_{11}}^{-1}\mathcal{S}_{10}$ is nothing but the matrix of the multiplication map by $f_0(x)$ in a canonical basis of the quotient ring $\mathbb{R}[x]/(f_1, \ldots, f_n)$. The point is that the eigenvalues of such a multiplication matrix are particularly interesting, they are the evaluation of f_0 at the common root of f_1 and f_2. If f_0 is a linear form one can thus easily solve the polynomial system $f_1(x) = f_2(x) = 0$.

Solving polynomial systems by hiding a variable. Another approach to solve a system of polynomial equations consists in *hiding* a variable (that is, in considering one of variables as a *parameter*), and in searching the value of this hidden variable for which the system has a solution. Typically, if we have n equations $f_1 = 0, \ldots, f_n = 0$ in n variables, we "hide" a variable, say x_n, and apply one of resultant constructions described before to the over-determined system $f_1 = 0, \ldots, f_n = 0$ in the $n-1$ variables $x_1, \ldots, x_{n-1}$ and a parameter x_n. This leads to a resultant matrix $\mathtt{S}(x_n)$ with polynomial entries in x_n. It can be decomposed as

$$\mathtt{S}(x_n) = \mathtt{S}_d\, x_n^d + \mathtt{S}_{d-1} x_n^{d-1} + \cdots + \mathtt{S}_0,$$

where $\mathtt{S}_i$ has coefficients in $\mathbb{R}$ and the same size as $\mathtt{S}(x_n)$. We look for the values ζ_n of x_n for which the system has a *solution* $\zeta' = (\zeta_1, \ldots, \zeta_{n-1})$ in the corresponding variety X' (of dimension $n-1$) associated with the resultant formulation. This implies that

$$\boldsymbol{v}(\zeta')^{\mathtt{t}}\, \mathtt{S}(\zeta_n) = \mathbf{0}, \tag{7}$$

where $\boldsymbol{v}(\zeta')$ is the vector of monomials indexing the rows of $\mathtt{S}$ evaluated at ζ'. Conversely, for generic systems of the corresponding resultant formulation there is only one point ζ' above the value ζ_n. Thus the vectors $\boldsymbol{v}$ satisfying $\mathtt{S}(\zeta_n)^{\mathtt{t}}\, \boldsymbol{v} = 0$ are scalar multiples of $\boldsymbol{v}(\zeta')$. From the entries of these vectors, we can deduce the other coordinates of the point ζ'. This will be assumed hereafter[6].

The relation (7) implies that $\boldsymbol{v}(\zeta')$ is a genearlized eigenvector of $\mathtt{S}^t(x_n)$. Computing such vectors can be transformed into the following linear generalized eigenproblem

$$\left(\begin{bmatrix} 0 & \mathbb{I} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mathbb{I} \\ \mathtt{S}_0^t & \mathtt{S}_1^t & \cdots & \mathtt{S}_{d-1}^t \end{bmatrix} - \zeta_n \begin{bmatrix} \mathbb{I} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \mathbb{I} & 0 \\ 0 & \cdots & 0 & -\mathtt{S}_d^t \end{bmatrix} \right) \boldsymbol{w} = 0. \tag{8}$$

[6] Notice however that this genericity condition can be relaxed by using duality, in order to compute the points ζ' above ζ_n (when they form a zero-dimensional fiber) from the eigenspace of $\mathtt{S}(\zeta_n)$.

The set of eigenvalues of (8) contains the values of ζ_n for which (7) has a solution. The corresponding eigenvectors $\boldsymbol{w}$ are decomposed as $\boldsymbol{w} = (\boldsymbol{w}_0, \ldots, \boldsymbol{w}_{d-1})$ so that the solution vector $\boldsymbol{v}(\zeta')$ of (7) is

$$\boldsymbol{v}(\zeta') = \boldsymbol{w}_0 + \zeta_n \boldsymbol{w}_1 + \cdots + \zeta_n^{d-1} \boldsymbol{w}_{d-1}.$$

Normal forms

Gröbner basis is a powerful tool to handle a lot of computations on polynomial systems. However their construction is not numerically stable, they may introduce artificial discontinuities due to the choice of a monomial order. A recent generalization of this notion has been proposed in [134, 138, 139]. It is based on a new criterion which gives a necessary and sufficient condition for a projection onto a vector subspace of R to be a normal form modulo the ideal I. More precisely we have:

Theorem 3. *Let B be a vector space in $R = \mathbb{R}[x_1, \ldots, x_n]$ connected to the constant polynomial 1[7]. If B^+ is the vector subspace generated by $B \cup x_1 B \cup \ldots \cup x_n B$, $N : B^+ \to B$ is a linear map such that N is the identity on B, we define for $i = 1, \ldots, n$, the maps*

$$\begin{aligned} M_i : B &\to B \\ b &\mapsto M_i(b) := N(x_i b). \end{aligned}$$

The two following properties are equivalent:

1. *For all $1 \leq i, j \leq n$, $M_i \circ M_j = M_j \circ M_i$.*
2. *$R = B \oplus I$, where I is the ideal generated by the kernel of N*

If this holds, the B-reduction along $\ker(N)$ is canonical.

This leads to a completion-like algorithm which starts with the linear subspace K_0 generated by the polynomials $f_1, \ldots, f_m$, that we want to solve and iterates the construction $K_{i+1} = K_i^+ \cap L$, where L is a fixed vector space. We stop when $K_{i+1} = K_i$. See [134, 138, 186, 139] for more details. This approach allows us to fix first the set of monomials on which we want to do linear operations and thus to treat more safely polynomials with approximate coefficients. It can be adapted very naturally to Laurent polynomials, which is not the case for Gröbner basis computation. Moreover it can be specialized very efficiently to systems of equations for which the basis of $\mathcal{A}$ is known a priori, such as in the case of a complete projective intersection [138]. Let us see how we can deduce the roots from this normal form computation. For this purpose, we will use the properties of the operators of multiplication by elements of $\mathcal{A} = R(f_1, .., f_m)$. For any $a \in \mathcal{A}$, we define

$$\begin{aligned} M_a : \mathcal{A} &\to \mathcal{A} \\ b &\mapsto M_a(b) := a\, b. \end{aligned}$$

[7] Any monomial $\boldsymbol{x}^\alpha \neq 1 \in B$ is of the form $x_i \boldsymbol{x}^\beta$ with $\boldsymbol{x}^\beta \in B$ and some i in $\{1, \ldots, n\}$.

We also consider its transpose operator

$$\begin{aligned} M_a^{\mathtt{t}} : \widehat{\mathcal{A}} &\to \widehat{\mathcal{A}} \\ \Lambda &\mapsto M_a^{\mathtt{t}}(\Lambda) = \Lambda \circ M_a, \end{aligned}$$

where the dual space $\widehat{\mathcal{A}}$ is the set of $\mathbb{R}$-linear forms from $\mathcal{A}$ to $\mathbb{R}$. The matrix of $M_a^{\mathtt{t}}$ in the dual basis of a basis B of $\mathcal{A}$ is the transpose of the matrix of M_a in B. The multiplication operators can be computed using a normal form algorithm, as described above.

Hereafter, $\boldsymbol{x}^E = (\boldsymbol{x}^\alpha)_{\alpha \in E}$ denotes a monomial basis of $\mathcal{A}$ (for instance obtained by a Gröbner basis). Then any polynomial can be reduced modulo $(f_1, \ldots, f_m)$ to a linear combination of monomials of $\boldsymbol{x}^E$.

The matrix approach to solve polynomial systems is based on the following fundamental theorem [3], [133]:

Theorem 4. *Assume that $\mathcal{Z}(I) = \{\zeta_1, \ldots, \zeta_d\}$. We have*

1. *Let $a \in \mathcal{A}$. The eigenvalues of the operator M_a (and its transpose $M_a^{\mathtt{t}}$) are $a(\zeta_1), \ldots, a(\zeta_d)$.*
2. *The common eigenvectors of $(M_a^{\mathtt{t}})_{a \in \mathcal{A}}$ are (up to a scalar) the evaluations $\mathbf{1}_{\zeta_1}, \ldots, \mathbf{1}_{\zeta_d}$.*

Since $\boldsymbol{x}^E = (\boldsymbol{x}^\alpha)_{\alpha \in E}$ is a basis of $\mathcal{A}$, the coordinates of $\mathbf{1}_{\zeta_i}$ in the dual basis of $\boldsymbol{x}^E$ are $(\zeta_i^\alpha)_{\alpha \in E}$. Thus if $\boldsymbol{x}^E$ contains $1, x_1, \ldots, x_n$ (which is often the case), we can deduce directly all the coordinates of the roots. We have the following algorithm:

Algorithm 1 *Solving in the case of simple roots. Let $a \in \mathcal{A}$ such that $a(\zeta_i) \neq a(\zeta_j)$ for $i \neq j$ (which is generically the case) and $\mathtt{M}_a$ be the matrix of multiplication by a in the basis $\boldsymbol{x}^E = (1, x_1, \ldots, x_n, \ldots)$ of $\mathcal{A}$.*

1. *Compute the eigenvectors $\Lambda = (\Lambda_1, \Lambda_{x_1}, \ldots, \Lambda_{x_n}, \ldots)$ of $\mathtt{M}_a^{\mathtt{t}}$.*
2. *For each eigenvector Λ with $\Lambda_1 \neq 0$, compute and output the point $\zeta = \left(\frac{\Lambda_{x_1}}{\Lambda_1}, \ldots, \frac{\Lambda_{x_n}}{\Lambda_1}\right)$.*

The set of output points ζ contains the simple roots (i.e. roots with multiplicity 1) of $f = 0$, since for such a root the eigenspace associated to the eigenvalue $a(\zeta)$ is one-dimensional and contains $\mathbf{1}_\zeta$. But as we will see in the next example, it can also yield in some cases the multiple roots.

In order to compute exactly the set of roots counted with their multiplicity, we use the following result. It is based on the fact that commuting matrices share common eigenspaces. [133, 135, 31].

Theorem 5. *There exists a basis of $\mathcal{A}$ such that for all $a \in \mathcal{A}$, the matrix of M_a in this basis is of the form*

$$\mathtt{M}_a = \begin{pmatrix} \mathtt{N}_a^1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathtt{N}_a^d \end{pmatrix} \quad \text{with} \quad \mathtt{N}_a^i = \begin{pmatrix} a(\zeta_i) & & \star \\ & \ddots & \\ \mathbf{0} & & a(\zeta_i) \end{pmatrix}.$$

We deduce the algorithm:

Algorithm 2 *Solving by simultaneous triangulation.*
INPUT: *Matrices of multiplication* $\mathtt{M}_{x_i}, i = 1, \ldots, n$, *in a basis of* $\mathcal{A}$.

1. *Compute a (Schur) decomposition* $\mathtt{P}$ *such that the matrices* $\mathtt{T}_i = \mathtt{P}\mathtt{M}_{x_i}\mathtt{P}^{-1}$, $i = 1, \ldots, n$, *are upper-triangular.*
2. *Compute and output the diagonal vectors* $\boldsymbol{t}_i = (t^1_{i,i}, \ldots, t^n_{i,i})$ *of triangular matrices* $\mathtt{T}_k = (t^k_{i,j})_{i,j}$.

OUTPUT: $\mathcal{Z}(I) = \{\boldsymbol{t}_i : i = 1, \ldots, \dim_{\mathbb{R}}(\mathcal{A})\}$.

The first step in this algorithm is performed by computing a Schur decomposition of M_l (where l is a generic linear form) which yields a matrix $\mathtt{P}$ of bases change. Then we compute the triangular matrices $\mathtt{T}_i = \mathtt{P}\mathtt{M}_{x_i}\mathtt{P}^{-1}$, $i = 1, \ldots, n$, since they commute with M_l. An implementation by Ph. Trébuchet of this algorithm is available in the SYNAPS library (see `solve(L,newmac<C>())`).

9 Conclusion

Shape interrogation methods are still of increasing interest in geometric modeling as well as in computer graphics. Originating 20 years ago from CAD/CAM applications where "class A" surfaces are required and no surface imperfections are allowed, shape interrogation has become recently an important tool for various other types of surface representations such as triangulated or polygonal surfaces, subdivision surfaces, and algebraic surfaces. In this chapter, we presented the state-of-the-art of shape interrogation methods including methods for detecting surface imperfections, surface analysis tools and methods for visualizing intrinsic surface properties. Furthermore we focused on stable numerical and symbolic solving of algebraic systems of equations, a problem that arises in most shape interrogation methods. Nevertheless, many issues are still open promising intensive research in various areas of shape interrogation. Let us focus on some of them now.

Discrete geometry representations are frequently used in many applications, especially for shapes acquired from real-world objects. Typically, surfaces are approximated by polygonal meshes, and we showed how to estimate differential properties for piecewise linear surfaces. Various methods exist so far, and recent approaches prove approximation and convergence properties. The design of robust methods coming with certain guarantees is still an area of active research.

In the area of algebraic and numerical polynomial system solvers, that provide one of the basic tools for shape interrogation methods, improvements are indispensable. Many critical problems in Computer Aided Geometric Deisgn, such as shape interrogation, are reduced to finding the zero set of a system of polynomial equations. Several root-finding methods for polynomial systems exist, even if we mainly presented resultant-based methods and subdivision methods. A wide choice of techniques and algorithms to solve polynomial systems are thus now available, but as a main drawback, all of these methods have difficulties in handling roots with high

multiplicities (or clusters of roots). They all have performance deterioration, lack of robustness in numerical computation and round-off errors during floating point arithmetic in such a situation. It is hence a crucial objective and an active research area to improve root-finding methods in this case which often occurs in practice.

In the area of symbolic curve and surface interrogation future/open problems can be addressed. Volumetric data sets are now used in many applications and serves as a prime candidate representation in medical applications. The extension of curve and surface interrogation methods to support volumetric representation, either as iso-surfaces in the volumes or direction analyze differential properties are highly desired. The degrees of many of these rational fields such as the Gaussian curvature, K, or the mean curvature square, M^2, are high. Methods to robustly handle these fields, in a more stable way, could further improve the quality of the result.

References

1. P. Alliez, M. Attene, C. Gotsman, and G. Ucelli. Recent advances in remeshing of surfaces. In L. De Floriani and M. Spagnuolo, editors, *Shape Analysis and Structuring*. Springer, 2007.
2. P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics*, 22(3):485–493, July 2003.
3. W. Auzinger and H. J. Stetter. An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations. In *Proc. Intern. Conf. on Numerical Math.*, volume 86 of *Int. Series of Numerical Math*, pages 12–30. Birkhäuser Verlag, 1988.
4. R Barnhill, G Farin, L. Fayard, and H Hagen. Twists, curvature and surface interrogation. *CAD*, 20:314–346, 1988.
5. J. Beck, R. Farouki, and J. Hinds. Surface analysis methods. *IEEE CG & Appl.*, 6:19–35, 1986.
6. K.-P. Beier. The porcupine technique: principles, applications, and algorithms. Technical report, University of Michigan, 1987.
7. K.-P. Beier and Y. Chen. Highlight-line algorithm for realtime surface quality assessment. *CAD*, 26(4):268–277, 1994.
8. A. G. Belyaev, E. V. Anoshkina, and T. L. Kunii. Ridges, ravines, and singularities. In A. T. Fomenko, and T. L. Kunii, *Topological Modeling for Visualization*, pages 375–383. Springer, 1997. Chapter 18.
9. A. G. Belyaev and Y. Ohtake. An image processing approach to detection of ridges and ravines on polygonal surfaces. In *Eurographics 2000, Short Presentations*, pages 19–28, August 2000.
10. A. G. Belyaev, A. A. Pasko, and T. L. Kunii. Ridges and ravines on implicit surfaces. In *Proc. Computer Graphics International 1998*, pages 530–535, 1998.
11. M. V. Berry and J. H. Hannay. Umbilic points on gaussian random surfaces. *J. Phys. A*, 10:1809–21, 1977.
12. P. J. Besl and R. C. Jain. Invariant surface characteristics for 3D object recognition in range images. *Comput. Vision Graph. Image Process*, 33(1):33–80, 1986.
13. I. A. Bogaevski, V. Lang, A. G. Belyaev, and T. L. Kunii. Color ridges on implicit polynomial surfaces. In *GraphiCon 2003 Proceedings*, pages 161–164, September 2003.

14. V. Borrelli, F. Cazals, and J. M. Morvan. On the angular defect of triangulations and the pointwise approximation of curvatures. *Computer Aided Geometric Design*, 20(6):319–341, 2003.
15. J. W. Bruce, P. J. Giblin, and F. Tari. Ridges, crests and sub-parabolic lines of evolving surfaces. *International Journal of Computer Vision*, 18(3):195–210, 1996.
16. J. W. Bruce, P. J. Giblin, and F. Tari. Families of surfaces: focal sets, ridges and umbilics. *Math. Proc. Camb. Phil. Soc.*, 125:243–268, 1999.
17. J. W. Bruce and T. C. Wilkinson. Folding maps and focal sets. In M. Dæhlen, T. Lyche, and L. L. Schumaker, editors, *Proceedings of Warwick Symposium on Singularities, Springer Lecture Notes in Math., vol 1462*, pages 63–72, Berlin and New York,, 1991. Springer-Verlag.
18. L. Busé. Residual resultant over the projective plane and the implicitization problem. In *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*, pages 48–55 (electronic), New York, 2001. ACM.
19. L. Busé, M. Elkadi, and B. Mourrain. Generalized resultants over unirational algebraic varieties. *J. Symbolic Comput.*, 29(4-5):515–526, 2000. Symbolic computation in algebra, analysis, and geometry (Berkeley, CA, 1998).
20. L. Busé, M. Elkadi, and B. Mourrain. Resultant over the residual of a complete intersection. *J. Pure Appl. Algebra*, 164(1-2):35–57, 2001. Effective methods in algebraic geometry (Bath, 2000).
21. L. Busé, M. Elkadi, and B. Mourrain. Using projection operators in computer aided geometric design. In *Topics in algebraic geometry and geometric modeling*, volume 334 of *Contemp. Math.*, pages 321–342. Amer. Math. Soc., Providence, RI, 2003.
22. J. Canny and P. Pedersen. An algorithm for the Newton resultant. Technical Report 1394, Comp. Science Dept., Cornell University, 1993.
23. C. Catalano and I. Ivrissimtzis. Subdivision surfaces and applications. In L. L. De Floriani and M. Spagnuolo, editors, *Shape Analysis and Structuring*. Springer, 2007.
24. F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. In *Symposium on Geometry Processing*, pages 177–187, 2003.
25. F. Cazals and M. Pouget. Ridges and umbilics of a sampled smooth surface: a complete picture gearing toward topological coherence. Rapport de Recherche RR-5294, INRIA, September 2004.
26. F. Cazals and M. Pouget. Smooth surfaces, umbilics, lines of curvatures, foliations, ridges and the medial axis: a concise overview. Rapport de Recherche RR-5138, INRIA, March 2004.
27. S. Chan and E. Purisima. A new tetrahedral tesselation scheme for isosurface generation. *Computers & Graphics*, 22(1):83–90, 1998.
28. E. Cohen, R. Riesenfeld, and G. Elber. *Geometric Modeling with Splines: An Introduction*. AK Peters, 2001.
29. D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905–914, August 2004. Proceedings of SIGGRAPH 2004.
30. D. Cohen-Steiner and J.-M. Morvan. Restricted delaunay triangulations and normal cycle. In *Proceedings of the nineteenth Conference on Computational Geometry (SCG-03)*, pages 312–321, June 8–10 2003.
31. R. M. Corless, P. M. Gianni, and B. M. Trager. A reordered Schur factorization method for zero-dimensional polynomial systems with multiple roots. In W.W. Küchlin, editor, *Porc. ISSAC*, pages 133–140, 1997.

32. P. Csákány and A. M. Wallace. Computation of local differential parameters on irregular meshes. In R. Cipola and R. Martin, editors, *The Mathematics of Surfaces IX*, pages 19–33. Springer, 2000.
33. C. D'Andrea. Macaulay style formulas for sparse resultants. *Trans. Amer. Math. Soc.*, 354(7):2595–2629 (electronic), 2002.
34. C. D'Andrea and A. Dickenstein. Explicit formulas for the multivariate resultant. *J. Pure Appl. Algebra*, 164(1-2):59–86, 2001. Effective methods in algebraic geometry (Bath, 2000).
35. G. Darboux. *Leçons sur la théorie générale des surfaces, Tome 4*. Gauthier-Villars, Paris, 1896.
36. D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella. Suggestive contours for conveying shape. *ACM Trans. on Graphics*, 22(3):848–855, 2003. Proc. ACM SIGGRAPH 2003.
37. M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Computer Graphics (Proceedings of SIGGRAPH 99)*, pages 317–324, 1999.
38. J. Dill. An application of color graphics to the display of surface curvature. *Computer Graphics*, 15(3):153–161, 1981.
39. P. M. Do Carmo. *Differential Geometry of curves and surfaces*. Prentice-Hall, Englewood Cliffs, 1976.
40. D. Eberly. *Ridges in Image and Data Analysis*. Kluwer, 1996.
41. A. Eigenwillig, V. Sharma, and C. K. Yap. Almost tight recursion tree bounds for the descartes method. In *ISSAC '06: Proceedings of the 2006 international symposium on Symbolic and algebraic computation*, pages 71–78, New York, NY, USA, 2006. ACM Press.
42. L. P. Eisenhart. *An introduction to differential geometry*. Princeton University Press, Princeton, N.J., 1976.
43. G. Elber. Free form surface analysis using a hybrid of symbolic and numerical computation. *PhD thesis, Department of Computer Science, The University of Utah*, 1992.
44. G. Elber. Freeform surface region optimization for three- and five-axis milling. *Computer Aided Design*, 27(6):465–470, June 1995.
45. G. Elber. Symbolic and numeric computation in curve interrogation. *Computer Graphics forum*, 14(1):25–34, March 1995.
46. G. Elber. Curve evaluation and interrogation on surfaces. *The Journal of Graphical Models*, 63(3):197–210, May 2001.
47. G. Elber and E. Cohen. Second-order surface analysis using hybrid symbolic and numeric operators. *ACM Trans. on Graphics*, 12(2):160–178, 1993.
48. G. Elber and M.-Soo Kim. Geometric shape recognition of freeform curves and surfaces. *Graphics Models and Image Processing*, 59(6):417–433, November 1997.
49. G. Elber and M.-Soo Kim. Geometric constraint solver using multivariate rational spline functions. In *Proceedings of the sixth ACM Symposium on Solid Modelling and Applicat ions*, pages 1–10. ACM Press, 2001.
50. I. Emiris and J. Canny. A practical method for the sparse resultant. In M. Bronstein, editor, *Proc. Intern. Symp. on Symbolic and Algebraic Computation*, pages 183–192, Kiev, July 1993.
51. I. Z. Emiris, B. Mourrain, and E. P. Tsigaridas. Real algebraic numbers: Complexity analysis and experimentations. Research Report 5897, INRIA, Avril 2006.
52. I. Z. Emiris and V. Y. Pan. Symbolic and numeric methods for exploiting structure in constructing resultant matrices. *J. Symbolic Comput.*, 33(4):393–413, 2002.

53. G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, New York, 4th edition, 1996.
54. G. Farin and N. Sapidis. Curvature and the fairness of curves and surfaces. *IEEE CG & Appl.*, 9:52–57, 1989.
55. R. Farouki and V. Rajan. On the numerical condition of polynomials in bernstein form. *Computer Aided Geometric Design*, 4(3):191–216, 1987.
56. R. T. Farouki. Graphical methods for surface differential geometry. In R. Martin, editor, *in Mathematics of surfaces*, pages 363–385. IMA Series, 1987.
57. R. T. Farouki and T. N. T. Goodman. On the optimal stability of the bernstein basis. *Mathematics of computation*, 65(216):1553–1566, October 1996.
58. J. D. Foley, A. van Damm, S. K. Feiner, and J. F. Hughes. *Computer Graphics. Principles and Practice*. Adison-Wesley, 1990. 2nd edition.
59. A. Forrest. On the rendering of surfaces. *Computer Graphics*, pages 253–259, 1979.
60. P. J. Frey and H. Boroucraki. Surface mesh quality evaluation. *International Journal for Numerical Methods in Engineering*, 45:101–118, 1999.
61. I. Friedel, P. Schröder, and A. Khodakovsky. Variational normal meshes. *ACM Transactions on Graphics*, 23(4):1061–1073, 2004.
62. I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky. *Discriminants, resultants, and multidimensional determinants*. Mathematics: Theory & Applications. Birkhäuser Boston Inc., Boston, MA, 1994.
63. A. S. Glassner. Computing surface normals for 3D models. In A. S. Glassner, editor, *Graphics Gems*, pages 562–566. Academic Press, 1990.
64. J. Goldfeather and V. Interrante. A novel cubic-order algorithm for approximating principal directions vectors. *ACM Transactions on Graphics*, 23(1):45–63, 2004.
65. G. G. Gordon. Face recognition from depth maps and surface curvature. In *Geometric Methods in Computer Vision, Proc. SPIE 1570*, pages 234–247, 1991.
66. H. Gouraud. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, C-20(6):623–629, 1971.
67. U. Grenader and M. I. Miller. Computational anatomy: An emerging discipline. *Quarterly of Applied Mathematics*, 56(4):617–694, 1998.
68. A. Guéziec, X. Pennec, and N. Ayache. Medical image registration using geometric hashing. *IEEE Comput. Sci. Eng.*, 4(4):29–41, 1997.
69. A. Gullstrand. Zur Kenntnis der Kreispunkte. *Acta Mathematica*, 29:59–100, 1904.
70. S. Gumhold, X. Wang, and R. McLeod. Feature extraction from point clouds. In *Proc. 10th International Meshing Roundtable*, pages 293–305, Sandia National Laboratories, Newport Beach, CA, 2001.
71. H. Hagen and S. Hahmann. Generalized focal surfaces : A new method for surface interrogation. In *Proceedings Visualization'92*, pages 70–76. IEEE, 1992.
72. P. L. Hallinan, G. G. Gordon, A. L. Yuille, P. Giblin, and D. Mumford. *Two- and Tree-Dimensional Patterns of the Face*. A K Peters, 1999.
73. B. Hamann. Curvature approximation for triangulated surfaces. *Computing Suppl.*, 8:139–153, 1993.
74. J. Harris. *Algebraic geometry*, volume 133 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1992. A first course.
75. E. Hartmann. On the curvature of curves and surfaces defined by normalforms. *Computer Aided Geometric Design*, 16(5):355–376, 1999.
76. R. Hartwig and H. Nowacki. Isolinien und schnitte in coonschen flächen. Geometrisches Modellieren 65, Informatik Fachberichte der GI, 1982.
77. R. A. Herman. *A Treatise on Geometrical Optics*. Cambridge University Press, 1900.

78. M. Higashi, T. Saitoh, and Y.. Watanabe. Analysis of aesthetic free-form surfaces by surface edges. In *Pacific Graphics '95*, pages 294–305, 1995.
79. D. Hilbert and S. Cohn-Vossen. *Geometry and the imagination*. Chelsea Publishing Company, New York, 1952.
80. K. Hildebrandt and K. Polthier. Anisotropic filtering of non-linear surface features. In *Proc. Eurographics*, pages 391–400, 2004.
81. K. Hildebrandt, K. Polthier, and M. Wardetzky. Smooth feature lines on surface meshes. In *Third Eurographics Symposium on Geometry Processing*, pages 85–90, July 2005.
82. M. Hisada, A. G. Belyaev, and T. L. Kunii. A skeleton-based approach for detection of perceptually salient features on polygonal surfaces. *Computer Graphics Forum*, 21(4):689–700, 2002.
83. D. D. Hoffman and W. A. Richards. Parts of recognition. *Cognition*, 18:65–96, 1985.
84. M. Hosaka. *Modeling of Curves and Surfaces in CAD/CAM*. Springer, Berlin, 1992.
85. J. Hoschek. Detecting regions with undesirable curvature. *CAGD*, 1:183–192, 1984.
86. J. Hoschek. Smoothing of curves and surfaces. *CAGD*, 2:97–105, 1985.
87. J. Hoschek, U. Dietz, and W. Wilke. A geometric concept of reverse engineering of shape: Approximation and feature lines. In M. Dæhlen, T. Lyche, and L. L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces II*, pages 253–262. Vanderbilt Univ. Press, 1998.
88. A. Hubeli and M. Gross. Multiresolution feature extraction from unstructured meshes. In *Proc. IEEE Visualization 2001*, pages 287–294, 2001.
89. V. Interrante, H. Fuchs, and S. Pizer. Enhancing transparent skin surfaces with ridge and valley lines. In *Proc. IEEE Visualization 1995*, pages 52–59, 1995.
90. J. P. Jouanolou. Formes d'inertie et résultant: un formulaire. *Adv. Math.*, 126(2):119–250, 1997.
91. S. Kapoor. Efficient computation of geodesic shortest paths. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 770–779, New York, NY, USA, 1999. ACM Press.
92. M. M. Kapranov, B. Sturmfels, and A. V. Zelevinsky. Chow polytopes and general resultants. *Duke Math. J.*, 67(1):189–218, 1992.
93. S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph.*, 22(3):954–961, 2003.
94. E. Kaufmann and R. Klass. Smoothing surfaces using reflection lines for families of splines. *CAD*, 20:312–316, 1988.
95. J. T. Kent, D. Lee, Mardia K. V., and A. D. Linney. Using curvature information in shape analysis. In K. V. Mardia, G. A. Gill, and I. L. Dryden, editors, *Proc. Image Fusion and Shape Variability Techniques*, pages 88–99. Leeds University Press, 1996.
96. J. T. Kent, K. V. Mardia, and J. West. Ridge curves and shape analysis. In *The British Machine Vision Conference 1996*, pages 43–52, 1996.
97. J. Kjellander. Smoothing of bicubic parametric surfaces. *CAD*, 15:288–293, 1983.
98. R. Klass. Correction of local irregularities using reflection lines. *CAD*, 12:73–77, 1980.
99. K. H. Ko, T. Maekawa, N. M. Patrikalakis, H. Masuda, and F.-E. Wolter. Shape intrinsic fingerprints for free-form object matching. In *Proc. of 8th ACM Symposium on Solid Modeling and Applications*, pages 196 – 207, 2003.
100. L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of SIGGRAPH 98*, pages 105–114, 1998.
101. J. J. Koenderink. *Solid Shape*. MIT Press, 1990.
102. I. Kreyszig. *Differential Geometry*. Univ. of Toronto Press, Toronto, 1959.

103. V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *SIGGRAPH 96 Conference Proceedings*, pages 313–324, New York, NY, USA, 1996. ACM Press.
104. P. Krsek, G. Lukacs, and R. R. Martin. Algorithms for computing curvatures from range data. In R. Cripps, editor, *The Mathematics of Surfaces VIII*, pages 1–16. IMA, 1998.
105. J. Lane and R. Riesenfeld. Bounds on a polynomial. *BIT*, 21:112–117, 1981.
106. T. Langer, A. Belyaev, and H.-P. Seidel. Asymptotic analysis of discrete normals and curvatures of polylines. In *SCCG '05: Proceedings of the 21st spring conference on Computer graphics*, pages 229–232, 2005.
107. T. Langer, A. Belyaev, and H.-P. Seidel. Exact and approximate quadratures for curvature tensor estimation. In *Vision, Modeling, and Visualization 2005 (VMV'05)*, pages 421–428, 2005.
108. R. B. Lee and D. A. Fredericks. Intersection of parametric surfaces and a plane. *IEEE CG & Appl.*, 4(8):48–51, 1984.
109. J. J. Little and P. Shi. Structural lines, TINs and DEMs. *Algorithmica*, 30(2):243–263, 2001.
110. A. M. López, F. F. Lumbreras, and J. Serrat. Creaseness from level set extrinsic curvature. In *Proc. ECCV'98*, pages 156–169. Springer, 1998.
111. W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH 87 Conference Proceedings*, pages 163–169, New York, NY, USA, 1987. ACM Press.
112. C. Lu, Y. Cao, , and D. Mumford. Surface evolution under curvature flows. *Journal of Visual Communication and Image Representation*, 13(1/2):65–81, March/June 2002.
113. G. Lukács and L. Andor. Computing natural division lines on free-form surfaces based on measured data. In M. Dæhlen, T. Lyche, and L. L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces II*, pages 319–326. Vanderbilt Univ. Press, 1998.
114. K.-L. Ma and V. Interrante. Extracting feature lines from 3D unstructured grids. In *Proc. IEEE Visualization 1997*, pages 285–292, 1997.
115. F. S. Macaulay. Some formulae in elimination. *Proc. London Math. Soc.*, 1(33):3–27, 1902.
116. T. Maekawa and Patrikalakis M. Interrogation of differential geometry properties for design and manufacture. *Visual Computer*, 10:216–237, 1994.
117. T. Maekawa, F.-E. Wolter, and N. Patrikalakis. Umbilics and lines of curvature for shape interrogation. *CAGD*, 13:133–161, 1996.
118. T. Maekawa, F.-E. Wolter, and N. M. Patrikalakis. Umbilics and lines of curvature for shape interrogation. *Computer Aided Geometric Design*, 13(2):133–161, 1996.
119. J.-L. Maltret and M. Daniel. Discrete curvatures and applications: a survey. Rapport de recherche 004.2002, Laboratoire des Sciences de l'Information et des Systèmes, 2002.
120. N. Max. Weights for computing vertex normals from facet normals. *Journal of Graphics Tools*, 4(2):1–6, 1999.
121. D. S. Meek and D. J. Walton. On surface normal and gaussian curvature approximations given data sampled from a smooth surface. *Computer Aided Geometric Design*, 17:521–543, 2000.
122. M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *International Workshop on Visualization and Mathematics*, Berlin-Dahlem, Germany, May 2002.
123. J. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, volume 334, pages 633–702. Elsevier Science, 2000.

124. J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitrou. The discrete geodesic problem. *SIAM J. of Computing*, 16(4):647–668, 1987.
125. N. J. Mitra and A. Nguyen. Estimating surface normals in noisy point cloud data. In *Symposium on Computational geometry*, pages 322–328. ACM Press, 2003.
126. N. J. Mitra, A. Nguyen, and L. Guibas. Estimating surface normals in noisy point cloud data. *International Journal of Computational Geometry and Applications*, 2004.
127. O. Monga, N. Armande, and P. Montesinos. Thin nets and crest lines: Application to satellite data and medical images. *Computer Vision and Image Understanding: CVIU*, 67(3):285–295, 1997.
128. O. Monga and S. Benayoun. Using partial derivatives of 3D images to extract typical surface features. *Computer Vision and Image Understanding: CVIU*, 61:171–195, 1995.
129. O. Monga, S. Benayoun, and O.D. Faugeras. From partial derivatives of 3-D density images to ridge lines. In *Proc. CVPR'92*, pages 354–359. IEEE, 1992.
130. H. P. Moreton and C. H. Sequin. Functional optimization for fair surface design. In *SIGGRAPH'92 Proceedings*, pages 167–176, August 1992.
131. R. Morris. The sub-parabolic lines of a surface. In G. Mullineux, editor, *Mathematics of Surfaces VI*, IMA new series 58, pages 253–262. Clarendon Press, 1996.
132. J. M. Morvan and B. Thibert. On the approximation of a smooth surface with a triangulated mesh. *Computational Geometry: Theory and Applications*, 33(3):337–352, 2002.
133. B. Mourrain. Computing isolated polynomial roots by matrix methods. *J. of Symbolic Computation, Special Issue on Symbolic-Numeric Algebra for Polynomials*, 26(6):715–738, Dec. 1998.
134. B. Mourrain. A new criterion for normal form algorithms. In M. Fossorier, H. Imai, Shu Lin, and A. Poli, editors, *Proc. AAECC*, volume 1719 of *LNCS*, pages 430–443. Springer, Berlin, 1999.
135. B. Mourrain and V. Y. Pan. Asymptotic acceleration of solving multivariate polynomial systems of equations. In *STOC '98 (Dallas, TX)*, pages 488–496. ACM, New York, 1999.
136. B. Mourrain and J.-P. Pavone. Subdivision methods for solving polynomial equations. Technical Report 5658, INRIA Sophia-Antipolis, 2005.
137. B. Mourrain, F. Rouillier, and M.-F. Roy. *Bernstein's basis and real root isolation*, pages 459–478. Mathematical Sciences Research Institute Publications. Cambridge University Press, 2005.
138. B. Mourrain and P. Trébuchet. Solving projective complete intersection faster. In C. Traverso, editor, *Proc. Intern. Symp. on Symbolic and Algebraic Computation*, pages 231–238. New-York, ACM Press., 2000.
139. B. Mourrain and Ph. Trébuchet. Generalised normal forms and polynomial system solving. In M. Kauers, editor, *Proc. Intern. Symp. on Symbolic and Algebraic Computation*, pages 253–260. New-York, ACM Press., 2005.
140. B. Mourrain, M. Vrahatis, and J. C. Yakoubsohn. On the complexity of isolating real roots and computing with certainty the topological degree. *J. of Complexity*, 18(2):612–640, 2002.
141. L. R. Nackman. Two-dimensional critical point configuration grpahs. *IEEE Trans. Pattren Analysis and machine Intelligence*, 6(4):442–450, 1984.
142. M. Nielsen, O. F. Olsen, M. Sig, and M. Sigurd. Koenderink corner points. In *Proceedings of the 4th International Workshop on Visual Form*, pages 420–430. Springer-Verlag, 2001.

143. G. M. Nielson and B. Hamann. The asymptotic decider: resolving the ambiguity in marching cubes. In *VIS '91: Proceedings of the 2nd conference on Visualization '91*, pages 83–91, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.
144. T. Nishita, T. W. Sederberg, and M. Kakimoto. Ray tracing trimmed rational surface patches. *Computer Graphics*, 24(4 (Proc. ACM Siggraph 90)):337–345, August 1990.
145. Y. Ohtake, A. Belyaev, and H.-P. Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Transactions on Graphics*, 23(3):609–612, August 2004. Proc. ACM SIGGRAPH 2004.
146. D. L. Page, A. Koschan, and M. Abidi. Perception-based 3D triangle mesh segmentation using fast marching watersheds. In *Proc. Intl. Conf. on Computer Vision and Pattern Recognition, Vol. II*, pages 27–32, 2003.
147. D. L. Page, A. Koschan, Y. Sun, J. Paik, and A. Abidi. Robust crease detection and curvature estimation of piecewise smooth surfaces from triangle mesh approximations using normal voting. In *Proceedings on Computer Vision and Pattern Recongition*, 2001.
148. D. L. Page, Y. Sun, A. Koschan, J. Paik, and M. Abidi. Normal vector voting: Crease detection and curvature estimation on large, noisy meshes. *Journal of Graphical Models*, 64:1–31, 2002.
149. N. M. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer-Verlag, Berlin and Heidelberg, 2002.
150. M. Pauly, R. Keiser, and M. Gross. Multi-scale feature extraction on point-sampled models. *Computer Graphics Forum*, 22(3):281–289, 2003. Eurographics 2003 issue.
151. X. Pennec, N. Ayache, and J. P. Thirion. Landmark-based registration using features identified through differential geometry. In I. N. Bankman, editor, *Handbook of Medical Imaging*, pages 499–513. Academic Press, 2000.
152. C. Petersen. Adaptive contouring of three-dimensional surfaces. *CAGD*, 1:61–74, 1984.
153. S. Petitjean. A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys*, 34(2), 2001.
154. G. Peyré and L. Cohen. Heuristically driven front propagation for geodesic paths extraction. In *Proceedings of VLSM'05*, pages 173–184. Springer LNCS, 2005.
155. V. Pham-Tron, N. Szafran, and L. Biard. Pseudo-geodesics on three-dimensional surfaces and pseudo-geodesic meshes. *Numerical Algorithms*, 26:305–315, 2001.
156. B. T. Phong. Illumination for computer generated pictures. *Communications of ACM*, 18(6):311–317, 1975.
157. T. Poeschl. Detecting surface irregularities using isophotes. *CAGD*, 1:163–168, 1984.
158. I. R. Porteous. Ridges and umbilics of surfaces. In R. R. Martin, editor, *The Mathematics of Surfaces II*, pages 447–458, Oxford, 1987. Clarendon Press.
159. I. R. Porteous. *Geometric Differentiation for the Intelligence of Curves and Surfaces*. Cambridge University Press, Cambridge, 1994.
160. I. R. Porteous and M. J. Puddephat. Landmarks of a surface. In R. Cipolla and R. R. Martin, editors, *Mathematics of Surfaces IX*, IMA new series, pages 114–125. Clarendon Press, 2000.
161. H. Pottmann. Visualizing curvature discontinuities of free-form surfaces. In *Proc. Eurographics'89*, pages 529–536, 1989.
162. E. Praun, H. Hoppe, and A. Finkelstein. Robust mesh watermarking. In *SIGGRAPH 99 Conference Proceedings*, pages 49–56, New York, NY, USA, 1999. ACM Press/ Addison-Wesley Publishing Co.
163. J. G. Ramsay. *Folding and Fracturing of Rocks*. McGraw Hill, 1967.
164. J. J. Risler. *Méthodes mathématiques pour la CAO*. Masson, 1991.
165. A. Rockwood. Accurate display of tensor product isosurfaces. In *IEEE Visualization '90 Conf.*, 1990.

166. C. Rössl, L. Kobbelt, and H.-P. Seidel. Extraction of feature lines on triangulated surfaces using morphological operators. In *Proceedings of the 2000 AAAI Symposium*, pages 71–75. AAAI Press, 2000.
167. S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Proc. of Second International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT)*, Thessaloniki, Greece, 2004.
168. M. A. Sabin. Contouring - the state of the art. In Earnshaw R.A., editor, *Fundamental Algorithms for Computer Graphics*, pages 411–482. Springer Verlag, 1985.
169. S. G. Scatterfield and D. F. Rogers. Contour lines from a b-spline surface. *IEEE CG & Appl.*, 5(4), 1985.
170. D. Schweitzer. Artificial texturing: an aid to surface visualization. *Computer Graphics*, 17(3):23–29, 1983.
171. T. W. Sederberg and R. J. Meyers. Loop detection in surface patch intersections. *Computer Aided Geometric Design*, 5(2):161–171, 1988.
172. T. W. Sederberg and J. Zheng. Algebraic methods for computer aided geometric design. In *Handbook of computer aided geometric design*, pages 363–387. North-Holland, Amsterdam, 2002.
173. E. C. Sherbrooke and N. M. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Comput. Aided Geom. Design*, 10(5):379–405, 1993.
174. K. Sloan. Surface normal (summary). In *Usenet comp.graphics article*, September 1991.
175. P.-P. J. Sloan, C. F. Rose, and M. F. Cohen. Shape by example. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 135–143, New York, NY, USA, 2001. ACM Press.
176. D. J. Struik. *Lectures on Classical Differential Geometry*. Dover Science, 1986.
177. G. Stylianou and G. Farin. Crest lines extraction from 3D triangulated meshes. In G. Farin, B. Hamann, and H. Hagen, editors, *Hierarchical and Geometrical Methods in Scientific Visualization*, pages 269–281. Springer, 2003.
178. Y. Sun, D. L. Page, J. K. Paik, A. Koschan, and M. A. Abidi. Triangle mesh-based edge detection and its application to surface segmentation and adaptive surface smoothing. In *Proc. Int. Conf. Image Processing, Vol. 3*, pages 825–828, 2002.
179. V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe. Fast exact and approximate geodesics on meshes. *ACM Trans. Graph.*, 24(3):553–560, 2005. Proceedings of SIGGRAPH'05.
180. G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proc. ICCV'95*, pages 902–907, 1995.
181. H. Theisel, C. Rössl, R. Zayer, and H.-P. Seidel. Normal based estimation of the curvature tensor for triangular meshes. In *Proc. Pacific Graphics*, pages 288–297, Seoul, South Korea, 2004.
182. J.-P. Thirion. The extremal mesh and the understanding of 3D surfaces. *International Journal of Computer Vision*, 19(2):115–128, 1996.
183. J.-P. Thirion. New feature points based on geometric invariants for 3D image registration. *International Journal of Computer Vision*, 18(2):121–137, May 1996.
184. J.-P. Thirion and A. Gourdon. The 3D marching lines algorithm and its application to crest lines extraction. *Graphical Models and Image Processing*, 58(6):503–509, 1996.
185. G. Thürmer and C. A. Wüthrich. Computing vertex normals from polygonal facets. *Journal of Graphics Tools*, 3(1):42–46, 1998.
186. P. Trébuchet. *Vers une résolution stable et rapide des équations algébriques*. PhD thesis, Université Pierre et Marie Curie, 2002.
187. B. L. van der Waerden. *Modern Algebra*. F. Ungar Publishing Co., New York, 3rd edition, 1950.

188. K. Watanabe and A. G. Belyaev. Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum*, 20(3):385–392, 2001. Eurographics 2001.
189. W. Welch and A. Witkin. Free–Form shape design using triangulated surfaces. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94*, pages 247–256, 1994.
190. F.-E. Wolter. Cut locus and medial axis in global shape interrogation and representation. Technical Report memorandum 92-2, MIT, Department of Ocean Engineering, January 1992.
191. A. L. Yuille. Zero crossings on lines of curvature. *Graphical Models and Image Processing*, 45(1):68–87, 1989.
192. A. L. Yuille and M. Leyton. 3D symmetry-curvature duality theorems. *Graphical Models and Image Processing*, 52(1):124–140, 1990.

Recent Advances in Remeshing of Surfaces

Pierre Alliez[1], Giuliana Ucelli[2], Craig Gotsman[3], and Marco Attene[4]

[1] INRIA, France pierre.alliez@sophia.inria.fr
[2] IGD / GraphiTech, Italy giuliana.ucelli@graphitech.it
[3] Technion, Israel gotsman@cs.technion.ac.il
[4] CNR, Italy attene@ge.imati.cnr.it

Summary. Remeshing is a key component of many geometric algorithms, including modeling, editing, animation and simulation. As such, the rapidly developing field of geometry processing has produced a profusion of new remeshing techniques over the past few years. In this paper we survey recent developments in remeshing of surfaces, focusing mainly on graphics applications. We classify the techniques into five categories based on their end goal: structured, compatible, high quality, feature and error-driven remeshing. We limit our description to the main ideas and intuition behind each technique, and a brief comparison between some of the techniques. We also list some open questions and directions for future research.

1 Introduction

Surface meshes are commonly used in many computer graphics applications to represent shapes. Many of these meshes are generated by scanning devices or by isosurfacing implicit representations. Unfortunately, such processes - especially if automated - are error-prone, and the resulting "raw" meshes are rarely satisfactory. Often they are oversampled and contain many redundant vertices. Besides needing to reduce the complexity of these meshes, which has stimulated a considerable amount of work in automatic mesh simplification [53], frequently the mesh quality, in terms of vertex sampling, regularity and triangle *quality*, must be improved. This improvement process is called *remeshing* (see, for example, Figure 1). The focus has been on ways to ease not only the display process, but also editing, animation, processing, storing and transmission. The following reviews several results of the past few years.

We invite the reader interested in related topics to read several comprehensive courses and tutorials on subdivision surfaces [72, 92], geometric modeling [41], digital geometry processing [81, 79] morphing [2], simplification and compression [53, 28, 3] and parameterization [22].

Fig. 1. Uniform remeshing of the Digital Michelangelo David model. Figure reproduced from [78].

1.1 Remeshing

The literature does not offer a precise universally accepted definition of remeshing. It often varies according to the targeted goal or application. Nonetheless, one possible definition could be: "Given a 3D mesh, compute another mesh, whose elements satisfy some quality requirements, while approximating the input acceptably". Quality herein has several meanings. It can refer to the sampling, grading, regularity, size or shape of elements. Often a combination of these criteria is desired in real applications. Some remeshing techniques proceed by altering the input, and some generate a new mesh from scratch.

1.2 Applications

Remeshing of surfaces is beneficial to a wealth of applications that take as input a meshed surface. These range from modeling to visualization through reverse engineering and simulation. All these applications execute some of the following, which require surface remeshing: creation and editing, animation, metamorphosis, approximation, simulation, denoising, smoothing and fairing, efficient rendering, compression, feature recovery and levels of detail.

1.3 Main Issues

We begin by listing briefly some general issues that arise during the remeshing process:

- **Validity.** The mesh has to be a *valid* mesh . This usually means that it should be a simple manifold. Typically it will also be closed; namely, it will not contain boundaries.

- **Quality.** The quality of mesh elements is crucial for robustness and numerical stability, required for numerical simulation as well as for geometry processing. Numerical computations, such as finite element analysis , require fairly regular meshes, both in terms of geometry and connectivity. These meshes are used to compute mechanical stress or solve heat and other differential equations. A high-quality mesh is required to minimize numerical errors and singularities that might otherwise arise (see [74]).
- **Fidelity.** The newly generated mesh should approximate the original shape geometry as closely as possible, while keeping the mesh complexity below a given budget. Ideally, "just enough" resolution for the problem being solved is sought. This involves choosing an error metric, as well as deciding between interpolation and approximation.
- **Discrete input.** The input is given as a discrete mesh, which is usually only an approximation of some (unknown) continuous shape. Having just this discrete approximation, and not the ideal shape, hampers most shape interrogation operations (e.g., normal, tangent plane, curvature estimations). Moreover, meshes generated from sampled point clouds by reconstruction algorithms may be contaminated by aliasing artifacts and lack important features present in the original.
- **Large data sets.** Modern 3D scanners generate very large datasets when the sampling rate is increased to ensure that no details are missed. As a result, the sampling and tessellation are insensitive to the shape, and the data is replete with redundancies.
- **Uncertainty.** Data obtained by an acquisition process such as laser scanning is often contaminated by electronic, mechanical or even optical noise present in the scanning pipeline.
- **Correspondence.** A central issue common to all remeshing techniques is to find the corresponding location of a new vertex on the input mesh surface. Such correspondence is typically found by parameterizing the input mesh. This is a complex problem, which can be computationally expensive, suffer from accuracy issues, and/or impose restrictions on the mesh. It is particularly problematic when performing the remeshing operations on a 2D parametric domain: the mapping of a nontrivial 3D structure (possibly a 3D mesh with arbitrary genus and holes) to a 2D parametric domain inevitably introduces some metric distortion, and may lead to the loss of important information. Furthermore, if the parameterization is combined with mesh segmentation, it is likely to encounter difficulties near the patch boundaries. Other parameter-free approaches work directly on the surface, and perform local modifications on the mesh (such as adding, removing, or relocating vertices). During these adaptations, the mesh vertices are forced to remain on the input mesh. This type of approach can be found in several different techniques [24, 23, 34, 33, 66, 83, 77]. The optimizations are either performed in 3D (which is computationally expensive), or in a tangent plane (which is faster, but less accurate). By using local operations, the approach may avoid the pitfalls of techniques based on global operations; and by performing the remeshing operations on a 2D plane, it is considerably faster than 3D optimizations. The distortion caused by mapping a 3D mesh to a 2D parametric domain can be considerably

reduced by using optimizations such as overlapping patches [77], and error accumulation (often caused by local operations) can be minimized by constantly comparing to a reference smooth approximation of the original geometry (e.g., by using triangular cubic Bzier patches such as PN triangles [85] or continuous patches [88]).

We now list some general desirable algorithmic functionalities of a remeshing algorithm:

- **Levels of detail.** Support for continuous Levels-of-Detail (*i.e.*, continuous-resolution representations) is often desirable for rendering and transmission applications. This poses a major challenge to remeshing algorithms.
- **Complexity.** With the increasing precision and resolution of modern acquisition devices, having to deal with meshes made of millions, or even billions, of faces is commonplace; thus, the speed of a remeshing algorithm is important. Often the main focus is on the trade-off between the quality of the result and the speed of the remeshing operation. Typically, close-to-linear runtime complexity is sought.
- **Theoretical guarantees.** Algorithms that guarantee the topology, matching of constraints, bounds on the distortion of geometry and normals, or bounds on the shape of elements are highly desirable for applications where certified results are required.

2 State of the Art

For each class of methods, this survey provides a definition of the characteristics that an algorithm must have, the motivations leading to the development of algorithms of each class, and a discussion of critical and open issues. To present state-of-the-art techniques as clearly as possible, we classify the remeshing techniques by their end goal rather than by the technique they employ. We identified five main categories of remeshing techniques:

- *Structured* remeshing (Section 2.1) - the connectivity graph of resulting meshes consists of regular patterns.
- *Compatible* remeshing (Section 2.2) - several meshes are modified to share a common connectivity structure.
- *High quality* remeshing (Section 2.3) - the shape of the elements as well as the vertex distribution are the main goals.
- *Feature* remeshing (Section 2.4) - preservation or even restoration of sharp features is the main focus when producing the resulting meshes.
- *Error-driven* remeshing (Section 2.5) - well-defined distances between the original and resulting surfaces are minimized (or bounded) while performing the remeshing.

Clearly, several of the characteristics mentioned above may be desirable simultaneously. In fact, some remeshing algorithms have been designed to produce a satis-

factory compromise within a particular application context. A neat separation, however, is necessary to produce a generic classification and a useful taxonomy, while trade-offs would need to be evaluated on a case-by-case basis.

2.1 Structured Remeshing

Definition

Structured remeshing replaces an unstructured input mesh with a structured one. In a structured mesh, sometimes called a regular mesh , all internal vertices are surrounded by a constant number of elements. A semi-regular mesh is obtained by regular subdivision of an irregular mesh (see [79]). All the vertices are regular except for a small number of *extraordinary* vertices (see Figure 2) . A highly regular mesh is one in which the vast majority of vertices are regular, yet the mesh has not necessarily been generated by subdivision.

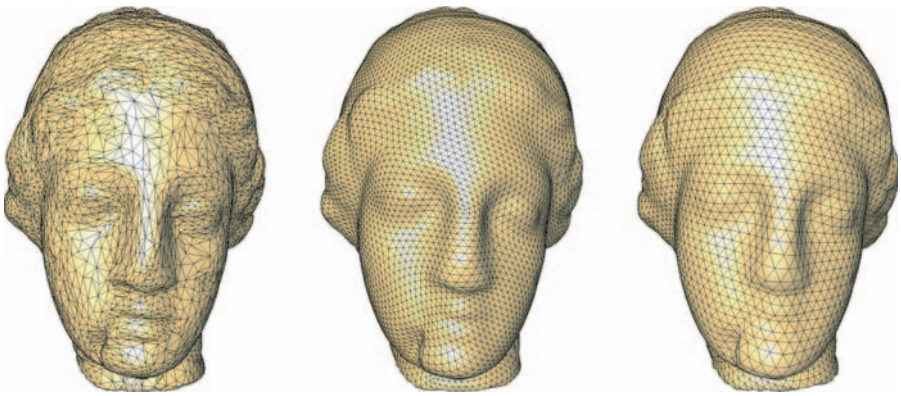


Fig. 2. Meshes: Irregular, semi-regular and regular.

Motivation

Structured meshes offer certain advantages over unstructured ones. Their connectivity graph is significantly simpler, hence allowing for efficient traversal and localization in the algorithms. Semi-regular meshes, which are essentially piecewise-regular, offer a trade-off between the simplicity of structured meshes and the flexibility of unstructured meshes.

Semi-Regular

Semi-regular meshes are obtained by recursive subdivision of an initial base mesh (Figure 3). Their hierarchical structure makes them ideal for multiresolution analysis (coarsification by downsampling and smoothing) and synthesis (subdivision and

adding of details). They have shown useful for modeling smooth or piecewise smooth surfaces, reverse engineering, multiresolution analysis and modeling, and morphing, editing and visualization with levels-of-detail applications.

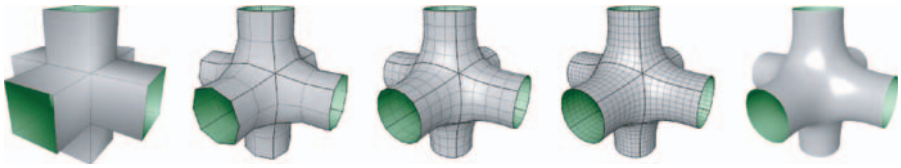


Fig. 3. Semi-regular mesh obtained by recursive subdivision of an initial base mesh.

The emerging field of geometry processing [79] has made significant use of semi-regular meshes. A fundamental question of geometry processing is the following: is it possible to extend the methods of classical digital signal processing (e.g., the discrete Fourier transform and wavelets), usually applied on regular uniform structures, to the irregular non-uniform setting? This question has only been partially solved, and the solution of choice consists of semi-regular remeshing of the original shape so that the geometric "signal" is resampled onto regular and uniformly sampled patches. One example of geometry processing is the set of discrete operators used for smoothing and fairing, applicable only in the regular and uniform setting.

The main techniques for semi-regular remeshing can be classified into two categories according to the way they find correspondences between the input and output meshes. The first class uses a parameterization to find a bijective correspondence. The techniques within this class differ mainly by the type of parameterization:

- Techniques that parameterize the input mesh on a global planar domain [35]. The parameter domain is then resampled, and the new mesh connectivity is projected back into 3D space, resulting in an improved version of the input (Figure 4). The main drawbacks of the global parameterization methods are the sensitivity to the specific parameterization used, and the metric distortion that may arise (due to the fact that the 3D structure is forced onto a foreign parameter plane). Furthermore, many of these techniques involve the solution of a large set of (sometimes nonlinear) equations, resulting in substantial computation. Sander et al. [69] used a hierarchical approach based on multigrid methods, which can accelerate the process to almost linear time even for large meshes. Nevertheless, numerical precision issues may arise for meshes with severe isoperimetric distortion.
- Techniques that parameterize the original model onto a set of base triangular domains, the latter obtained either by simplification, or by partitioning the original mesh into regions using a discrete analogue of the notion of a Voronoi tiling. This technique, used by [20, 47, 30], yields excellent results while being sensitive to the patch structure (see example Figure 5 and its colour version CP-1 in Appendix B). The vertex sampling is also sensitive to control.

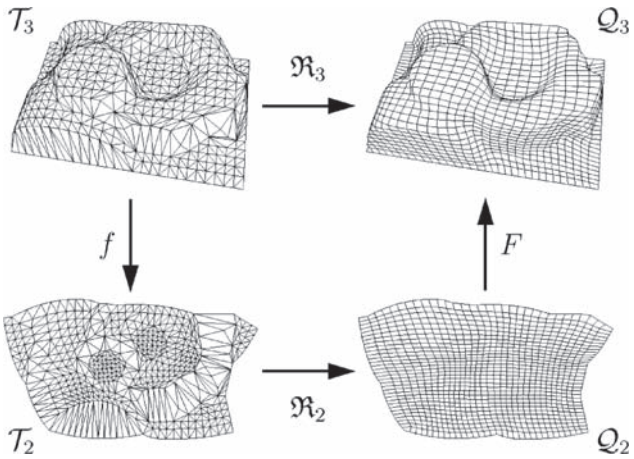


Fig. 4. Quadrilateral remeshing: The main idea of the algorithm is to circumvent the three-dimensional remeshing problem by flattening the 3D mesh T_3 to a 2D version T_2, and solving the two-dimensional problem instead. The deflation function f is then defined by linearly mapping each triangle of T_3 to the corresponding triangle in T_2 while the inverse inflation function F enables to get back from 2D to 3D. Figure reproduced from [35].

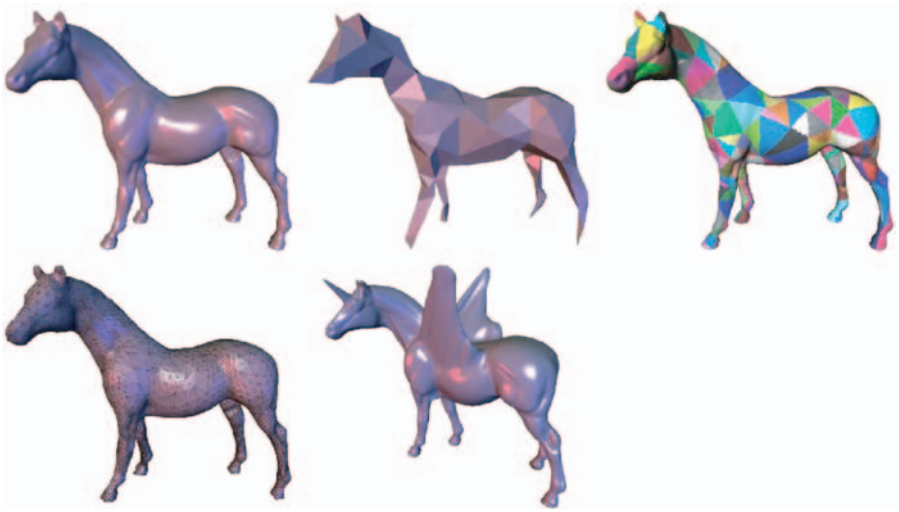


Fig. 5. Multiresolution adaptive parameterization of surfaces. Overview of the algorithm. Top left: a scanned input mesh (courtesy Cyberware). Next the parameter or base domain, obtained through mesh simplification. Top right: regions of the original mesh colored according to their assigned base domain triangle. Bottom left: adaptive remeshing with subdivision connectivity. Bottom middle: multiresolution edit. Figure reproduced from [47].

The second class of techniques does not rely on any parameterization but instead uses ray shooting [43] to find correspondences. These are then used to shrink wrap the new mesh onto the input mesh (Figure 6).

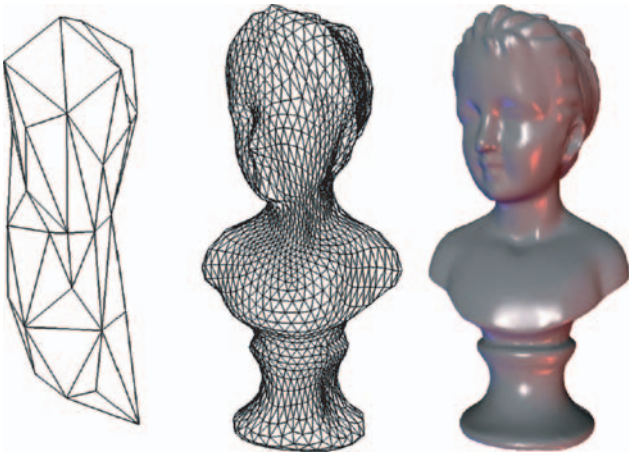


Fig. 6. Remeshing by shrink wrapping. The original bust model has 61K triangles. The base mesh with 72 triangles is subdivided three times to generate the center mesh and 5 times to generate the right image. Figure reproduced from [43].

Shape compression techniques employing *semi-regular remeshing* are among the best reported to date. The main idea behind these techniques [40, 30, 38, 60] is the observation that a mesh representation has three components: geometry, connectivity and parameterization, of which the latter two (*i.e.* connectivity and parameterization) are not important for the representation of the geometry. The goal is, therefore, to reduce the "volume" of these two components as much as possible by semi-regular remeshing (see [3] for a more detailed description of this shape compression technique).

Discussion

In all mapping-based methods, parameterization plays a critical role, and any deficiencies in it will be amplified in the output. In particular, building globally smooth parameterization is notoriously difficult [39]). Having subdivision connectivity is still necessary for multiresolution analysis, which has proved to be a powerful tool for many geometric modeling and processing applications. The challenge remains in how to handle irregular meshes directly. Moreover, this stumbling block will remain as long as current geometry processing approaches are designed in analogy to their continuous counterparts.

Completely Regular

In a regular mesh (a grid, triangle or hexagonal tessellation) the connectivity is implicit, the compactness and regularity of the data structure improve the efficiency and facilitate the implementation of many algorithms. Regular remeshing has been shown to be useful for efficient rendering (no cache indirection), texture and other modulation mapping (e.g., normal, transparency maps).

Gu et al. [29] remeshed irregular triangle meshes using a regular rectangular grid. The input mesh of arbitrary genus is initially cut to reduce it to a single topological disc. It is then parameterized on the unit 2D square while minimizing a geometric-stretch measure. This is then represented as a so-called *geometry image* that stores the geometry as well as any modulation map required for visualization purposes (see Figure 7 and its colour version CP-2 in the in Appendix B). Such a compact grid structure drastically simplifies the rendering pipeline since all cache indirections usually found in irregular mesh rendering are eliminated. Despite its obvious importance for efficient rendering, this technique has a few drawbacks due to the inevitable surface cutting: each geometry image has to be homeomorphic to a disk, therefore, closed or genus> 0 models have to be cut along a cut graph. In particular, it introduces unacceptably high parameterization distortion for high genus models or shapes with high isoperimetric ratios (e.g., long extremities). To alleviate these drawbacks, Sander et al. [70] used an atlas construction to map the input mesh onto charts of arbitrary shape. Those charts are then packed in a geometry image in parameter space, and a zippering algorithm is used to remove the discontinuities across chart boundaries and create a watertight surface. Another way to minimize seams due to cutting is to first parameterize the mesh to a sphere [27], which is then mapped in a highly structured way to the square.

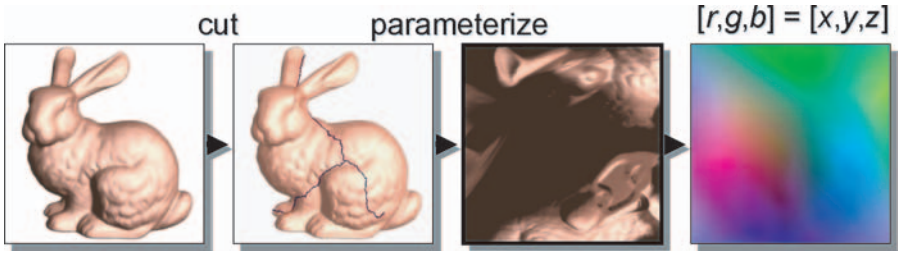


Fig. 7. Construction of a Geometry Image: Original mesh (70k faces, genus 0), original mesh with cut, parameterization and Geometry Image (257×257). Figure reproduced from [29].

Discussion

The concept of geometry images follows the recent trend in computer graphics to represent all surface modulation signals as "texture images" (normal maps, bump maps, transparency maps, color maps, light maps, reflection maps), instead of using

a fine mesh with attributes at each vertex. The key idea is to represent the shape geometry itself using regular grids, assuming the cost of 3D transformations to be negligible with respect to the cost of "decorating" the mesh using a complex multi-texturing process. Research on geometry images, mainly driven by Hoppe and co-workers, anticipates the unification of vertex and image buffers.

Highly Regular

Szymczak et al.[80] described a remeshing method for the creation of piecewise regular meshes . Based on their orientation, this algorithm partitions the triangles into six sets. The set of triangles whose normal is closest to the positive x-direction is sampled using a regular grid in the y-z plane. The other five sets are sampled similarly using the appropriate grids. Finally, these re-sampled pieces are connected into one valid mesh. The result typically contains a large fraction of regular vertices; specifically, all the internal vertices of each piece are regular by construction, while some irregular vertices may appear along the seams.

Surazhsky and Gotsman [77] performed local modifications directly on the mesh surface in order to obtain a highly regular mesh. One key feature of their method is the use of overlapping patches to locally parameterize the surface (which overcomes the problems of global parameterization and of remeshing that usually arise near the patch boundaries when parameterizing based on mesh segmentation). Another key feature is a series of edge-collapse and edge-flip operations combined with area-based mesh optimization to improve regularity and to produce well-shaped triangles (without the problem of long and skinny triangles typically created if mesh generation is based on triangle areas). As the overlapping parameterization allows 2D mesh optimization methods to be applied to 3D meshes (while minimizing the distortion problem, typical of mapping a 3D mesh to a 2D parametric domain), this algorithm is fast as well as robust (see an example in Figure 8).

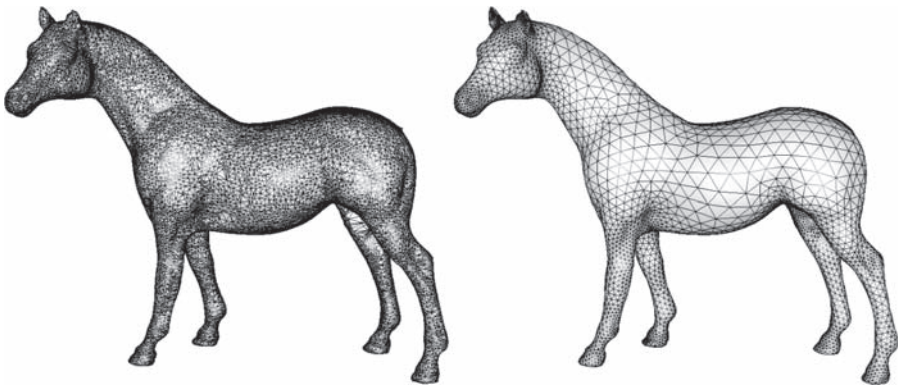


Fig. 8. Highly regular remeshing. Figure reproduced from [77].

Discussion

Highly regular meshes are frequently obtained by tessellating on a regular grid. Surazhsky and Gotsman [77] demonstrated that highly regular meshes cannot be generated simply by local mesh adaptation, unless some semi-global operations, such as drifting edges, are performed. One challenge is to obtain semi-regular meshes with a prescribed number of irregular vertices (up to that required by the Euler formula) by semi-global adaptation instead of subdivision.

2.2 Compatible Remeshing

Definitions

Given a set of 3D meshes with a partial correspondence between them, *compatible remeshing* amounts to generating a new set of meshes that are remeshes of the input set, such that they have a common connectivity structure, well-shaped polygons, approximate the input well, and respect the correspondence.

Motivation

Motivating applications are morphing between shapes and attributes, multi-model shape blending, synchronized model editing, fitting template models to multiple data sets and principal component analysis. In these applications the common connectivity is usually more important than the mesh element quality.

Joint Parameterization

Much of the work done on compatible meshing focuses on morphing as the target application. This first requires the computation of a *joint parameterization* (sometimes called *cross parameterization*) , namely, a bijective mapping between the two meshes, possibly subject to some constraints. Alexa [2] provided a good review of joint parameterization and compatible remeshing techniques developed for morphing. Joint parameterization is typically computed by parameterizing the models on a common base domain. One popular choice is the sphere. A number of algorithms for spherical parameterization exist, e.g., [1, 27, 64]. Of these, only Alexa's method addresses feature correspondence (see Figure 9). However, it does not guarantee a bijective mapping and is not always capable of matching the features. An inherent limitation of a spherical parameterization is that it can only be applied to closed, genus zero surfaces.

A more general approach is to parameterize the models over a common base mesh [46, 48, 55, 65]. This approach splits the meshes into matching patches with an identical inter-patch connectivity. After the split, each set of matching patches is parameterized on a common convex planar domain. An advantage of this approach is that it naturally supports feature correspondence by using feature vertices as corners of the matching patches. The main challenge in mapping the models to a single base mesh is to construct identical inter-patch connectivities. The vast majority of the

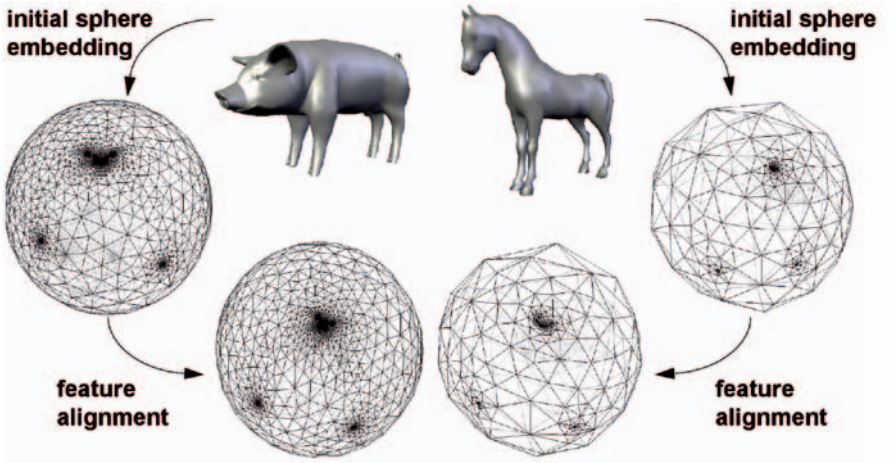


Fig. 9. Joint spherical parameterization: First, an initial sphere embedding is computed for each mesh. Second, the initial subdivision is deformed such that the common features coincide on the spheres. The two connectivities are then merged. Figure reproduced from [1].

methods use heuristic techniques that work only when the models have nearly identical shape. Praun et al. [65] provided a robust method for partitioning both meshes into patches given user-supplied base mesh connectivity. A common disadvantage of existing techniques for constructing base meshes is that the patch structure severely restricts the freedom of the parameterization. As a result, the shape of the patches has a huge influence on the amount of mapping distortion.

Given joint parameterization, many techniques [1, 37] generate the common connectivity for the models by overlaying the meshes in this parameter domain and computing a common intersection mesh. The new mesh captures the geometry of the models. However, the new mesh is typically much larger than the input meshes and has very badly shaped triangles. The overlaying algorithm is also extremely tricky to implement, as it requires multiple intersection and projection operations. An alternative is to remesh the models using a regular subdivision connectivity derived from the base mesh [46, 55, 65]. Due to the rigid connectivity structure, the shape of the mesh triangles reflects the shape of the base mesh. Thus, if the shape of the triangles is poor (because, for example, the user picked unevenly spaced feature vertices), the shape of the mesh triangles will reflect this. More importantly, a model that contains features interior to the base mesh triangles will require a very dense subdivision mesh over the entire model.

Inter-Surface Mapping

Kraevoy and Sheffer [45] developed a technique for joint parameterization and compatible remeshing of two genus-0 meshes with partial correspondence (Figure 10). The input of the algorithm is a pair of triangle meshes and a set of corresponding

feature vertices. The first stage of the algorithm constructs a common base domain by incrementally adding pairs of matching shortest edge paths. Care is taken to avoid intersections and blocking, as well as to preserve cyclic orders so as to obtain matching patch layouts. Face paths are then added until all patches are triangulated, and an additional path flip procedure improves the connectivity of the patch layout. The second stage computes a shape preserving parameterization with smooth transitions between patches using the mean-value parameterization followed by an adjacency preserving smoothing procedure. The last stage constructs compatible meshes by alternating vertex relocation to attract vertices toward areas of higher error, and error-driven mesh refinement. The approximation of normals is improved by an additional pseudo edge-flip refinement procedure. The meshes generated by this procedure contain significantly fewer elements than those generated by simple overlaying methods, while approximating well the geometry and normals of the input model.

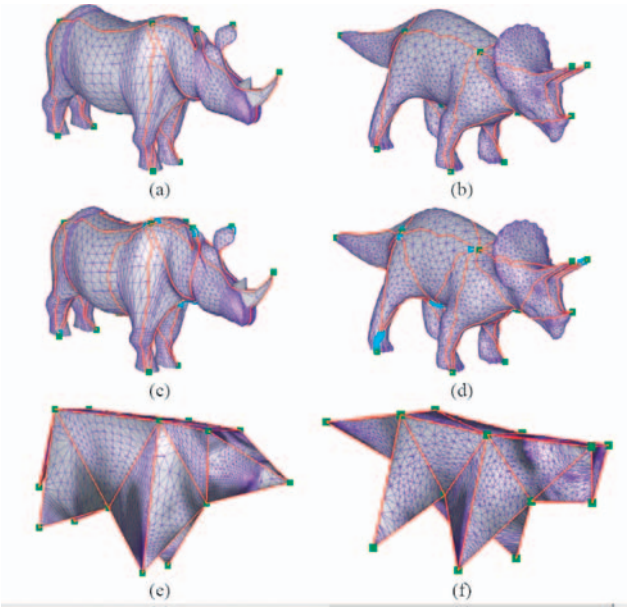


Fig. 10. Base domains construction for joint parameterization and compatible remeshing of two genus-0 meshes (feature vertices are dark green): (a),(b) edge paths; (c),(d) face paths, new vertices are highlighted (turquoise); (e),(f) base meshes. Figure reproduced from [45].

Schreiner et at. [71] used a procedure similar to that of Kraevoy and Sheffer for base mesh construction, handling models of arbitrary genus more robustly. To generate a smooth joint parameterization, they used a symmetric, stretch based relaxation procedure, which trades off high computational complexity for quality of the mapping. The common mesh is generated using an overlay of the input meshes,

as described above. To avoid artifacts, the method has to relax the feature vertex correspondence in some cases.

Discussion

While compatible remeshing is becoming increasingly important in computer graphics animation applications, where a sequence of meshes is available, it is still plagued by a number of problems. The selection of pairs of corresponding feature points is still manual. Very few methods extend easily to arbitrary genus surfaces and long animation sequences. Lastly, the results are still highly dependent on the parameterization method used to perform the joint parameterization.

2.3 High Quality Remeshing

Definitions

In our taxonomy *high quality remeshing* means to generate a new discretization of the original geometry with a mesh that exhibits the following three properties: well-shaped elements, uniform or isotropic sampling and smooth gradation sampling. A well-shaped triangle has an aspect ratio as close to 1 as possible, and a well-shaped quadrilateral contains angles between two consecutive edges as close to $\pi/2$ as possible. Isotropic sampling means that the sampling is locally uniform in all directions. Requiring uniform sampling is even more restrictive since it mandates the sampling to be uniform over the entire mesh. Smooth gradation means that if the sampling density is not uniform - it should vary in a smooth manner [13].

Motivation

High quality remeshing is motivated by numerical stability and reliability of computations for simulation. Efficient rendering, interactive free-form shape modeling, as well as a few geometry processing algorithms such as compression, fairing or smoothing also benefit from high quality meshes. The shape of mesh elements [61] has a direct impact on the numerical stability of numerical computations for finite element analysis, as well as for efficient rendering. For popular triangle meshes, it is desirable to have no small angles and/or no large angles, depending on the targeted computations (see [74]).

We restrict our description to *point-based* sampling techniques, although other primitives can be evenly distributed on surfaces for meshing (*e.g.* bubble packing [90], square cell packing [75], placement of streamlines [6]). Uniform (resp. isotropic) point sampling for remeshing amounts to globally (resp. locally) distributing a set of points on the input model in as even a manner as possible. We may distinguish between *greedy* sample placement methods that insert one point at a time to refine the newly generated model, and *relaxation-based* methods that improve an initial placement either locally or globally through point relocation.

Farthest point sampling.

The *farthest point* paradigm [49] advocates inserting one sample point at a time, as far as possible from previously placed samples, *i.e.*, at the center of the biggest void. Its main advantage is in retaining the uniformity while increasing the density. In contrast to stochastic approaches, it can guarantee some uniformity by bounding the distance between samples [12]. This paradigm, called Delaunay refinement [15, 68, 56] or sink insertion [21] as a variant, has shown effective in producing uniform as well as isotropic sample placements. Recently it has been extended using the geodesic distance estimated on the input mesh to find the center of the biggest voids [63, 57]. From an initial point set sampled on the input mesh, a Delaunay-like triangulation is created by taking the dual of a geodesic-based Voronoi diagram constructed using the Fast Marching method of Sethian and Kimmel [73].

Advancing front.

A popular method for evenly-spaced placement is the advancing front paradigm commonly used for meshing [11, 31, 82]. This method has recently been extended using an approximation of the geodesic distance for remeshing by Sifri et al. [76]. A more general approach was introduced by Dong et al. [18], who computed a harmonic Morse function on the mesh surface. Drawing isocontours of this function, and placing a set of orthogonal streamlines results in a good quad remesh (Figure 11). Another quasi-uniform remeshing approach based on an advancing front is implicit in the SwingWrapper compression scheme [10]. To reduce the number of bits needed to encode the vertex locations, SwingWrapper partitions the surface into geodesic triangles that, when flattened, constitute a new, strongly compressible mesh. The remeshing is performed so that for each vertex of the new mesh there is at least one incident isosceles triangle having a prescribed height. Though not optimally uniform, the remeshing performed by SwingWrapper might effectively be used as an initial guess for iterative processes that try to optimize uniformity.

Attraction-repulsion.

One of the first remeshing techniques to surface in the graphics community was described by Turk [83]. It places a (user defined) number of new vertices on the input mesh, and arranges the new vertices with the help of an attraction-repulsion particle relaxation procedure, followed by an intermediate mutual tessellation that contains both the vertices of the original mesh and the new vertices. This simple approach produced quite remarkable results, although it had several limitations. Most notably, it is not suitable for models that have sharp edges and corners, as it does not precisely approximate such surfaces.

Umbrella operator.

Another popular method commonly used for even placement of samples consists of repeatedly moving each sample point to the barycenter of its neighbors, and updating

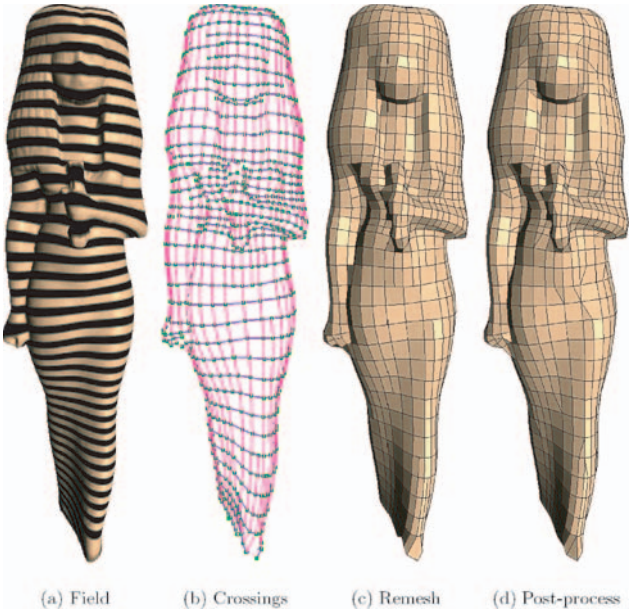


Fig. 11. Quadrilateral remeshing of arbitrary manifolds: (a) A harmonic function is computed over the manifold. (b) A set of crossings along each flow line is constructed. (c) A non-conforming mesh is extracted from this net of flow crossings. (d) A post-process produces a conforming mesh composed solely of triangles and quadrilaterals. Figure reproduced from [18].

the mesh connectivity. This procedure tends to generate globally uniform edges in the simple case, and locally uniform edges (*i.e.* isotropic sampling) if weights are assigned to edges [87].

The interactive remeshing technique introduced by Alliez et al. [5] is based on global parameterization. It represents the original mesh by a series of 2D maps in parameter space, and allows the user to control the sampling density over the surface patch using a so-called control map, the latter created from the 2D maps. First, an initial isotropic resampling is performed using an error-diffusion sampling technique originally designed for image half-toning [58], followed by relaxation using the umbrella operator. This method is a hybrid between a greedy method and a variational method since the coefficients used for error diffusion are optimized during an off-line procedure that seeks a placement with a so-called blue-noise profile, related to the notion of isotropic sampling. The initial sample placement is then performed in a single pass at run time; see Figure 12.

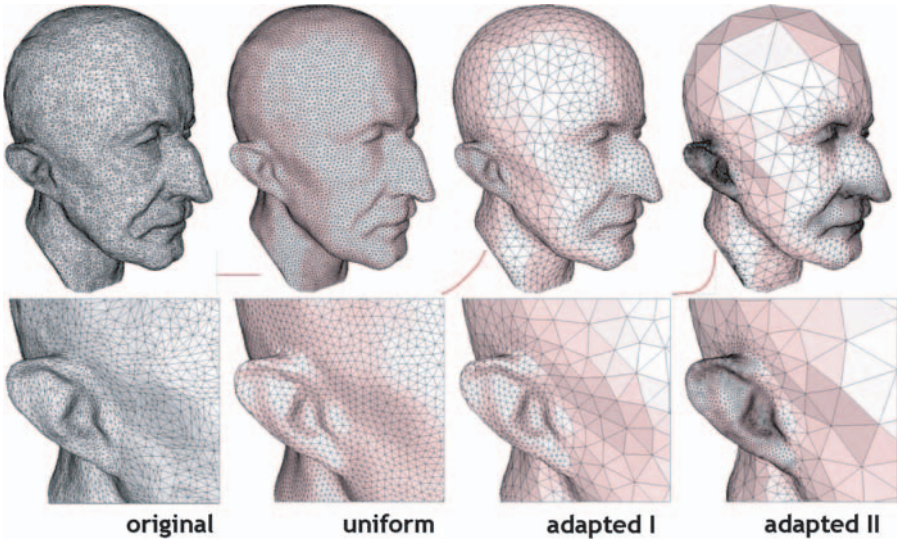


Fig. 12. Interactive geometry remeshing: Remeshing of the MaxPlanck model with various distribution of the sampling with respect to the curvature. The original model (left) is remeshed uniformly and with an increasing importance placed on highly curved areas (left to right) as the magnified area shows. Figure reproduced from [5].

Local area equalization.

Precise uniform sampling can be achieved through local area equalization. Assuming that the neighbors of the vertex to be relocated is fixed, the new position is computed by solving a linear system in order to minimize the area dispersion among all incident triangles [77]. This technique has recently been extended to local equalization of the Voronoi areas of the vertices in order to symmetrize a linear system used for multiresolution modeling [14]. The system is solved efficiently using a Cholesky-based solver that takes advantage of symmetric band-limited matrices. Although efficient and robust, these area equalization techniques do not provide an easy way to globally distribute a set of samples in accordance to a density function.

Lloyd relaxation.

Isotropic sample placement can be achieved by applying the Lloyd clustering algorithm [50], which consists of alternating Voronoi partitioning with relocation of the generators to the centroid of their respective Voronoi cell (Figure 13). Such a relaxation procedure generates centroidal Voronoi diagrams [19], where the generators coincide with the centroid of their respective cells. Lloyd relaxation minimizes energy related to the compactness of the Voronoi cells (and hence to isotropic sampling) while equi-distributing the energy within each cluster, as shown by Gersho in the late seventies [26]. Contrary to other methods, this method allows the definition

of a density function related to the desired size of each Voronoi cell. It will then generate a distribution of energy which globally matches the local size while achieving isotropic sampling.

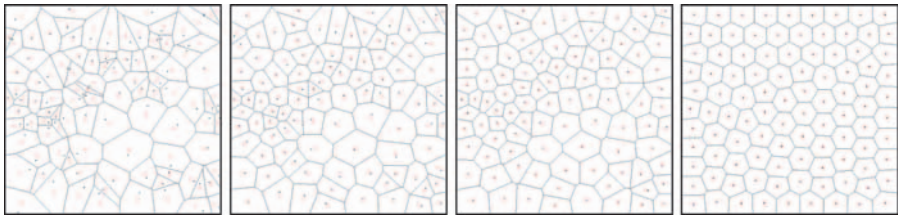


Fig. 13. Lloyd relaxation: A set of generators (black dots) are randomly generated (the centroid of each Voronoi cell is depicted as a red circle). Each iteration of the Lloyd algorithm moves each generator to its associated centroid, and updates the Voronoi diagram.

Alliez *et al.* [7], and Surazhsky *et al.* [78] proposed two remeshing techniques based on Lloyd relaxation. The first uses a global conformal planar parameterization and then applies relaxation in the parameter space using a density function designed to compensate for the area distortion due to flattening (Figure 14). To alleviate the numerical issues for high isoperimetric distortion, as well as the artificial cuts required for closed or genus models, the second approach applies the Lloyd relaxation procedure on a set of local overlapping parameterizations (Figure 15). More recently, the Lloyd-based isotropic remeshing approach has been extended in two directions: one uses the geodesic distance on triangle meshes to generate a centroidal geodesic-based Voronoi diagram [62], while the other is an efficient discrete analog of the Lloyd relaxation applied on the input mesh triangles [84].

Discussion

As expected, relaxation-based sample placement methods achieve better results than greedy methods, at the price of lengthier computations. Nevertheless, the only methods that provide certified bounds on the shape of elements are the greedy approaches based on Delaunay refinement. The Lloyd-based isotropic sampling method combined with local overlapping parameterization has been successful at isotropically distributing a point set in accordance with a density function [78]. Two remaining challenges related to the Lloyd relaxation method are to prove or to give sufficient conditions for achieving convergence to a global optimum, and to accelerate convergence. Another promising direction for efficient isotropic sampling is the hierarchical Penrose-based importance sampling technique developed by Ostromoukhov [59], which is deterministic and several orders of magnitude faster than relaxation methods.

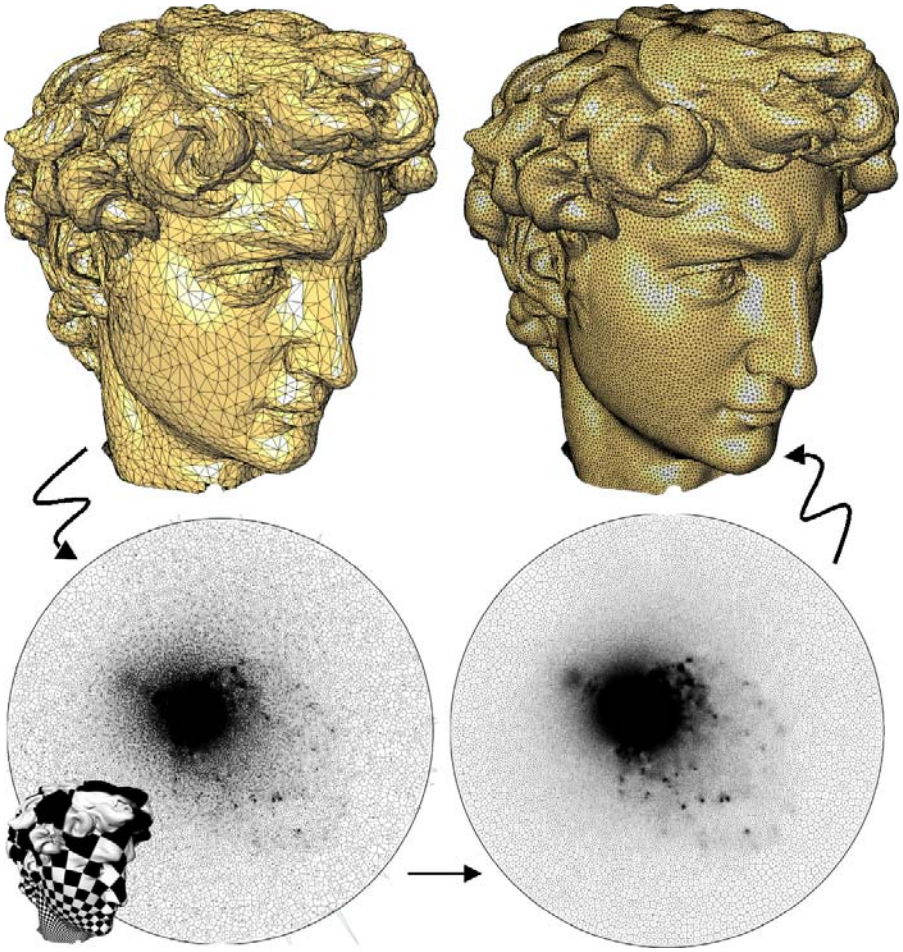


Fig. 14. Uniform remeshing of the David head: a planar conformal parameterization is computed (bottom left). Then Lloyd relaxation is applied in parameter space in order to obtain a weighted centroidal Voronoi tessellation, with which the mesh is uniformly resampled. Figure reproduced from [7].

2.4 Feature Remeshing

Definitions

Assume that a triangle mesh is an approximation of a curved shape, possibly with sharp edges and corners. We call the process that takes such a triangle mesh and generates a new tessellation in which the original sharp features are preserved, feature remeshing. In this context, the quality of the approximation may be measured either using a purely geometric metric (the L^∞ norm, for example, is strongly affected by

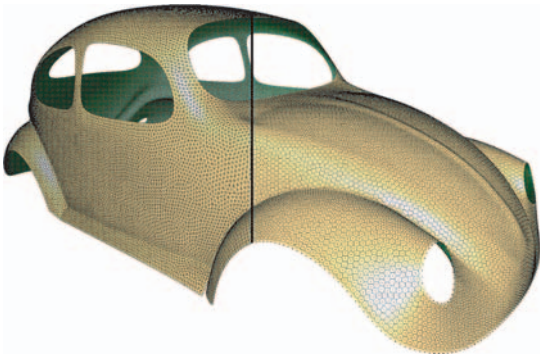


Fig. 15. Uniform remeshing of the Beetle: Lloyd relaxation is applied over local overlapping parameterizations as described in [78].

badly-approximated sharp corners), or by a metric that reflects visual-quality (e.g., normal deviation), or a combination of both.

Motivation

Most acquisition techniques, as well as several recently developed remeshing algorithms [67, 80, 29, 10], restrict each sample to lie on a specific line or curve whose position is completely defined by a pre-established pattern. In most cases, such a pattern cannot be adjusted to coincide with a model's sharp edges and corners, and almost none of the samples will lie on such sharp features. Thus, the sharp edges and corners of the original shape are removed by the sampling process and replaced by irregularly triangulated chamfers, which often result in a poor-quality visualization and high L^∞ distortion.

Feature-preserving

When the original shape is available, the error between such a shape and the approximating triangle mesh may be reduced by dense sampling. Over-sampling, however, will significantly increase the number of vertices, and thus the associated complexity, transmission and processing costs. Furthermore, as observed by Kobbelt et al. [44], the associated aliasing problem will not be solved by over-sampling, since the surface normals in the reconstructed model will not converge to the normal field of the original object. To cope with this problem, an extended marching cubes algorithm was proposed in [44]. The input shape is first converted into a signed distance field. This representation is then polygonized using a variant of the marching-cubes [51] algorithm in which vertex normals are derived from the distance field and used to decide whether a voxel contains a sharp feature or not. If they do appear, additional vertices are created within the voxel and placed at intersections between the planes defined by the vertices and their associated normals. Another feature-preserving approach was proposed in [36]. It is able to accurately polygonize models with sharp

features using adaptive space subdivision (an octree), resulting in polygonal models with fewer faces. In a different setting, an original triangulation may be remeshed without converting it into a scalar distance field, and the aliasing problem may be avoided by snapping some of the evenly distributed vertices onto sharp creases, as proposed in [86].

Feature-enhancing

When the original shape is not available, the EdgeSharpener method [9] provides an automatic procedure for identifying and sharpening the chamfered edges and corners. In a first step, the mesh is analyzed and the average dihedral angle at the edges is computed. Based on this value, "smooth" regions are grown on the mesh, and the strips of triangles separating neighboring smooth regions are considered "aliasing artifacts" made of chamfer triangles . The growing process results in a number of smooth regions in which all the internal edges have a nearly flat dihedral angle. EdgeSharpener infers the original sharp edges and corners by intersecting planar extrapolations of the smooth regions. Then, each chamfer triangle is subdivided, and the newly inserted vertices are moved to the intersections, which are assumed to better approximate the original sharp features (see Figure 16). Unless the input contains significant amounts of noise, EdgeSharpener does not introduce undesirable side-effects, and limits the modifications to the portions of the mesh that are actually chamfer artifacts. EdgeSharpener has been tested on the results of several feature-insensitive remeshing algorithms [10, 80, 67], and has been shown to significantly reduce the L^∞ distortion introduced by the remeshing process.

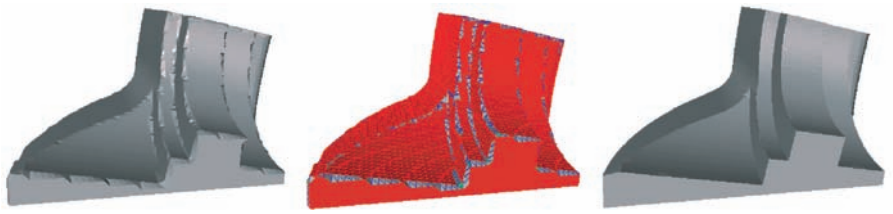


Fig. 16. EdgeSharpener: A triangle mesh reconstructed from a point cloud (left) is improved by EdgeSharpener [9]. Smooth regions are identified (red) and chamfer triangles (gray and green with blue edges) are sharpened (right).

To give designers more flexibility, an interactive remeshing approach has been proposed [42] for restoring corrupted sharp edges. The user is required to construct a number of fish bone structures (spine and orthogonal ribs) that will be automatically tessellated to replace the original chamfers. Though not automatic, this method is particularly suitable for simple models with few sharp edges, and allows the designer to sharpen the chamfers as well as to modify the swept profiles to produce blends or decorated edges.

One may argue that an application of the extended marching cubes [44] to a polygonal mesh may be used to infer, and hence reconstruct, the sharp features. In [44], this application to remeshing is discussed and, in fact, it is useful in improving the quality of meshes having degenerate elements or other bad characteristics. In some cases, the information at the edge-intersections makes it possible to reconstruct sharp features in an EdgeSharpener-like manner. For example, if a cell contains an aliased part that does not intersect the cell's edges, the normal information at the intersections is used to extrapolate planes, and additional points are created on the inferred sharp feature. If, on the other hand, the cell's edges do intersect the aliased part, the normal information is contaminated, and nothing can be predicted about any possible feature reconstruction. Moreover, remeshing the whole model through the extended marching cubes approach can introduce an additional error on the regions without sharp features, while the local remeshing produced by EdgeSharpener only affects the aliased zones by subdividing the triangles that cut through the original solid (or through its complement) near sharp edges.

Discussion

The ability to preserve or reconstruct sharp features is undoubtedly important. Methods that do not assume the availability of the original surface, however, must necessarily rely on heuristics to infer and restore sharp edges and corners in an aliased model. Thus one of the main challenges in this context is the definition of a formal framework for sampling non-smooth surfaces. Although such a framework has been defined for smooth models [8, 12], the problem of dealing with tangential discontinuities remains open, even for the 2D case [17].

2.5 Error-driven Remeshing

Definitions

Error-driven remeshing amounts to generating a mesh that maximizes the trade-off between complexity and accuracy. The complexity is expressed in terms of the number of mesh elements, while the geometric accuracy is measured relative to the input mesh and according to a predefined distortion error measure. The efficiency of a mesh is qualified by the error per element ratio (the smaller, the better). One usually wants to minimize the approximation error for a given budget of elements, or conversely, minimize the number of elements for a given error tolerance. Another challenging task consists of optimizing the efficiency trade-off at multiple levels-of-detail.

Motivation

Efficient representation of complex shapes is of fundamental importance, in particular for applications dealing with digital models generated by laser scanning or isosurfacing of volume data. This is mainly due to the fact that the complexity of numerous

algorithms is proportional to the number of mesh primitives. Examples of related applications are modeling, processing, simulation, storage or transmission. Even for most rendering algorithms, polygon count is still the main bottleneck. Being able to automatically adapt the newly generated mesh to the local shape complexity is of crucial importance in this context.

Mesh simplification or refinement methods are obvious ways of generating efficient meshes. In this survey we will not pretend to survey the plethora of polygonal simplification techniques published in the last decade, and instead refer the interested reader to the a multitude of comprehensive course notes and surveys [32, 25, 52, 53, 28]. We complement these documents by focusing on techniques that proceed by optimization or by recovering a continuous model from the input mesh. These include techniques specifically designed to exploit a shape's local planarity, symmetry and features in order to optimize its geometric representation. We focus in more detail on techniques that construct efficient meshes by extracting, up to a certain degree, the "semantical content" of the input shape.

Hoppe et al. [34] formulated the problem of efficient triangle remeshing as an optimization problem with an energy functional that directly measures the $\mathcal{L}^2$ error deviation from the final mesh to the original one. They showed that optimizing the number of vertices, as well as their geometry and connectivity, captures the curvature variations and features of the original geometry. Despite a spring force restricting the anisotropy of the results and an approximate point-to-surface Euclidean $\mathcal{L}^2$ distance measure, this technique results in particularly efficient meshes. Alliez et al. [4] described another optimization method that minimizes the volume between the simplified mesh and the input mesh using a gradient-based optimization algorithm and a finite-element interpolation model implicitly defined on meshes. The volume-based error metric is shown to accurately fit the geometric singularities on 3D meshes by aligning edges appropriately, without any distinction required between smooth and sharp areas.

Following previous work on feature remeshing (see Section 2.4), the remeshing technique introduced by Alliez et al. [6] pushes the idea of aligning edges on features further by generalizing it to the entire surface. They generated a quad-dominant mesh that reflects the symmetries of the input shape by sampling the input shape with curves instead of the usual points. The algorithm has three main stages. The first stage recovers a continuous model from the input triangle mesh by estimating one 3D curvature tensor per vertex. The normal component of each tensor is then discarded and a 2D piecewise linear curvature tensor field is built after computing a discrete conformal parameterization. This field is then altered to obtain smoother principal curvature directions. The singularities of the tensor field (the umbilics) are also extracted. The second stage consists of resampling the original mesh in parameter space by building a network of lines of curvatures (a set of "streamlines" approximated by polylines) following the principal curvature directions. A user-prescribed approximation precision in conjunction with the estimated curvatures is used to define the local density of lines of curvatures at each point in parameter space during the integration of streamlines. The third stage deduces the vertices of the new mesh

by intersecting the lines of curvatures on anisotropic areas and by selecting a subset of the umbilics on isotropic areas (estimated to be spherical). The edges are obtained by straightening the lines of curvatures in-between the newly extracted vertices on anisotropic areas, and simply deduced from the Delaunay triangulation on isotropic areas. The final output is a polygon mesh with mostly elongated quadrilateral elements on anisotropic areas, and triangles on isotropic areas. Quads are placed mostly on regions with two (estimated) axis of symmetry, while triangles are used to either tile isotropic areas or to generate conforming convex polygonal elements. On flat areas the infinite spacing of streamlines will not produce any polygons, except for the sake of convex decomposition (see example Figure 17). This approach has recently been extended to reduce its dependence on any parameterization [54].

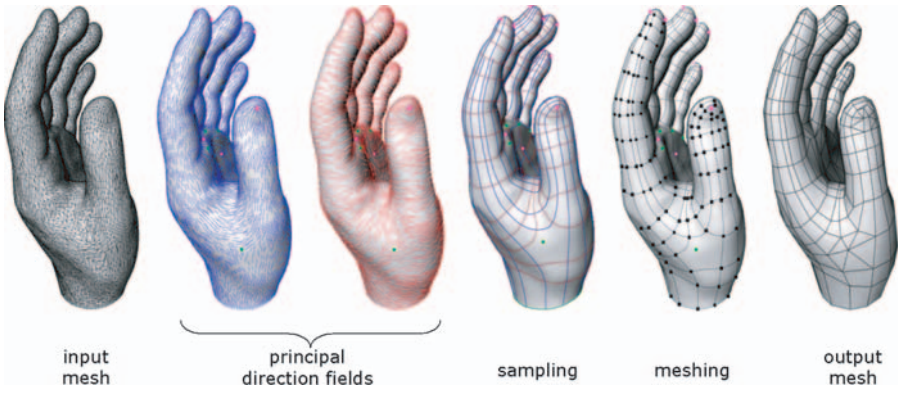


Fig. 17. Anisotropic remeshing: From an input triangulated geometry, the curvature tensor field is estimated, then smoothed, and its umbilics are deduced (colored dots). Lines of curvatures (following the principal directions) are then traced on the surface, with a local density guided by the principal curvatures, while usual point-sampling is used near umbilic points (spherical regions). The final mesh is extracted by subsampling, and conforming-edge insertion. The result is an anisotropic mesh, with elongated quads aligned to the original principal directions, and triangles in isotropic regions. Figure reproduced from [6].

Although the edge sampling strategy described above increases the mesh efficiency by matching the conditions of optimality for the $\mathcal{L}^2$ metric in the limit, there is no guarantee of its efficiency at coarse scales. Moreover, this technique involves local estimation of curvatures, known to be difficult on discrete meshes. The estimator itself requires the definition of a scale that remains elusive (intuitively, the scale itself should depend on the approximation tolerance). These observations motivate an efficient remeshing approach based exclusively on the approximation error. Thus Cohen-Steiner et al. [16] proposed an error-driven clustering approach that does not resort to any estimation of differential quantities or any parameterization. Error-driven remeshing is now cast as a variational partitioning problem where a set of planes (so-called proxies) are iteratively optimized using Lloyd's heuristic to minimize a predefined approximation error (see Figure 2.5 and its colour version CP-3 in

Appendix B). As in the original Lloyd algorithm, the key idea hinges on alternating partitioning and moving each representative to the centroid of its region. The partitioning is generated triangle by triangle using a region growing procedure driven by a global priority queue. The queue is sorted by the error between each new triangle candidate for expansion and the proxy (representative) of the corresponding region. The analog of the centroid in the metric space is now simply the best fit proxy for each region. Closed forms for the errors between one triangle and one proxy, as well as for the best fit proxy are given for regions consisting of a set of triangles, both for the $\mathcal{L}^2$ and $\mathcal{L}^{2,1}$ ($\mathcal{L}^2$ deviation of normals) error metric. A polygonal remeshing technique is proposed based on a discrete analog of a Voronoi diagram implemented with a two-pass partitioning algorithm over the input triangle mesh. The elements of the resulting polygonal meshes will then exhibit orientation and elongation guided by the minimization of the approximation error instead of being the result of a curvature estimation process as in [6]. This technique has been extended by Wu and Kobbelt to handle non planar proxies such as spheres, cylinders, and rolling ball blend patches [89], and by Yan et al. to handle quadric proxies [91].

Fig. 18. Error-driven remeshing: Through repeated error-driven partitioning (left), a set of geometric proxies (represented as ellipses, center) is optimized. These proxies are then used to construct an approximating polygonal mesh (right). Figure reproduced from [16].

Discussion

In this section we narrowed our scope to the study of methods that best preserve the shape geometry during the remeshing stage of the geometry processing pipeline. Despite the considerable amount of work done on mesh approximation through error-driven simplification or refinement, there is much less work on approximating shapes by using geometric analysis to guide the remeshing process.

Observations have shown that for sketching, artists implicitly exploit the symmetry of a shape as they sketch images that best convey the desired model. Simple symmetric primitives such as planes, spheres, ellipses, saddles, cylinders and cones are also exploited by artists as basic components for modeling a shape. For reverse

engineering, remeshers such as [89, 91] help, to a certain degree, to automatically capture the "semantical" structure of a measured shape by inferring a smooth model and extracting its main traits. The local symmetries and main traits of the shape should ideally be deduced from the elements of the mesh, facilitating structuring and analysis.

References

1. M. Alexa. Merging polyhedral shapes with scattered features. In *Proceedings of the International Conference on Shape Modeling and Applications (SMI-99)*, pages 202–210, 1999.
2. M. Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2):173–196, 2002.
3. P. Alliez and C. Gotsman. Recent advances in compression of 3d meshes. In *Proceedings of the Symposium on Multiresolution in Geometric Modeling*, 2003.
4. P. Alliez, N. Laurent, H.Sanson, and F. Schmitt. Mesh approximation using a volume-based metric. In *Proceedings of the 7th Pacific Conference on Computer Graphics and Applications 1999*, pages 292–301, Los Alamitos, 1999. IEEE Computer Society.
5. P. Alliez, M. Meyer, and M. Desbrun. Interactive geometry remeshing. *Acm Transactions on Graphics*, 21(3):347–354, 2002.
6. Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Levy, and Mathieu Desbrun. Anisotropic polygonal remeshing. *ACM.Transactions on Graphics*, 22:485–493, 2003.
7. Pierre Alliez, Eric Colin de Verdiere, Olivier Devillers, and Martin Isenburg. Isotropic surface remeshing. In M.S. Kim, editor, *SMI '03: Proceedings of Shape Modeling International 2003*, pages 49–58, Los Alamitos, 2003. IEEE Computer Society.
8. N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, pages 415–422, Jul 1998.
9. M. Attene, B. Falcidieno, J. Rossignac, and M. Spagnuolo. Edge-sharpener: recovering sharp features in triangulations of non-adaptively re-meshed surfaces. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing 2003*, pages 62–69. ACM Press, 2003.
10. M. Attene, B. Falcidieno, M. Spagnuolo, and J. Rossignac. Swingwrapper: Retiling triangle meshes for better edgebreaker compression. *Acm Transactions on Graphics*, 22(4):982–996, 2003.
11. M. Attene, B. Falcidieno, M. Spagnuolo, and G. Wyvill. A mapping-independent primitive for the triangulation of parametric surfaces. *Graphical Models*, 65(5):260–273, 2003.
12. J. D. Boissonnat and S. Oudot. Provably good surface sampling and approximation. In *Proc. of Symp. on Geo. Processing*, pages 9–18, 2003.
13. Houman Borouchaki, Frederic Hecht, and J.Frey Pascal. Mesh gradation control. In *Proceedings of 6th International Meshing Roundtable*, pages 131–141. Sandia National Laboratories, 1997.
14. Mario Botsch and Leif Kobbelt. A remeshing approach to multiresolution modeling. In R. Scopigno and D. Zorin, editors, *Proceedings of 2nd Eurographics Symposium on Geometry Processing*, pages 189–196. Eurographics, 2004.
15. L. Paul Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proceedings of the ninth annual symposium on Computational geometry*, pages 274–280. ACM Press, 1993.

16. D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, 2004.
17. T.K. Dey and R. Wenger. Reconstructing curves with sharp corners. *Computational Geometry Theory and Applications*, 19:89–99, 2001.
18. S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer Aided Geometric Design*, 2005. To appear.
19. Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi Tesselations: Applications and Algorithms. *SIAM review*, 41(4):637–676, 1999.
20. M. Eck, T. De Rose, T. Duchamp, H. Hoppe, M.Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, pages 173–182, 1995.
21. H. Edelsbrunner and D. Guoy. Sink-insertion for mesh improvement. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 115–123. ACM Press, 2001.
22. M.S. Floater and K. Hormann. *Surface Parameterization: a Tutorial and Survey*. Springer, 2004.
23. P. J. Frey and H. Borouchaki. Geometric surface mesh optimization. *Computing and Visualization in Science*, 1:113–121, 1998.
24. Pascal J. Frey. About surface remeshing. In *Proceedings of the 9th International Meshing Roundtable*, pages 123–136. Sandia National Laboratories, 2000.
25. M. Garland. Multiresolution modeling: Survey & future opportunities. In *Eurographics '99, State of the Art Report (STAR)*, pages 111–131. Eurographics, 2000.
26. A. Gersho. Asymptotically optimal block quantization. *IEEE Transactions on Information Theory*, IT-25(4):373–380, July 1979.
27. C. Gotsman, X.F. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3d meshes. *Acm Transactions on Graphics*, 22(3):358–363, 2003.
28. C. Gotsman, S. Gumhold, and L. Kobbelt. *Simplification and compression of 3D-meshes*. 2002.
29. X. Gu, S.J. Gortler, and H. Hoppe. Geometry images. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, pages 355–361, 2002.
30. I. Guskov, K. Vidimce, W. Sweldens, and P. Schroeder. Normal meshes. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, pages 95–102, 2000.
31. E. Hartmann. A marching method for the triangulation of surfaces. *the Visual Computer*, 14(3):95–108, 1998.
32. P. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms, 1997.
33. H. Hoppe. Progressive meshes. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, pages 99–108, 1996.
34. Hugues Hoppe, Tony De Rose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, pages 19–26, 1993.
35. K. Hormann and G. Greiner. Quadrilateral remeshing. In *Proceedings of Vision, Modeling, and Visualization 2000*, pages 153–162, 2000.
36. T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, pages 339–346, 2002.
37. T. Kanai, H. Suzuki, and F. Kimura. Metamorphosis of arbitrary triangular meshes. *Ieee Computer Graphics and Applications*, 20:62–75, 2000.

38. A. Khodakovsky and I. Guskov. *Compression of Normal Meshes*. Springer-Verlag, 2003.
39. A. Khodakovsky, N. Litke, and P. Schröder. Globally smooth parameterizations with low distortion. *ACM.Transactions on Graphics*, 22:350–357, 2003.
40. A. Khodakovsky, P. Schroeder, and W. Sweldens. Progressive geometry compression. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, pages 271–278, 2000.
41. L. Kobbelt, S. Bischoff, M. Botsch, K. Kahler, C. Rössl, R. Schneider, and J. Vorsatz. Geometric modeling based on polygonal meshes. In *Euroraphics 2000 Tutorial*, 2000.
42. L. Kobbelt and M. Botsch. Feature sensitive mesh processing. In *SCCG '03: Proceedings of the 19th Spring Conference on Computer Graphics*, pages 17–22. ACM Press, 2003.
43. L. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel. A shrink wrapping approach to remeshing polygonal surfaces. *Computer Graphics Forum*, 18:119–130, 1999.
44. L.P. Kobbelt, M. Botsch, U. Schwanecke, and H.P. Seidel. Feature sensitive surface extraction from volume data. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, pages 57–66, Aug 2001.
45. V. Kraevoy and A. Sheffer. Cross-parameterization and compatible remeshing of 3d models. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, 2004.
46. A. Lee, D. Dobkin, W. Sweldens, and P. Schröder. Multiresolution mesh morphing. In *Siggraph 1999, Computer Graphics Proceedings*, pages 343–350, 1999.
47. A.W.F. Lee, W. Sweldens, P. Schroeder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parameterization of surfaces. *Computer Graphics*, 32:95–104, 1998.
48. J. L. Lin, J. H. Chuang, C. C. Lin, and C. C. Chen. Consistent parametrization by quinary subdivision for remeshing and mesh metamorphosis. In *GRAPHITE '03: Proceedings of the 1th International Conference on Computer Graphics and Interactive Techniques in Austalasia and South East Asia 2003*, pages 151–158. ACM Press, 2003.
49. M.l Lindenbaum, M. Porat, Y. Y. Zeevi, and Y. Eldar. The farthest point strategy for progressive image sampling, 1996.
50. S. Lloyd. Least square quantization in PCM. *IEEE Trans. Inform. Theory*, 28:129–137, 1982.
51. W.E. Lorensen and H.E. Cline. Marching cubes: a high resolution 3d surface reconstruction algorithm. *Computer Graphics*, 21:163–169, 1987.
52. D. Luebke. A developer's survey of polygonal simplification algorithms. *Ieee Computer Graphics and Applications*, 2001.
53. D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Morgan-Kaufmann, San Francisco, 2002.
54. M. Marinov and L. Kobbelt. Direct anisotropic quad-dominant remeshing. In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications*, pages 207–216, 2004.
55. T. Michikawa, T. Kanai, M. Fujita, and H. Chiyokura. Multiresolution interpolation meshes. In *Proceedings of the 9th Pacific Conference on Computer Graphics and Applications 2001*, pages 60–69, Los Alamitos, 2001. IEEE Computer Society.
56. G. L. Miller. A time efficient delaunay refinement algorithm. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 400–409. Society for Industrial and Applied Mathematics, 2004.
57. C. Moenning and N. A. Dodgson. Fast marching farthest point sampling. Technical Report UCAM-CL-TR-562, University of Cambridge, Computer Laboratory, 2003.
58. V. Ostromoukhov. A Simple and Efficient Error-Diffusion Algorithm. In *Proceedings of SIGGRAPH*, pages 567–572, 2001.

59. V. Ostromoukhov, C. Donohue, and P. M. Jodoin. Fast hierarchical importance sampling with blue noise properties new york, ny, usa. *ACM.Transactions on Graphics*, 23, Aug 2004.
60. F. Payan and M. Antonini. 3d mesh wavelet coding using efficient model-based bit allocation. In *Proceedings of the 1st International Symposium on 3D Data Processing Visualization and Transmission 2002*, pages 391–394, 2002.
61. P. P. Pebay and T. J. Baker. A comparison of triangle quality measures. In *Proceedings, 10th International Meshing Roundtable*, pages 327–340, 2001.
62. G. Peyré and L. Cohen. Surface Segmentation Using Geodesic Centroidal Tesselation. In *Proceedings of 2nd International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 995–1002, 2004.
63. Gabriel Peyre and Laurent Cohen. Geodesic remeshing using front propagation. In *Proceedings of 2nd IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision 2003*, pages 33–40, Los Alamitos, 2003. IEEE Computer Society.
64. E. Praun and H. Hoppe. Spherical parametrization and remeshing. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, pages 340–349, 2003.
65. E. Praun, W. Sweldens, and P. Schröder. Consistent mesh parameterizations. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, pages 179–184, 2001.
66. A. Rassineux, P. Villon, J.M. Savignat, and O. Stab. Surface remeshing by local hermite diffuse interpolation. *International Journal for Numerical Methods in Engineering*, 49:31–49, 2000.
67. C. Rocchini, P. Cignoni, F. Ganovelli, C. Montani, P. Pingi, and R. Scopigno. Marching intersections: an efficient resampling algorithm for surface management. In *Proceedings of the International Conference on Shape Modeling and Applications*, pages 296–305, 2001.
68. J. Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, 1995.
69. P. Sander, S. Gortler, J. Snyder, and H. Hoppe. Signal-specialized parametrization. In *EGWR '02: 13th Eurographics Workshop on Rendering 2002*. Eurographics, 2002.
70. P. Sander, Z. Wood, S. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing 2003*, pages 246–255. ACM Press, 2003.
71. J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe. Inter-surface mapping. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, 2004.
72. P. Schröder. Subdivision for modeling and animation, 1998.
73. J. Sethian. *Level Sets Methods and Fast Marching Methods*. Cambridge University Press, 2nd edition, 1999.
74. J. R. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. In *Proceedings of 11th International Meshing Roundtable*, 2002.
75. K. Shimada and J. Liao. Quadrilateral Meshing with Directionality Control through the Packing of Square Cells. In *7th Intl. Meshing Roundtable*, pages 61–76, oct 1998.
76. Oren Sifri, Alla Sheffer, and Craig Gotsman. Geodesic-based surface remeshing. In *Proceedings of 12th International Meshing Roundtable*, pages 189–199. Sandia National Laboratories, 2003.
77. V. Surazhsky and C. Gotsman. Explicit surface remeshing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing 2003*, pages 20–30. ACM Press, 2003.

78. Vitaly Surazhsky, Pierre Alliez, and Craig Gotsman. Isotropic remeshing of surfaces: a local parameterization approach. In *Proceedings of 12th International Meshing Roundtable*, pages 215–224. Sandia National Laboratories, 2003.
79. W. Sweldens and P. Schröder, editors. *Digital Geometry Processing*. SIGGRAPH Conference course notes, 2001.
80. A. Szymczak, J. Rossignac, and D. King. Piecewise regular meshes: Construction and compression. *Graphical.models.*, 2003.
81. G. Taubin. Geometric signal processing on polygonal meshes. In *Euroraphics 2000: State of the Art Report (STAR)*. Eurographics, 2000.
82. J.R. Tristano, S.J. Owen, and S.A. Canann. Advancing Front Surface Mesh Generation in Parametric Space Using a Riemannian Surface Definition. In *Proceedings of the 7th Int. Meshing Roundtable*, 1998.
83. Greg Turk. Re-tiling polygonal surfaces. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, pages 55–64, 1992.
84. Sebastien Valette and Jean Marc Chassery. Approximated centroidal voronoi diagrams for uniform polygonal mesh coarsening. *Computer Graphics Forum*, 2004.
85. A. Vlachos, J. Peters, C. Boyd, and J. L. Mitchell. Curved PN triangles. In *Symposium on Interactive 3D Graphics*, pages 159–166, 2001.
86. J. Vorsatz, C. Rössl, L. Kobbelt, and H.-P. Seidel. Feature sensitive remeshing. *Computer Graphics Forum*, pages 393–401, 2001.
87. J. Vorsatz, C. Rössl, and H.-P. Seidel. Dynamic remeshing and applications. In *SMA '03: Proceedings of the 3th ACM Symposium on Solid Modeling and Applications 2003*, pages 167–175. ACM Press, 2003.
88. D.J. Walton and D.S. Meek. A triangular G^1 patch from boundary curves. *Computer Aided Design*, 28(2):113–123, 1996.
89. Jianhua Wu and Leif Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24(3):277–284, 2005.
90. Soji Yamakawa and Kenji Shimada. Triangular/quadrilateral remeshing of an arbitrary polygonal surface via packing bubbles. In *Proceedings of Geometric Modeling and Processing 2004*, Los Alamitos, 2004. IEEE Computer Society.
91. Dong-Ming Yan, Yang Liu, and Wenping Wang. Quadric surface extraction by variational shape approximation. In *Proceedings of Geometric Modeling and Processing 2006*, 2006.
92. D. Zorin and P. Schröder. Subdivision for modeling and animation. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, 2000.

Multiresolution Analysis

Georges-Pierre Bonneau[1], Gershon Elber[2], Stefanie Hahmann[3], and Basile Sauvage[3]

[1] Université Joseph Fourier, Grenoble, France,
Georges-Pierre.Bonneau@imag.fr
[2] Technion, Israel Institute of Technology, Israel,
gershon@cs.technion.ac.il
[3] Institut National Polytechnique de Grenoble, France,
Stefanie.Hahmann@imag.fr, basile.sauvage@imag.fr

Summary. Multiresolution analysis has received considerable attention in recent years by researchers in the fields of computer graphics, geometric modeling and visualization. They are now considered a powerful tool for efficiently representing functions at multiple levels-of-detail with many inherent advantages, including compression, Level-Of-Details (LOD) display, progressive transmission and LOD editing.

This survey chapter attempts to provide an overview of the recent results on the topic of multiresolution, with special focus on the work of researchers who are participating in the AIM@SHAPE European Networks of Excellence [4].

1 Introduction

Multiresolution analysis has received considerable attention in recent years by researchers in the fields of computer graphics, geometric modeling and visualization [83]. Its attraction is its utility as a powerful tool for efficiently representing functions at multiple levels-of-detail with many inherent advantages, including compression, Level-Of-Details (LOD) display, progressive transmission and LOD editing. A plethora of publications can be easily found on these topics.

This survey chapter attempts to provide an overview of the recent results on the topic of multiresolution, with special focus on the work of researchers who are participating in the AIM@SHAPE European Networks of Excellence.

In Section 2, hierarchical freeform representations are introduced and discussed. Multiresolution methods for freeform spline spaces are discussed in Section 3, including linear and non-linear constraints, and intrinsic multiresolution decomposition. Multiresolution representation of piecewise linear and triangular irregular meshes are considered in Section 4. Finally, we conclude this chapter, in Section 5.

[4] AIM@SHAPE Project, http://www.aimatshape.net

2 Hierarchical Freeform Representations

Forsey and Bartels pioneered the idea of hierarchical B-splines [25]. B-splines can be locally refined using overlays. Based on this model, these researchers created a complex surface such as a dragon's head from a rectangular domain with a hierarchical edition. However, since this model was established over tensor product splines, it is restricted to tensor product mesh and topology.

Localized-hierarchical surface splines [32] extended the hierarchical spline paradigm to surfaces of arbitrary topology. These are defined locally on a hierarchy of meshes using the "reference plus offset" model of Forsey and Bartels for encoding the details. Since they are based on C^1-surface-splines [72], the surface is defined explicitly by low-degree triangular and quadrangular Bézier patches, while requiring the structure to satisfy a particular regularity property through all levels of the hierarchy.

Hierarchical triangular splines [90] are the most recent method for hierarchical modeling of smooth surfaces of arbitrary topology. Based on the previously developed triangular interpolating scheme [45], this method enables LOD construction and surface editing by interpolating the vertices of a hierarchy of locally refined meshes. The initial mesh, referred to as the base mesh, can be any triangular two-manifold mesh. Given a base mesh, a polynomial interpolating surface is computed. It is smooth in the sense that it is overall tangent-plane continuous. LOD is then added by iterative local refinement and editing of the surface. Each local surface refinement replaces a set of coarse surface patches by a set of finer surface ones, while maintaining both the overall tangent-plane continuity and the shape. The user can add detail by editing the refined surface patches. A hierarchical editing tool is also provided thanks to a "reference plus offset" representation. The main features include:

- Any *triangular mesh* can be handled, which means there are *no restrictions* on topology, geometry, genus or boundaries. It only has to be a two-manifold mesh.
- The surface *interpolates* a hierarchy of meshes, thus offering direct control for surface modifications, in different resolutions.
- *Uniform surface model*: The same interpolant is applied to both the initial surface and its different refinement steps, in order to locally recompute the new surface part. Thus, only a *few geometric quantities* are stored for each surface patch in order to completely evaluate the surface.

Further properties that the hierarchical surface inherits from the underlying surface model include:

- Overall tangent-plane continuity.
- Each surface patch is represented as a parametric, polynomial triangular Bézier patch of degree five.
- The surface has local control, i.e., the modification of a mesh vertex modifies only the surrounding surface patches, leaving the surface unchanged outside this region.

Figure 1 presents a hierarchical editing process for a surface composed of triangular patches. The different colors, from white through yellow to red, denote the different levels of detail (see Figure CP-1 in Appendix C).

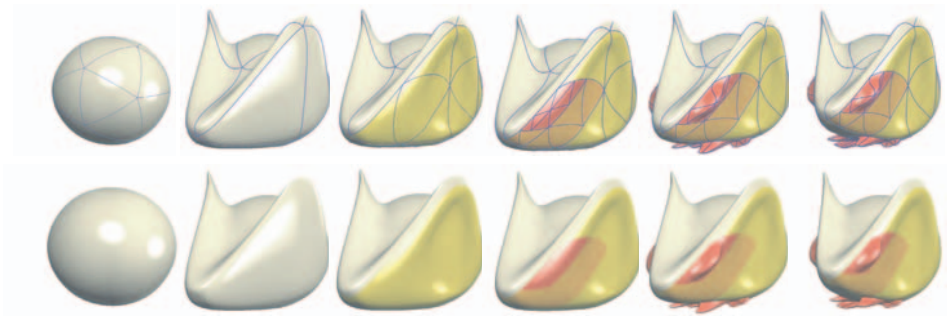


Fig. 1. Hierarchical surface representation and editing: From left to right, a hierarchical editing of an object is shown. Colors correspond to different levels of detail. Starting with an initial surface at the coarsest level, local refinements and local editings are gradually introduced. Finally, the surface is edited at a vertex of the coarsest level, thus naturally deforming all finer details depending hierarchically on this vertex.

3 Multiresolution Methods for Freeform Representations

In the literature, the term multiresolution (MR) is employed in different contexts, including MR-based wavelets, subdivision and hierarchies or multigrids. Multiresolution representations based on wavelets have been developed for parametric curves [14, 22, 62], and can be generalized to tensor-product surfaces [22, 52], to surfaces of arbitrary topological type [61], to spherical data [80], and to volume data [15]. Wavelets provide a rigorous unified framework. Herein, a complex function is decomposed into "coarser" low-resolution parts, together with a collection of detail coefficients and different resolution levels, necessary to recover the original function. Other multiresolution representations exist for data defined for tensor-product surfaces, known as hierarchical B-splines [25], and for volumetric data sets represented using tri-variate functions [75].

In the context of geometric modeling, LOD editing is an attractive MR application because it allows modification of the overall shape of a geometric model at any scale while automatically preserving all fine details. In contrast to classical control-point-based editing methods where complex detail-preserving deformations need to manipulate a lot of control points, MR methods can achieve the same effect by manipulating only a few control points of some low-resolution representation; see [22, 83]. However, there are application areas, including Computer Aided Geometric Design (CAGD) and computer animation, where deformations under constraints

are required. It is obvious that constraints offer additional and finer controls over the deformations applied to curves and surfaces.

The remainder of this section surveys recent results on MR methods in the context of freeform spline geometry, with and without constraints. In Section 3.1 we briefly describe B-wavelets, or wavelets of B-spline functions. In Section 3.2, direct manipulation of freeform curves and surfaces are presented whereas in Sections 3.3 and 3.4 linear and non-linear constraints are discussed, respectively. In Section 3.5, intrinsic MR decomposition of freeform geometry is considered, employing curvature signatures of the shapes. The application of MR to metamorphosis is considered in Section 3.6 and finally, variational design that aims at optimizing and/or fairing the shape is discussed in the context of MR representations, in Section 3.7.

3.1 Wavelet Decomposition of B-spline Functions

Multiresolution manipulation of geometry draws from the ability to project geometry G_i in space $\mathcal{S}_i$ onto another subspace $\mathcal{S}_{i+1} \subset \mathcal{S}_i$. Spline spaces are solely defined by the knot sequences $\tau_{\mathbf{i}}$ (and the orders o_i). In [14, 62], wavelet decomposition of spline spaces, both uniform and non-uniform, were presented. Subspaces are typically selected by removing every second knot, preserving the uniformity of the knot sequence or possibly by weighing the importance of the knots, as is done, for example, in knot removal algorithms [63].

Consider a curve $C(t) \in \mathcal{S}_i$ with a uniform knot sequence $\tau_j = j$. The removal of a single knot, τ_k, creates a sub space $\mathcal{S}_{i+1}$ in which no discontinuity can be present at parameter value τ_k. The B-spline wavelet (also known as B-wavelet) Ψ_k that corresponds to knot τ_k spans the complementary subspace of $\mathcal{S}_i - \mathcal{S}_{i+1}$. While many ways exist to define the function that spans the complementary space, seeking a unique orthogonal representation to Ψ_k, we constrain Ψ_k to be orthogonal to all the B-spline basis functions in $\mathcal{S}_{i+1}$. Since $\Psi_k \in \mathcal{S}_i$, Ψ_k has one additional degree of freedom, which is typically used to normalize Ψ_k, for example with the constraint of $\langle \Psi_k, \Psi_k \rangle = 1$. By using only uniform knot sequences, all B-wavelets are just translations (and scales) of each other. Yet, nothing in the above prevents one from using non-uniform knot sequences with the cost of no possible precomputations. All B-wavelets must now be reevaluated for every new knot sequence. Figures 2 and 3 show several B-wavelet functions for the quadratic and cubic cases, respectively. Both uniform (computed once up to translation and scale!) and non-uniform B-wavelets are shown.

A B-spline curve $C(t)$ is typically decomposed into a low-resolution curve $C_0(t)$ and a sequence of detail curves $D_i(t)$ at different resolutions so that

$$C(t) = C_0(t) + \sum_{i=1}^{n} D_i(t). \tag{1}$$

$C_0(t)$ is the lowest or coarsest resolution and typically contains no interior knots in its subspace. Every additional detail curve $D_i(t)$ contains additional knots all the way to $D_n(t)$. These knots are all shared by the original space of $C(t)$. The vector

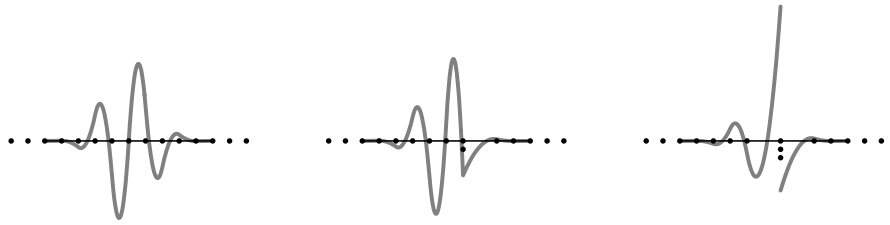


Fig. 2. Quadratic B-wavelets for the uniform case (left) and multiple knots (middle and right). Note a triple knot renders the quadratic B-wavelet discontinuous.

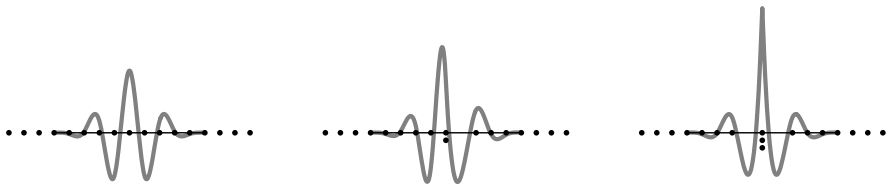


Fig. 3. Cubic B-wavelets for the uniform case (left) and multiple knots (middle and right). Note a triple knot renders the cubic B-wavelet C^0 continuous.

function addition in Equation (1) is always possible since both $C_0(t)$ and $D_i(t)$ remain in the subspace of the original space. In other words, by refinement, one can always elevate $C_0(t)$ and $D_i(t)$ to the original space, where the sum presented in Equation (1) reduces to adding the respective control points of the curves. Figure 4 presents one example of a multiresolution decomposition of a freeform curve.

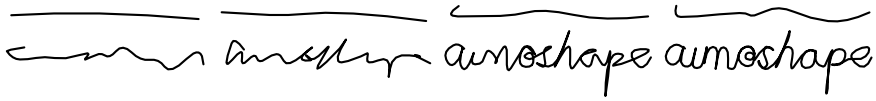


Fig. 4. A decomposition of a B-spline curve into various resolutions. The original quadratic curve is shown at the bottom right and contains over a hundred control points.

By modifying a single control point in $C_0(t)$, the entire shape of $C(t)$ is affected. By modifying the $D_i(t)$ vector functions, one is able to create modifications in different resolutions, from a coarse resolution for $D_1(t)$ all the way to fine details in $D_n(t)$. Figure 5 presents an example of manipulating a freeform curve at different MR levels.

One typical application for MR analysis of spline geometry could be found in the direct manipulation of a freeform shape (see also Section 3.2 below). The local support of the B-spline representation is also the weakest point of the representation. Global modifications are no longer possible in a highly refined B-spline curve. Recognizing this deficiency, in [36, 22], wavelet decomposition was proposed for

Fig. 5. Modification of a B-spline curve at various resolutions. A vertical select-and-drag operation at the top of the 's' character at four different resolutions. The original curve is presented in gray.

uniform B-spline curves toward MR editing control of the shape. When the user wishes to add small details to the shape, a fine subspace is used during the manipulation whereas when global changes are necessary, a coarse resolution is employed.

One clear advantage of using uniform knot sequences is that it allows wavelet decomposition to be performed a-priori, as the decomposition depends solely on the subspaces of the splines and is completely independent of the control points of the shapes. Yet, in reality, many curves and surfaces that are created using contemporary geometric modeling tools possess non-uniform knot sequences. Further, in order to preserve the uniformity of the knots, in a given curve with a uniform knot sequence, every subspace must present half the number of knots of its immediate containing space. That is, $\tau_{\mathbf{i+1}}$ of $\mathcal{S}_{i+1}$ will consist of half the knots in $\tau_{\mathbf{i}}$, with every second knot in $\tau_{\mathbf{i}}$ being removed, preserving the uniformity in the knot spacing.

The work of [36, 22] was extended to non-uniform knot sequences for curves and surfaces, in [52]. Direct manipulation of non-uniform B-spline curves and surfaces is presented in [52] with the aid of a B-wavelet decomposition [62]. Figure 6 shows an example of MR interactive editing, in different resolutions, of a freeform tensor-product B-spline surface in the shape of a chess knight (see also Figure CP-2 in Appendix C).

3.2 Direct Freeform Curve and Surface Manipulation

As already stated, direct manipulation of freeform shapes is a crucial and vital tool in any modern geometric modeling environment. Being able to sculpt the geometry allows novice users to intuitively and interactively manipulate the shape.

Direct manipulation of freeform surfaces is not new and, for example, in [25], a hierarchical representation of B-spline surfaces is presented that allows local and focused manipulation of freeform geometry. Adding degrees of freedom to a freeform surface is usually translated into the insertion of new knots into the shape—an action that affects a whole row or column in the mesh of the surface, and hence is not

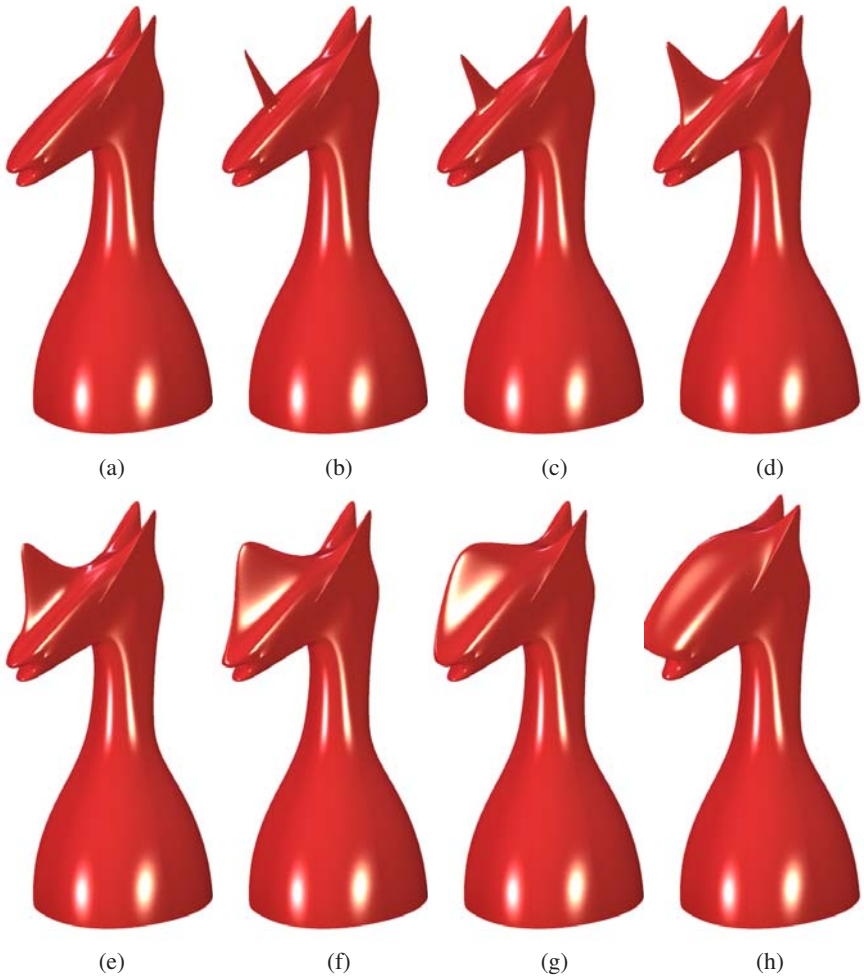


Fig. 6. An example of multiresolution editing of a tensor-product B-spline surface in the shape of a chess knight. A forehead location is selected and dragged upwards in several different resolutions. (a) shows the original surface, while (b) to (h) present the results of the select-and-drag operations in the different resolutions from the highest (b) to to lowest (h).

really local. In contrast, in [25], a hierarchy of partially independent surfaces is imposed that allows the end user to locally affect only a small region in a given surface, by applying a small patch with the new detail at the desired location. This occurs while fixing the outermost rows and columns of the new small patch to preserve the proper continuity. A related scheme for volumetric representations was offered in [75]. Here, a hierarchy of tri-variate functions of different resolution is used to define the sculpted surface. The (iso) surface itself is defined as the zero set of the sum of these tri-variates.

Rather than manipulating control points, Bartels and Beatty showed, in [2], how to select any point on a B-spline curve and change its location, i.e., the curve is constrained to pass through a user-specified location. The new curve shape is computed by minimizing the control points' offset. In [28] Fowler and Bartels controlled the shape of a B-spline curve by enforcing prescribed geometric constraints, such as the position of a curve point, tangent direction and magnitude, or curvature magnitude. An extension to tensor-product B-spline surfaces is given in [26]. This satisfies the user-defined position of surface points, normal direction, tangent-plane rotation (twisting effect), and the first partial derivative's magnitude (tension effect). Borel and Rappoport [9] deformed B-spline surfaces by determining the displacement and radius of influence for each constrained surface point. Hsu et al. [50] proposed points selection for freeform deformations. Curve constraints, i.e., enforcing the surface to contain a given curve or to model a character line, were considered in [12, 38, 71]. Direct shape manipulation techniques are closely related to variational design, where the objective of obtaining fair and graceful shapes is achieved by minimizing some energy; see Section 3.7. In general, a freeform shape has many more degrees of freedom than constraints to satisfy. In order to compute a new shape, the remaining degrees of freedom are stipulated by minimizing some energy functional, such as bending. For example, Welch et al. [89] maintained the imposed constraints while calculating a surface that is as smooth as possible. Celniker and Welch [12] derived interactive sculpting techniques for B-spline surfaces based on energy minimization, keeping some linear geometric surface-constrained features unchanged. Celniker and Gossard [11] enforced linear geometric constraints for shape design of finite elements governed by some surface energy. While energy minimization affects the surface globally, finite element methods allow for local control. Forsey and Bartels [25] later used the technique of hierarchical B-splines in an attempt to overcome this drawback for B-spline surfaces.

In the context of MR, [36, 22] offered direct multiresolution manipulation of uniform B-spline curves and surfaces. While no constraint support was offered in these publications, they demonstrated, for the first time, the hidden power in MR editing and direct manipulation of freeform curves and surfaces. Exact B-spline wavelet (B-wavelets) decomposition was used to perform the MR analysis. In [52] and using the results of [62], the approach of using precise B-wavelet decomposition in direct curve and surface manipulation was extended to non-uniform B-spline space. Also demonstrated in [36, 22] were abilities to add details of different shapes to an existing curve—another modeling feature of high interest.

The work of [22, 36, 52] computed the exact orthogonal projections of the freeform geometry into lower dimensional spaces, employing the B-wavelet decomposition of uniform and non-uniform B-spline representations. While fairly simple to compute in the case of uniform knot sequences, this decomposition, in the non-uniform case, must be recomputed for each newly defined space and is computationally intensive. Fortunately, one can recognize that the explicit orthogonal decomposition is not really necessary [35], alleviating these computational difficulties. In [20], an MR curve editor that is based on a non-orthogonal decomposition was also presented. The major deficiency of this non-orthogonal decomposition lies in its ambiguous representation, by offering the user, for example, the option of conducting many fine high-resolution operations, which can, in fact, be represented as a few low-resolution operations. The (approximated) projection of a curve to a low-dimensional space is simple, and for direct manipulation purposes, it might be sufficient.

3.3 Linear Constraints

In [27, 29, 89], surface editing schemes that satisfy zero-dimensional constraints such as positions, tangents and normals, were presented. The constraints, being linear, are efficiently solved, allowing for the interactive manipulation of the freeform geometry. [89] also considered *transfinite constraints* where the constraints might have a non zero dimensionality. While some cases might be of a finite dimension, such as the containment of a polynomial curve in a polynomial surface when posed as a composition, other cases might necessitate an approximation. The composition of the polynomial curve $\gamma(t) = (u(t), v(t))$ and polynomial surface $S(u, v)$ yields $S(t) = S(u(t), v(t))$, a curve over S, which is a polynomial as well. The degree of $S(t)$ equals the product of the sum of the degrees of S and the degree of $\gamma(t)$. Hence, m linear constraints, where m is the order of $S(t)$, fully prescribe a polynomial curve over a polynomial surface. This result also extends to rationals.

Other finite linear constraints are treated with ease. A positional constraint, following [27, 29, 89, 28, 19], could be prescribed as, for curves,

$$P = C(t_p) = \sum_j Q_j B_{i,n}(t_p),$$

and for surfaces,

$$P = S(u_p, v_p) = \sum_{jk} Q_{jk} B_{j,n}(u_p) B_{k,m}(v_p).$$

Similarly, a normal constraint could be written as

$$0 = \langle N, C'(t_n) \rangle = \sum_j \langle N, Q_j \rangle \, B'_{i,n}(t_n),$$

for curves and the normal or tangent-plane constraint yields

$$0 = \left\langle N, \frac{\partial S(u_n, v_n)}{\partial u} \right\rangle = \sum_{jk} \langle N, Q_{jk} \rangle \, B'_{j,n}(u_n) B_{k,m}(v_n),$$

$$0 = \left\langle N, \frac{\partial S(u_n, v_n)}{\partial v} \right\rangle = \sum_{jk} \langle N, Q_{jk} \rangle \, B_{j,n}(u_n) B'_{k,m}(v_n).$$

For surfaces, the normal constraint is related to tangency constraints. The two partials of S, which span the tangent-plane if S is regular, also uniquely determine the orthogonal, normal space, of S. That is, $\frac{\partial S(u_n,v_n)}{\partial u} \times \frac{\partial S(u_n,v_n)}{\partial v} \neq 0$. Hence, the normal constraints as listed above could be similarly written as $C'(t_n) = T$ with one important difference. By coercing $C'(t_n) = T$, the length of the tangent field is exactly fixed, achieving C^1 continuity at this point. By posing the constraint as $\langle N, C'(t_n) \rangle$, G^1 continuity is gained, necessitating fewer degrees of freedom.

3.4 Bi-Linear and Non-Linear Constraints

The advantage of having linear constraints is obvious. The solution is much simplified and is typically more robust to compute. Several types of non-linear constraints could also be expanded and solved with little effort. The preservation of the area enclosed by a closed planar curve is important, for example, when one designs a cross-section of an airplane's fuselage that is assumed to hold a fixed volume. This area (and volume in R^3) constraint could be represented as a bi- (tri-) linear constraint [19, 47].

Consider again $C(t) = (x(t), y(t))$ being a regular, closed planar parametric curve. Employing Green's theorem, the (signed) area, $\mathcal{A}$, enclosed by $C(t)$, equals (see, for example [16, 31]),

$$\mathcal{A} = \frac{1}{2} \oint -x'(t)y(t) + x(t)y'(t)dt = \frac{1}{2} \oint |C(t) \times C'(t)|dt. \tag{2}$$

Equation (2) is clearly quadratic in t. Yet, in [19, 47], it is recognized that Equation (2) could be decomposed into a bi-linear form in t as $\mathcal{A} = x(t)My(t)$. This decomposition eases the incorporation of an area constraint into an MR framework. In [19], $x(t)$ and $y(t)$ are interleavingly fixed while solving the remainder of the linear constraint in $y(t)$ and $x(t)$, respectively. In Figure 7, a nose in an outline of a face is pulled without constraints, and then with positional, and positional and area constraints. This comparison shows how positional constraints could anchor the shape at certain points, and how the fixed area constraints have a global effect on the shape even for local changes. A local nose expansion automatically reacts by shrinking the entire shape, in order to keep the area constant.

In [47] another area preserving MR editing method for uniform B-splines was introduced. Herein, a wavelet-based MR analysis similar to [22] has been used. It enables in particular the derivation of a multiresolution representation of the area functional for the curve at any level of resolution. Let us briefly introduce this MR framework here, since it is different from the non-uniform multiresolution setting of Section 3.1. In this setting we are given some functional space E and some nested

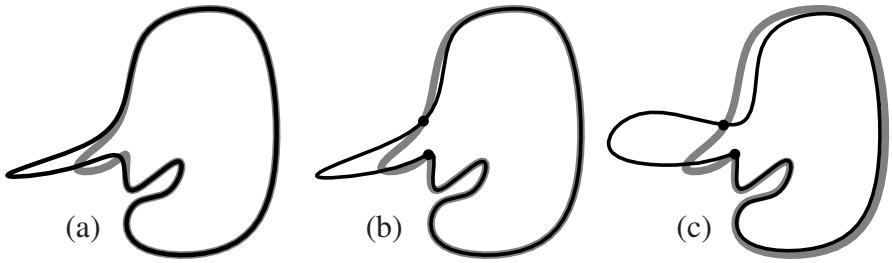


Fig. 7. Multiresolution editing with linear and bi-linear area constraints, before (wide gray) and after (thin black) the editing operation. In (a), the nose is interactively pulled to the left with no additional constraints. In (b), two positional constraints are placed at the base of the nose, while in (c), the area is fixed.

linear *approximation spaces* $S^j \subset E$ with $S^0 \subset S^1 \subset \cdots \subset S^n$. Since we are dealing with closed curves, these spaces have finite dimension. Let S^j be spanned by a set of basis functions $(\varphi_i^j)_i$, called *scaling functions*. A space W^j being the complement of S^j in S^{j+1} is called the *detail space*. Its basis functions $(\psi_i^j)_i$ are such that together with φ^j they form a basis of S^{j+1}. The functions ψ_i^j are called *wavelets*. The space S^n can, therefore, be decomposed as follows:

$$S^n = S^{n-1} \oplus W^{n-1} = S^{n-2} \oplus \bigoplus_{j=n-2}^{n-1} W^j = \cdots = S^0 \oplus \bigoplus_{j=0}^{n-1} W^j. \tag{3}$$

Condition (3) implies that the scaling functions are refinable; that is, for all $j \in \{0, \ldots, n\}$ there must exist some matrices P^j and Q^j such that the following refinement equations hold:

$$\begin{aligned} \varphi^{j-1} &= (P^j)^T \varphi^j, \\ \psi^{j-1} &= (Q^j)^T \varphi^j. \end{aligned} \tag{4}$$

On the other hand, the "fine" scaling functions φ^j can be constructed from the coarser scaling functions and wavelets with the aid of some matrices A^j and B^j:

$$\varphi^j = (A^j)^T \varphi^{j-1} + (B^j)^T \psi^{j-1}. \tag{5}$$

Note that $[P^j \mid Q^j]$ and $\begin{bmatrix} A^j \\ B^j \end{bmatrix}$ are both square matrices, and that

$$\begin{bmatrix} P^j \; Q^j \end{bmatrix} \begin{bmatrix} A^j \\ \\ B^j \end{bmatrix} = I. \tag{6}$$

The choice of the scaling functions determines the structure of the matrices P^j, Q^j, A^j, and B^j . Sparse matrices are desirable for most of the applications.

Based on this framework let an *MR curve* $C(t)$, $t \in I$ be defined as an element of some functional space S^n. Let $D2^n$ be the dimension of the space S^n, with $D = dim(S^0)$. $C(t)$ can be written as:

$$C(t) = \sum_{i=0}^{D2^n-1} x_i^n \varphi_i^n = (\mathbf{x}^n)^T(\varphi^n), \tag{7}$$

where $\mathbf{x}^n$ is a column vector of $D2^n$ scaling coefficients $x_0^n, \ldots, x_{D2^n-1}^n \in \mathbb{R}^2$. In a more general setting, the vector of scaling coefficients $\mathbf{x}^n$ can be thought of as a discrete signal with $D2^n$ samples.

The relations (4) and (5) now allow us to create a low-resolution signal $\mathbf{x}^{n-1}$ with less samples by using the low pass filter matrix A^n:

$$\mathbf{x}^{n-1} = A^n \mathbf{x}^n,$$

where A^n is of size $(D2^{n-1} \times D2^n)$. The details, which have been lost in this filtering process, can be captured as another signal $\mathbf{d}^{n-1}$ using the high-pass filter matrix B^n:

$$\mathbf{d}^{n-1} = B^n \mathbf{x}^n,$$

where B^n is also a $(D2^{n-1} \times D2^n)$ matrix. The process of splitting a signal $\mathbf{x}^n$ into a coarser signal $\mathbf{x}^{n-1}$ with details $\mathbf{d}^{n-1}$ can now be repeated recursively with the new signal $\mathbf{x}^{n-1}$. Finally, the original signal will be decomposed into a low-resolution signal $\mathbf{x}^0$ and details $\mathbf{d}^0, \ldots, \mathbf{d}^{n-1}$. This recursive process is known as a **filter bank** [64]; see Figure 8. At any intermediate level of resolution, L, the original signal is decomposed into a coarser signal $\mathbf{x}^L$ and details $\mathbf{d}^L, \ldots, \mathbf{d}^{n-1}$. Note that the vector $\mathbf{d}^j$ is of size $D2^j$, $j = 0, \ldots, n-1$.

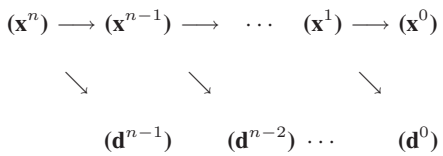


Fig. 8. A filter bank

Using the matrices P^j and Q^j the original signal $\mathbf{x}^n$ can be reconstructed recursively with

$$\mathbf{x}^j = P^j \mathbf{x}^{j-1} + Q^j \mathbf{d}^{j-1}, \qquad \text{for } j = 1, \ldots, n.$$

Since the filter bank also applies to the scaling coefficients of an MR curve (7), such a curve can be represented at any *level of resolution*, $L \in \{0, \ldots, n\}$, by some coarse coefficients $\mathbf{x}^L$ that form approximations of the initial coefficients and by the detail coefficients $\mathbf{d}^L, \ldots, \mathbf{d}^{n-1}$ as follows:

$$C(t) = (\mathbf{x}^L)^T(\varphi^L) + (\mathbf{d}^L)^T(\psi^L) + \cdots + (\mathbf{d}^{n-1})^T(\psi^{n-1}), \quad L = 0, \ldots, n. \tag{8}$$

In this wavelet-based MR framework, the area functional (2) of an MR curve (8) can now be evaluated at any level of resolution L. This leads to the bi-linear equation

$$2\mathcal{A} = (\mathbf{X}^L) \left[M^L \right] (\mathbf{Y}^L)^T, \quad \forall L \in \{0, \ldots, n\}, \tag{9}$$

where X^L and Y^L are the line vectors of the x- and y-coordinates, respectively, of all $D2^n$ coefficients (coarse and wavelet coefficients) of the MR representation of the curve, i.e.,

$$\begin{pmatrix} \mathbf{X}^L \\ \mathbf{Y}^L \end{pmatrix} = (\mathbf{x}^L, \mathbf{d}^L, \mathbf{d}^{L+1}, \ldots, \mathbf{d}^{n-1}),$$

and

$$M^L = \begin{bmatrix} I(\varphi^L, \varphi^L) & I(\varphi^L, \psi^l)_{l=L}^{n-1} \\ I(\psi^k, \varphi^L)_{k=L}^{n-1} & I(\psi^k, \psi^l)_{k,l=L}^{n-1} \end{bmatrix} = \begin{bmatrix} A & B \\ -B^T & C \end{bmatrix}. \tag{10}$$

The MR area constraint is then linearized during the optimization process in order to locally or globally deform a curve at any level of resolution while preserving the enclosed area; see Figure 9.

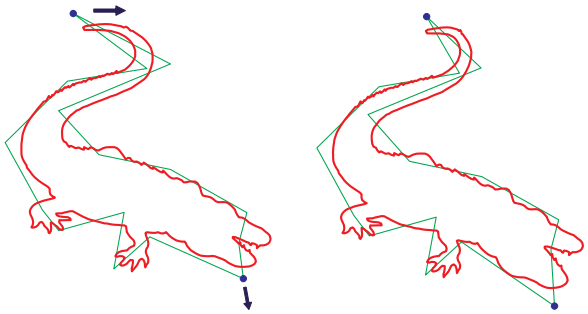


Fig. 9. Multiresolution editing of a coarse level of resolution with preservation of the enclosed area and a positional constraint.

Some works that preserve **volume** while manipulating the shape are also available. In [74], a cuboid volume was manipulated while preserving its volume, handling the problem as a non-linear optimization problem. In [19], it was also shown that the volume constraint, which is cubic in general, could also be posed as a trilinear constraint. Volume-preserving editing of MR surfaces represented by wavelets for uniform tensor-product B-splines following the MR setting described above has been developed in [77]. An example of volume-preserving MR editing is shown in Figure 10.

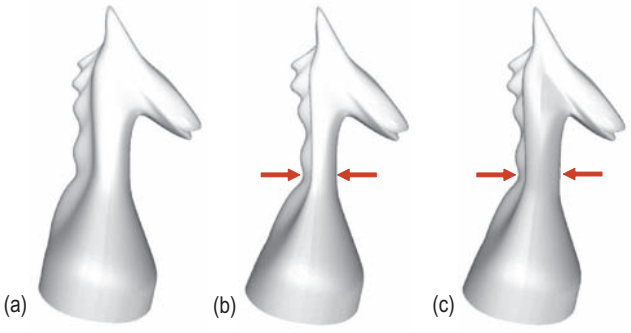


Fig. 10. An example of multiresolution editing with volume-preservation. (a) shows the original tensor-product B-spline surface. In (b), a deformation is applied at a coarse level of resolution without volume-preservation. In (c), the same deformation is applied but the volume of the original surface is preserved.

Other non-linear constraints that are commonly considered are second-order differential constraints such as convexity [51], and first- and second-order fairing constraints, typically in the form of strain and stress surface shape optimization functionals [89]. Another non-linear constraint of high interest is the preservation of the arc-length of the curve. In [78], the arc-length of a curve was presented as a non-linear constraint that is preserved during the curve's manipulation. Herein the constraint is integrated into an MR editing system that allows intuitive control of the deformation's extent and aspect. In [79] this length-constrained MR deformation has been integrated in a wrinkling tool for soft tissue modeling.

The exploitation of first and second differential order constraints, in real-time, is also highly intensive computationally. In [73], an interactive surface editing system that supports real-time surface manipulation with convexity/developability constraints was reported. It exploits a careful symbolic pre-computation of the curvature fields.

3.5 Intrinsic Multiresolution Decomposition of Freeform Shapes

The fundamental problem of MR decomposition is that the decomposition is typically not intrinsic. A curve or a surface could be arbitrary closely approximated using different knot sequences and even different control points. Likewise, two similarly looking objects could be represented using completely different polygonal meshes, as it is evident by the vast remeshing results that have been published in recent years.

It is, therefore, plausible to try and execute this MR decomposition in a way that is independent of the representation underneath, taking into account only the intrinsic geometry, and ignoring, for example, the parameterization.

One such possibility with regard to a planar C^2 freeform curve is to represent the shape by its curvature signature, $\kappa(t)$:

$$\kappa(t) = \frac{x'(t)y''(t) - x''(t)y'(t)}{\left(x'^2(t) + y'^2(t)\right)^{3/2}}, \tag{11}$$

assuming $C(t)$ is regular or $||C'(t)|| \neq 0$.

Assume $C(s)$ is an arc-length parameterized curve. Then, $\kappa(s) = x'(s)y''(s) - x''(s)y'(s)$. Further,

$$C'(s) = T(s), \quad C''(s) = T'(s) = \kappa(s)N(s),$$

where $T(s)$ and $N(s)$ are the unit tangent and normal fields of $C(s)$. $T(s) = (x'(s), y'(s))$ is a unit size vector and hence is always on the unit circle. Let θ be the angle between $T(s)$ and the x-axis,

$$\theta(s) = tan^{-1}\left(\frac{y'(s)}{x'(s)}\right),$$

and consider $\theta'(s)$,

$$\begin{aligned}
\theta'(s) &= \left(tan^{-1}\left(\frac{y'(s)}{x'(s)}\right)\right)' \\
&= \frac{1}{1+\left(\frac{y'(s)}{x'(s)}\right)^2}\left(\frac{y'(s)}{x'(s)}\right)' \\
&= \frac{x'^2(s)}{x'^2(s)+y'^2(s)}\frac{x'(s)y''(s)-x''(s)y'(s)}{x'^2(s)} \\
&= \frac{x'(s)y''(s)-x''(s)y'(s)}{x'^2(s)+y'^2(s)} \\
&= x'(s)y''(s)-x''(s)y'(s).
\end{aligned}$$

In other words, $\theta'(s) = \kappa(s)$ or a curve $C(s)$ could be reconstructed from $\kappa(s)$ by (see also [10])

$$C(s) = \int_0^s T(\bar{s})d\bar{s} = \int_0^s Circ\left(\int_0^{\bar{s}} \kappa(\tilde{s})d\tilde{s}\right)d\bar{s},$$

up to a rigid-motion, where $Circ(\cdot)$ is an arc-length parameterized unit circle. We are now able to switch back and forth between a regular parametric form of a planar curve $C(t)$ and its curvature signature $\kappa(t)$, up to rigid-motion.

While polynomial parametric curves are not arc-length, in general, one can approximate a given polynomial parametric curve as an arc-length polynomial parametric curve to an arbitrary precision; see, for example [18]. Figure 11 shows one example of a curvature signature computed to an approximation of an arc-length polynomial curve.

Multiresolution decomposition could now be applied to $\kappa(s)$ instead of $C(s)$. Alternatively, details could be added to low-resolution shapes by modulating the base $\kappa(s)$ signature and reconstructing the curve. Practical attempts of this procedure turned out to be quite slow and a large number of $\kappa(s)$ samples were necessary to achieve a reasonable reconstruction.

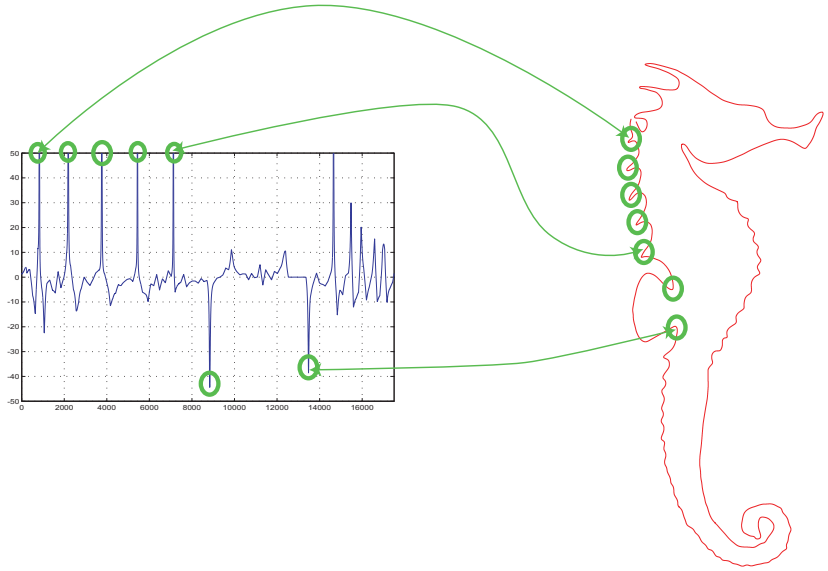


Fig. 11. A (portion of a) curvature signature (left) of an approximately arc-length parameterized curve (right).

An actual intrinsic MR decomposition of a freeform curve using its curvature signature is presented in Figure 12. A multiresolution analysis of $\kappa(t)$ was performed using Haar wavelets. New curvature functions κ_{small}, κ_{mean}, κ_{large} were computed by partially reconstructing the wavelet decomposition using only detail coefficients greater than a given (small, mean, large) threshold. The curves were then obtained by the integration of the new curvature functions.

Further research in this direction of intrinsic MR decomposition of freeform geometry is in order. One such research direction should seek an ability to preserve the continuity of closed, periodic curves throughout the intrinsic MR process.

3.6 Multiresolution Morphing

Morphing (or metamorphosis) is known as the smooth and progressive transformation of one shape into another. The shape can be an image or a planar curve in 2D space, or it can be a surface or a volume in 3D space. The problem is to create an aesthetic and intuitive transition between two shapes. The intermediate shapes should preserve the appearance and the properties of the input shapes. A trivial linear interpolation is often not appropriate, since the intermediate shapes tend to vary a lot in their volume or they lose the proportions of their shape features. Another negative effect is that the geometric details can disappear and re-appear later during the transition. Good results are generally achieved not by interpolating the positions of the boundary representation but by interpolating elements of alternative representations. In the case of 2D polygonal shapes, Sederberg et al. [81] represented polygons by a

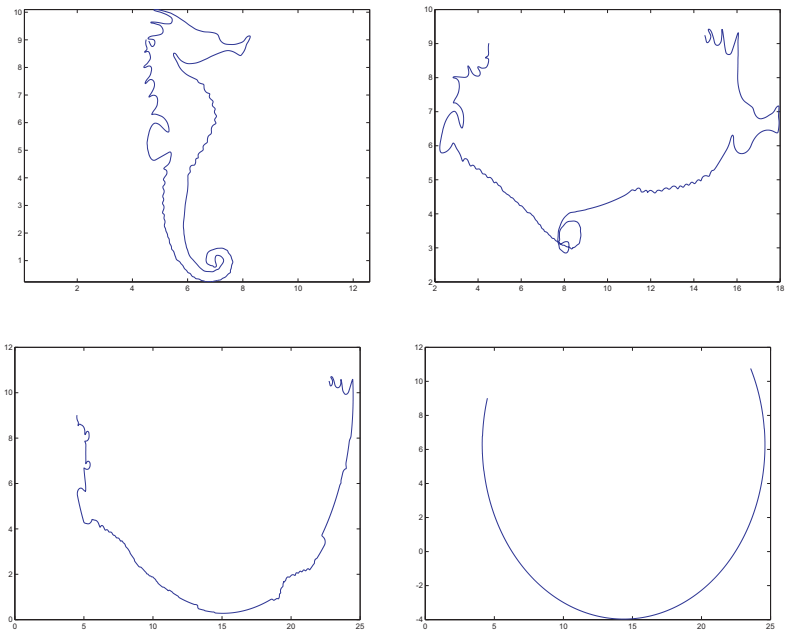


Fig. 12. Multiresolution analysis and partial synthesis of a seahorse curve based on an intrinsic curvature signature.

set of lengths and angles. Shapira and Rappoport [82] used a star-skeleton representation. Goldstein and Gotsman [30] used an MR representation based on curve evolution. Alexa et al. [1] morphed compatible triangulations by locally least-distorting maps. There is also the MR mesh morphing technique by Lee et al. [59]. The key to a successful method thus seems to be the use of a representation based on intrinsic properties of the object geometry such that interpolation of its elements achieves automatically pleasing morphs.

The morphing method we report on in this section is based on a new intrinsic MR representation. It decomposes the source and the target shapes into a coarse approximation and a set of detail coefficients. It computes separately the sequence of coarse intermediate shapes and details, and then reconstructs all intermediate shapes. The choice of the MR representation is crucial for the quality of the resulting shapes. For example, a wavelet-based MR analysis would not preserve the orientation of the details during deformation. In fact, the details here are encoded in a global coordinate system. An MR representation that encodes the details using local frames similar to [22] solves this problem, but the coarse representation of the curve in [22] is still not intrinsic.

In [46] a curvature-based MR representation for 2D polygonal curves was introduced. This MR representation is based on an intrinsic parameterization of both the coarse shape and the detail coefficients. All coefficients will be represented intrinsi-

cally by lengths and angles. Similar to local frames, the MR representation preserves the orientation of the details during deformation.

Let $P_i = (x_i, y_i)$, $i = 0, \ldots, N-1$ denote the vertices of a polygon to be MR-analyzed. The initial polygon needs to be transformed from (x, y)-coordinates into so-called (θ, l)-coordinates, where $\theta_i = \angle(\overline{P_{i-1}P_i}, \overline{P_iP_{i+1}})$ is the counterclockwise angle of the two consecutive polygon segments at P_i and $l_i = \|\overline{P_iP_{i+1}}\|$, $i = 0, \ldots, N-2$. The (x, y)-coordinates of the control points P_i can be recovered directly using, for example, P_0 as an anchor point and $\overline{P_0P_1}$ as an anchor line (determining the translation and rotation, rigid-motion, degrees of freedom). Note the (θ, l)-coordinates are rigid-motion invariant.

Following the filter bank algorithm presented in Figure 8, an *angle-length MR representation* can be computed as follows. From a polygon with 2^{n+1} segments, one analysis step creates a polygon with 2^n segments and 2^n detail coefficients, which are represented by two-dimensional vectors of the form:

$$\begin{array}{ccc} (\theta^{n+1}, \mathbf{l}^{n+1}) & \rightarrow & (\theta^n, \mathbf{l}^n) \\ & \searrow & \\ & & (\alpha^n, \beta^n), \end{array}$$

where $\theta^n = (\theta_0^n, \ldots, \theta_{2^n-1}^n)$ and is analogous for $\mathbf{l}^n, \alpha^n, \beta^n$.

The MR analysis is the recursive procedure of splitting the vector of coefficients of a polygon (θ^{n+1}, l^{n+1}) into a vector of coarse coefficients of a lower resolution polygon (θ^n, l^n) and a vector of detail coefficients; see Figure 13. Let the upper index n denote the level of resolution. Both coarse shape and detail coefficients of level n must be computed directly from (θ^{n+1}, l^{n+1}) and vice-versa. The coarse shape and detail coefficients are computed using the basic cosine trigonometric rule for triangles (also known as the Al-Kashi formula for triangles); see [46] for more details.

Given two polygons with the same number of corresponding control points P_S and P_T, called source and target polygons, the MR morphing algorithm constructs in-between polygons P_t that gradually change P_S into P_T for $t \in [0, 1]$, where $P_S = P_0$ and $P_T = P_1$. For both polygons one disposes of two sets of coarse coefficients and of two sets of detail coefficients. In principle, the in-between morphs are now generated by interpolating the coefficients of this intrinsic MR representation. However, the coarse coefficients are interpolated using the locally least distorting interpolation [1] in order to get "as-rigid-as-possible" intermediate morphs. Figure 14 shows two examples of curves with a complex shape and with a lot of fine details that are difficult to interpolate with standard morphing techniques.

3.7 Variational Multiresolution Methods for Freeform Surface

The variational modeling paradigm is used in order to find the "best" curve or surface amongst all solutions that meet some constraints. The constraints may result from the particular modeling technique used, for example, sample point approximation, or direct curve manipulation (see Section 3.2), or they may be one of the constraints

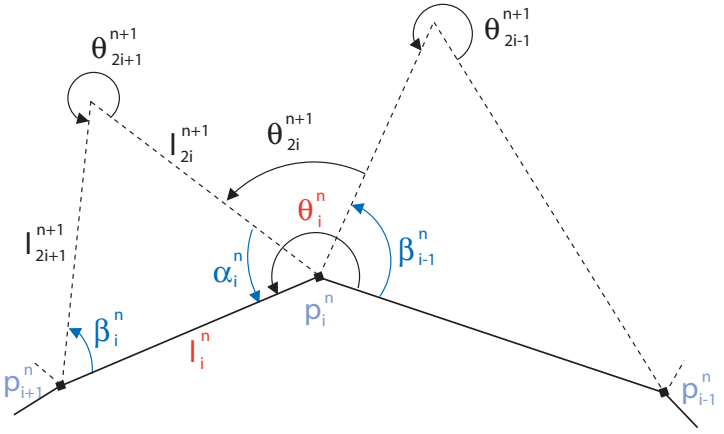


Fig. 13. The analysis. The dotted polygon belongs to resolution level $n+1$, the fat polygon belongs to level n.

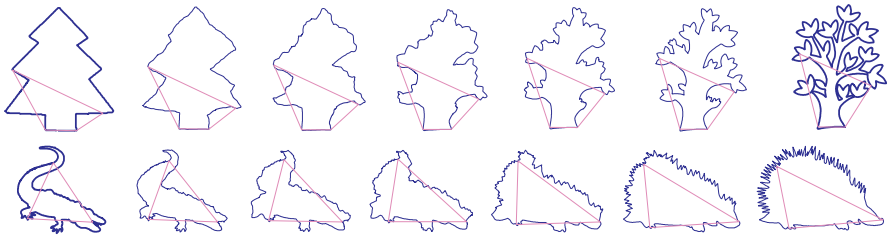


Fig. 14. Two examples of intrinsic multiresolution morphing. The tree curves have 2048 control points. The animal curves have 1536 control points. The algorithm applies a multiresolution analysis to the source (left) and target (right) polygons. Then, the coarse shape and detail coefficients are independently interpolated, and the intermediate curves are reconstructed. The resulting curves are shown alongside the coarse polygons.

described in Section 3.3. In the context of smooth curve and surface design, the notion of "best" is formulated by minimizing some energy functional.

Variational Shape Design

Although it is difficult to define exactly, in mathematical terms, what *fairness* of a curve or surface is, it is commonly accepted that smooth and graceful shapes are obtained by minimizing the amount of energy stored in the surface. The energy functionals originating from elasticity theory, such as the bending energy for curves $\int \kappa^2(t)dt$ or the thin-plate energy for surfaces $\int \kappa_1^2+\kappa_2^2 dA$, are in general non-linear. These and other higher order, non-linear, energy functionals were used in [67, 37].

In order to accelerate computations, linearized versions of these energy functionals are generally used; see, for example, [11, 12, 89, 33]

$$\mathcal{E} = \int_{\sigma} (\alpha \text{ stretch} + \beta \text{ bend}) d\sigma,$$

where α and β are weights on the stretching and bending energies. These produce a surface that tends to minimize its area to avoid folding and to distribute curvature over large regions in order to result in fair shapes. The stretch-and-bend functionals are typically approximated via the following quadratic terms: $\alpha_{11}X_u^2 + \alpha_{12}X_uX_v + \alpha_{22}X_v^2$ and $\beta_{11}X_{uu}^2 + \beta_{12}X_{uv}^2 + \beta_{22}X_{vv}^2$, respectively, only to be linearized in the optimization process.

Historically, the use of such energy functionals goes back to early spline and CAGD literature [65, 76] and has today led to a research area called variational design (of smooth curves and surfaces) [21, 43, 42, 7, 44].

Variational Multiresolution Modeling

Gortler and Cohen [33] showed how the variational constraint, which generalizes least-squares, can be solved through an MR formulation of a planar curve. A wavelet-based MR curve satisfying some linear constraints and minimizing a linearized bending energy functional may be found by solving the following linear system [89]:

$$\begin{bmatrix} \bar{H} & \bar{A}^T \\ \bar{A} & 0 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}} \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix},$$

where $\bar{A}$ is the constraint matrix, $\bar{H}$ is the Hessian matrix of the basis functions, and λ is the vector of Lagrange multipliers. The bars signify that the variables are wavelet coefficients. The wavelets allow acceleration of the iterative conjugate gradient-solving of the variational problem.

Variational subdivision is another modeling technique where constraints are combined with classical subdivision. Instead of applying explicit rules for the new vertices, Kobbelt's variational subdivision scheme [56] computes the new vertices such that a fairness functional is minimized. At each step a linear system has to be solved. The resulting curves have minimal total curvature. Furthermore, [58] showed how wavelets can be constructed by using the Lifting Scheme [85], which is appropriate for variational subdivision curves. Weimer and Warren [86, 87, 88] developed variational subdivision schemes that satisfy partial differential equations for, for instance, fluid or thin-plate equations.

4 Multiresolution Analysis for Irregular Mesh-based Representations

A lot of work has been done in the past ten years on MR analysis of models based on a decomposition of the shape into triangles. This section will focus on two types, scalar data defined on triangulations and mesh-based freeform surfaces.

Only certain types of data sets can be analyzed by wavelet MR analysis. One principal restriction is that the grid on which the data is defined has to be obtained

by successive subdivisions of a coarse grid. These subdivisions define a sequence of grids such that the cells of one grid are subdivided by the cells of the next grid. Such a sequence of grids is deemed "connected by subdivision". This restriction is due to the fact that wavelet analysis needs a *nested* sequence of approximating spaces; see Section 3.1. In the case of quadrilateral or triangular grids, the regular four-way split, as illustrated in Figure 15, is generally used to create a grid with subdivision connectivity, since the grids will not tend to degenerate after several subdivisions.

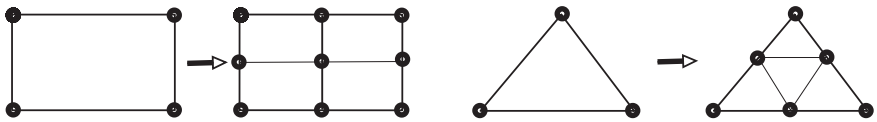


Fig. 15. Regular four-way split for a quad mesh and a triangle mesh.

However, data defined on triangulations as well as freeform surface meshes are generally of more complex structure due to acquisition techniques such as observation and laser range scanning. Thus, classical wavelet theory cannot be adapted directly to these so-called *irregular meshes*, since it is impossible to associate a sequence of grids with subdivision connectivity to this data. In the case of quadrilateral grids with subdivision connectivity, the one-dimensional wavelet-based MR analysis applies directly by tensor-product [84, 66]. If the data is defined on a regular triangular grid, classical wavelet theory can, and has also been adapted as well [68, 80]. Nevertheless, the case of freeform surfaces is more complicated since surfaces of arbitrary topology cannot be parameterized on regular quad or triangle meshes.

The aim of the present section is to focus on MR analysis for irregular mesh-based representations. Section 4.1 addresses the simplification of numerical data attached to an irregular mesh. Section 4.2 covers the simplification of surface meshes.

4.1 Irregular Triangulations

Wavelet methods assume that the mesh on which the data is defined can be reached by recursive subdivision of a basic mesh. Thus, every wavelet-based scheme is associated with hierarchies that have a tree structure (where every parent node is subdivided into a set of child nodes). Wavelet volume visualization [39] is related to Octree structures. Wavelet radiosity [34] and wavelets over triangulated domains [61, 68, 80] are based on Quadtree structures.

On the other hand, irregular triangular meshes cannot be reached by subdivision rules, therefore hierarchical structures that have been developed to handle them are more complicated than trees. These include, for example, hierarchical Delaunay triangulations [55, 23], or progressive meshes [48, 49]. These data structures are appropriate for LOD models, see [24], but not for MR analysis as described in the present chapter.

The approach introduced in [3] fills the gap between wavelet methods (on subdivision hierarchies) and hierarchical structures on irregular triangular meshes for a special type of data set, i.e., for piecewise constant data defined on irregular planar or spherical triangulations.

The basic idea is to relax the restrictions imposed by classical wavelet-based MR analysis, while preserving good properties such as constant memory requirements, linear computational time, and the ability to accurately approximate data with few detail coefficients. The relaxed restrictions are related to the approximation spaces associated with the MR analysis. These spaces are the functional spaces that correspond to each level of resolution where the original function is successively approximated during the analysis. These spaces have to be nested, i.e., the space corresponding to one resolution has to be a subspace of all spaces corresponding to finer resolutions. This property of nested spaces is the reason why the grids of data have be to connected by subdivision.

The generalized framework of MR analysis for irregular triangulation introduced in [3] does not require the nested property. The latter is replaced by a weaker condition that is related to the growth of the approximation spaces. If the data is defined on irregular triangulations, it becomes possible to associate them with a sequence of approximation spaces corresponding to coarser irregular triangular grids. There exist numerous algorithms for reducing the number of triangles in a mesh—independently of the data that is defined on this mesh. Delaunay-removal can be applied to planar or convex triangular meshes, and edge-collapse to general triangular meshes. If the mesh comes from the recursive four-way split of some triangles in a base mesh, then the obvious way to simplify it, is to replace each group of four sub-triangles by their parent triangle. The common setting of these decimation algorithms is that a set of n triangles is replaced by a set of m triangles covering the same domain, with $m < n$, as is shown in Figure 16.

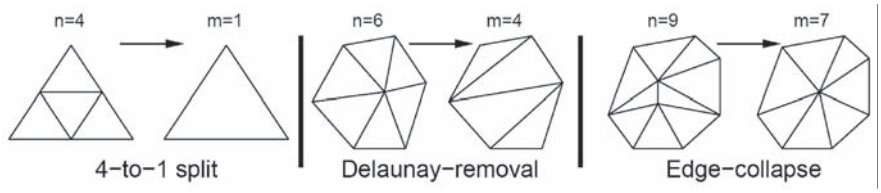


Fig. 16. Local triangle decimation.

A sequence of triangulations obtained by successive decimation is generally not nested because it is not connected by subdivision. However, the generalized framework allows the development of MR analysis algorithms that generate a coarse approximation of the original data and a set of detail coefficients. Therefore, the same types of applications that are mentioned in Section 3.1 are possible by selecting a subset of the detail coefficients and by synthesizing the data set using only the se-

lected coefficients. This idea has been used for different types of data sets defined on irregular triangulations in [8, 4, 5, 6]. Below, we describe the basic principle.

Let T denote a triangle of the domain and s a data value (scalar) defined on a triangle. The superscript f (fine) denotes quantities before the local decimation algorithm, and c (coarse), after the decimation. Bold letters denote vectors. The pair $(\mathbf{T}, \mathbf{s})$ denotes the piecewise constant function equal to s_i on the triangles T_i. If $\mathbf{Q}$ is a matrix, then $\mathbf{Q}_k$ denotes the k-th column vector of $\mathbf{Q}$.

Let us focus here on the following setting: we are given a piecewise constant function $(\mathbf{T}^f, \mathbf{s}^f)$ on n triangles, and a set of m triangles $\mathbf{T}^c$ covering the same domain on the surface, with $m < n$.

The essence of an MR analysis is the filter bank algorithm [64] explained in Section 3.4. It consists of the local decomposition and reconstruction algorithm. In the present irregular setting this algorithm is the same as in wavelet theory: a function living in a fine space (in our case, the piecewise constant function on the finer triangulation) is decomposed into a coarser approximating function (piecewise constant on the coarser triangulation) and error functions (piecewise constant on the finer triangulation). These error functions have two properties: they can be used to recover the original data exactly, and their norm is a measure of the error between the input function and the approximation.

Intuitively, we have to define one smoothing operator that maps the input function onto its approximation, and one error operator that captures the difference between the input function and its approximation. These two operators are defined by two rectangular matrices $\mathbf{A}$ and $\mathbf{B}$ of size $m \times n$ and $(n - m) \times n$, respectively:

$$\mathbf{s}^c = \mathbf{A}\mathbf{s}^f \tag{12}$$

$$\mathbf{d} = \mathbf{B}\mathbf{s}^f \tag{13}$$

The smoothing operator (12) computes the coarser coefficients $\mathbf{s}^c$ from the finer coefficients $\mathbf{s}^f$, and the error operator (13) computes the detail coefficients $\mathbf{d}$. The actual computation of the so-called analysis matrices $\mathbf{A}$ and $\mathbf{B}$ is detailed in [3]. One step of the filter bank algorithm can thus be illustrated as follows:

$$\begin{array}{ccc} \mathbf{s}^f & \longrightarrow & \mathbf{s}^c \\ & \searrow & \\ & & \mathbf{d} \end{array}$$

In order to keep a constant memory size for the data values, the original coefficients $\mathbf{s}^f$ are cleared from memory after the decimation, and replaced by the coarse and detailed coefficients $\mathbf{s}^c$ and $\mathbf{d}$. Of course, the sum of the sizes of $\mathbf{s}^c$ and $\mathbf{d}$ equals the size of $\mathbf{s}^f$. Since $\mathbf{s}^f$ is cleared from memory, the decomposition formulas (12) and (13) have to be invertible, in order to be able to recover the original data values. This is the purpose of the reconstruction formula:

$$\mathbf{s}^f = \mathbf{P}\mathbf{s}^c + \mathbf{Q}\mathbf{d}. \tag{14}$$

The so-called synthesis matrices $\mathbf{P}$ and $\mathbf{Q}$ are of sizes $n \times m$ and $n \times (n - m)$, respectively. Intuitively, the operator $\mathbf{P}$ is the inverse of the smoothing operator $\mathbf{A}$:

$\mathbf{P}$ acts as a subdivision operator, although subdivision is not possible if the triangular domains are non-nested. The operator $\mathbf{Q}$ adds the details $\mathbf{d}$ to the oversampled data $\mathbf{P}\mathbf{s}^c$, in order to recover the original data $\mathbf{s}^f$. The matrices $\mathbf{P}$ and $\mathbf{Q}$ can be computed from $\mathbf{A}$ and $\mathbf{B}$ by:

$$(\mathbf{P}\ \mathbf{Q}) = \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix}^{-1}.$$

To be more precise about the properties of our decomposition, let us rewrite the reconstruction formula (14) with a functional point-of-view instead of a coefficient point-of-view:

$$(\mathbf{T}^f, \mathbf{s}^f) = (\mathbf{T}^f, \mathbf{P}\mathbf{s}^c) + \sum_{k=1}^{n-m} d_k(\mathbf{T}^f, \mathbf{Q}_k), \tag{15}$$

where $\mathbf{Q}_k$ denotes the k-th column vector of $\mathbf{Q}$, and d_k denotes the k-th detail coefficient of $\mathbf{d}$.

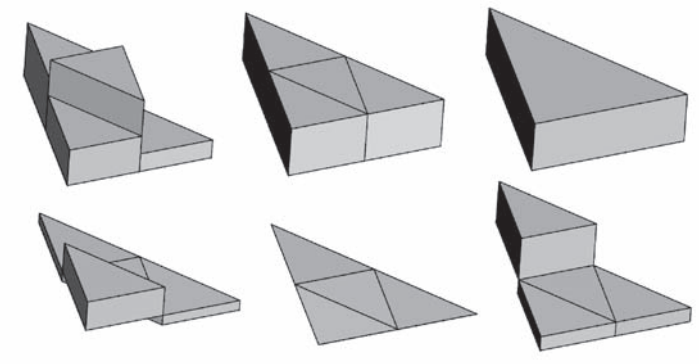


Fig. 17. Local decomposition by 4-to-1 split: finer, intermediate and coarser approximations on top, detail coefficients times wavelet functions on bottom. In this case, the coarse and fine triangular domains are nested, and therefore, the intermediate approximation equals the coarser approximation. The relative high magnitudes of the detail coefficients (bottom part) show the large L^2 error between the fine and coarse approximations.

Figs. 17 and 18 illustrate the local decomposition on two examples. In both figures, the top part shows, from left to right, the finer function $(\mathbf{T}^f, \mathbf{s}^f)$, the intermediate function $(\mathbf{T}^f, \mathbf{P}\mathbf{s}^c)$, and the final coarser function $(\mathbf{T}^c, \mathbf{s}^c)$. The bottom part shows the detail coefficients times the wavelet functions: $\mathbf{d}(\mathbf{T}^f, \mathbf{Q}^f)$. Figure 17 shows the local decomposition on a 4-to-1 split example. This leads to a traditional Haar wavelet decomposition for irregular triangular meshes. Note in this case that the intermediate function $(\mathbf{T}^f, \mathbf{P}\mathbf{s}^c)$ (top-middle), although defined over a finer mesh, equals the coarser function $(\mathbf{T}^c, \mathbf{s}^c)$ (top-right). In Figure 18, the block results from the removal of one interior vertex. Therefore, two detail coefficients are computed (bottom part).

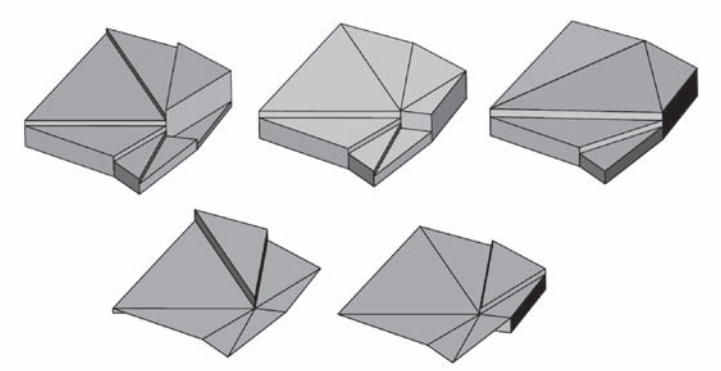


Fig. 18. Local decomposition by Delaunay-removal: finer, intermediate and coarser approximations on top, detail coefficients times wavelet functions on bottom. Since the triangular domains are non-nested, the intermediate approximation differs from the coarse approximation. The relative low magnitude of the detail coefficients (bottom part) shows the small L^2 error between the fine and coarse approximations.

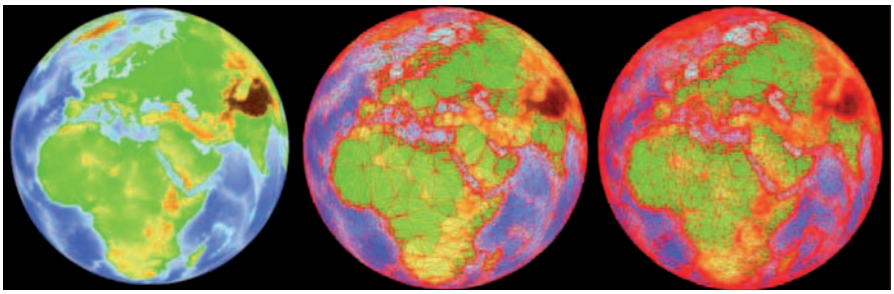


Fig. 19. Original data set with 1.3 M faces. Middle: partial reconstruction with 60000 faces. Right: partial reconstruction with 150000 faces.

Combining successive removal of sets of non-adjacent vertices with the filter bank algorithm leads to the construction of a hierarchy of triangulations that corresponds to an MR analysis of the initial data set. Threshold reconstruction is one of the possible applications of such a wavelet-based MR analysis. It consists of applying the reconstruction formula (15) only to the blocks of the hierarchy whose wavelet coefficients are greater than a fixed threshold. The visual effect is illustrated in Figure 19. The original data set (ETOP05 data set) in Figure 19 (left) has been fully analyzed with the preservation of the coastlines. Figures 19 (middle and right) show two partial threshold reconstructions. See also Figures CP-3 in Appendix C.

4.2 Surface Meshes

Developing MR modeling methods for large manifold surface meshes has been the subject of a great many research papers. Motivated by the ever-increasing size of polygonal meshes resulting from laser range scanners, researchers have tried to generalize the theory of wavelet MR analysis, successfully applied to 2D images, in order to compress, efficiently render, transmit, and *edit* such large meshes. One innovative application of MR methods for surface meshes is the ability to perform editing operations at different resolutions, as illustrated in Section 3.1 for curves. However, these surface meshes are generally irregular meshes, i.e., they do not have subdivision connectivity. Different approaches exist in the literature to define an MR analysis for manifold surface meshes.

For example, one of the early papers [17] first computes an approximation of the original data, a two-dimensional manifold triangular mesh, using new data defined on a grid with subdivision connectivity. The new data can then be analyzed with a wavelet analysis.

The groundbreaking work in the area of MR mesh representation was done by [61]. This paper also proposed a theory close to wavelet MR analysis. Its MR surface mesh model is closely related to subdivision surfaces. In general, in every MR analysis/synthesis scheme, the synthesis (or reconstruction) process can be seen as the combination of a subdivision step with a correction step. Based on this observation, [61] built an MR analysis/synthesis scheme on top of well known surface subdivision schemes, including Loop and Butterfly subdivision schemes (see the Chapter on Subdivision surfaces and applications in this volume). It turns out that only interpolating subdivision schemes lead to a linear time analysis process, while the synthesis process can always be performed in linear time. In order to build an MR analysis/synthesis scheme on top of a surface subdivision scheme, Lounsberry et al. [61] introduced a scalar product for functionals defined on the surface domain, and used this scalar product in order to define wavelet functions with good approximating properties. Based on this MR scheme, a fine mesh with subdivision connectivity can be represented on a wavelet basis. Thus, compression of the mesh can be performed by neglecting small wavelet coefficients. Progressive transmission is efficiently implemented by sorting the wavelet coefficients and transmitting them, starting with the most significant. The progressive transmission and its application to MR mesh viewing is the topic of [13].

This pioneering work was followed by [91] and [57]. These two papers did not rely on a genuine wavelet decomposition of the meshes. Rather, they mimicked the analysis process of the wavelet MR representation, by using a smoothing procedure to convert fine meshes into coarser meshes, and by encoding the error occurring during this smoothing procedure. [91] introduced highly adaptive procedures, with the aim of being able to edit large meshes in real-time. While [91] was still restricted to meshes with subdivision connectivity, [57] proposed a generalization to arbitrary meshes.

The idea of remeshing the irregular surface mesh into a semi-regular mesh (see Figure 20) with subdivision connectivity [60, 41] before computing wavelet-

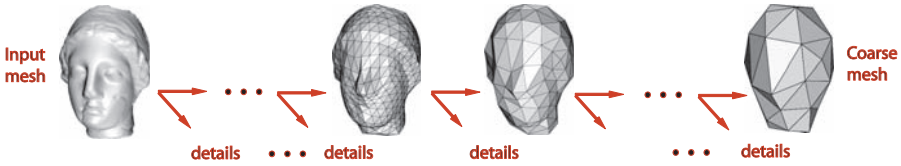


Fig. 20. Multiresolution analysis of a semi-regular mesh.

based MR analysis has also been used for signal processing applications, such as coding and compression of surface meshes. The compression allows a compact storage or a fast transmission of these surface data in a bandwidth-limited application. Wavelets are now frequently exploited to perform efficient compression. Based on MR analysis, wavelet coders do not only achieve better compression rates [61, 54, 40, 53, 69, 70] than methods based on signal quantization, but also make the progressive transmission and adaptive display easier.

5 Conclusions and Open Issues

Multiresolution representations play an increasing role in geometric design. Their use for both polygonal meshes and freeform polynomial and rational representations is expected to increase as their usefulness is further recognized. Nevertheless, many issues are still open and need to be resolved before the full power of this representation can be revealed.

Starting with MR representation for B-spline curves, both uniform and non-uniform knot spacings are, by now, fully supported and understood. Yet, the computation of the B-wavelet basis functions for spaces with non-uniform knot sequences is expensive and methods should be sought to reduce this cost.

Another related problem is the question of the inherent imprecisions of interactive MR editing of freeform curves. Being imprecise, it is difficult to employ in precise engineering design. In order to improve the precision, linear constraints are already embedded with interactive MR editing as well as a few non-linear constraints such as area and arc-length. The efficiency in solving linear constraints makes them attractive but also limited. Other families of non-linear constraints should be embedded with MR as well. Examples include curvature prescriptions, fairing requirements, higher order moments, etc.

Features are, many times, viewed as high frequency details of the geometry. A smooth shape of some animation could be combined with high frequency details of hair, thorns, or just scales. However, the simple algebraic sum of the two shapes would yield a result that is not necessarily the most appealing one. This is due to the fact that the details, when added algebraically, are not oriented along the smooth shape's geometry. Different MR decomposition schemes, which are intrinsically geometric and not algebraic, might be able to resolve such problems. In this sense,

intrinsic MR decomposition schemes, such as the presented curvature signatures, should be further explored.

References

1. M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.
2. R. Bartels and J. Beatty. A technique for the direct manipulation of spline curves. *Graphics Interface '89*, pages 33–39, 1989.
3. G.-P. Bonneau. Multiresolution analysis with non-nested spaces. *Computing Supplementum*, 13:51–66, 1996.
4. G.-P. Bonneau. Multiresolution analysis on irregular surface meshes. *IEEE Transactions on Graphics and Visualization*, 4(4):365–378, 1998.
5. G.-P. Bonneau and A. Gerussi. Hierarchical decomposition of datasets on irregular surface meshes. In *Proceedings of CGI'98,*, pages 59–63, Hannover, Germany, June 1998.
6. G.-P. Bonneau and A. Gerussi. Level of detail visualization of scalar data sets on irregular surface meshes. In *Proceedings Visualization'98,*, pages 73–77. IEEE, 1998.
7. G. P. Bonneau and H. Hagen. Variational design of rational bézier curves and surfaces. In L. Laurent and L. Schumaker, editors, *Curves and Surfaces*, volume II, pages 51–58. AK Peters, 1994.
8. G.-P. Bonneau, S. Hahmann, and G.M. Nielson. Blac-wavelets: a multiresolution analysis with non-nested spaces. In *Proceedings Visualization'96,*, pages 43–48. IEEE, 1996.
9. P. Borel and A. Rappoport. Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics*, 13(2):137–155, 1994.
10. M. P. Do Carmo. *Differential Geometry of Curves and Surfaces.* Cambridge University Press, 1976.
11. G. Celniker and D. Gossard. Deformable curve and surface finite-elements for free-form shape design. In *ACM SIGGRAPH Conference Proceedings*, pages 257–266. ACM, 1991.
12. G. Celniker and W. Welch. Linear constraints for deformable b-spline surfaces. In *Symposium on Interactive 3D Graphics*, pages 165–170, 1992.
13. A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle. Interactive multiresolution surface viewing. *Computer Graphics*, 30(Annual Conference Series):91–98, 1996.
14. C. K. Chui and E. G. Quak. Wavelets on a bounded interval. In D. Braess and L. Schumaker, editors, *Numerical Methods of Approximation Theory, Volume 9*, pages 53–75. Birkhäuser Verlag, Basel, 1992.
15. P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Multiresolution representation and visualization of volume data. *IEEE Trans. on Visualization and Comp. Graph*, 3(4):352–369, 1997.
16. D. Eberly and J. Lancaster. On gray scale image measurements: I. arc length and area. *CVGIP: Graphical Models and Image Processing*, 53(6):538–549, 1991.
17. M. Eck, T. DeRose, T. Duchamp, H. Hoppe, T. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Computer Graphics Proceedings (SIGGRAPH 95)*, pages 173–182, 1995.
18. G. Elber. Symbolic and numeric computation in curve interrogation. *Computer Graphics forum*, 14(1):25–34, March 1995.

19. G. Elber. Multiresolution curve editing with linear constraints. *The Journal of Computing & Information Science in Engineering*, 1(4):347–355, December 2001.
20. G. Elber and C. Gotsman. Multiresolution control for nonuniform bspline curve editing. In *The Third Pacific Graphics Conference on Computer Graphics and Applications, Seoul, Korea*, pages 267–278, [August 1995.
21. G. Farin, G. Rein, N. Sapidis, and A. J. Worsey. Fairing cubic b-spline curves. *Computer Aided Geometric Design*, 4:91–103, 1987.
22. A. Finkelstein and D. H. Salesin. Multiresolution curves. *Computer Graphics Proceedings (SIGGRAPH 94)*, pages 261–268, 1994.
23. L. De Floriani. A pyramidal data structure for triangle-based surface description. *IEEE Computer Graphics and Applications*, 9(2):67–78, 1989.
24. L. De Floriani, M. Alexa, Marie-Paule Cani, Paolo Cignoni, Emanuele Danovaro, Thomas Di Giacomo, HyungSeok Kim, Nadia Magnenat-Thalmann, and Enrico Puppo. Level-of-detail shape modeling. In *chapter 5 of this book*. Springer, 2005.
25. D. Forsey and R. Bartels. Hierarchical b-spline refinement. *Proceedings of SIGGRAPH'88, ACM New York*, pages 205–212, 1988.
26. B. Fowler. Geometric manipulation of tensor product surfaces. In *1992 Symposium on Interactive 3D Graphics*, pages 101–108, 1992.
27. B. Fowler. Geometric manipulation of tensor product surfaces. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 101–108. ACM Press, 1992.
28. B. Fowler and R. Bartels. Constraint-based curve manipulation. *IEEE Computer Graphics and Applications*, 13(5):43–49, 1993.
29. M. Gleicher. Integrating constraints and direct manipulation. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 171–174. ACM Press, 1992.
30. E. Goldstein and Craig Gotsman. Polygon morphing using a multiresolution representation. In *Graphics Interface '95*, pages 247–254. Canadian Inf. Process. Soc., 1995.
31. C. Gonzalez-Ochoa, S. Mccammon, and J. Peters. Computing moments of objects enclosed by piecewise polynomial surfaces. *ACM Transaction on Graphics*, 17(3):143–157, July 1998.
32. C. Gonzalez-Ochoa and J. Peters. Localized-hierarchy surface splines (less). In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 7–15. ACM Press, 1999.
33. S. Gortler and M. Cohen. Hierarchical and variational geometric modeling with wavelets. In *1995 Symposium on 3D Interactive Graphics*, pages 35–41, 1995.
34. S. Gortler, P. Schröder, M. Cohen, and P. Hanrahan. Wavelet radiosity. *Computer Graphics Proceedings (SIGGRAPH 93)*, pages 221–230, 1993.
35. S. J. Gortler. Private communications.
36. S. J. Gortler. Wavelet methods in computer graphics. *PhD thesis, Department of Computer Science, Princeton*, 1994.
37. G. Greiner. Variational design and fairing of spline surfaces. In *Proc. Eurographics 1994*, pages 143–154, 1994.
38. G. Greiner and J. Loos. Data dependent thin plate energy and its use in interactive surface modeling. *Eurographics '96 (1996)*, 15:176–185, 1996.
39. M. Gross, L. Lippert, R. Dietrich, and S. Häring. Two methods or wavelet-based volume rendering. *Computers & Graphics*, 21(2):237–252, 1997.
40. I. Guskov, A. Khodakovsky, P. Schröder, and W. Sweldens. Hybrid meshes: Multiresolution using regular and irregular refinement. In *Proceedings of SoCG 2002*, 2000.
41. I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal meshes. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 95–102. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

42. H. Hagen and P. Santarelli. Variational design of smooth b-spline surfaces. In H. Hagen, editor, *Topics in Geometric Modeling*, pages 85–94. SIAM Philadelphia, 1992.
43. H. Hagen and G. Schulze. Automatics smoothing with geometric surface patches. *Computer Aided Geometric Design*, pages 231–236, 1987.
44. S. Hahmann. Shape improvement of surfaces. *Computing Suppl.*, 13:135–152, 1998.
45. S. Hahmann and G.-P. Bonneau. Polynomial surfaces interpolating arbitrary triangulations. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):99–109, 2003.
46. S. Hahmann, G.-P. Bonneau, B. Caramiaux, and M. Cornillac. Multiresolution morphing of planar curves. *Computing*, 2007. to appear.
47. S. Hahmann, B. Sauvage, and G.-P. Bonneau. Area preserving deformation of multiresolution curves. *Computer Aided Geometric Design*, 22(4):349–367, 2005.
48. H. Hoppe. Progressive meshes. *Computer Graphics Proceedings (SIGGRAPH 96)*, pages 99–108, 1996.
49. H. Hoppe. View-dependent refinement of progressive meshes. *Computer Graphics Proceedings (SIGGRAPH 97)*, pages 189–198, 1997.
50. W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. In *Computer Graphics (SIGGRAPH 92 Proceedings)*, pages 177–184, July 1992.
51. P. D. Kaklis and N. S. Sapidis. Convexity-preserving interpolatory parametric splines of nonuniform polynomial degree. *Comput. Aided Geom. Des.*, 12(1):1–26, 1995.
52. R. Kazinnik and G. Elber. Orthogonal decomposition of non-uniform bspline spaces using wavelets. *Computer Graphics forum*, 16(3):27–38, September 1997.
53. A. Khodakovsky and I. Guskov. Compression of normal meshes, 2003.
54. A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 271–278. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.
55. D. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983.
56. L. Kobbelt. A variational approach to subdivision. *Computer Aided Geometric Design*, 13:743–761, 1996.
57. L. Kobbelt, S. Campagna, J. Vorsatz, and HP. Seidel HP. Interactive multiresolution modeling on arbitrary meshes. In *Computer Graphics Proceedings (SIGGRAPH 98)*, pages 105–114, 1998.
58. L. Kobbelt and P. Schröder. A multiresolution framework for variational subdivision. *ACM Trans. on Graph.*, 17(4):209–237, 1998.
59. A. W. F. Lee, D. Dobkin, W. Sweldens, and P. Schröder. Multiresolution mesh morphing. *Computer Graphics Proceedings (SIGGRAPH 99)*, pages 343–350, 1999.
60. Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. *Computer Graphics*, 32(Annual Conference Series):95–104, 1998.
61. M. Lounsbery, T. De Rose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transaction on Graphics*, 16(1):34–73, 1997.
62. T. Lyche and K. Morken. Spline wavelets of minimal support. In D. Braess and L. Schumaker, editors, *Numerical Methods of Approximation Theory*, pages 177–194. Birkhäuser Verlag, Basel, 1992.
63. Tom Lyche and Knut Morken. Knot removal for parametric b-spline curves and surfaces. *Comput. Aided Geom. Des.*, 4(3):217–230, 1987.
64. S. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:674–693, 1989.
65. E. Mehlum. Non-linear spline. In R. E. Barnhill and R. F.R iesenfeld, editors, *Computer Aided Geometric Design*, pages 173–208. Academic Press, 1974.

66. M.Gross, O. Staadt, and R. Gatti. Efficient triangular surface approximations using wavelets and quadtree data structures. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):130–143, 1996.
67. H. P. Moreton and C. H. Séquin. Functional optimisation for fair surface design. *Computer Graphics*, 26(2):167–176, 1992.
68. G. Nielson, IH. Jung, and J. Sung. Haar-wavelets over triangular domains with applications to multiresolution models for flow over a sphere. In *IEEE Visualization'97*, pages 143–150, november 1997.
69. F. Payan and M. Antonini. An efficient bit allocation for compressing normal meshes with an error-driven quantization. *Computer Aided Geometric Design*, 22:466–486, July 2005.
70. F. Payan and M. Antonini. Mean square error approximation for wavelet-based semiregular mesh compression. *IEEE Transactions on on Visualization and Computer Graphics (TVCG)*, 2006. to appear.
71. J. P. Pernot, S. Guillet, J. C. Leon, F. Giannini, B. Falcidieno B., and E. Catalano. A shape deformation tool to model character lines in the early design phases. In *Proceedings Shape Modeling International 2002, Banff, Canada*, 2002.
72. J. Peters. C1-surface splines. *SIAM J. Numer. Anal.*, 32(2):645–666, 1995.
73. M. Plavnik and G. Elber. urface design using global constraints on total curvature. In *The VIII IMA Conference on Mathematics of Surfaces*, September 1998.
74. A. Rappoport, A. Sheffer, and M. Bercovier. Volume-preserving free-form solids. In *Proceedings of Solid Modeling 95*, pages 361–372, May 1995.
75. A. Raviv and G. Elber. Three dimensional freeform sculpting via zero sets of scalar trivariate functions. *CAD*, 32(8/9):513–526, July/August 2000.
76. C. H. Reinsch. Smoothing by spline functions ii. *Num. Math.*, 16:451–454, 1967.
77. B. Sauvage. *Déformation de courbes et surfaces multirésolution sous contraintes*. Phd, Institut National Polytechnique de Grenoble (INPG), December 2005.
78. B. Sauvage, S. Hahmann, and G.-P. Bonneau. Length preserving multiresolution editing of curves. *Computing*, 72:161–170, 2004.
79. B. Sauvage, S. Hahmann, and G.-P. Bonneau. Length constrained multiresolution deformation for surface wrinkling. In *International Conference on Shape Modeling and Applications, SMI'06*, pages 113–121, Matsushima, June 2006. IEEE Computer Society Press.
80. P. Schröder and W. Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. *Computer Graphics Proceedings (SIGGRAPH 95)*, pages 161–172, 1995.
81. T.W. Sederberg, P. Gao, G. Wang, and H. Mu. 2-d shape blending: An intrinsic solution to the vertex path problem. *Computer Graphics,(SIGGRAPH 93 Proceedings)*, 27:15–18, 1993.
82. M. Shapira and A. Rappoport. Shape blending using the star-skeleton representation. *IEEE Comput. Graph. Appl.*, 15(2):44–50, 1995.
83. E. Stollnitz, T. DeRose, and D. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan-Kaufmann, 1996.
84. Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for computer graphics: A primer, part 2. *IEEE Computer Graphics and Applications*, 15(4):75–85, 1995.
85. W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546, 1997.
86. J. Warren and H. Weimer. Variational subdivision for natural cubic splines. *Approximation Theory IX*, 2:345–352, 1998.
87. H. Weimer and J. Warren. Subdivision schemes for thin plate splines. *Computer Graphics Forum (Proceedings of Eurographics 98)*, pages 303–313, 1998.

88. H. Weimer and J. Warren. Subdivision schemes for fluid flow. *Computer Graphics (SIGGRAPH 99 Conference Proceedings)*, pages 111–120, August 1999.
89. W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics (SIGGRAPH '92 proceedings)*, 26:157–166, July 1992.
90. A. Yvart, S. Hahmann, and G.-P. Bonneau. Hierarchical triangular splines. *ACM Transactions on Graphics*, 24(4):1374–1391, 2005.
91. D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. *Computer Graphics Proceedings (SIGGRAPH 97)*, pages 259–268, 1997.

Subdivision Surfaces and Applications

Chiara Eva Catalano[1], Ioannis Ivrissimtzis[2], and Ahmad Nasri[3]

[1] Istituto di Matematica Applicata e Tecnologie Informatiche, Italian National Research Council, Genova (Italy) chiara.catalano@ge.imati.cnr.it
[2] Durham University, Department of Computer Science (UK) ioannis.ivrissimtzis@durham.ac.uk
[3] American University of Beirut, Department of Computer Science (Lebanon) anasri@aub.edu.lb

Summary. After a short introduction on the fundamentals of subdivision surfaces, the more advanced material of this chapter focuses on two main aspects. First, shape interrogation issues are discussed; in particular, artifacts, typical of subdivision surfaces, are analysed. The second aspect is related to how structuring the geometric information: a multi-resolution approach is a natural choice for this geometric representation, and it can be seen as a possible way to structure geometry. Moreover, a first *semantic* structure can be given by a set of meaningful geometric constraints that the shape has to preserve, often due to the specific application context. How subdivision surfaces can cope with constraint-based modelling is treated in the chapter with a special attention to applications.

1 Introduction

The problem of generating a smooth surface from a coarse mesh is one of the central themes of Shape Modelling. In a typical application scenario, the smoothness of the final surface is required for visually appealing renderings; on the other hand, many times only a coarse mesh can be created and maintained. The latter is usually the case with the manual creation and editing of a shape, or when there is limited memory and bandwidth for the storage and transmission of the shape.

Subdivision employs a simple and intuitive procedure to solve this problem. The surface is completely defined by an initial coarse mesh which is progressively refined by inserting new vertices and connecting them with edges and faces, until it converges to a smooth surface in the limit. Figure 1 shows two examples, one of a geometric and one of a natural shape.

The capability of expressing a shape at incremental levels of resolution causes a natural structuring of the geometric information according to different modelling requirements. In this chapter, shape interrogation and structuring issues will be specialised for subdivision surfaces.

In the last decades great attention has been put on these surfaces and plenty of literature has been produced. The biggest challenge still remains their practical usage

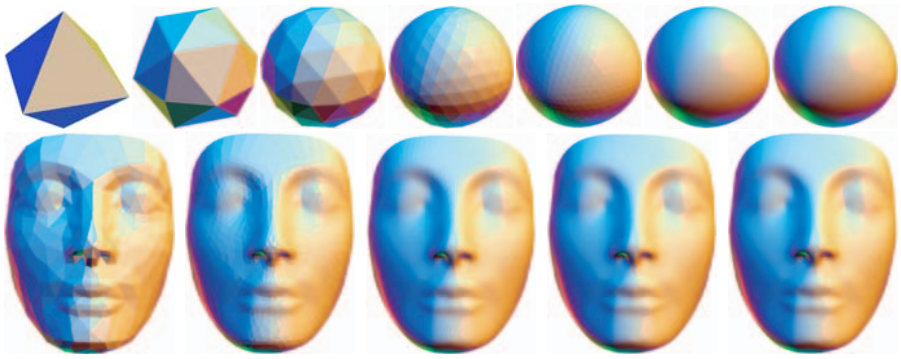


Fig. 1. Two examples of Loop subdivision. All the triangle meshes in the first four sections of the chapter were rendered with Yutaka Ohtake's *MeshViewer*, which is available online [54].

as an alternative geometric representation in application domains, Computer Aided Design (CAD) above all. For such reason, subdivision surfaces will be treated here mainly from the application point of view.

Section 2 gives some background on subdivision, introducing the main subdivision schemes and analysing their smoothness properties. The high-order continuity of these schemes means that they generally produce high-quality surfaces. Nevertheless, this does not automatically make subdivision an intuitive design tool. Section 3 studies subdivision artifacts, defined as unexpected features on the subdivision surface which cannot be intuitively controlled by repositioning the vertices of the original coarse mesh. In Section 4, usability issues are discussed. Besides the aliasing effects, another limit that prevents subdivision surfaces from being a powerful modelling tool for applications is the insufficient shape predictability. In Section 5, the constrained based subdivision is introduced, which permits to overcome such a restriction by fulfilling specific modeling needs.

2 Subdivision basics

2.1 Subdivision schemes

Traditionally, subdivision rules are defined in two stages. First, a refinement rule produces a sequence of ever increasing connectivities. Then, the actual positions of the new vertices are computed, usually as an affine combination of the old vertices. The coefficients of these affine combinations are described either in the form of *stencils*, which give the weights of all the old vertices influencing the position of a single new vertex, or *masks*, describing the influence of a single old vertex on the position of all the new vertices it affects.

Even though the basic idea behind subdivision surfaces is extremely simple, it can be implemented in a multitude of ways. Thus, over the years many different subdivision schemes have been proposed and next we briefly describe the most popular.

Following the standard terminology, a vertex of a quadrilateral mesh is called *regular* if it has valence four, that is, there are four edges incident to it. Similarly, the regular vertices of a triangular mesh are those with valence six, while all the other vertices are called *extraordinary*.

The *Catmull-Clark* scheme [12] is a quadrilateral, approximating scheme. The regular parts of the subdivision surface are tensor products of cubic B-splines and have C^2 continuity. At the extraordinary vertices the surface is C^1 continuous. Figure 2 shows the stencils of the scheme.

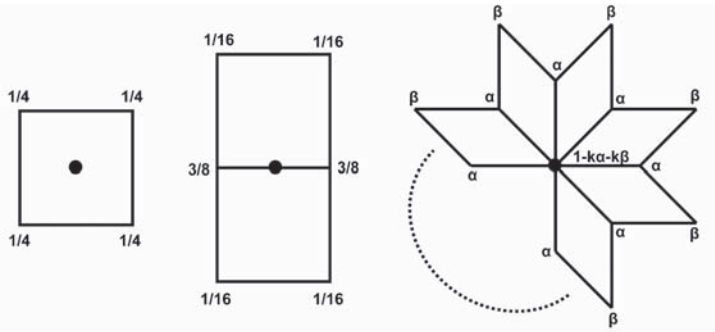


Fig. 2. The three stencils of the Catmull-Clark scheme. The new vertices correspond to old faces, edges and vertices, respectively. $\alpha = \frac{3}{2k^2}$ and $\beta = \frac{1}{4k^2}$, where k is the valence of the vertex.

The *Doo-Sabin* scheme [16] is a quadrilateral, dual, approximating scheme. Its regular parts are tensor products of quadratic B-splines and have C^1 continuity. At the extraordinary vertices the surface is also C^1 continuous.

The *Loop* scheme [41] is a triangular, primal, approximating scheme. Its regular parts are quartic box splines and have C^2 continuity. At the extraordinary vertices the surface is C^1 continuous. Special boundary rules for the Loop scheme were proposed in [7] and a ternary version was proposed in [43].

The *Butterfly* scheme [21] is a triangular, interpolating scheme. It is based on the univariate 4-point scheme [20] and gives C^1 continuous surfaces in regular patches. The original scheme has C^0 continuity at the extraordinary vertices, while the *Modified Butterfly* scheme [75] has C^1 continuity everywhere.

The above are the most commonly used schemes. Other, less frequently used schemes include the *Kobbelt* scheme [30], the *simplest* scheme [57], the $\sqrt{3}$-scheme [31, 35] and the *Quad/Triangle* scheme [70]. For an in-depth description of subdivision schemes we refer the reader to [74] and [72].

2.2 Subdivision analysis

As the main objective of subdivision is to produce a smooth surface out of a coarse mesh, the study of the properties of the limit surfaces focuses on the issue of analytical smoothness. In other words, we seek proofs that, given a generic control polyhedron as input, the subdivision scheme will produce a C^k continuous surface.

Laurent polynomials

The continuity properties of subdivision curves and their tensor products (regular subdivision surfaces) has been studied through the associated *Laurent polynomial* [17, 19]. The vertices of the polygon correspond to monomial components of the Laurent polynomial, and the effect of one subdivision step corresponds to polynomial multiplication.

Laurent polynomials are convenient tools for dealing with the dynamic nature of a subdivision mesh because operations such as addition, multiplication and convolution are defined over polynomials of any order. In contrast, in square matrix multiplication, all the matrices must have the same dimension and then they correspond to a fixed part of the connectivity of the subdivision mesh (see the next subsection).

Here we outline the univariate case following the notation in [18]. Let the subdivision rule be

$$f_i^{k+1} = \sum_{j \in \mathbf{Z}} \alpha_{i-2j} f_j^k \tag{1}$$

where $\{f_j^k\}$, $j \in \mathbf{Z}$ is the control polygon at step k. The mask of the scheme corresponds to a Laurent polynomial

$$\alpha(z) = \sum_{j \in \mathbf{Z}} a_j z^j. \tag{2}$$

We define the *generating function* of the initial control polygon f^0 as the Laurent polynomial

$$F(z; f^0) = \sum_{j \in \mathbf{Z}} f_j^0 z^j. \tag{3}$$

The generating function of the initial control polygon and the mask suffice to describe the final subdivision curve. In fact, simple computations involving these two Laurent polynomials give the control polygon after k subdivision steps and its m-th order divided differences.

In particular, the effect of one subdivision step on the control polygon is equivalent to polynomial multiplication by $q(z)$, where $q(z)$ can be easily derived from the mask $\alpha(z)$. The information contained in $q(z)$ can also be written in the form of a subdivision matrix S_q, with the odd and even coefficients of the polynomial corresponding to odd and even rows of the matrix. The convergence of the subdivision scheme is equivalent to the convergence of

$$\lim_{k \to \infty} S_q^k. \tag{4}$$

For the latter, it is enough to show that the maximum row norm $|\cdot|$ of the matrix S_q is less than one. If this is not the case, we proceed by computing the polynomial of the double subdivision step and checking if the norm of the corresponding matrix is less than 1. If not, n-multiple steps are considered. If the norm is less than one for some value of n, then the scheme is convergent.

Spectral analysis of the subdivision matrix

The analytical properties of the subdivision surface around an extraordinary vertex O of valence n are studied through the spectral analysis of a part of the subdivision matrix. The approach was introduced in [16], where necessary conditions for C^1 continuity were found and the coefficients of the Catmull-Clark and the Doo-Sabin schemes were tuned according to these conditions. It can be noticed that for irregular meshes we cannot use Laurent polynomials to describe the transformations of the subdivision mesh. Instead, we use matrices and study a part of the mesh around O, which contains enough information to determine a similar part with the same connectivity on the next mesh.

Due to symmetry assumptions, which hold for any reasonable scheme, the part of the mesh we study has rotational symmetry of order n around O. That means that, with the exception of one row and one column corresponding to O, the rest of the matrix has a circulant-block or block-circulant structure, depending on the labelling of the vertices. The convergence of the subdivision scheme requires the largest eigenvalue to be equal to one and the corresponding eigenvector to be $(1, 1, \ldots, 1)^T$.

The next question is if the surface is tangent-plane continuous at the limit of O. This depends on the second and the third eigenvalues and eigenvectors. These two eigenvalues are complex conjugates (in particular, if they are real, then they are equal). Sufficient conditions for a subdivision scheme to give tangent-plane continuous surfaces for generic initial inputs are given in [3].

The next question is about C^1 continuity, i.e., the existence of a local 1-1 continuous map, called the *characteristic map*, between the surface and the tangent plane. The study of the characteristic map is facilitated by the observation that the commonly used mesh refinement rules create around an extraordinary vertex a ring structure with regular connectivity, which has a natural parameterisation. Conditions guaranteeing the existence of the characteristic map were first obtained in [61]. In [58, 59] practical criteria are proposed for verifying the existence of the characteristic map, thus, guaranteeing C^1 continuity of the subdivision surfaces for generic initial inputs. Even though the sufficient conditions for C^1 continuity seem to be complex, in practice almost all of the proposed schemes are C^1 continuous. The reason relies on the symmetry assumptions taken into account in the design of a scheme, which implicitly satisfy the C^1 conditions.

The case is very different with C^2 continuity. The most popular subdivision schemes guarantee C^2 continuity at the regular part of the surface, but they present diverging second derivatives at the extraordinary points. Modifications in the masks may lead to bounded but discontinuous second derivatives, or to vanishing ones.

This behaviour of the second derivatives means that the surfaces have either unbounded, zero or discontinuous curvature. Analyses of the curvatures can be found in [62, 42, 64]. No C^2 stationary scheme has been found yet.

The continuity degree of subdivision surfaces

As we discussed above, the analysis of subdivision focuses on the question of C^k continuity of a given subdivision scheme. There are several limitations in this approach. First, all the subdivision surfaces produced by a scheme are treated as a single class, i.e., trying to find theorems that apply to all of them. Secondly, C^k continuity is a discrete measure of smoothness, as long as k is required to be an integer. A comparison with a continuous measure of a curve's smoothness, such as the Hölder regularity, shows that the discrete one can lead to very conservative estimates of smoothness. For example, the 4-point scheme for subdivision curves is not C^2, but nevertheless, it has Hölder regularity $2 - \epsilon$ for arbitrary small $\epsilon > 0$ [14].

Moreover, the analytical tools described above can mostly handle stationary schemes, that is, schemes where the subdivision coefficients depend on the connectivity only. For the evaluation of geometry-sensitive schemes, experimentation seems to be the only available practical tool. In [55], the experimental computation of Hölder exponents was proposed as an alternative way to estimate the degree of continuity of a subdivision scheme. In [45], a large scale experiment over a diverse sample of subdivision curves gave conclusive evidence for the behaviour of a non-stationary subdivision scheme.

3 Artifacts in subdivision surfaces

As seen in 2.2, the behaviour of subdivision surfaces depends on few eigenvalues and eigenvectors of the subdivision matrix. In many cases the spectral properties of the subdivision matrix dominate the surface generation process, overwriting the intent of the designer who is unable to use intuitively the vertices of the initial mesh to create the shape he/she has in mind. Even on very simple meshes, designers encounter unwanted *artifacts* in the form of spikes, ripples, oscillations, which they are not able to remove by repositioning the control points. The existence of artifacts is the main obstacle preventing the adoption of subdivision surfaces as a standard in industrial design and several other applications.

The definition of the artifacts as features on the subdivision surface that were not intended by the designer introduces an element of subjectivity [63]. Nevertheless, we can identify several distinct types of artifacts and classify them according to the eigenvalues and eigenvectors of the subdivision matrix that are responsible for them, see [22].

We call *first order artifacts* the artifacts related to the second and third eigenvalues of the subdivision matrix. These two eigenvalues and the corresponding eigenvectors affect tangent plane properties of the subdivision surfaces. The corresponding

artifacts appear in the form of uneven mesh structure with the neighbourhoods of low valence vertices being more densely meshed than those of the high valence vertices.

We call *second order artifacts* the artifacts related to the fourth, fifth and sixth eigenvalues of the subdivision matrix, that is, the eigenvalues and eigenvectors affecting the curvature behaviour of the surface. They appear as unwanted ripples of the surface around irregular vertices.

In principle, any type of artifact can be classified into one or both of the above categories. However, designers have at their disposal some higher level operations besides positioning control points: the two most common operations is curve *extrusion* and curve *revolution* around an axis. Thus, one can also study artifacts at that higher level.

In the following we discuss these types of artifacts using the Loop subdivision scheme to produce all the pictures with the exception of Figure 12 where $\sqrt{5}$-subdivision was used. Even though certain schemes may outperform the others with regard to a particular artifact, the Loop scheme has generally the best behaviour among all the triangular schemes and can be used as a benchmark.

3.1 First order artifacts

Around a vertex P, the subdivision mesh shrinks in the tangential direction by a ratio λ which is equal to the norm of the second and third eigenvalues of the subdivision matrix. In all major subdivision schemes (i.e., Catmull-Clark, Doo-Sabin, Loop, Butterfly) the value of λ at regular vertices is $\frac{1}{2}$, which is compatible with the binary refinement rule for the connectivity. The result is mesh evenly refined around regular vertices. Figure 3 shows the mesh structure around a valence six vertex in Loop subdivision.

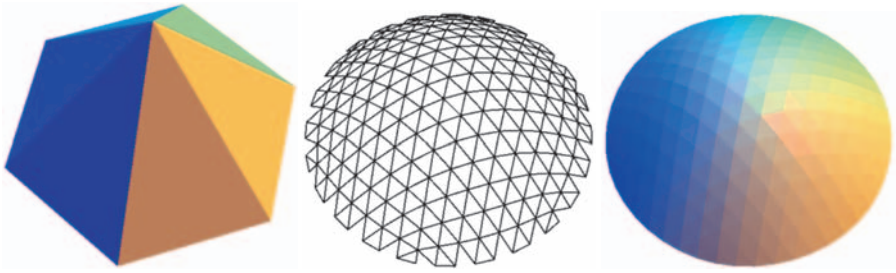


Fig. 3. In the vicinity of a valence six vertex the mesh is evenly refined because the second and third eigenvalues are equal to $\frac{1}{2}$.

On the other hand, in all major subdivision schemes, λ increases with the valence k. That means that in the tangential direction the subdivision mesh shrinks slower around high valence vertices than in the regular parts, resulting to an uneven mesh structure. Figure 4 shows the slower refinement around a vertex of valence 18. It can

be noticed that the slower refinement around high valences does not affect the quality of the limit surface. Nevertheless, very dense meshes may still produce low quality renderings as a result of it.

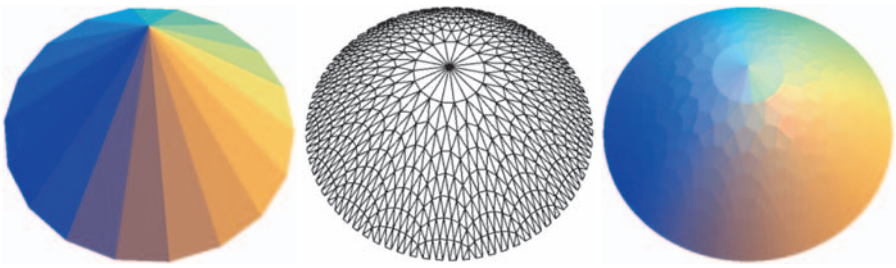


Fig. 4. In the vicinity of a valence 18 vertex the mesh is unevenly refined because the second and third eigenvalues are larger than $\frac{1}{2}$. Another artifact, in the form of a small spike at the valence 18 vertex, is also visible.

In dual schemes the uneven refinement problem takes the form of high valence faces with disproportionately large area. In some schemes the uneven refinement problem can be severe, e.g., in the *simplest scheme*. There, the problem of slow refinement of high valence faces is explicitly dealt with and modified subdivision rules are proposed to rectify it.

One solution to the uneven mesh structure artifact is to modify the subdivision coefficients, so that the second and third eigenvalues match the connectivity refinement ratio for all valences. As the initial subdivision rules were found to be optimal in some sense, such modifications require extra degrees of freedom, as for example using special subdivision rules, or working with larger stencils. Without extra degrees of freedom, we can expect that any modification ameliorating one kind of problem may create other types of artifacts. The trade-off between uneven refinement and ripples on the subdivision surface was shown in [74], pp.95-97, and studied in [4] for the Loop scheme.

A second type of first order artifact concerns the normal of the tangent plane at the limit of the vertex P. In many schemes, that normal depends only on the 1-ring polygon of P and not on the position of P itself. Thus, changes in the position of P do not change its normal. This is shown in Figure 5 where the high valence vertex of Figure 4 has been repositioned, creating a spike on the mesh. The limit position of P is not the point on the subdivision surface with the highest curvature, and its normal does not point in the direction of the spike as one would intuitively expect.

The same artifact arises in the regular case, see Figure 6. Therefore, it is also present when designing with B-splines. We consider this phenomenon to be an artifact because the resulting refined mesh is contrary to the designer's intuition, and this can hinder the design process. Nevertheless, it does not create any visible defect on the subdivision surface.

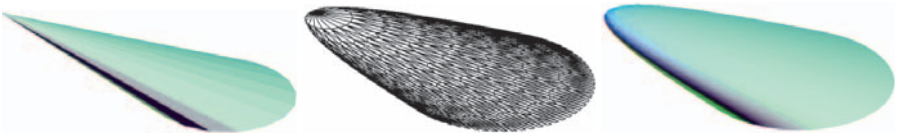


Fig. 5. The normal at the limit of the valence 18 vertex is the normal of its planar 1-ring neighbourhood.

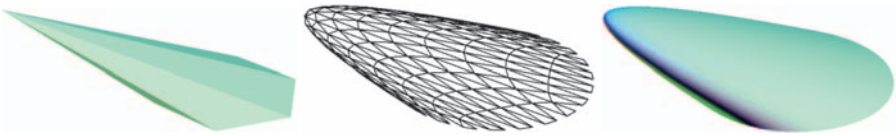


Fig. 6. The limit position of the spike on the coarse mesh is not in the highest curvature area of the refined mesh.

3.2 Second order artifacts

Although the designer can intuitively use the vertices of the coarse mesh to control the general shape of the subdivision surface, the same is not true for the local behaviour of the surface curvature. Thus, around extraordinary vertices the curvature oscillates in a way that cannot be intuitively controlled by the designer. The visual effect of the curvature oscillations is the rippling of the subdivision surface near large valence vertices, see [74], pp. 94. Figure 7 shows this artifact around a vertex of valence 12. Figure 11 shows curvature colourmaps for the subdivided meshes of this section (see also Figure CP-1 in Appendix D).

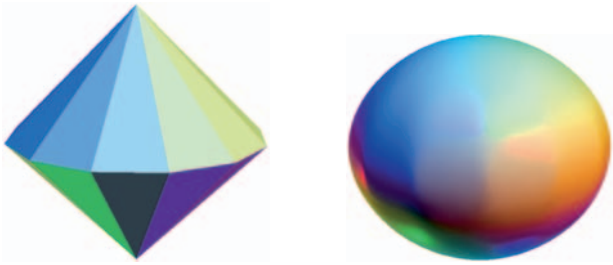


Fig. 7. A dipyramid with valence 12 apices. Loop subdivision creates a feature near the equatorial line which was not intended by the designer.

As it was pointed out in [63], these artifacts are related both to the high valence apices of the dipyramid and the 4-valent vertices in the equatorial line. In Figure 8, in order to separate the effect of the high valence vertices from the low valence ones, we first performed two steps of linear subdivision on the coarse mesh. We notice that

while both high and low valence vertices are responsible for the artifact, the impact of the low valences is larger.

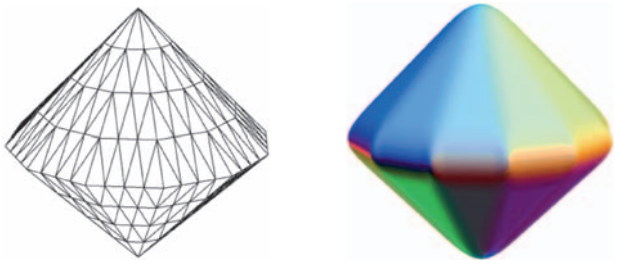


Fig. 8. The initial coarse mesh has been linearly subdivided twice. The artifact in the equatorial line is even more pronounced.

Apart from the irregularities of the connectivity, curvature artifacts are also related to the geometry of the coarse mesh. As mentioned above, the curvature behaviour of the subdivision scheme depends on the fourth, fifth and sixth eigenvalues and the corresponding eigenvectors. These eigenvalues and eigenvectors determine how the corresponding eigencomponents of the coarse mesh scale at each subdivision step.

In Figure 8 the negative Gaussian curvature areas in the equatorial line of the dipyramid are due to the hyperbolic eigencomponent of the 1-ring polygons around the 4-valent vertices, see [27]. By repositioning the apices of the dipyramid, the 1-ring polygon of one of the 4-valent vertices can become planar, thus it will have no hyperbolic component and the negative Gaussian curvature around that vertex will disappear. This is shown in Figure 9, where we can eliminate the artifacts on the right hand side of the model by moving the apices of the dipyramid to the right. On the other hand, the artifact on the left hand side of the model is now worse because the 1-ring polygons of the 4-valent vertices on the left have larger hyperbolic components.

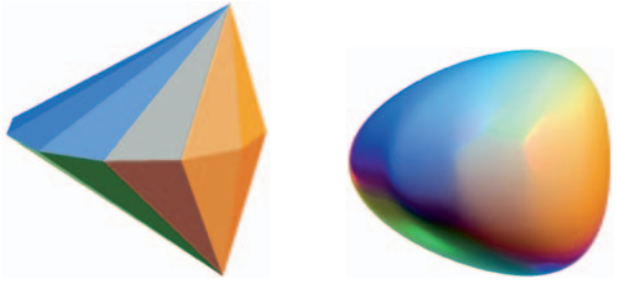


Fig. 9. The 4-valent vertex on the right hand side of the skew dipyramid has planar 1-ring polygon. Around this vertex there is no curvature artifact.

Obviously the elimination of artifacts through the elimination of hyperbolic components on the initial mesh is possible in very few cases only. In Figure 10 (a)-(b), the two apices of the dipyramid have valence four and the coarse mesh is a regular octahedron. It is one of the few meshes with the property that the 1-ring polygons around all the vertices are planar. We notice that there are no curvature artifacts. In Figure 10 (c)-(d), the apices of the dipyramid have valence 6 and, even though the connectivity of the mesh is more regular than that of the octahedron, curvature artifacts start to appear.

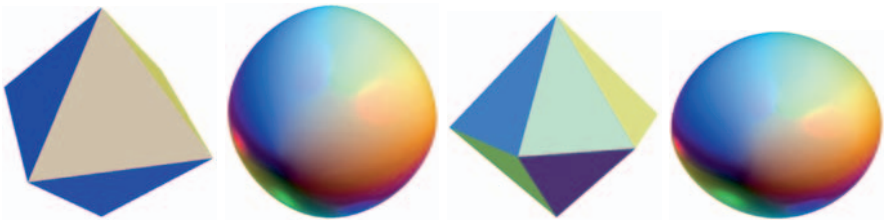


Fig. 10. (a)-(b): A regular octahedron. (c)-(d): A dipyramid with valence 6 apices.

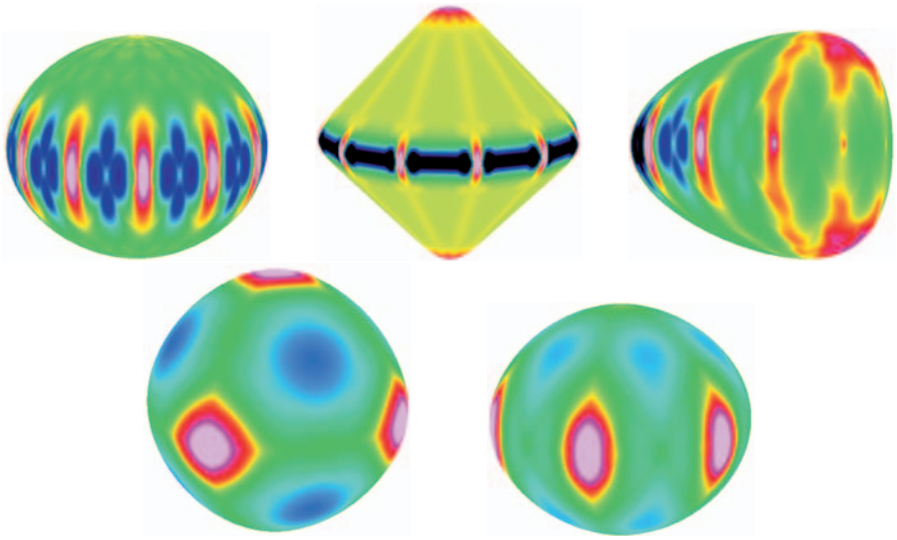


Fig. 11. Gaussian curvature colourmaps for the subdivided meshes shown in Figures 7, 8, 9 and 10, respectively. They are drawn in a relative scale, starting with red for the highest curvature values and going through yellow, green, blue and black to the lowest values.

3.3 Higher level artifacts

Curve extrusion is an operation commonly used in surface design. In most implementations, extrusion creates a regular mesh with the extrusion path being one of the mesh directions. When the extrusion path is a straight line, the designer expects the subdivision mesh to respect it.

The behaviour of subdivision schemes on extrusion meshes is well understood and it is known that, for a given mesh direction z_i, a subdivision scheme will not have extrusion artifacts along that direction if there is a term $1 + z_i$ in the corresponding z-transform, see [63]. For example, in Loop subdivision all three mesh directions are safe for extrusion.

However, this property does not hold for all subdivision schemes. Figure 12 shows that the $\sqrt{5}$-scheme for quad meshes proposed in [25] does suffer from extrusion artifacts. The reason is that it does not have a term $1 + z_i$ in either of the two mesh directions. The same is true for the $\sqrt{3}$-scheme proposed in [31]. Both schemes have totally fractal support [26], therefore they have extrusion artifacts in all mesh directions [63].

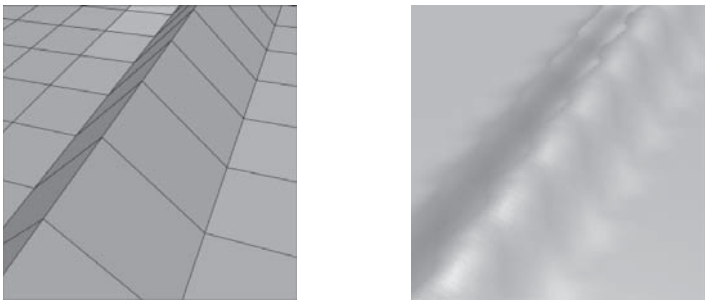


Fig. 12. A coarse mesh obtained through extrusion. The $\sqrt{5}$-scheme produces an artifact along the extrusion line.

Safe extrusion directions correspond to the connectivity, not to the geometry of the mesh. Figure 13 (top) shows three simple planar meshes which can be used as examples illustrating this point (see also Figure CP-2, top, in Appendix D).

In all the three cases we perform two steps of linear subdivision, and then lift in the direction of the normal all the points lying on the bold line. The results are shown in Figure 13 (see also Figure CP-2 in Appendix D).

The artifact in Figure 13 (b) (see also Figure CP-2 (b) in Appendix D) appears because the extrusion line does not follow the mesh direction. It is clearly visibly in the first subdivision step in the form of a surface feature perpendicular to the extrusion line. In the limit surface it is revealed by the mean curvature colourmap.

Two of the coarse meshes in Figures 13 (a)-(b) (see also Figures CP-2 (a)-(b) in Appendix D) make an interesting comparison because they have exactly the same geometry, i.e., their sets of vertices are identical, as well as the underlying surfaces.

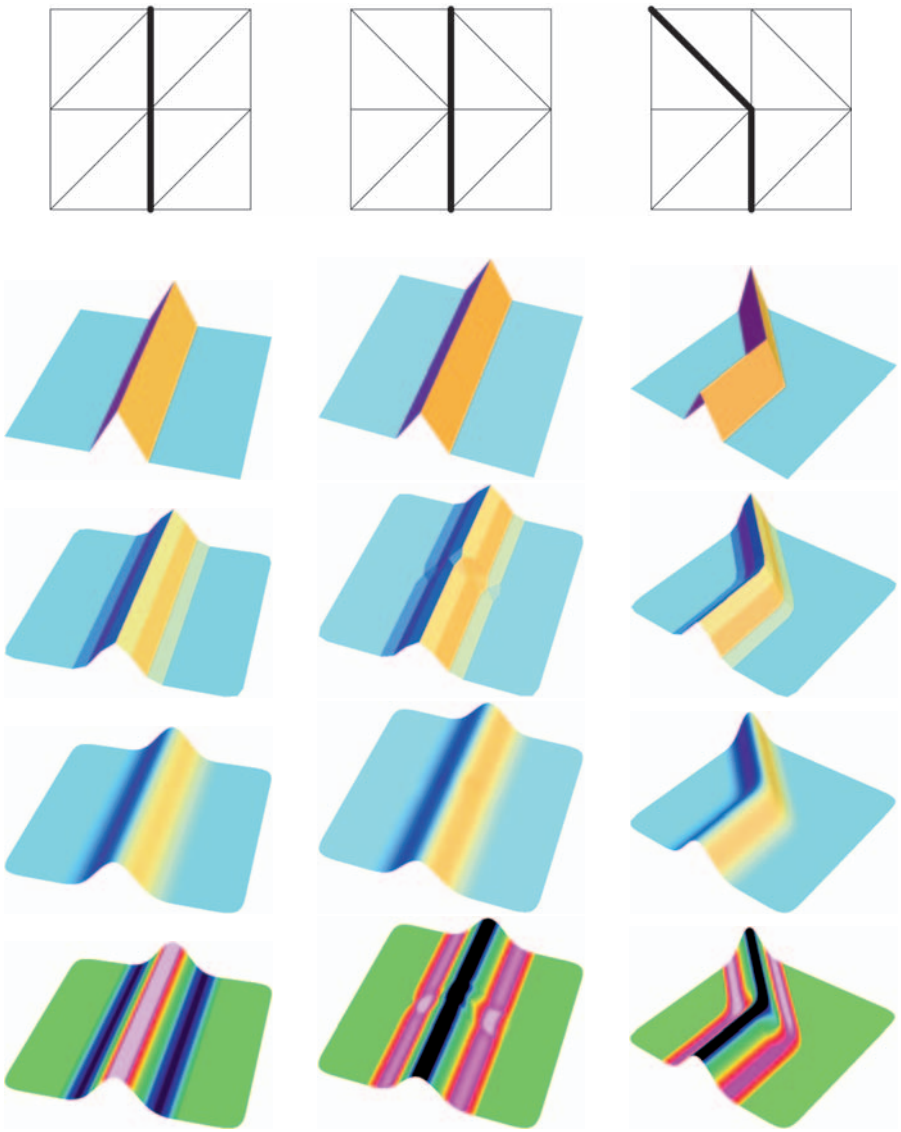


Fig. 13. Loop subdivision. Left: The extrusion path is a straight line on a mesh direction. Middle: The extrusion line is a straight line but not on a mesh direction. The artifact appears in the form of a feature perpendicular to the extrusion line at the point where it stops following the mesh direction. It is more visible in the first subdivision step. Right: The extrusion line is not straight but follows the mesh direction. There is no extrusion artifact. From top to bottom: The extrusion path, the coarse mesh, the mesh after one step of subdivision, the subdivision surface and its mean curvature colourmap.

Also, inside the boundary, that is in the area of interest, both meshes have regular connectivity (i.e., all the vertices have valence 6). Nevertheless, the 1-ring neighbourhoods on these two meshes are different, which explain the artifact in Figure 13 (b) (see also Figure CP-2 (b) in Appendix D).

Another commonly used design operation is the rotation of a curve along an axis to produce a surface of revolution. In this case, we expect artifacts similar to those shown in Figure 7, as the dipyramid is a typical surface of revolution.

4 Implementation and usability issues

Subdivision fundamentals have been treated so far from a theoretical point of view. In applications, designers needs some more practical clues to use comfortably subdivision surfaces as an alternative shape representation. In fact, it is often not easy and intuitive to get good results in terms of both shape and regularity when defining the control polyhedron of the surface desired. As a result, subdivision surfaces have made significant inroads mainly in the entertainment industry, where the typical mesh is relatively dense and natural, the latter meaning also that the artifacts will cancel each other.

A discussion will follow, where surface modelling and evaluation issues are faced from a more practical perspective.

4.1 Direct evaluation of subdivision surfaces

Since the subdivision surface is continuous only at the limit, the evaluation at arbitrary points is not always straightforward: in fact, every algorithm evaluating the limit surface can provide values that are an approximation of the real value on the discrete surface at a certain refinement level. Nevertheless, satisfying solutions have been provided.

If an interpolation scheme is used, evaluation is simple since all the points at each step belong to the final surface. If the subdivision scheme is a scheme extending splines, the coordinates of the regular points at the limit are the values of the spline at the corresponding points and the estimation is easy. As usual, the problem is given by the extraordinary points. The Stam's algorithm [69] is the work which answers almost completely this issue. It allows for the calculation of points and derivatives on the limit surface at arbitrary parameter values if using Catmull-Clark scheme. He showed that the surface and its derivatives can be calculated in terms of a set of eigenbasis functions depending only on the subdivision scheme. After treating the regular part, the behaviour of extraordinary vertices is studied: using some manipulations on the eigenstructure of the scheme, it is possible to analytically compute the surface everywhere with an algorithm costing as much as the evaluation of bi-cubic splines. This method can be analogously applied to Loop surfaces.

This work was extended by Zorin and Kristjansson [73] by considering the subdivision rules for piecewise-smooth surfaces with boundaries depending on parameters. They introduced a different set of basis vectors for evaluation, which, unlike eigenvectors, depend continuously on the coefficients of the subdivision rules.

Thanks to that, it is possible to define an evaluation procedure for parametric families of rules without considering an excessive number of special cases. In particular, the authors demonstrate how such bases are computed for a specific parametric family of subdivision rules extending Loop subdivision to meshes with boundaries.

4.2 Visual quality and shape predictability

Smoothness is important in many applications, being related to the quality of the shape. CAD is probably one of the environments where such an issue is crucial. Not only in the creation phase, but also at the manufacturing stage, high precision is required: shape must be as much correct as possible and surfaces must often be C^2. For this reason, surface evaluation usually follows the creation step, necessary in other phases as well, e.g., for estimating a tool path of numerical control machines and for simulation purposes.

For the visual inspection of the smoothness of the surface, light lines, such as reflection or shadow lines, are widely used in CAD and are mathematically related to the C^2-continuity of the surface. In [32] the treatment of light lines in generic discrete geometry, useful for example for visualisation purposes, is described.

However, not only tools for high regularity are needed. Common objects are not smooth everywhere and at the same rate; moreover, there are various contexts, such as manufacturing [34], finite element analysis and the design of thin-shell structure [13, 23], reverse engineering and the fitting of subdivision surfaces from clouds of data points [24, 44], where the lack of smoothness is required. Conversely, subdivision techniques tend to smooth the mesh and can be also used just as smoothing operators. Hence, it is important to have methods to decrease the smoothness at some points, or along some lines, e.g., when sharp edges are desired: using subdivision, special rules are applied to edges or vertices to sharpen [24].

When modelling industrial products, the "pleasantness" of a surface to the eye is not guaranteed by high smoothness only, but *fairing* issues have been considered in order to optimise the shape. Such a formulation has been extended to discrete representations as well, even specialised for subdivision models (i.e. variational subdivision surfaces, [33]).

For applications other than CAD, such as animation, smoothness and exact predictability are less crucial. In these cases, artifacts constitute the main drawback for visualisation and "pleasantness" purposes, thus requiring a satisfactory solution.

The research on artifacts follows two directions. One is the further tuning of the subdivision scheme which can be done either by conceiving more degrees of freedom [4] or by making the subdivision coefficients sensitive to the geometry of the control mesh [45]. The second direction is the *reverse engineering* of the initial input so that the subdivision surface has good properties even though the structure of the initial control mesh is counter-intuitive [2], as shown in Fig 14.

A different solution to the artifact problem of the subdivision surfaces is given by the *multi-resolution surfaces*, where subdivision surfaces are organised hierarchically, as naturally inspired by the recursive structure. They will be described in the section 4.3, where also other properties of this formulation will be outlined.

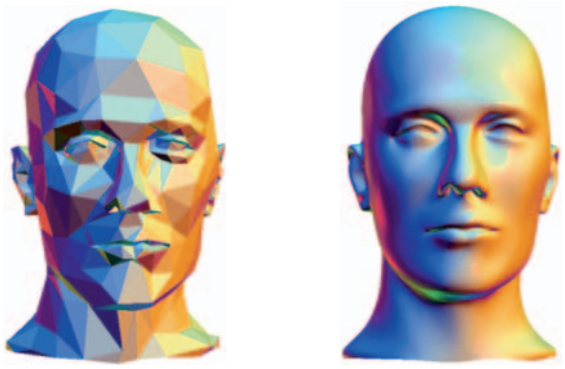


Fig. 14. An initial control mesh of 428 vertices gives a high quality smooth surface under Loop. Notice that the vertices of the initial mesh are not placed where would have been intuitively.

The problem of the control of the shape as required in product design is only partially solved by the three approaches mentioned above, especially in CAD/CAM, where it is very important that the shape of an object is represented as correctly as possible. In the case of subdivision, it is not often straightforward finding the right initial control net which will produce a good approximation of the real object when refined. However, a similar problem also appears when using splines, since the shape is not built directly but through a control polyhedron.

To have a better guess of the surface shape, it is generally convenient to use schemes where the convergence rates is a-priori known. Catmull-Clark, Doo-Sabin and Loop schemes converge to certain splines: envisaging the final shape is easier in this case. On the other hand, they are approximating schemes, thus subject to a shrinking effect which cannot be measured. By definition, interpolating schemes allow for a more predictable final surface, but at the expenses of smoothness; for this reason, approximating schemes are preferable. In addition, various studies have been done about the adaptation of the refinement depth to the features of the shape (see, for example, [56]). Naïve local refinement produces inconsistencies in the connectivity of the mesh, which affect further rendering, processing and editing of the shape. Thus, more efficient algorithms are required to address different levels of refinement on adjacent areas. Criteria to select automatically the regions requiring more density are usually based on the (high) local curvature of the shape. When specific needs of a user play a role, such as in the insertion of different kinds of features on a shape, more complex strategies must be devised and no completely automatic solutions may be possible.

An effective way to reduce the problem can be constraining the surface, where some important features must be preserved, and this is the most adopted strategy in editing subdivision surfaces. This can be seen as an alternative way of structuring the shape accordingly to its characterising elements. Such entities can be points, normals, curves and surfaces: they are geometrical, but -especially in applications-

have strong relationship with the modelling intent and the semantics of the object. In Section 5, a detailed treatment of constraint-based subdivision will be provided.

4.3 Multi-resolution subdivision surfaces

Subdivision surfaces belong to the continuous regular mesh-based LOD (Level-Of-Detail) representation due to their refinement process. This naturally hierarchical structure can be further enriched using multi-resolution techniques, which permit to store *details* according to the modelling requirements of the application context (see Chapter [9] in this volume).

In order to define *multi-resolution subdivision surfaces*, at each step of the algorithm, the surface subdivision rule is used first to compute an initial estimate for the position of a vertex, and then a displacement vector is added, which is stored separately. Clearly, the multi-resolution surfaces have higher quality as the displacement vectors improve the geometry of the original subdivision mesh. On the other hand, they are more memory intensive, requiring the storage of displacement vectors. Moreover, multi-resolution surfaces can be used to model a given set of data but not for creating a shape from scratch in a free-form design application. The latter would require a tedious input of displacement vectors.

Nevertheless, multi-resolution surfaces retain several of the good properties of subdivision surfaces. Most importantly, they allow the user to change the resolution of an object and represent it at a coarser or finer level. A typical application using this property is the real time rendering of large scenes, where an object near the viewer is represented in detail to increase the visual quality of the rendering, while an object farther away is represented at a coarser resolution to save rendering time. Another application is editing, where the user can either use a coarse resolution to make large scale changes at the shape or work with finer resolution and edit the detail as in [6]. In [71] multi-resolution surfaces are used as a basis for multi-scale operations which perform local and global deformations able to merge models with different shape and textural characteristics.

Another application of multi-resolution surfaces is shape compression. Indeed, the two steps of a computation of the position of a point can also be interpreted as a form of predictive mesh encoding. That is, the subdivision rule predicts the position of a vertex while the displacement vector stores the error of the prediction. For the shapes commonly found in practical applications, the error of the prediction is low and the displacement vectors can be compressed efficiently. If we can tolerate some more error, the displacement vector can be substituted by a scalar representing the length of the projection of the vector on the vertex normal. This latter method works because most of the error lies in the direction of the vertex normal, being the displacement vector usually almost parallel to it [36].

5 Constraint-based subdivision surfaces

To fulfil the need of various interpolation constraints, fast multi-level finite element solution strategies for subdivision meshes have been designed, exploiting the natural

hierarchy of subdivision. These attempts are only a natural step towards the adaptation of subdivision in these applications, but there are many open problems that still have to be addressed. In this section, we face some of these problems and describe possible solutions. While passing, we outline some of the remaining challenging issues. For a deeper analysis, a complete taxonomy of interpolation conditions for both subdivision curves and surfaces may be found in [51, 52].

5.1 Point interpolation

In subdivision surfaces, interpolation of vertices has so far been restricted to

1. Vertices that are part of the defining polyhedron of the limit surface, or
2. Vertices to which a corresponding polyhedron is constructed whose limit surface interpolates the given vertices.

In either case, for each given vertex v_i to be interpolated, a new vertex w_i is defined by a linear combination of some of the vertices v_i, or the ones obtained from one level of refinement. A system of linear equations is then obtained whose solution gives the vertices of the constructed polyhedron. Note that the topology of the constructed vertices is the same as the given polyhedron, or the one formed by the given set of vertices.

Such interpolation is not enough for the CAD industry. Subdivision surfaces should be able to interpolate arbitrary vertices that are not necessarily part of the given polyhedron. Furthermore, the topology of the vertices to be interpolated should not be a requirement for subdivision surfaces. Qin et al. [60] introduced dynamic Catmull-Clark subdivision surfaces where a physical-based approach is coupled with subdivision to locally deform an initial surface towards some point constraints. The limits here are typical of the physical-based models: the deformation cannot be controlled both in shape and in size.

5.2 Interpolation with normal constraints

Interpolation of vertices with given normal is also a major requirement for many of subdivision-based applications. The initial attempt for generating subdivision surfaces with such constraints was reported in [48] for Doo-Sabin surfaces which was later extended to Catmull-Clark in [7].

Again, based on the restrictions imposed in Sect. 5.1, normals are only interpolated on vertices of the control polyhedron and the issue of interpolating normals at *arbitrary* points remains to be addressed.

5.3 Interpolation of isolated curves

The next interpolation constraint often required in subdivision-based applications is the curve constraint, in particular where shape is prescriptive. There are two cases to be considered depending on the nature of the cross derivative needed. If only C^0 is

required the interpolated curve is then called a *crease*. This can be achieved by treating the curve as a boundary curve where two pieces may join with C^0 as suggested in [46] or by modifying the subdivision coefficient on either side of the tagged control polygon as described in [24]. The previous work has been extended in [15], permitting to obtain semi-sharp edges, useful for modelling fillets and blends. They used subdivision surfaces in the animation field, modifying Catmull-Clark scheme by introducing an edge sharpness parameter, whose values vary from zero (completely smooth) to infinite (completely sharp) through an interpolation function.

Direct applications of curve interpolation are feature curves or even lofted subdivision surfaces [50, 11, 10, 53]. One major approach for interpolating curves by subdivision surfaces is the polygonal complex approach [47]. Under subdivision, a polygonal complex P generates a curve C. The idea is that, by embedding such a complex in the polyhedron defining a subdivision surface, the curve C will automatically be interpolated by the limit surface. Typically, the topology of a polygonal complex depends on the subdivision scheme to be used. In Catmull-Clark setting, a polygonal complex can be defined by three rows of vertices (t_i), (m_i), and (b_i), not necessarily of equal size.

To be useful, the limit curve C of a complex should be identified. For Catmull-Clark, two cases can be considered depending on whether the rows have the same number of vertices or not. In the former case, the curve C is simply the cubic uniform B-spline whose control polygon is give by:

$$\frac{1}{8}\begin{pmatrix}1 & 6 & 1\end{pmatrix} \times \begin{pmatrix} t_0 & t_1 & .. & t_{n-1} \\ m_0 & m_1 & .. & m_{n-1} \\ b_0 & b_1 & .. & b_{n-1} \end{pmatrix}.$$

If the rows have different number of vertices, the limit is also a uniform B-spline curve obtained by using the vertices of first refinement of P. Since after one step of Catmull-Clark all faces become quads, the refined mesh of a Camull-Clark polygonal complex will have equal number of vertices. As such, a similar equation to 5 can be defined. Figure 15 shows an example of such a complex.

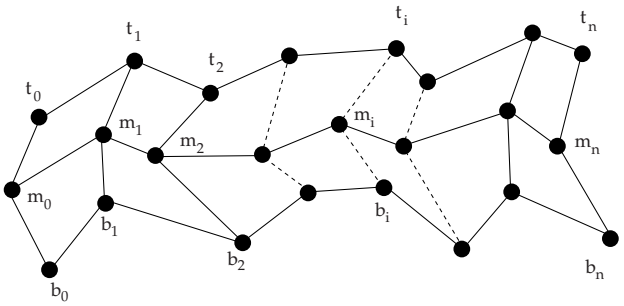


Fig. 15. A basic Catmull-Clark polygonal complex.

A straightforward application of this property is to constrain a subdivision surface to interpolate the uniform B-spline curve defined by a tagged control polygon on its polyhedron. One solution can be found by building a polygonal complex from the faces sharing the edges of the tagged control polygon. However, the limit curve of this complex will not lie on the surface. To achieve that, a polygonal complex can be constrained to have its limit curve defined by its middle control polygon (m_i) which will play the role of the tagged control polygon. This can be done by computing a new middle row $\hat{m}_i$ using the following equation:

$$\begin{pmatrix} \hat{m}_0 \\ \hat{m}_1 \\ . \\ . \\ \hat{m}_{n-1} \end{pmatrix}^T = \frac{1}{4}\begin{pmatrix} -1 & 6 & -1 \end{pmatrix} \times \begin{pmatrix} t_0 & t_1 & .. & t_{n-1} \\ m_0 & m_1 & .. & m_{n-1} \\ b_0 & b_1 & .. & b_{n-1} \end{pmatrix}.$$

To interpolate the uniform cubic B-spline curve of a tagged control polygon on a polyhedron, we simply reposition the tagged vertices by the formula proposed above. It can be noted that if the faces of the complex obtained from the tagged edges are not 4-sided, then one refinement is needed before we apply Eq. 5.

The limitation of this approach is that only uniform curves are interpolated, which is not adequate for subdivision-based applications. Moreover, it addresses only the creation of new surfaces, while editing existing shapes cannot be faced.

Techniques for editing curve constraints applicable both in the creation and in the manipulation phase have been devised [39], based on the concept of combined subdivision schemes, which include local samples of the desired curve as subdivision control points [38]. This approach permits to fit curve constraints also at the boundary of the surface, performing also trimming operations [40]: in fact, the trim curve is curved as a boundary of a new subdivision surface; conversely, a hole can be filled with the same philosophy.

Alternative approaches to curve-driven surface modification are followed in [29], and, more accurately, in [6]. In both cases the constraint line is drawn by the user arbitrarily onto the subdivision surface itself, but only a displacement operation is performed on the points localised on the mesh. In a multi-resolution framework, the first approach proposes a mesh editing technique which does not eventually provide a pure multi-resolution surface; on the contrary, the second one includes the curve constraints reparameterising the subdivision surface: in this way, sharp features (as well as trim curves) are placed within the multi-resolution model, permitting to further consistent manipulation.

An evolution of the latter work can be found in [5], where the authors, starting from the method proposed in [6], solve the problem of pasting a given portion of surface on another one. The area to paste is parameterised through a spine and distances from the boundary: with a proper projection, the area is mapped and blended on the second surface.

5.4 Interpolation of isolated curves with cross derivative

The polygonal complex defining an interpolated curve carries with its cross derivatives information such as tangent plan and cross curvature. A challenging problem is to construct a polygonal complex to interpolate a given curve with predefined tangent plane and/or cross curvature (see Figure 16, and also Figure CP-3 in Appendix D). The interpolated curve can thus be used as a feature curve on the underlying subdivision surface as suggested in [53].

Certainly it is hard to define the curvature at any point on the curve in terms of the control vertices of the polyhedron. A possible solution is using a 2D polygon to define a running feature on a subdivision surface, which can be seen as an interface to control the cross curvature of the interpolated curve [11, 10]. The idea is then to reposition the vertices of the polyhedron defining a subdivision surface in the vicinity of a tagged control path. The intention of the work developed in [10] is capturing the semantics of the styling activity through the concept of styling features described and applied to subdivision surfaces. Since a curve-oriented design methodology fulfils designers' attitude to sketch, features obtainable by means of generalised sweep operations (named *sweep-like features*) have been defined and treated. The novelty here is a proper support of the stylists' creativity using subdivision surfaces as underlying geometric representation. Such an approach can lead to different behaviours of the feature along the interpolated curve such as those shown in Figure 17 (see also Figure CP-4 in Appendix D).

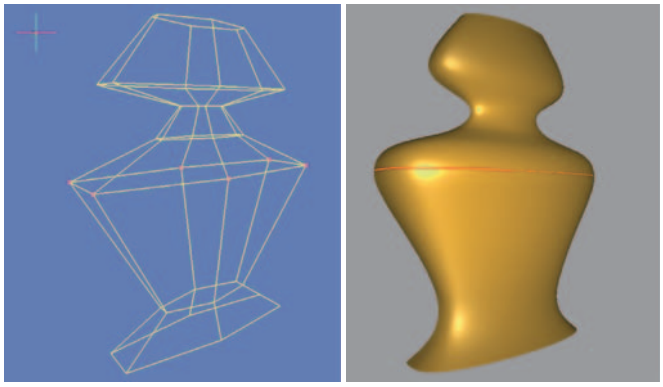


Fig. 16. One curve interpolated with predefined cross curvature (With kind permission of Springer Science and Business Media).

5.5 Lofted subdivision surfaces

The natural flow of constraints is the generation of lofted subdivision surfaces. The constraint here is a set of cross section curves defined in terms of B-spline control

Fig. 17. Insertion of different Sweep-like features on a car model.

vertices. What is needed is a subdivision surface that interpolates the given curves. In [50] a polygonal complex is constructed for each of the given curves; such complexes are then merged into a polyhedron that can be passed to a subdivision kernel which simply does not see the interpolation constraints and generates a limit surface interpolating the given curves. Figure 18 shows an example of such a surface with its 14 limit curves.

Fig. 18. A lofted surface from 14 given curves.

Building the control polyhedron after constructing the polygonal complexes remains an open problem. A heuristic approach was used in [50] based on short distance distribution between complex boundaries.

5.6 Interpolation of a net of curves

One further constraint for subdivision surfaces that was introduced in [47], and later considered in [37, 66, 65] is the generation of surfaces interpolating a net of curves. To solve this problem, a solution to the interpolation of multiple intersecting curves through an extraordinary point was needed. In the combined scheme proposed in [37] no more than two intersecting curves can be interpolated, which is not adequate in many applications. The polygonal complex approach was extended to interpolate an unlimited number of curves through an extraordinary point using the Doo-Sabin scheme [49].

This was recently extended to Catmull-Clark surfaces in [1] using the notion of *X-configuration*. An X-configuration is composed of an even number of quads, all adjacent around the same extraordinary point, see Figure 19. Multiple intersecting curves can then be interpolated using an X-complex where a group of two or more Catmull-Clark polygonal complexes can be connected to a common X-configuration. In this way, an X-configuration can be visualised as a docking station where the complexes may be connected to any of its available ports. Figure 19 shows one of these complexes connected to an X-configuration, and a surface interpolating five cubic B-spline curves through an extraordinary point.

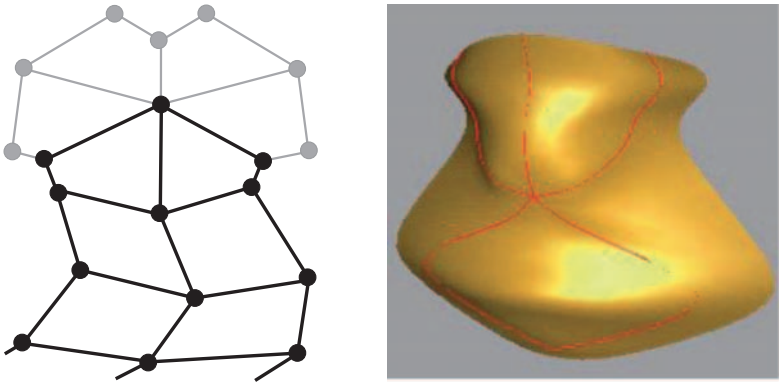


Fig. 19. Left: an X Configuration with one complex attached to it. Right: A surface interpolating five intersecting curves.(images from [1], ©2005 IEEE)

In all approaches above the surface is not C^2 through the extraordinary point, but some sort of bounded curvature should be possible.

5.7 Non-uniform subdivision surfaces

Most subdivision-based applications concentrated on the uniform based subdivision. Future work should emphasise the non-uniform subdivision in order to compete with NURBS. Along this direction, we should note the NURCCS (Non Uniform Catmull-Clark) scheme [67] which is a stationary scheme compared to the initial non-stationary NURSS suggested in [68]. Constrained based non-uniform subdivision is not fully explored. For example, the notion of polygonal complexes can be extended to interpolate non-uniform curves as suggested in [53].

Assume that the vertices (a_i) correspond to a knot u_{i-1}, (b_i) to u_i and (c_i) to u_{i+1}, the following equation can then be used:

$$V = \begin{pmatrix} f_1 & f_2 & f_3 \end{pmatrix} \times \begin{pmatrix} a_0 & a_1 & .. & a_{n-1} \\ b_0 & b_1 & .. & b_{n-1} \\ c_0 & c_1 & .. & c_{n-1} \end{pmatrix}$$

where the f_i coefficients are computed using the polar form of B-splines, see [53] for more details,

$$\begin{aligned} f_1 &= \frac{(u_{i+1} - u_i)^2}{(u_{i+1} - u_{i-2})(u_{i+1} - u_{i-1})} \\ f_2 &= \frac{(u_{i+1} - u_i)(u_i - u_{i-2})}{(u_{i+1} - u_{i-2})(u_{i+1} - u_{i-1})} + \frac{(u_i - u_{i-1})(u_{i+2} - u_i)}{(u_{i+1} - u_{i-1})(u_{i+2} - u_{i-1})} \\ f_3 &= \frac{(u_i - u_{i-1})^2}{(u_{i+1} - u_{i-1})(u_{i+2} - u_{i-1})}. \end{aligned}$$

Figure 20 shows an example of a Non-Uniform Catmull-Clark interpolating a non-uniform B-spline curve. (see also Figure CP-5 in Appendix D)

All other constraints such as normals, and even nets of curves need to be considered as future work.

6 Conclusions

Subdivision surfaces are powerful geometric representation being in-between continuous and discrete ones. Allowing for arbitrary connectivity while guaranteeing smoothness properties, they are typically viewed as an alternative to NURBS. In fact, subdivision surfaces are easier to treat, with the advantage of discarding multi-patch representations. The literature is very wide about manipulation techniques for discrete surfaces, even suitable for CAD applications. The subdivision surface representation inherits all this background, with the further capability of a refinement process. This permits to refine the surface only where necessary and to apply multi-resolution approaches, which optimise the storage of the surface. In addition, some work started on non-uniform configurations.

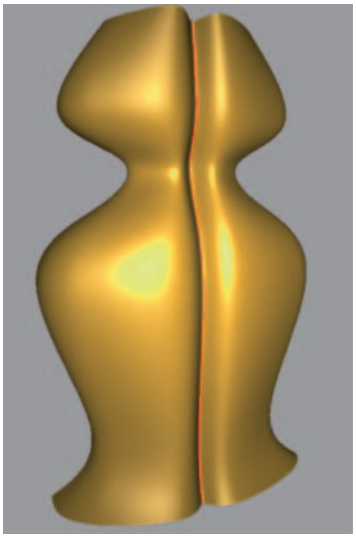


Fig. 20. A non-uniform surface interpolating a non-uniform B-spline curve

As a consequence, animation and 3D effects companies started first to move to subdivision schemes. In more recent times, CAD is following the same strategy [8]. In fact, in addition to discarding multi-patch models, recent literature related to the product development process has been moving towards tessellations in phases where continuous surfaces had been adopted traditionally, such as manufacturing. Considering that different phases of the process usually require discrete surfaces, an ideal unified geometric framework for different phases appear promising and profitable [10]. Conversion and transfer problems might be reduced in integrated modelling, together with time to market and costs.

Among the more promising applications of subdivision surfaces, we can mention free-form deformations, where the surface is embedded in a collection of solids. Subdivision defines a map from each point enclosed in that solid to another point depending on the position of the control points. In this way, intuitive animations can be successfully obtained.

Finally, an application in the bio-medical area has been proposed [28], where subdivision surfaces have been used to generate an atlas of a mouse brain. The surface is overlayed on an image of the brain to generate a multi-resolution parameterisation which permits the brain to be partitioned into key anatomical regions. A geometric database collected the models to allow biologists to organise and search gene expression data.

Acknowledgement. A grant from the American University of Beirut Research Board, 2005-2006, supported Ahmad Nasri

References

1. Abbas A and A. Nasri. Interpolating meshes of curves by catmull-clark subdivision surfaces with a shape parameter. In *The Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05)*, CAD-CG, pages 107–112, Hong Kong, 2005. IEEE Press.
2. N. Alkalai and N. Dyn. Optimizing 3D triangulations: Improving the initial triangulation for the butterfly subdivision scheme. In Neil Dodgson, Michael Floater, and Malcom Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 231–244. Springer, 2004.
3. A. A. Ball and D. J. T. Storry. Conditions for tangent plane continuity over recursively generated B-spline surfaces. *ACM Transactions on Graphics*, 7(2):83–102, 1988.
4. L. Barthe and L. Kobbelt. Subdivision scheme tuning around extraordinary vertices. *Computer Aided Geometric Design*, 21(6):561–583, 2004.
5. H. Biermann, I. Boier-Martin, F. Bernardini, and D. Zorin. Cut-and-paste editing of multiresolution surfaces. In *Proc. of SIGGRAPH*, pages 312–321, 2002.
6. H. Biermann, I. Boier-Martin, D. Zorin, and F. Bernardini. Sharp features on multiresolution subdivision surfaces. In *Conference Proceedings of Pacific Graphics*, 2001.
7. H. Biermann, A. Levin, and D. Zorin. Piecewise smooth subdivision surfaces with normal control. In *SIGGRAPH 00 Conference Proceedings*, pages 113–120, 2000.
8. I. Boier-Martin and F. Bernardini. Subdivision-base representations for surface styling and design. In *DIMACS Workshop on Computer Aided Design and Manufacturing*, October 2003. DIMACS Center, Rutgers University, Piscataway, New Jersey.
9. G.P. Bonneau, G. Elber, S. Hahmann, and B. Sauvage. Multiresolution analysis. In L. De Floriani and M. Spagnuolo, editors, *Shape Analysis and Structuring*. Springer, 2007.
10. C.E. Catalano. *Feature Based Methods for Free Form Surface Manipulation in Aesthetic Engineering*. PhD thesis, Genova, 2004.
11. C.E. Catalano. Introducing design intent in discrete surface modelling. *International Journal of Computer Applications in Technology (IJCAT)*, 23(2/3/4):108–119, 2005. Special Issue on Models and methods for representing and processing shape semantics.
12. E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10:350–355, 1978.
13. F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: a new paradigm for thin-shell finite-element analysis. *Numerical Methods in Engineering*, 47(12):2039–72, April 2000.
14. I. Daubechies, I. Guskov, and W. Sweldens. Regularity of irregular subdivision. *Constructive Approximation*, 15(3):381–426, 1999.
15. T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *SIGGRAPH 98 Conference Proceedings*, pages 85–94, 1998.
16. D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10:356–360, 1978.
17. N. Dyn. Subdivision schemes in Computer-Aided Geometric Design. In W. Light, editor, *Advances in numerical analysis*, volume 2, chapter 2, pages 36–104. Clarendon Press, 1992.
18. N. Dyn. Analysis of convergence and smoothness by the formalism of Laurent polynomials. In A. Iske, E. Quak, and M. S. Floater, editors, *Tutorials on Multiresolution in Geometric Modelling*, chapter 3, pages 51–68. Springer, 2002.
19. N. Dyn and D. Levin. Subdivision schemes in geometric modelling. *Acta Numerica*, 11:73–144, 2002.
20. N. Dyn, D. Levin, and J. A. Gregory. A 4-point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design*, 4:257–268, 1987.

21. N. Dyn, D. Levin, and J. A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, 1990.
22. G. Farin, J. Hoschek, and M. Kim, editors. *Handbook of Computer Aided Geometric Design*. Elsevier, 2002.
23. S. Green and G. Turkiyyah. Second order accurate constraints for subdivision finite elements. *Numerical Methods in Engineering*, 60(13), 2004.
24. H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. *Computer Graphics*, 28:295–302, 1994.
25. I. Ivrissimtzis, N. Dodgson, and M. Sabin. $\sqrt{5}$-subdivision. In N. Dodgson, M. Floater, and M. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 285–300. Springer, 2004.
26. I. Ivrissimtzis, M. Sabin, and N. Dodgson. On the support of recursive subdivision. *ACM Transactions on Graphics*, 23(4):1043–1060, 2004.
27. I. Ivrissimtzis and H.-P. Seidel. Evolutions of polygons in the study of subdivision surfaces. *Computing*, 72(1-2):93–104, 2004.
28. T. Ju, J. Warren, G. Eichele, C. Thaller, W. Chiu, and J. Carson. A geometric database for gene expression data. In *Eurographics Symposium on Geometric Processing*. L. Kobbelt, P. Shroeder, H. Hoppe (Editors))p, 2003.
29. A. Khodakovsky and P. Schröder. Fine level feature editing for subdivision surfaces. In *Proc. ACM Solid Modeling*, pages 203–211, 1999.
30. L. Kobbelt. Interpolatory subdivision on open quadrilateral nets with arbitrary topology. *Computer Graphics Forum*, 15(3):409–420, 1996.
31. L. Kobbelt. $\sqrt{3}$ subdivision. In *SIGGRAPH 00, Conference Proceedings*, pages 103–112, 2000.
32. L. Kobbelt, M. Botsch, K. Kaehler, C. Rössl, R. Schneider, and J. Vorsatz. Geometric modeling based on polygonal meshes. *Tutorial T4, Eurographics 2000*, 2000.
33. L. Kobbelt and P. Schröder. A multiresolution framework for variational subdivision. *ACM Trans. on Graph.*, 17(4):209–237, 1998.
34. J. Kuragano, H. Suzuki, and F. Kimura. Generation of NC tool path for subdivision surfaces. In *Proceedings of CAD/Graphics' 2001, Kunming China*, pages 676–682, 2001.
35. U. Labsik and G. Greiner. Interpolatory $\sqrt{3}$-subdivision. *Computer Graphics Forum*, 19(3):131–138, 2000.
36. A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 85–94, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.
37. A. Levin. Interpolating nets of curves by smooth subdivision surfaces. In *Computer Graphics Proceedings, ACM SIGGRAPH 1999*, pages 57–64, 1999.
38. A. Levin. Combined subdivision schemes. *PhD thesis, School of Mathematical Science, Tel Aviv University*, 2000.
39. N. Litke, A. Levin, and P. Schröder. Fitting subdivision surfaces. *IEEE Visualization*, pages 319–324, October 1998.
40. N. Litke, A. Levin, and P. Schröder. Trimming for subdivision surfaces. *Computer Aided Geometric Design*, 18(5):463–481, June 1998.
41. C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.
42. C. Loop. Bounded curvature triangle mesh subdivision with the convex hull property. *The Visual Computer Journal*, 18(5-6):316–325, 2002.

43. C. Loop. Smooth ternary subdivision of triangle meshes. In *Proceedings of the 5th Conference on Curves and Surfaces*, pages 295–302. Nashboro Press, 2003.
44. W. Ma, X. Ma, S. Tso, and Z. Pan. Subdivision surface fitting from a dense triangular mesh. In *Proc. of Geometric Modeling and Processing*, pages 94–103, July 1999.
45. M. Marinov, N. Dyn, and D. Levin. Geometrically controlled 4-point interpolatory schemes. In N. Dodgson, M. Floater, and M. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 301–315. Springer, 2004.
46. A. Nasri. Polyhedral subdivision methods for free-form surfaces. *ACM Transactions on Graphics*, 6(1):29–73, 1987.
47. A. Nasri. Recursive subdivision of polygonal complexes and its applications in CAGD. *Computer Aided Geometric Design*, 17:595–619, 2000. Presented also at The $5th$ Siam Conferene On Geometric Design, Nashville, 1997.
48. A. Nasri. Constructing polygonal complexes with shape handles for curve interpolation by subdivision surfaces. *Computer Aided Design*, 33:753–765, 2001.
49. A. Nasri. Interpolating an unlimited number of curves meeting at extraordinary points on subdivision surfaces. *Computer Graphics Forum*, 22(1):87–97, 2003.
50. A. Nasri, Abbas A, and I. Hasbini. Skinning Catmull-Clark subdivision surfaces with incompatible cross-sectional curves. In *Pacific Graphics 2003*, pages 102–111, Canmore, Canada, 2003. IEEE Press. ISBN 0-7695-2028-6.
51. A. Nasri and M. Sabin. Taxonomy of interpolation conditions in recursive subdivision curves. *The Visual Computer*, 18(4):259–272, 2002.
52. A. Nasri and M. Sabin. Taxonomy of interpolation conditions in recursive subdivision surfaces. *Journal Visual Computer*, 18(6):382–403, 2002.
53. A. Nasri, M. Sabin, R. Abu Zaki, N. Nassiri, and R. Santina. Feature curves with cross curvature control on Catmull-Clark subdivision surfaces. volume 4035 of *Lecture Notes in Computer Science*, pages 761–768. Springer, 2006.
54. Y. Ohtake. Mesh Viewer. http://www.mpi-sb.mpg.de/∼ohtake/software. Last access October 2006.
55. P. Oswald and P. Schröder. Composite primal/dual $sqrt(3)$-subdivision schemes. *CAGD*, 20(3):135–164, 2003.
56. H.R. Pakdel and F.F. Samavati. Incremental Catmull-Clark subdivision. In *5th International Conference on 3-D Digital Imaging and Modeling*, pages 95–102, Canada, June 2005. IEEE Computer Society Press.
57. J. Peters and U. Reif. The simplest subdivision scheme for smoothing polyhedra. *ACM Transactions on Graphics*, 16(4):420–431, 1997.
58. J. Peters and U. Reif. Analysis of algorithms generalizing B-spline subdivision. *SIAM Journal on Numerical Analysis*, 35(2):728–748, 1998.
59. H. Prautzsch. Smoothness of subdivision surfaces at extraordinary points. *Advances in Computational Mathematics*, 9(3-4):377–389, 1998.
60. H. Qin, C. Mandal, and B. C. Vemuri. Dynamic Catmull-Clark subdivision surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):216–229, 1998.
61. U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design*, 12(2):153–174, 1995.
62. U. Reif and P. Schröder. Curvature integrability of subdivision surfaces. *Advances in Computational Mathematics*, 14(2):157–174, 2000.
63. M. Sabin and L. Barthe. Artifacts in recursive subdivision surfaces. In *Curve and Surface Fitting*, pages 353–362. Nashboro Press, 2003.
64. M. Sabin, N. Dodgson, M. Hassan, and I. Ivrissimtzis. Curvature behaviours at extraordinary points of subdivision surfaces. *Computer Aided Design*, 35(11):1047–1051, 2003.

65. S. Schaefer, D. Zorin, and J. Warren. Lofting curve networks with subdivision surfaces. In *Proceedings of Eurographics Symposium on Graphics Processing*, pages 105–116, 2004.
66. J. Schweitzer. *Analysis And Applications of Subdivision Surfaces*. PhD thesis, The University of Washington, 1991.
67. T. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-Splines and T-NURCCs. *ACM Transaction on Graphics*, 22(3):477–484, 2003. ACM SIGGRAPH 2003, ACM Press.
68. T. W. Sederberg, J. Zheng, D. Sewell, and M. Sabin. Non-uniform subdivision surfaces. In *ACM Siggraph 1998*, volume 17, pages 387–394, 1998.
69. J. Stam. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH 98 Conference Proceedings*, pages 395–404, 1998.
70. J. Stam and C. Loop. Quad/triangle subdivision. *Computer Graphics Forum*, 22(1):79–85, 2003.
71. L. Velho, K. Perlin, L. Ying, and H. Biermann. Procedural shape synthesis on subdivision surfaces. In *SIBGRAPI 2001*, 2001.
72. J. Warren and H. Weimer. *Subdivision Methods for Geometric Design*. Morgan Kaufmann, 2001.
73. D. Zorin and D. Kristjansson. Evaluation of piecewise smooth subdivision surfaces. *The Visual Computer Journal*, 18(5-6):299–315, 2002.
74. D. Zorin, P. Schröder, A. DeRose, L. Kobbelt, A. Levin, and W. Sweldens. SIGGRAPH 00 Course Notes, Subdivision for modeling and animation, 2000.
75. D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *SIGGRAPH 96 Conference Proceedings*, pages 189–192, 1996.

Skeletal Structures

Silvia Biasotti[1], Dominique Attali[2], Jean-Daniel Boissonnat[3], Herbert Edelsbrunner[4], Gershon Elber[5], Michela Mortara[1], Gabriella Sanniti di Baja[6], Michela Spagnuolo[1], Mirela Tanase[7], and Remco Veltkamp[7]

[1] CNR. - Ist. di Matematica Applicata e Tecnologie Informatiche, Genova, Italy. silvia.biasotti,michela.mortara, michela.spagnuolo@ge.imati.cnr.it
[2] LIS-CNRS, Domaine Universitaire, BP 46, 38402 Saint Martin d'Hères, France. Dominique.Attali@lis.inpg.fr
[3] INRIA, 2004 Route des Lucioles, BP 93, 06904 Sophia-Antipolis, France. Jean-Daniel.Boissonnat@sophia.inria.fr
[4] Department of Computer Science, Duke University, Durham, and Raindrop Geomagic, Research Triangle Park, North Carolina, USA. edels@cs.duke.edu
[5] Technion, Israel Institute of Technology, Israel. gershon@cs.technion.ac.il
[6] CNR - Ist. di Cibernetica "E. Caianello", Pozzuoli, Napoli, Italy. gsdb@imagm.cib.na.cnr.it
[7] Universiteit Utrecht (UU), The Netherlands. remco.veltkamp@cs.uu.nl

Shape Descriptors are compact and expressive representations of objects suitable for solving problems like recognition, classification, or retrieval of shapes, tasks that are computationally expensive if performed on huge data sets. *Skeletal structures* are a particular class of shape descriptors, which attempt to quantify shapes in ways that agree with human intuition. In fact, they represent the essential structure of objects and the way basic components connect to form a whole.

In the large amount of literature devoted to a wide variety of skeletal structures, this Chapter provides a concise and non-exhaustive introduction to the subject: indeed the first structural descriptor, the medial axis, dates back to 1967, which means forty years of literature on the topic.

1 Introduction

The main issue in high-level structuring is to extract an abstract description of the shape that can be more useful for many purposes. For instance, the search in a data base for an object similar to a query shape can be nearly impossible if approached comparing bulks of thousand triangles. Conversely, the process is extremely facilitated when two descriptors of the shapes are compared instead. Of course the performance and the quality of results depends on the conciseness and on the expressiveness of the description. A shape descriptor may be any number, property or function that can be used to discriminate between shapes. For instance, the edge number can be used to classify polygons. Depending on the application tasks and on the shape domain, usually more sophisticated descriptors are needed. In this Chapter we introduce and describe a particular class of shape descriptors, i.e., skeletal structures.

As everybody knows, the word "skeleton" generally indicates the bone structure of vertebrates; in general, skeleton recalls a support structure (e.g., the skeleton of a ship), or the scheme of something (the skeleton of an opera). Translating the concept in the digital context is not straightforward. Intuitively, the skeleton can be defined following two different philosophies: one privileges the aspect of the skeleton of being a *medial structure*, i.e., an entity that always falls inside the shape and is in each point equidistant from the shape boundary. From this point of view, the skeleton of a planar shape is a linear graph, and each point on the skeleton is equidistant from the boundary points of the shape. In the $3D$ space things change: a cylinder with circular base sufficiently far from the bases exhibits a linear skeleton, while the skeleton of an elongated box is conversely a medial surface, i.e., a two dimensional sheet, which extents in the longitudinal direction.

From the other point of view, the skeleton can be regarded as the explicit representation of how the basic components of the shape are glued together to form a whole. A strictly tubular shape has normally one skeletal line, which lays medial to the object and acts as a symmetry axis, usually referred to as a *centreline*. Furthermore, complex objects formed by the arrangement of tubular-like components can be abstracted to a collection of centrelines which split and join, following the object topology.

The definition of skeleton as a medial structure privileges the geometric aspect of the descriptor. Therefore the skeleton retains a strong correspondence with the shape, so that the boundary can be exactly reconstructed, or at least approximated, from the information encoded in the skeletal structure.

Conversely, the second paradigm regards the skeleton as an abstract adjacency graph of salient shape features and relies on shape decomposition in a way that agree with human intuition: recent cognitive research, alongside with new developments in digital imaging and computer vision, has led to a growing consensus that decomposition of shapes into their constituent parts is fundamental to human vision as an early stage of the cognitive process.

Following the previous considerations, a unique formal definition of skeleton in the context of *digital shapes*, i.e., n-dimensional data having a visual representation, can not be given. In this Chapter we will distinguish between *geometric skeletons*, like the medial axis transform, which give a richer encoding of the spatial extent of the shape, and *topological skeletons*, that dismiss some geometric information but make explicit higher level properties of the shape (main features, adjacency relations among parts, number of components, holes, ...).

The choice on which descriptor should be preferred relies on the application context it must cope with. In a variety of applications it is desirable for the skeleton to be linear (e.g., in medical imaging for vascular narrowing detection, in computer-aided screening for early detection of polyps, and so on). Conversely, other applications may require that the skeleton retains a full correspondence with the shape geometry. This is the case of many CAD/CAM applications, where medial surfaces are exploited, for instance, for subdivision of complex solids into simpler pieces for automatic mesh generation and also for the generation of simpler idealised models such as shells and beams for stress analysis.

1.1 Overview

An exhaustive review of the existing literature on skeletal structures would require an effort which is beyond the scope of the Chapter. The goal here is to provide a selection of the methods that are more relevant for subsequent applications in shape modelling. For the classes of methods reviewed we will provide basic definitions and an overview of the structure with respect to different discrete settings. Comparative remarks and examples of their applications will also

be given. The presentation is organized into two main classes: *geometric skeletons* including the medial axis and other medial structures like bisectors, and *skeletons derived from topological structures*, possibly enriched by geometric information to retain a strong correspondence to the shape; the Reeb graph belongs to this category.

Maybe the best known of geometric skeletal descriptors is the *Medial Axis Transform, (MAT)* defined by Blum in the sixties [21]; he first described the medial axis extraction for a 2D shape by analogy with a fire front which starts at the boundary of the shape and propagates isotropically towards the interior. The medial axis is defined by the locations at which the fire fronts collide.

In the planar case the medial axis is a graph, while for shapes in $\mathbb{R}^3$ the MAT is a dimensionally heterogeneous entity composed by curves and surface patches. Small modifications of the input shape can induce large modifications of its medial axis; nonetheless they do not affect the entire medial axis. Typical effects for shapes in $\mathbb{R}^2$ are spurious branches that leave the rest of the medial axis unchanged.

The exact computation of the medial axis is extremely complex in the domain of freeform shapes. Nonetheless, results exist for computing *bisectors* between rational entities exactly. The concept of bisector is strictly related to the medial axis, but while the medial axis can be computed for a given object, the bisector involves more entities, being the locus of points equidistant from two (or more) shapes. We present this approach since bisectors can be effectively used as primitives to construct the MAT and the Voronoi diagram of rational curves (see Section 3.2).

Conversely, many approaches have been adopted to implement Blum's original definition in the discrete case. Basically, we can distinguish them into four categories, depending on the adopted skeletonisation method: *skeleton extraction from Voronoi diagrams; simulation of the grassfire; topological thinning; skeleton extraction from distance maps.* The medial axis of a planar curve can be thought of as the Voronoi diagram generalized to an infinite set of points (the boundary points) [3, 83, 84]. It has been formally shown [28] that the Voronoi diagram becomes an increasingly precise approximation of the continuous medial axis as the density of boundary samples increases. Algorithms which actually try to implement the grassfire process are quite rare; examples are the straight skeleton, first introduced by [1], and the linear axis [97]. Thinning and distance map computation can be directly applied to volumetric discrete representations that are widely used especially in medical applications: most acquisition techniques produce in fact voxel grids, like the Computed Tomography or the Magnetic Resonance Imaging. All these skeletonisation methods are detailed in Section 4.

Concerning topological structures, the *Reeb graph* was defined much before the MAT [89], but its potential in shape description has been understood and formalized later on [94]. Reeb graphs act as a tool for studying shapes through the evolution and the arrangement of the level sets of a real function defined over the shape. This fact relies to Morse theory, [80], that studies the link between the differential properties of a shape and its algebraic topology (in the sense of the number of connected components, number and type of holes, etc.). From this point of view, an object can be partitioned into protrusions, holes and other characteristics and can be efficiently represented as a collection of features with a set of adjacency relations between them. These facts raise the idea that topology-based descriptors, maybe integrated with geometric information, are suitable for dealing with the definition of basic models to represent, generate and manipulate shapes without forgetting the feasibility and the computational complexity of the problem, [15]. In fact, a recent work by Goswami et al. [68] exploits topological structures to locate flat and tubular shaped regions on 3D shapes. Focusing on the level set evolution, we obtain a discrete description which effectively represents the shape

and can be encoded in a topological graph. Some methods follow this paradigm and compute skeletons joining the barycentres of adjacent sections [81, 76, 72].

The remainder of this Chapter is organized as follows: the MAT and geometric skeletons are treated first. In Section 2 the definitions of the main concepts are given of medial axis, Voronoi diagram, shock graphs and bisectors, while the Reeb Graph definition is shifted to the topological skeleton Section (5) for a better reading. Techniques that construct an exact representation of medial structures for particular classes of shapes are detailed in Section 3, while approximated methods are described in Section 4. Skeleton derived from topological structures including the Reeb graph are presented in Section 5. Finally, some concluding remarks and future developments are given in Section 6.

2 Definitions of geometric medial structures

In this Section the concepts of medial axis transform, shock graphs, Voronoi diagrams and bisectors are introduced. All these entities share the property of being *medial* with respect to the shape boundary (medial axis and shock graphs) or to two or more objects (Voronoi diagrams, bisectors); therefore they can be referred to as *medial structures*.

The *medial axis transform (MAT)* has been introduced by Blum [21] as a tool in image analysis. To get an intuitive feeling for this concept, consider starting a grass fire along a curve in the plane. The fire starts at the same time, everywhere along the curve, and it grows at constant speed in every direction. The medial axis is the set of locations where the front of the fire meets itself. Formally, let X be a bounded open subset of the Euclidean k-dimensional space, $\mathbb{R}^k$. The *medial axis*, $\mathcal{M}[X]$, is the set of points that have at least two closest points in the complement of X [78], see Figure 1.

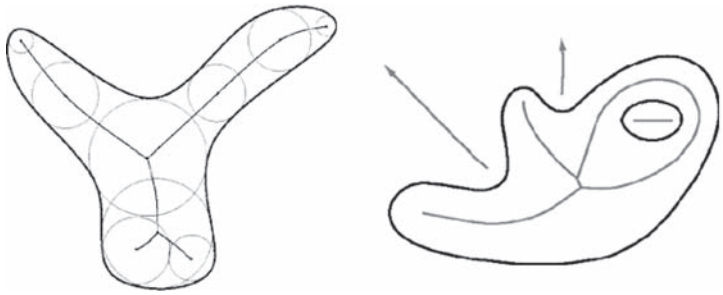


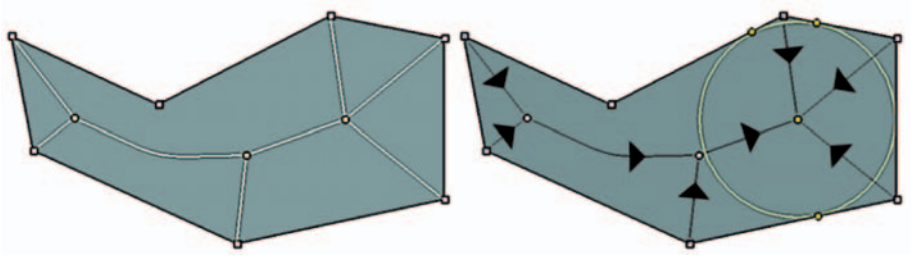
Fig. 1. Medial axis of two planar shapes. In the second example the medial axis is shown also for the external part of the shape.

The medial axis of a shape captures its connectivity, ignoring local dimensionality. More precisely, a shape and its medial axis are homotopy equivalent [78, 91, 101]. In $\mathbb{R}^k$, the medial axis has generically dimension $k-1$, one less than the dimension of the space. In the plane, the medial axis is a (one-dimensional) graph whose branches correspond to regions of the shape it represents. The MAT of planar polygons consists of straight lines and parabolic arcs; each convex vertex of the polygon has an edge of the MAT terminating in it. The MAT structure is very sensitive to noise: the insertion of a new vertex in the boundary of the shape will

cause new edges to appear in the skeleton. In $\mathbb{R}^3$, it is composed of pieces of surfaces, and is sometimes called a *medial surface*. When each point x of the medial axis is weighted with the radius $\rho(x)$ of the maximal ball centered at x, then we have enough information to reconstruct the shape. In other words, the medial axis together with the map ρ provides a reversible coding of shapes. This coding is not necessarily minimal and some shapes, such as finite union of balls, can be reconstructed from proper subsets of their weighted medial axes.

Another medial structure is the *shock graph*, [75], which is obtained by viewing the medial axis as the locus of singularities (shocks) generated during the fire front propagation from the shape boundary. This dynamic view of the medial axis associates a direction and an instantaneous speed of flow to each shock point, [67]. In particular, shock points may be classified according to the number of contact points and to the flow direction, as described in [66]: *source* and *sink* points determine the nodes of the graph while the *links* connect source points to sink ones and define the arcs of the graph. In addition, attributes are associated to the shock graph to store both the intrinsic geometry of the portion of shape corresponding to a link and the radius and the flow direction of each node. Analogously to the MAT, the shock graph structure and the corresponding point classification have been extended to $3D$ shapes [67]. Also, in this case the shock graph structure contains dimensionally heterogeneous components and it is not a planar graph.

The medial axis and the shock graph differ for the interpretation of the structure entities rather than for the geometric abstraction they provide. For example, the shock graph and the MAT of a curve have the same arcs and nodes, but the shock graph associates also to each arc the growing direction of the radius of the bi-tangent spheres, see Figure 2(b). In general, we may consider that the shock graph is a finer partition of the medial axis.



(a) (b)

Fig. 2. The medial axis (a) and the shock graph (b) of two simple curves.

Shock graphs are widely used for image matching, recognition and curve alignment, therefore methods proposed in literature mainly address the problem in the bi-dimensional case and the shape is supposed to be a closed curve.

Strictly related to the medial axis is the *Voronoi diagram* . Given a finite set of points S in $\mathbb{R}^k$, for each point p in $\mathbb{R}^k$ there is at least one point in S closest to p; a point p may be equally close to two or more points in S. For each point in S its Voronoi cell is defined as the subset of $\mathbb{R}^k$ of points closest to it than to any other point in S. The union of Voronoi cells of all points in S is a partition of $\mathbb{R}^k$ called Voronoi Diagram corresponding to the set S.

For instance, in the planar case, given two points a and b, the set of points equidistant from a and b is an infinite line l, the perpendicular bisector of the segment joining a and b. l represents the boundary between the two infinite Voronoi cells of a and b (two half-planes).

The concept of Voronoi diagram is much correlated to the MAT: indeed the MAT of a shape can be approximated by the Voronoi diagram of a finite set of boundary points, as

detailed in Section 4.1; on the other hand, while the MAT is the skeleton of a shape, the Voronoi diagram represents a medial structure between two or more entities (points or objects).

Indeed the Voronoi diagram definition can be easily generalized to set of objects: given m different objects O_1, ... O_m, the Voronoi cell of an object O_i, $(1 \leq i \leq m)$ is defined as the set of points that are closer to the object O_i than to any other object Oj $(1 \leq j \leq m))$. The *bisector* of two objects is the locus of points that are equidistant from the two shapes. The Voronoi cell that contains all points in space that are closer to some object than to any other in space is, therefore, formed out of these bisectors. Similarly, the Voronoi diagram and the medial axis transform are also prescribed by subregions of these bisectors. Therefore, bisectors can be seen as building blocks for the MAT and the Voronoi diagram in such cases where a direct computation of these structures is too complex.

3 Exact representation of medial structures

Indeed, the exact MAT computation was considered for long time affordable only for polygons [77, 65], and more recently for polyhedra [92, 42]. Recently, a few researchers have tackled the problem in the context of freeform (piecewise) rational entities.

Today' accepted approach for computing the planar arrangements of freeform geometry approximates the geometry using piecewise lines and arcs, but this method has noteworthy disadvantages. First, the approach is only an approximation. Second, it is also erroneous. The MAT of a planar shape enclosed by two concentric circles is another mean circle in between them. Yet, by tessellating the two input circles into lines, one introduces numerous C^1 discontinuities along these circles. The resulting MAT will consist of numerous and erroneous edges from the mean circle toward all the C^1 discontinuities in the two boundary circles.

Fortunately, methods exist to compute bisectors of rational entities exactly. For these reason, new approaches aim at computing bisectors between basic freeform shapes as building blocks of every Voronoi Diagram or Medial Axis Transform.

Beyond computing the bisectors between points, lines and arcs in the plane, the current state-of-the-art not only provides complete answers on when an analytic bisector exists between rational manifolds in $\mathbb{R}^n$, but also proposes tractable computational schemes to derive it, as described in Section 3.1.

3.1 Bisectors for freeform shapes

In the following, we will restrict our discussion to rational parametric curves and surfaces, only. Since the rational representation is fully capable of representing all the simple primitives common to the Constructive Solid Geometry (CSG) modelling technique, such as cones, cylinders, spheres, and torii, we will focus on this representation. The fundamental question is whether the bisector between two rational manifolds in $\mathbb{R}^n$ is rational, hence retaining a closure that enables the precise representation of the bisector sheet in the same geometric modelling environment.

The building blocks of every Voronoi diagram or medial axis transform computed in the plane or 3-space must include all cases. These include point-point and point-curve bisectors that are rational in both $\mathbb{R}^2$ and $\mathbb{R}^3$, point-surface and curve-curve bisectors that are rational in $\mathbb{R}^3$, and curve-surface and surface-surface bisectors that are not rational in either space. These non rational curve-curve bisectors in the plane must be differently represented or approximated and such approximations are considered in [54, 62]. [54] maps the problem of computing

the bisector between two planar curves $C(t)$ and $C(r)$ to a zero-set-finding problem in the parameter space of the two curves (t, r). In [62], the planar curve-curve bisector problem is reduced to an envelope of a continuum of point-curve bisectors. The rational surface bisector cases in $\mathbb{R}^3$ are considered in [56].

While the bisectors between points, lines, and arcs have been known for thousands of years, the first real step toward support of freeform geometry was made by Farouki [61]. He showed that the bisector between a point and a rational curve in the plane is indeed rational.

Let $C(t) = (c_x(t), c_y(t))$ be a rational plane curve and $P = (p_x, p_y)$ a point in the plane. The planar bisector sheet, $B(t) = (b_x(t), b_y(t))$, could then be characterized as,

$$\begin{aligned} &< B(t) - P, B(t) - P > = < B(t) - C(t), B(t) - C(t) >, \\ &< B(t) - C(t), C'(t) > = 0. \end{aligned} \tag{1}$$

The first constraint above merely states that the distance between the bisector B and point P should equal the distance between the bisector and curve $C(t)$. The second constraint ensures we measure the distance in an orthogonal direction to the curve, or in the normal space of $C(t)$. It is simple to show that the set of Equations (1) is linear in $B(t)$. Hence one can rewrite Equations (1) as,

$$\begin{bmatrix} c_x(t) - p_x & c_y(t) - p_y \\ c'_x(t) & c'_y(t) \end{bmatrix} \begin{bmatrix} b_x \\ b_y \end{bmatrix} \begin{bmatrix} \langle C(t), C(t) \rangle - \langle P, P \rangle \\ < C(t), C'(t) > \end{bmatrix}. \tag{2}$$

Clearly $B(t)$ in Equation (2) has a rational representation, employing the Cramer rule.

The fact that the number of degrees of freedom equals the number of constraints is a strong hint that the point-rational curve in the plane has a rational representation. Generally speaking, the $(n-1)$-manifold bisector between two input manifolds in $\mathbb{R}^n$ must satisfy three sets of constraints:

1. It must be at an equal distance from the two manifolds.
2. It must be in the normal space of the first manifold.
3. It must be in the normal space of the second manifold.

The distance equality 1 is always there and always imposes one constraint. Constraints 2 and 3 depend on the dimensions of the normal spaces of the two input manifolds. Interestingly enough, Constraints 1-3 are all linear in the bisector function. Hence, the number of constraints for bisectors between zero-, one-, and two-manifolds inputs equal (written as equality constraint + first manifold normal space constraints + second manifold normal space constraints) is listed in Table 1.

	Point	Curve	Surface
Point	1=1+0+0	2=1+0+1	3=1+0+2
Curve	2=1+1+0	3=1+1+1	4=1+1+2
Surface	3=1+2+0	4=1+2+1	5=1+2+2

Table 1. Number of constraints in the bisector computations between points, curves and surfaces. Constraints are listed as distance constraint plus orthogonality constraint(s) to first manifold plus orthogonality constraint(s) to second manifold.

Rational solutions exist whenever the number of constraints, as prescribed in Table 1, is less than or equal to the number of degrees of freedom of the bisector, which is always the

same as the dimension of the space. Every case for which the total number of constraints is less than or equal to two has a rational bisector representation in the plane. The point-point and point-curve bisectors are both rational in the plane. Further, every case for which the number of constraints is less than or equal to three has a rational bisector representation in $\mathbb{R}^3$. Consequently, in $\mathbb{R}^3$, one has a rational representation for point-point, point-curve [61], curve-curve [55] and point-surface [56] bisector cases. Interestingly enough, after inspecting Table 1, we can see that the bisector between two curves is not rational in the plane ($\mathbb{R}^2$), yet is rational in all higher dimensional spaces ($\mathbb{R}^n$, $n > 2$); specifically it is rational in $\mathbb{R}^3$. Figure 3 (see also Figure CP-1 in Appendix E) shows two examples of rational curve-curve and point-surface bisectors in $\mathbb{R}^3$.

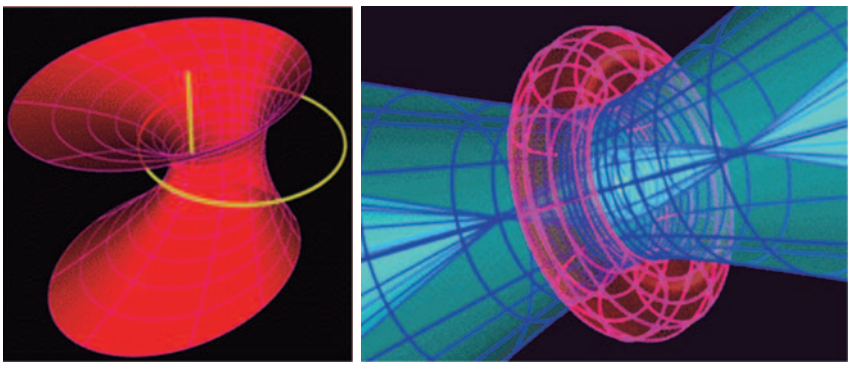


Fig. 3. Curve-curve (left) and point-surface (right) bisector examples in $\mathbb{R}^3$. The curve-curve bisector (in red) on the left is between a horizontal circle and a vertical line (in yellow). The point-surface bisector (in blue) on the right is between a torus (in magenta) and a point at its centre (in yellow). This bisector has two sheets that extend all the way to infinity.

If the number of constraints is less than the number of degrees of freedom, a rational solution still exists. Further readings on these rational cases can be found in [55].

The following set of constraints is defined for the surface-surface bisector, $B = (bx, by, bz)$, in $\mathbb{R}^3$:

$$
\begin{aligned}
0 &= \left\langle \mathcal{B} - S_1(u,v), \frac{\partial S_1(u,v)}{\partial u} \right\rangle, \\
0 &= \left\langle \mathcal{B} - S_1(u,v), \frac{\partial S_1(u,v)}{\partial v} \right\rangle, \\
0 &= \left\langle \mathcal{B} - S_2(s,t), \frac{\partial S_2(s,t)}{\partial s} \right\rangle, \\
0 &= \left\langle \mathcal{B} - S_2(s,t), \frac{\partial S_2(s,t)}{\partial t} \right\rangle, \\
0 &= \langle \mathcal{B} - S_1(u,v), \mathcal{B} - S_1(u,v) \rangle - \langle \mathcal{B} - S_2(s,t), \mathcal{B} - S_2(s,t) \rangle .
\end{aligned}
$$

These five (linear in B) constraints also have seven degrees of freedom: $u, v, s, t, b_x, b_y, b_z$. Hence, having two more degrees of freedom than constraints, the solution space is a two-

manifold, the bisector sheet in $\mathbb{R}^3$ (recall that the bisector sheet in $\mathbb{R}^n$ is an $(n-1)$-manifold). One needs to solve these five equations in seven degrees of freedom – by any means, a non trivial task. In [58], a special non-linear multivariate solver has been employed, presented in [59], which supports cases with non-zero dimensional solution spaces. The solution is given as a dense set of $(u, v, s, t, b_x, b_y, b_z)$ points in $\mathbb{R}^7$. Then, exploiting the given (u, v) parameterisation of S_1, a two-manifold in $\mathbb{R}^3$ is fitted to this data, satisfying the interpolation constraints of $x(u, v) = b_x, y(u, v) = b_y, z(u, v) = b_z$. The non rational bisector between a curve and a surface in $\mathbb{R}^3$ is computed using a similar approach. In Figure 4 (see also Figure CP-2 in Appendix E), the solution point set is shown as yellow points on the fitted bisector sheet in red/magenta.

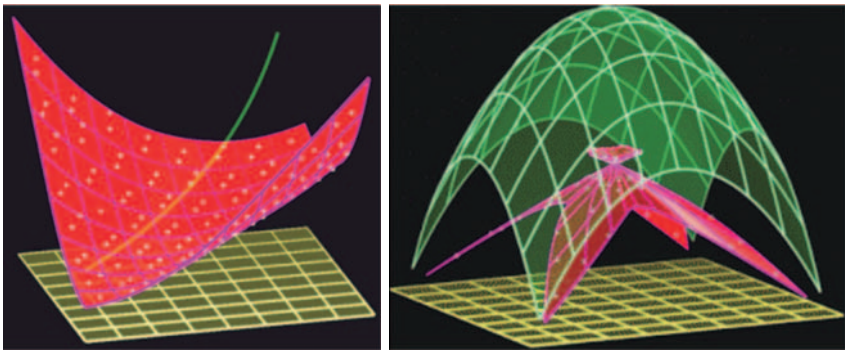
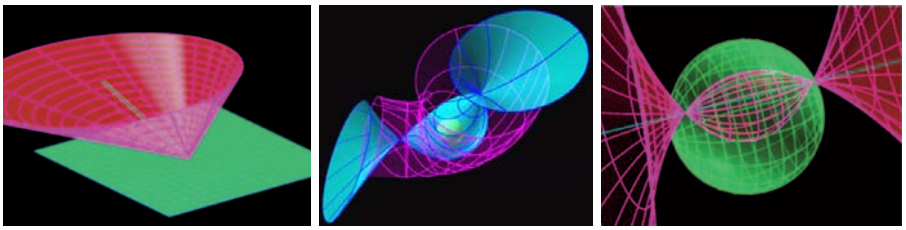


Fig. 4. Curve-surface (left) and surface-surface (right) approximations to the bisector sheets (in red/magenta) in $\mathbb{R}^3$. The dense solution point set is shown as yellow points.

Clearly, being an approximation, the curve-surface and surface-surface bisectors are further more difficult to compute than their analytic counterparts. They become even more difficult when the result is numerically unstable – a not an uncommon case when dealing with bisectors. In many cases, the bisector sheets introduce poles as they vanish at infinity (see the bisector in Figure 3 (right)), and cusps, and hence, self-intersections when the bisector sheet is not regular (see the bisector in Figure 4 (right)). Luckily, many important cases exist where the bisector between a curve and a surface or between two surfaces is indeed rational. One notable simple case is the plane-plane bisector that is another (bisector) plane.

In [57, 86], more special curve-surface and surface-surface rational bisectors in $\mathbb{R}^3$ are identified. The full details of these results are beyond this survey but we will describe a few of the approaches that are presented in [57, 86]. The bisector between a line and a plane in a general position is simply a cone. This is obvious if the line is orthogonal to the plane but also holds for any non-coplanar line (see Figure 5 (a) and Figure CP-3 (a) in Appendix E). An offset is an operation to which the bisector is invariant. The bisector between a sphere and any surface that yields a rational offset could be reduced to a point-surface bisector via the simultaneous offset of the sphere and the other surface by the sphere's radius (see Figure 5 (b) and Figure CP-3 (b) in Appendix E). The bisector computation between a sphere and a canal surface that yields a rational offset is reduced to a bisector computation between a point and rational surface representing the offset of a canal surface. In Figure 5 (c) (see also Figure CP-3 (c) in Appendix E), the line-sphere bisector is similarly reduced to a cylinder-point bisector



(a) (b) (c)

Fig. 5. The bisector sheets (in red/magenta) of a plane and a line (a), a sphere and a canal surface (b), and a line and a sphere (c).

computation, again via an offset operation. Table 2 summarizes the cases known to be rational, as presented in all above references. As can be seen from Table 2, pretty much all CSG primitive shapes yield a rational bisector in $\mathbb{R}^3$ with the exception of the torus, which in most cases has a rational bisector only in special arrangements.

	Point	Line	Plane	Cylinder	Sphere	Cone	Torus
Point	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Line		Yes	Yes	Yes	Yes	Yes	Partial
Plane			Yes	Yes	Yes	Yes	Partial
Cylinder				Yes	Yes	Yes	Partial
Sphere					Yes	Yes	Yes
Cone						Yes	Partial
Torus							Partial

Table 2. The existence of rational bisectors between CSG primitives in $\mathbb{R}^3$.

3.2 Exact computation of the medial axis

As noted above, the construction of the Voronoi diagram and MAT for freeform curves in the plane is more difficult because of the complexity of the bisectors. Ramamurthy and Farouki [63, 64] implemented an incremental algorithm in which the bisectors are inserted one by one and the Voronoi diagram of the curves is updated after each insertion; the MAT is derived from the Voronoi diagram and is represented as a piecewise linear approximation of the actual bisector, computed as the envelope of the point-curve rational bisectors. Ramanathan and Gurumoorthy [87] implemented a different tracing algorithm for the construction of the MAT of a freeform shape. This implementation also approximates the edges of the MAT by computing samples of bisector points on the edges and interpolating these sample points. Piecewise linear curves involve the comparison of expressions with two nested square roots [29]. Efficient and fully robust implementations are few [71]. An exact algorithm for not-necessarily convex polyhedra in $\mathbb{R}^3$ can be found in [41].

A fairly general class of shapes for which it is possible, in principle, to compute the medial axis exactly are the *semi-algebraic sets* . These sets are the solutions of a finite system of

algebraic equations and inequalities. The medial axis of such a set is itself semi-algebraic and can be computed with tools from computer algebra. To describe this, let X be a shape in $\mathbb{R}^3$ whose boundary is a C^1-smooth manifold. We introduce the *symmetry set* of X, consisting of the centers of spheres tangent to the boundary of X at two or more points. It contains all points of the medial axis but also possibly additional points since the spheres are not constrained to bound balls contained in X. Suppose now the boundary of X is defined by the algebraic equation $f(x) = 0$ and 0 is a regular value of f. It follows that the gradient for all points of the boundary is non-zero, $\nabla f(x) \neq 0$. In this case, the symmetry set is the closure of the set of points z for which there exists points x and y that satisfy the following system of algebraic equations:

$$\begin{cases} f(x) &= 0, \\ f(y) &= 0, \\ (x-z) \times \nabla f(x) &= 0, \\ (y-z) \times \nabla f(y) &= 0, \\ \|x-z\|^2 &= \|y-z\|^2, \\ t\|x-y\|^2 &= 1. \end{cases}$$

In the last condition, t is an additional free variable that ensures that x and y are distinct. If 0 is not a regular value of f, we need to add $\nabla f(x) \nabla f(y) s = 1$ as yet another equation, with s as a free variable. Finally, the medial axis is obtained by imposing the additional conditions that $\|u-z\|^2 \geq \|x-z\|^2$, for all points u on the boundary, and z be contained in X. Considering u to be a new free variable, it is possible to remove points from the solution, namely the points z for which $f(z) < 0$ or for which there exists u with $f(u) = 0$ and $\|u-z\|^2 < \|x-z\|^2$. This new set is still semi-algebraic since it is the difference between two semi-algebraic sets.

In [70], the fact that one can express the bisectors of rational curves and the MAT of rational curves as (semi-) algebraic sets is used to derive an algorithm that computes the precise Voronoi cells of rational curves in the plane. Using the precise low degree algebraic formulation offered in [54] to represent the bisector of two planar curves, trimming conditions based on orientation and curvature properties are formulated for these bisectors. The trimmed bisectors are then fed into a lower envelope computation stage in which the Voronoi cells are precisely extracted. The bisector segments are represented as implicit B-spline bivariate forms and hence are algebraic. Further, the locations where adjacent bisectors intersect, and therefore define the corners of the Voronoi cells, are also representable as a set of algebraic constraints. The end result is a precise representation of the Voronoi cells of planar rational curves. Figure 6 shows a few examples of precise Voronoi cells of rational curves.

Dutta and Hoffman [51] proposed a scheme to compute the Voronoi diagram and MAT of CSG primitives. As noted above, their results on bisectors of CSG primitives were partial and, therefore, their work was theoretical and never implemented. A recent result by Ramanathan and Gurumoorthy [88], which is based on their work in [87], constructs the MAT of extruded and revolved shapes. In their work, they exploit the fact that the 3D MAT of an extruded or revolved shape is closely related to the 2D MAT of its creating section curve. This is the only implementation, as far as we can determine, that constructs a MAT in $\mathbb{R}^3$ of surfaces that are not polyhedra.

For the complement of a union of balls in $\mathbb{R}^k$, the medial axis can be derived from the Apollonius diagram of the corresponding spheres or from convex hulls of finitely many points in $\mathbb{R}^{k+2}$ [12, 23]. Perhaps surprisingly, the medial axis of the union of finitely many balls is simpler than that of the complement. As first described in [5], it is piecewise linear and can be constructed from the Voronoi diagram of a finite set of points. As discussed in more detail shortly, the cells of dimension less than k in this diagram may be interpreted as the medial

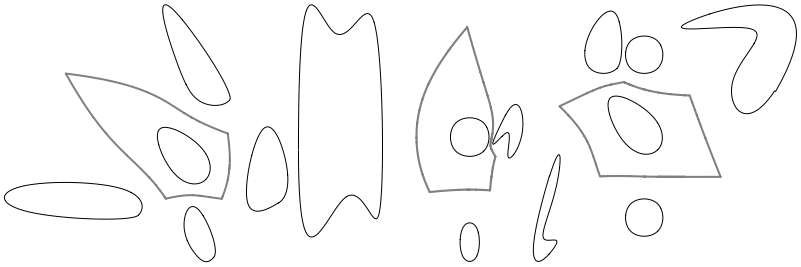


Fig. 6. Three examples of precise Voronoi regions (in gray wide lines) of rational closed parametric curves. The Voronoi region of one curve (the curve inside the Voronoi region) is shown in each example.

axis of a punctured Euclidean space, a case that permits particularly simple exact algorithms. Finally, the MAT of spheres in $\mathbb{R}^3$ was also recently considered in [74].

4 Approximation of the medial axis

Except for the few cases described in the previous Section, when effectively computing a medial representation of a shape, we face the problem of extracting a finite representation of the medial axis. Let $\mathcal{M}[X]$ be the MAT of the shape X. In most cases, we apply an *approximation* of $\mathcal{M}[X]$ that may be either *numerical*, in the sense that our output is always "near" or exactly $\mathcal{M}[X]$, or *geometric*, in the sense that we define new descriptors that are geometrically similar to the skeleton of the shape. The approximation techniques discussed in Section 4.1 refer to numerical approximations of the medial axis. These techniques compute the medial axis as a subset of the Voronoi Diagram of a set of points sampled on the shape boundary. Geometric approximations of the medial axis are shown in Sections 4.2, 4.3 and 4.4. Such approaches are classified on the basis of the skeletonisation method adopted, i.e. implementation of the grassfire propagation, distance map computation and thinning .

Moreover, while techniques based on approximating the Voronoi diagram and on simulating the grassfire represent *continuous* methods that manipulate points with real coordinates (see also Section 4.2), distance maps and thinning constitute *discrete* methods: the object is stored as a collection of pixels/voxels and the resulting skeleton is a connected subset of such pixels/voxels. Working in the discrete space means that we have to face problems specific to this space, which are relevant for medial axis extraction and skeletonisation. It is well known that a different connectivity type has to be used for the shape and for its complement to avoid topological paradoxa. The connectivity type depends on which, among the neighbors of a pixel/voxel, are considered as directly connected to each other. In two dimensions, each pixel p has four neighbors sharing an edge with p, and other four neighbors sharing a vertex with p. The 4-connectivity considers as directly connected to each other pixels sharing an edge, while the 8-connectivity considers both kinds of neighbors. In three dimensions, a voxel v has six neighbors sharing a face with v, twelve neighbors sharing an edge and eight neighbors sharing a vertex. Three connectivity types are hence possible: 26-connectivity, when all three kinds of neighbors are considered, 18-connectivity, when neighbors sharing a face or an edge are considered, and 6-connectivity, when only the neighbors sharing a face are considered. If the

same connectivity type is used for both the object and its complement, a closed curve/surface would not divide its complement into disjoint parts, or an open curve/surface would divide its complement into disjoint parts. For discrete space in two dimensions, the 8-connectivity and the 4-connectivity are generally adopted for the object (and, hence, its skeleton) and for its complement, respectively. In three dimensions, the 26-connectivity and the 6-connectivity are generally used for the object and its complement. Another problem relevant for skeletonisation is strictly related to the nature of the discrete space. In correspondence with regions whose thickness is expressed by an even number of pixels/voxels, the set of centers of maximal balls is 2-pixel/voxel wide. This means that whenever a discrete solution to medial axis extraction or skeletonisation is desired, the resulting set can locally be 2-pixel/voxel wide. Alternatively, which is generally regarded as preferable, the nearly-thin medial axis or skeleton can be reduced to a 1-pixel/voxel thick set by means of final thinning, but the complete reversibility is lost. We remark that the loss in object recovery exclusively regards pixels/voxels on the boundary of the original object. The loss in recovery is generally considered as acceptable, since the actual belonging of pixels/voxels to the boundary of an object obtained after acquisition and digitisation of a continuous object is questionable. We also remark that, in the two-dimensional space, the skeleton is a union of arcs and curves and reversibility is almost completely guaranteed, starting from the 1-pixel wide linear skeleton. In turn, in the three-dimensional space, reversibility is possible only if the so called surface-skeleton, consisting of surfaces and curves, is computed. For solid objects, i.e., objects having no cavities, the surface-skeleton can be furthermore compressed to obtain a linear shape representation (the so called curve-skeleton.) In this case, reversibility is no longer possible. In fact, a large number of centers of maximal balls is unavoidably removed from the surface-skeleton to reduce it to the curve-skeleton. In Sections 4.3 and 4.4 we will mainly focus on linear skeletons.

A more detailed analysis of medial axis extraction and skeleton computation can be found in [31] for objects in the two-dimensional space and in [47] for the three-dimensional case. Other recent contributions on this topic are provided in [40, 43].

4.1 Skeletons from Voronoi Diagrams

We have pointed out that the exact computation of the medial axis runs into obstacles except for certain classes of shapes. Another approach is to approximate the smooth shape with a discrete one, for which the medial axis can be computed exactly.

Despite the intuitive correlation between the Voronoi diagram of a set of points sampling the boundary of a planar shape and its MAT, the formal proof of the Voronoi diagram convergence to the MAT as the number of samples goes to infinite has come rather recently [27]. In this Section we introduce methods that approximate the medial axis of a shape using the Voronoi graph of points sampling its boundary. The role of these methods is twofold: they can either compute the MAT on an approximation of smooth shapes or be applied directly to discrete representations such as triangulations.

In the following we introduce the approximation paradigm; for more details about the stability and computation of medial axes see [6].

Instability and semi-continuity

We think of $\mathcal{M}$ as a transform that maps the shape X to its medial axis, $\mathcal{M}[X]$. As emphasized in [78], geometric shapes are usually not known exactly and represented by approximations of one kind or another. For example, the boundary of a shape may be approximated by a

triangulation obtained by software for surface reconstruction or segmentation. Under these circumstances, it would be important that the transform be continuous. In other words, one should be able to compute an arbitrarily accurate approximation of the output for a sufficiently accurate approximation of the input. Most commonly, one would use the Hausdorff distance to quantify the difference between two inputs and two outputs and this way define what it means for the transform to be continuous. Unfortunately, the medial axis transform is not continuous under this notion of distance: small modifications of the input shape can induce large modifications of its medial axis. This effect is illustrated in Figure 8, where we compare the medial axis of an oval on the left with the medial axis of a set whose Hausdorff distance to the oval is bounded from above by $\epsilon > 0$. The difficulty of approximating the medial axis due to its instability with respect to the Hausdorff distance is a well-known but until recently not well-understood problem.

One can observe experimentally that small modifications of a shape do not affect the entire medial axis. Typical effects for shapes in $\mathbb{R}^2$ are fluctuating branches that leave the rest of the medial axis unchanged. Similarly, for shapes in $\mathbb{R}^3$ we notice fluctuating spikes, added to or removed from the otherwise stable structure. This observation is consistent with the fact that the medial axis is semi continuous with respect to the Hausdorff distance [79, chapter 11]. To explain this concept, we let A and B be subsets of $\mathbb{R}^k$ and write $d_H(A \mid B) \sup_{x \in A} d(x, B)$ for the *one-sided Hausdorff distance* of A from B, where $d(x, B)$ is the infimum of the Euclidean distances between x and points y in B. Observe that $d_H(A \mid B) < \epsilon$ if and only if A is contained in the offset $B^{+\epsilon} = \{x \in \mathbb{R}^k \mid d(x, B) < \epsilon\}$. The *Hausdorff distance* between A and B is $d_H(A, B) \max\{d_H(A \mid B), d_H(B \mid A)\}$. We write A^c and B^c for the complements of A and B and note that the Hausdorff distance between A^c and B^c is generally different from that between A and B. Indeed, $d_H(A^c, B^c)$ is forgiving for small islands of A far away from B, while $d_H(A, B)$ is forgiving for small holes of A far away from B^c. With this notation, we are ready to define the concept of semi continuity. Specifically, a transform $\mathcal{T}$ is *semi continuous* if for every bounded open subset $X \subseteq \mathbb{R}^k$ and for every $\delta > 0$, there exists $\epsilon > 0$ such that for every open subset Y of $\mathbb{R}^k$,

$$d_H(X^c, Y^c) < \epsilon \implies d_H(\mathcal{T}[X] \mid \mathcal{T}[Y]) < \delta. \tag{3}$$

Note that ϵ depends on X. In words, small Hausdorff distance between the complements of X and Y implies that $\mathcal{T}[X]$ is contained in a tight parallel body of $\mathcal{T}[Y]$. As mentioned earlier, this condition is satisfied for $\mathcal{T} = \mathcal{M}$.

Approximation paradigm for the medial axis

The difficulty of computing the medial axis exactly (see Section 3) motivates a serious look at approximation algorithms. A framework that captures a common line of attack to approximating the medial axis is sketched in Figure 7. First, Y that belongs to a class of shapes for which the medial axis can be constructed exactly is found such that it approximates X . Second, the medial axis of Y is constructed. Third, the medial axis of Y is pruned to get a subset $\mathcal{P}[\mathcal{M}[Y]] \subseteq \mathcal{M}[Y]$ that approximates the medial axis of X. The composition of the three steps provides the approximation of the medial axis of X. The most challenging step in this paradigm is the extraction of a subset $\mathcal{P}[\mathcal{M}[Y]$ of $\mathcal{M}[Y]$ that indeed approximates $\mathcal{M}[X]$. Recent mathematical results that rationalize this approach are discussed shortly.

The notion of approximation used in the first step varies between different implementations of the approximation paradigm. It either means that Y is the image of X under a small C^m-perturbation [36], or that the Hausdorff distance between the complements of X and Y is small, as in [35]. Other notions of approximation are conceivable.

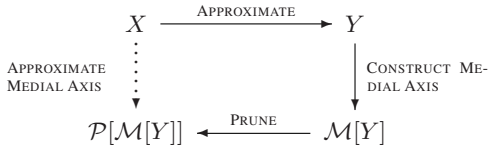


Fig. 7. An approximation $\mathcal{P}[\mathcal{M}[Y]]$ of the medial axis of a shape X can be found as part of the medial axis of a shape Y approximating X.

Punctured Euclidean spaces

We start by identifying a class of shapes for which the medial axis can be constructed exactly and efficiently. We obtain shapes in this class by puncturing the k-dimension real space at a discrete set of locations. Equivalently, we consider the complement of a discrete set of points P in $\mathbb{R}^k$. The medial axis of this space is the *Voronoi graph* of P. Algorithms for constructing the Voronoi graph are well-studied in computational geometry and implementations are available from the geometric software library CGAL [33]. For a set P of n points in $\mathbb{R}^k$, the graph can be constructed in time $O(n^{\lceil k/2 \rceil} + n \log n)$, which is optimal in the worst case because the graph can consist of a constant times $n^{\lceil k/2 \rceil}$ faces. In most practical applications, the number of faces, F, is much less and the output-sensitive algorithm in [34] constructs the graph in $\mathbb{R}^3$ in time $O((n + F) \log^2 F)$. Examples of point sets with provable small Voronoi graphs are so-called κ-light ϵ-samples of compact smooth generic surfaces in $\mathbb{R}^3$, with $F = O(n \log n)$ [9], and κ-light ϵ-samples of polyhedral surfaces in $\mathbb{R}^3$, with $F = O(n)$ [8]. Such samples will be studied in more detail shortly.

Consider a finite point set P whose Hausdorff distance to the boundary of a shape X is less than ϵ and write $\mathrm{Vor}[P]$ for the Voronoi graph of P. Using the semi continuity of the medial axis expressed in (3), the subset of $\mathrm{Vor}[P]$ inside X contains an approximation of the medial axis of X. In the approximation paradigm for medial axes, this subset can be interpreted as part of the medial axis of a shape Y close to X. Following [35], Y is defined to be the parallel body $X^{+\epsilon}$ of X minus the points in P; see Figure 8. Since the Hausdorff distance between P and the boundary of X is less than ϵ, the same is true for the complements of X and the thus constructed space: $d_H(X^c, Y^c) < \epsilon$. In summary, we have $\mathcal{M}[Y] \cap X = \mathrm{Vor}[P] \cap X$.

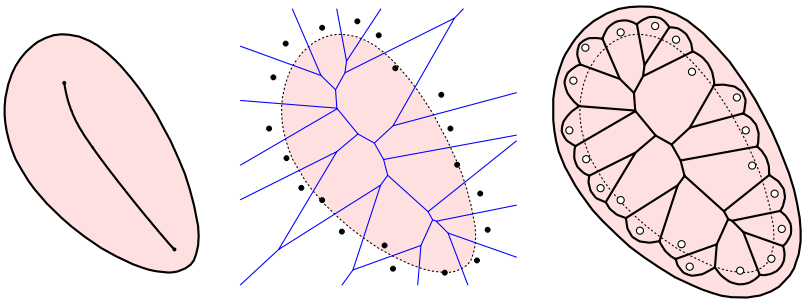


Fig. 8. On the left, a shape X and its medial axis. In the middle, a finite set of points P whose Hausdorff distance to the boundary of X is less than ϵ and its Voronoi graph. On the right, $X^{+\epsilon} - P$ and its medial axis.

Pruning the Voronoi graph

We now consider results that focus on the detailed relationship between the Voronoi graph of a finite point set and the medial axis of the shape whose boundary the points sample. A *sample* of the boundary of a shape X is a finite set of points (exactly and not just approximately) on that boundary. An ϵ-*sample* is a sample whose Hausdorff distance to the boundary of X is less than ϵ. In other words, every point of the boundary is less than distance ϵ away from a point in the ϵ-sample. The ϵ-sample is κ-*light* if the number of sample points within distance ϵ is never more than κ. The ϵ-sample is *noisy* if points are not necessarily on the boundary but at Hausdorff distance less than ϵ to the boundary.

An early result on the connection between the Voronoi graph and the medial axis is due to Brandt [27]. Given a shape in $\mathbb{R}^2$, he takes an ϵ-sample on the boundary curve and considers the Voronoi edges and vertices that are completely contained in the shape; see Figure 9.

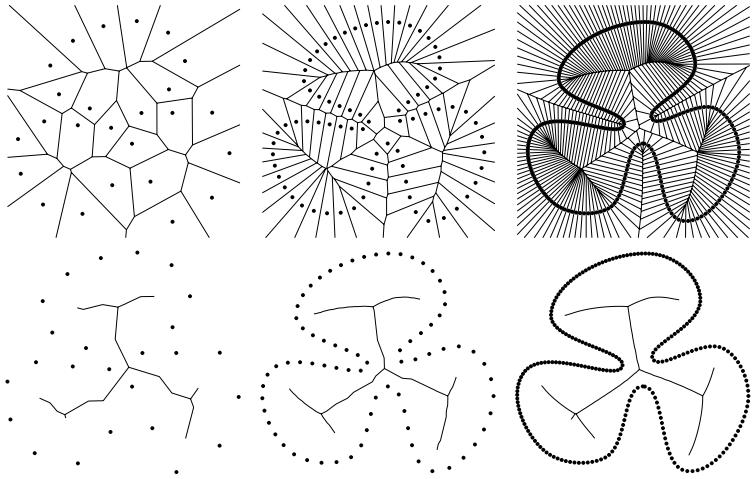


Fig. 9. In $\mathbb{R}^2$, vertices and edges lying inside a shape and extracted from the Voronoi graph of an ϵ-sample of the boundary approximate the medial axis (courtesy of Attali and Montanvert [10]).

Brandt then proves that under some technical conditions on the boundary curve, the portion of the Voronoi graph defined by these edges and vertices approximates the medial axis. Amenta and Bern [2] point out that the direct extension of this result to shapes in $\mathbb{R}^3$ does not hold; see Figure 10. The validity of the extension is spoiled by the existence of slivers in three-dimensional Delaunay triangulations, which occur for ϵ-samples with arbitrarily small $\epsilon > 0$. Roughly, a *sliver* is a tetrahedron whose four vertices are almost co-circular. The location of the Voronoi vertex corresponding to the sliver depends on the four vertices but is generally unrelated to any feature of the surface and does not necessarily lie near the medial axis. As a first step to cope with slivers, Amenta and Bern eliminate all except a few Voronoi vertices they refer to as poles. Every sample point p generates a Voronoi polyhedron and the vertices furthest away from p on the two sides of the surface are the *poles* of p. Clearly, there are at most $2n$ poles for a sample of n points. As proved in [3], for a shape whose boundary is

a smooth C^1-manifold, the poles tend to the medial axis of the shape and its complement as ϵ goes to zero.

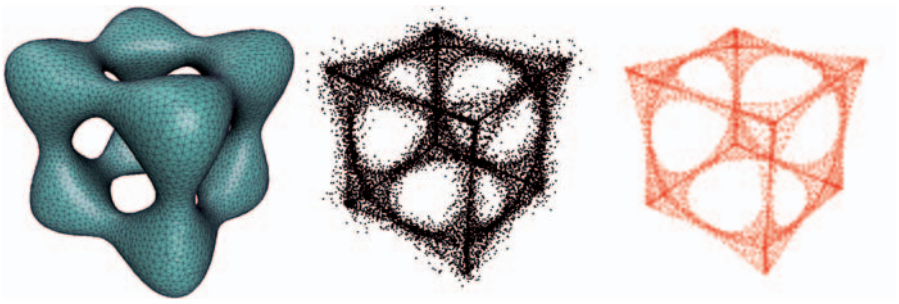


Fig. 10. On the left we see a triangulation of the boundary of a shape in $\mathbb{R}^3$. Its vertices determine a Voronoi diagram whose vertices inside the shape are shown in the middle. The subset of poles inside the shape is shown on the right.

To extend the result of Brandt to $\mathbb{R}^3$, we need more than just points (the poles) near the medial axes, we also need to connect them to form a geometric structure approximating the medial axis. In [3], Amenta, Choi and Kolluri use simplexes of the (weighted) Delaunay triangulation of the poles. To avoid the construction of this weighted Delaunay triangulation and connect the poles directly inside the Voronoi graph, we need to know about its local distance from the medial axis. Bounds on this distance can be found in [7, 22, 38]. Assuming the boundary of the shape is a smooth C^1-manifold and using these bounds, among other things, Dey and Zhao [44] give an algorithm that identifies a sub graph of the Voronoi graph that approximates the medial axis for the Hausdorff distance. We note that the above results are limited to smooth surfaces and to samples of points that lie on that surface. In [35], Chazal and Lieutier obtain a similar result but for more general data: shapes are bounded open subsets and samples are noisy. They introduce a subset of the Voronoi graph, called the λ-Voronoi graph , that approximates the medial axis for a particular sequence of decreasing λ [6]; see Figure 11. Furthermore, for small enough values of λ, this subset is homotopy equivalent to the shape [35].

4.2 Skeleton trough the simulation of the grassfire

Beside methods for the exact computation of a polygon like that proposed in [77], several approximate variations of the medial axis have been proposed in the literature. In particular, in this Section we focus on the straight skeleton and on one of its approximation: the linear axis.

Straight skeleton

Aichholzer and Aurenhammer [1] introduced the *straight skeleton*, a new type of skeleton for polygons. It is closely related to the medial axis, being also based on a wavefront propagation. The wavefront consists of straight line segments and circular arcs (see Figure 12 (a)) and, as it propagates inwardly, the breakpoints between consecutive line segments and circular arcs trace the Voronoi diagram of the polygon. By removing the segments in the diagram incident

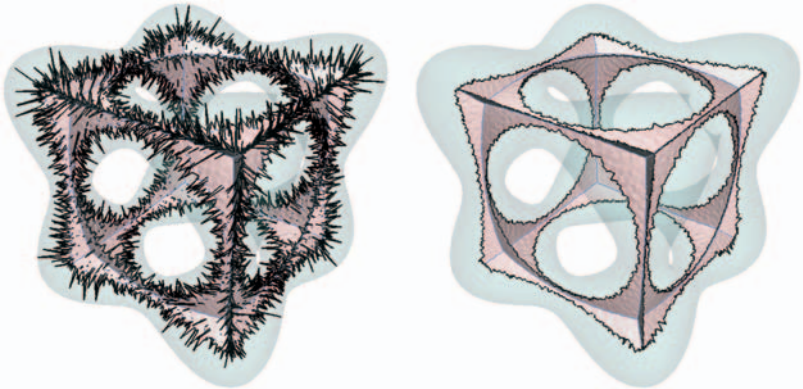
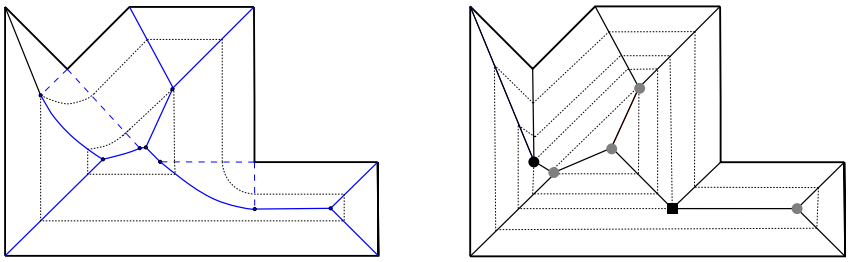


Fig. 11. Two λ-Voronoi graph of the same shape, with λ increasing from left to right, constructed as a subset of the Voronoi graph of a sample of the boundary.



(a) (b)

Fig. 12. Medial Axis (a) vs. Straight Skeleton (b). In (b) the black disk marks a reflex edge annihilation, while gray disks mark convex edge annihilations. An edge-edge collision generates the arc between the black box (vertex-edge collision) and a gray disk (convex edge annihilation)

to the reflex vertices, we obtain the medial axis, which consists of straight line segments and parabolic arcs.

To construct the straight skeleton, we let wavefront edges move parallel to the polygon sides. In contrast to the medial axis, edges incident to a reflex vertex will grow in length. The front remains a polygon, whose vertices during the process trace out the skeleton (see Figure 12(b)). As its name suggests, it consists of straight line segments only. It also has a smaller combinatorial complexity ($n - 2$ internal nodes, with n the number of polygon vertices) than the medial axis ($n + r - 2$ nodes, with r the number of reflex vertices).

A straightforward computation of the straight skeleton consists of simulating the sequence of events occurring in the propagation process described above. Possible edge events are given by intersections of the bisectors of adjacent vertices of the current wavefront. If we maintain a priority queue $\mathbf{E}$ of all these events, indexed by the moment in time when they occur, the next edge event can be detected in constant time. Also after each event occurring in the propagation, only a constant number of updates in $\mathbf{E}$ are necessary. These updates come from changes in the wavefront at the location of the newly occurred event. The priority queue can be created

in $O(n\log(n))$ time, and each update requires $O(\log(n))$ time, where n is the number of vertices in P. Unlike for the edge events, the computation of possible split events can not be done locally. For this purpose we maintain a priority queue $\mathbf{S}$ of all pairs (reflex vertex, wavefront edge), indexed by the moment in time when a split between them would occur. After each event in the propagation a linear number $O(n)$ of updates in $\mathbf{S}$ are necessary. Thus $\mathbf{S}$ can be created in $O(nr\log(n))$ time, and the updates after each event take $O(n\log(n))$ time, where r is the number of reflex vertices of P. The straight skeleton $S(P)$ can thus be computed in $O(nr\log(n))$ time, and the above algorithm requires $O(nr)$ space.

A faster algorithm that uses more complex data structures can be found in [60]. It runs in $O(n^{1+\epsilon} + n^{8/11+\epsilon}r^{9/11+\epsilon})$ time with a similar space complexity, where ϵ is an arbitrarily small positive constant. Eppstein's algorithm simulates the sequence of interactions between edges and vertices in the propagation process. If the polygon P is interpreted as the outline of a building's groundwalls, the straight skeleton is the projection of a roof over P, whose facets are all of equal slope. In simulating the events defining the skeleton, they view time as a third spatial dimension, so that the propagation process becomes an upward sweep of the roof of the polygon with a horizontal plane.

A more recent algorithm by Cheng and Vigneron [37] computes the straight skeleton of a non-degenerate simple polygon in $O(n\log^2 n + r\sqrt{r}\log r)$ expected time. For a degenerate simple polygon, its expected time bound is $O(n\log^2 n + r^{17/11+\epsilon})$.

The Linear Axis

When a simple polygon contains sharp reflex angles with short incident edges, its straight skeleton gives counterintuitive results (see the left column of Figure 14). In [97], Tanase and Veltkamp introduce the *linear axis*. It is based on a linear wavefront propagation like the straight skeleton, but the discrepancy in the speed of the points in the propagating wavefront, though never zero, can decrease as much as wanted.

More formally, let $\{v_1, v_2, \ldots, v_n\}$ denote the vertices of a simple polygon P and let $\kappa = (k_1, k_2, \ldots, k_n)$ be a sequence of natural numbers. If v_i is a convex vertex of P, $k_i = 0$, and if it is a reflex vertex, $k_i \geq 0$. Let $\mathcal{P}^\kappa(0)$ be the polygon obtained from P by replacing each reflex vertex v_i with $k_i + 1$ identical vertices, the end points of k_i zero-length edges, which will be referred to as the *hidden edges* associated with v_i. The directions of the hidden edges are chosen such that the reflex vertex v_i of P is replaced in $\mathcal{P}^\kappa(0)$ by $k_i + 1$ "reflex vertices" of equal internal angle.

Then, the linear axis $\mathbf{L}^\kappa$ of P, corresponding to a sequence κ of hidden edges, is the trace of the convex vertices of the linear wavefront $\mathcal{P}^\kappa$ in the above propagation process. $\mathbf{L}^\kappa$ is a subset of the straight skeleton of $\mathcal{P}^\kappa(0)$; it is sufficient to remove the bisectors traced by the reflex vertices of the wavefront (see Figure 13 (a)). If each reflex vertex v_j of internal angle greater than $3\pi/2$ has at least one associated hidden edge ($k_j \geq 1$), then $\mathbf{L}^\kappa$ is a connected graph. This is because only bisectors incident to reflex vertices of P are removed from the straight skeleton of $\mathcal{P}^\kappa(0)$ in order to obtain $\mathbf{L}^\kappa$.

Obviously, the larger the number of hidden edges, the better the linear axis approximates the medial axis. A thorough analysis of the relation between the number of the inserted hidden edges and the quality of this approximation is given in [97]. They introduce the notion of *ε-equivalence* between two skeletons. Nodes in the two skeletons are clustered based on a proximity criterion, and the ε-equivalence between the two skeletons is defined as an isomorphism between the resulting graphs with clusters as vertices. This allows to compare skeletons based on their main topological structure, ignoring local details. In [97], an algorithm is given

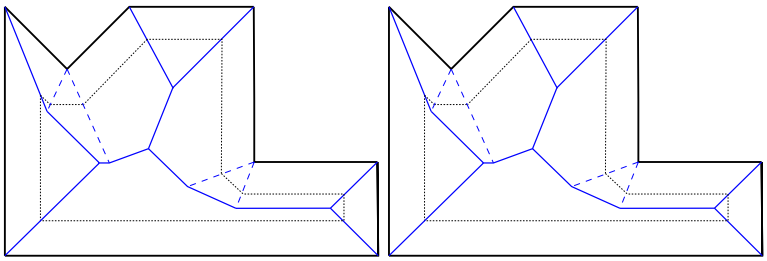


Fig. 13. (a) The linear axis in the case when one hidden edge is inserted at each reflex vertex. A linear wavefront is drawn in dotted line style; the dashed lines are the bisectors that are not part of the linear axis. (b) The linear offset (solid line) of a reflex vertex with 3 associated edges is made of 5 line segments tangent to the uniform offset (dotted line) of this vertex.

for computing the number of hidden edges for each reflex vertex such that the resulting linear axis is ε-equivalent to the medial axis. The whole linear axis computation takes linear time for polygons with a constant number of nodes in any cluster. There is only a limited category of polygons not having this property. Implementation results suggest that in practice only a few hidden edges are necessary to yield a linear axis that is ε-equivalent to the medial axis.

4.3 Skeletons based on topological thinning

Thinning refers to the process of removing pixels or voxels from a discretised object in an attempt to whittle the object down in topological fashion to a more simple representation consisting of connected, unit-wide pathways of pixels or voxels. This process, applied to elongated objects characterized by nearly constant thickness (e.g., printed or hand-written characters, line drawings, blood vessels, or branching patterns of air passageways in the lungs), leads to a set of lines centered within the object and retaining the relevant structural and shape information of the object. For this reason, the main focus of thinning is the preservation of topology, with the primary purpose being to aid in the identification of a basic structure.

Solutions for different grid types such as the rectangular, the triangular and the hexagonal grid have been proposed. Rosenfeld [90] provides a list of over 160 papers on thinning; note, however, that the vast majority of these papers deal with the problem in two dimensions. Ideally, thinning is an isotropic compression process. Since compression takes place from all directions at the same rate, its implementation by means of a parallel algorithm is a natural choice. Actually, both parallel and sequential algorithms have been developed and the literature includes a huge number of papers on this subject (for a survey of two-dimensional thinning algorithms, see e.g. [46]). In parallel algorithms, the processing done at each iteration is a function of the object resulting from the previous iteration only. In sequential algorithms, the elements are processed one after another and are updated in terms both of the object resulting from the previous iteration, and of the modifications produced so far in the current iteration. Thus, the structure of the set resulting from a sequential algorithm depends on the order in which pixels/voxels are processed. Sometimes, spurious branches appear in a particular order of processing, but do not appear in a different order. End-point detection criteria have great importance in this case, to guarantee isotropic object compression and to avoid unwanted shortening of branches in the resulting set.

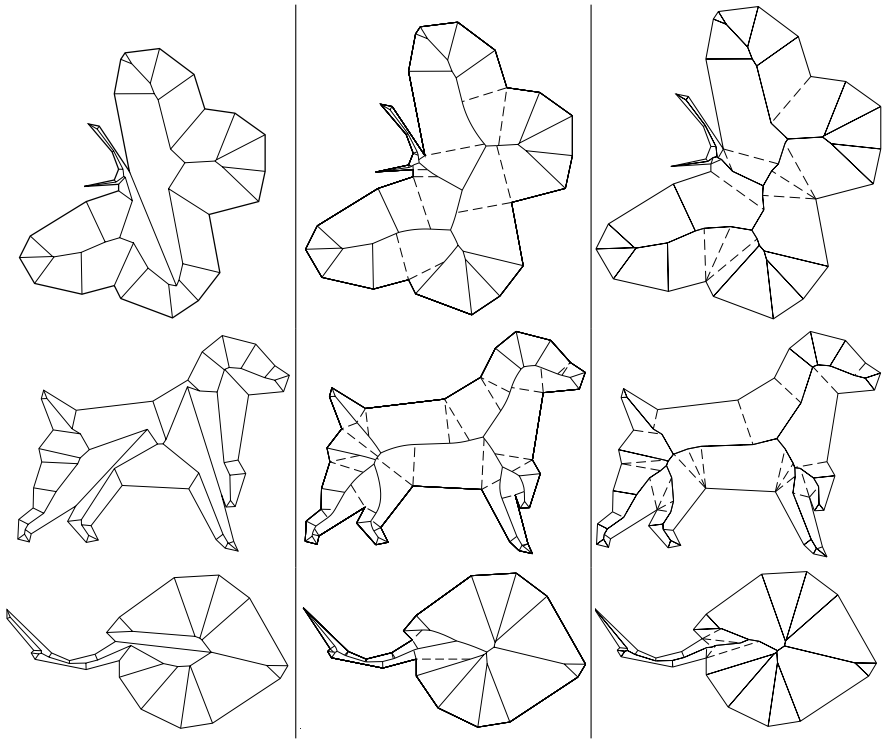


Fig. 14. A comparison of the straight skeleton (left column), the medial axis (middle column), and the linear axis (right column). The skeletons are drawn in solid line style. The dashed lines in the medial axis figures are the Voronoi edges, which are not part of the medial axis. The dashed lines in the linear axis figures represent the bisectors traced by the reflex vertices of the wavefront, which are not part of the linear axis. In these examples, the linear axis is isomorphic with the medial axis ($\varepsilon = 0$).

Topologically oriented thinning consists of repetitive testing and subsequently deletion of pixels or voxels on the boundary of the object, whenever their removal does not alter the topology of the thinned shape. However, as said above, in order the resulting set reflects the geometrical structure of the object, removal operations should be combined with suitable preservation criteria to avoid non isotropic object compression and unwanted shortening of branches in the resulting representation. Practically, in correspondence with every significant protrusion of the object, a branch is expected to be found in the thinned shape. To correctly map protrusions with branches, the tip of each protrusion should be identified and an element in correspondence of each tip (i.e. the end-point of a branch) should be preserved from removal. Unfortunately, most of the existing thinning algorithms do not ensure that the previous correspondence between tips of protrusions and end-points always holds, so their performance is likely to become unacceptable when a wide repertory of objects is to be processed. This behaviour is imputable to the fact that removal occurs during a 'blind' sequential process, that uses the property satisfied by

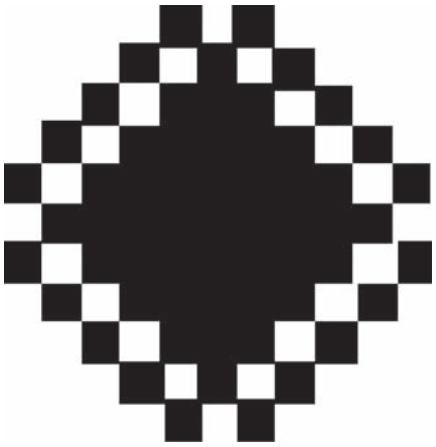


Fig. 15. A lace-shaped 2D object that cannot be reduced to one-pixel wide subset.

the end-points in the resulting set, i.e. the property of having only one neighbor in the skeleton branch, as a criterion to detect the end-points during thinning. This may cause end-points to be originated or not, depending on the order in which the chosen sequence of removal operations is applied to the object's elements. To overcome this problem, the boundary configurations that are assumed to be sufficiently significant to originate end-points, should be identified at the beginning of each iteration of the object compression process, before applying the removal operations. In the opposite case, the sequential way of examining and deleting elements would change the geometry of the neighbourhood the elements are embedded in and may allow the creation of spurious end points, as well as an excessive shortening of significant branches. Effective criteria to correctly identify the tips and mark therein the elements, which will be the end points in the resulting thin set, can be based on the distance of boundary elements from the interior of the object at each iteration of thinning. Boundary subsets including elements whose distance from the interior of the object overcomes a given threshold are preserved from removal, as they correspond to significantly elongated object protrusions [30]. Alternatively, effective criteria can be based on the selection and preservation from removal of all centers of maximal balls in the distance map of the object. In fact, in correspondence with the tip of an object protrusion, a maximal ball of the object exists, whose boundary fits the boundary of the object protrusion for a (wide) connected portion. The center of such a maximal ball can be selected as the endpoint of the branch corresponding to the protrusion(for more details, see next Section).

Topological thinning guarantees connected skeletons; on the other hand, topological thinning does not obligatory produce perfectly thinned output (i.e. one-pixel/voxel-wide paths) since there exist arrangements of pixels/voxels which cannot be further eroded, unless altering object's topology. A 2D example is the lace-shaped object shown in Figure 15, whose border pixels are all non-removable. The alternative approach is based on distance map computation.

4.4 Skeletons from distance maps

Like thinning, skeletonisation based on distance maps is especially suitable for image processing and pattern recognition, and in general for the analysis of discrete objects represented

by grids of pixels or voxels. While thinning is mainly suited to elongated objects, distance map based skeletonisation is also suited to objects that are not elongated as well as to objects that have variable thickness, and provides a representation including also surfaces/branches originating from significant convexities of the boundary of the objects. Distance map based skeletonisation is more directly related to the Blum's notions of a symmetric point and a growth process. In fact, in the distance map the centers of the maximal balls can be easily detected and assigned to the skeleton. The detection of the remaining pixels/voxels necessary to guarantee that the skeleton has the same homotopy type as the object is also an easy task, due to the structure provided by the distance map to the portion of space occupied by the object. Differently from iterative thinning, which requires a number of iterations proportional to the object thickness and, hence, a generally large number of scans of the image when sequential computers are used, distance map based skeletonisation requires a small number of scans, independent of object thickness. Distance map based skeletonisation directly identifies and marks on the distance map of the object the elements that are recognized as belonging to the skeleton, due to the local configuration they are embedded in. The set of elements detected on the distance map includes all the centers of maximal balls (which implies skeleton reversibility), is symmetrically placed within the object and has the same topology as the object. This set is likely to be 2-element wide, in correspondence with object parts characterized by a thickness expressed by an even number of elements. To obtain the unit-wide skeleton, a final thinning, based on topology preserving removal operations, is necessary. We point out that in the three-dimensional case, the skeleton computed by means of the distance map is actually a surface-skeleton. For solid objects, the surface-skeleton can be furthermore compressed to a linear structure, the curve-skeleton, by using an iterative thinning, based on topology preserving removal operations, see e.g. [96]. The so obtained curve-skeleton, though providing a significant representation of the object's shape, does no longer allow object recovery.

In the distance map, each object point is labeled with its distance to the nearest background point. The distance of an element measures the length of a shortest path from that element to the background, where the path consists of elements linked to each other according to the selected connectivity type. Good approximations to the Euclidean distance are obtained by using weighted distances, where suitable integer weights are employed to compute the contribution given to the length of the path by the elements, depending on their relative positions (see, e.g., [24, 25]).

Ridges of the distance map are expected to belong to the skeleton, since they are centrally located within the object. Almost all the elements of a ridge are centers of maximal balls. As such, they can be identified by comparing the distance label of the element z at hand with the distance label of its neighbors, since this is equivalent to comparing the radii of the balls centered on z and on its neighbors. The extrema of a ridge, which are not necessarily centers of maximal balls, can be identified by taking into account that they are placed in saddle configurations. Their detection can be accomplished by counting for each element z, the number of components consisting of neighbors of z with distance labels larger than or equal to the distance label of z, and the number of components of neighbors of z with distance labels smaller than the distance label of z, respectively. Slopes connecting the ridges in the distance map are also expected to belong to the skeleton, to guarantee that the skeleton has the same homotopy type as the object. These linking elements can be found by growing, from the already detected skeletal elements, connecting paths according to the increasing value of the gradient in the distance map. For skeletonisation in the three-dimensional space, besides the linking elements necessary to guarantee skeleton connectedness, also further voxels necessary to prevent the creation of spurious tunnels have to be assigned to the surface skeleton. Roughly speaking, a distance map based skeletonisation algorithm includes three steps:

- distance map computation;
- identification of ridges and slopes;
- reduction of the set of ridges and slopes to unit width.

Obviously different skeletons are obtained depending on the chosen distance function. A number of algorithms can be found in the literature, each of which tailored to a specific distance function (as an example, see [4, 45, 48] for the two-dimensional case, and [49], for the three-dimensional case). Although all distance map based skeletonisation algorithms follow more or less the above scheme, ad hoc rules are often used (for instance to identify the centers of the maximal balls, or to obtain skeleton connectedness through the linking elements), which apply only to the specific distance case.

An important post-processing step is devoted to skeleton simplification [26] and pruning [45, 95]. Simplification is done in the three-dimensional case only, to remove from the surface-skeleton short peripheral curves, whose presence would only make the curve-skeleton structure unnecessarily complex. Pruning is done both in three and in two dimensions and should not be simply regarded as an optional step for a skeletonisation algorithm. In fact, pruning is useful to get rid of superfluous noisy branches and is indispensable to make the linear skeleton stable under object rotation, by eliminating those branches whose presence in the skeleton depends on object orientation. In turn, a post-processing aimed at improving skeleton aesthetics by removing zigzags mostly created by final thinning, can also be performed to favour the use of the skeleton for shape analysis.

In Figure 16, the skeleton of an object in the two-dimensional space is shown, which has been computed according to different distance functions. Namely, the Manhattan distance $d(1,2)$, the chessboard distance $d(1,1)$, the weighted distance $d(3,4)$, which assigns weights 3 and 4 to the steps in the path via the edge-neighbors and the vertex-neighbors respectively, and the weighted distance $d(5,7,11)$, which also consider as possible neighbors along the path pixels that can be reached with the knight move in the game of chess and assigns weights 5, 7 and 11 to the steps via edge-, vertex- and knight-neighbors along the path. In each row, from left to right, the nearly thin skeleton, the unit-wide skeleton and the skeleton resulting after pruning non significant peripheral branches are shown.

In Figure 17, a $3D$ object, its surface skeleton, computed according to D^6, and the curve skeleton obtained by furthermore compressing the surface skeleton by the algorithm [96] are shown. As said before, in the three-dimensional case only the surface skeleton is reversible.

5 Skeletons from topological structures

Methods grouped in this Section have in common the property of coding the evolution and the arrangement of the level set curves of a real, at least continuous, function defined on the shape. The most popular representative of this class of descriptors is the Reeb graph [89].

In principle, topological graphs give an abstract representation of the shape structure, with no information about the geometric embedding. Nevertheless, salient geometric information can be extracted from the shape and attached to the skeleton, thus obtaining a representation that is not only topological but retains also a geometric correspondence with the original shape. In this Section we overview the most popular techniques for constructing skeletons from topological structures related to level sets, distinguishing them in two main classes: those that derive from wave-like expansion techniques and those that more explicitly refer to the Reeb graph definition. Contour trees, a specific kind of Reeb Graphs for scalar fields, are treated in [19].

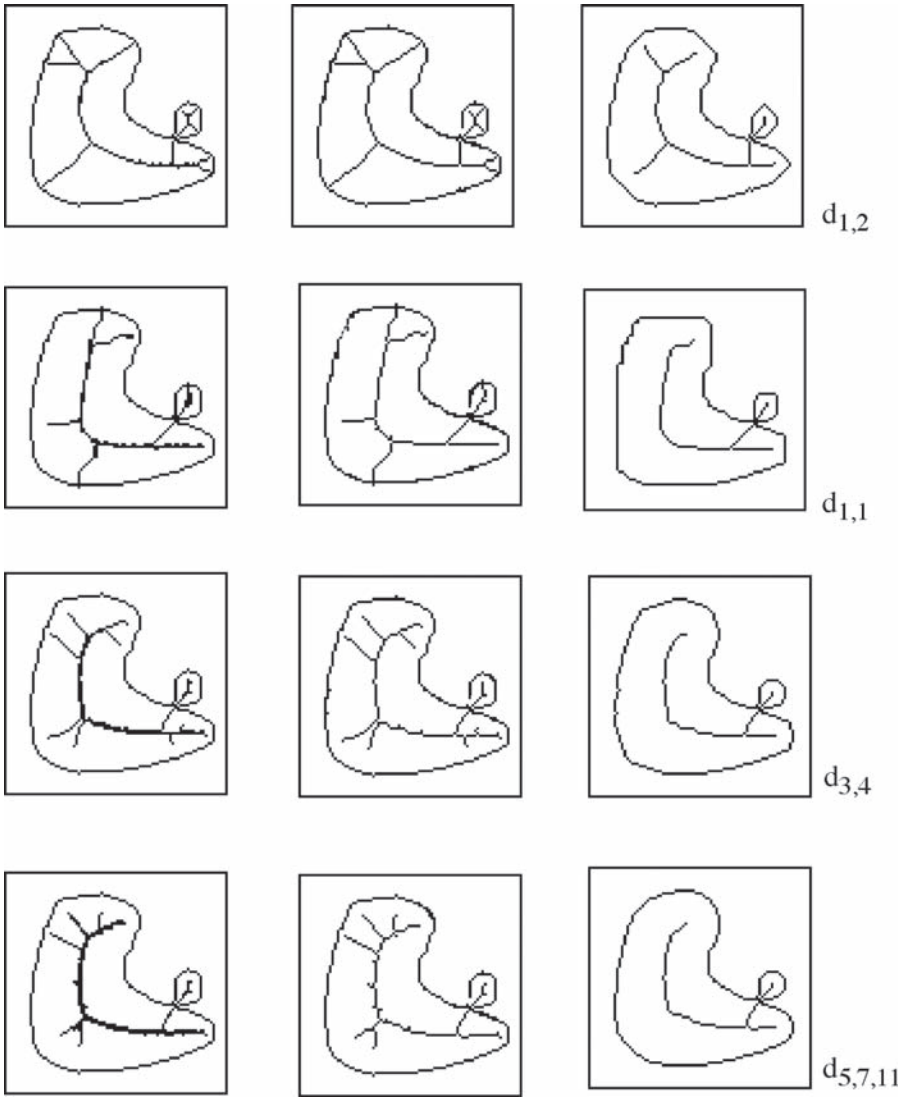
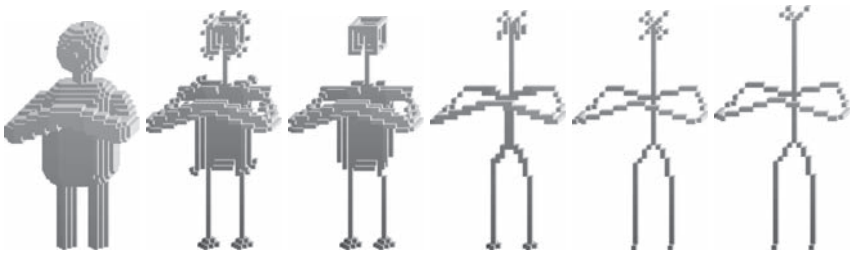


Fig. 16. From top to bottom, skeletons computed by using d(1,2), d(1,1), d(3,4) and d(5,7,11). From left to right, the nearly thin skeleton, the unit-wide skeleton and the skeleton resulting after pruning non significant peripheral branches.

5.1 Methods based on wavefront propagation

Algorithms belonging to this category compute each level set of a continuous function defined over the paradigm of a wave that originates in one point and propagates isotropically with

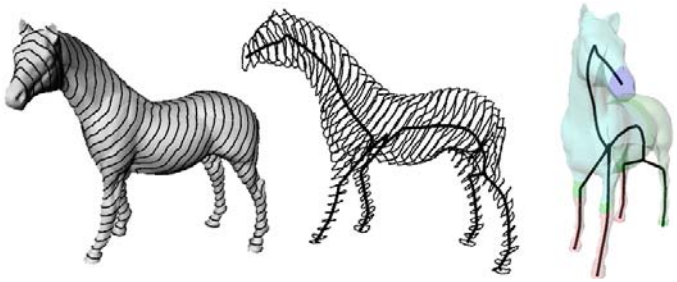


(a) (b) (c) (d) (e) (f)

Fig. 17. A 3D object, a), its surface-skeleton computed according to D6, b), the simplified surface-skeleton, c), the nearly thin curve-skeleton, d), the unit-wide curve-skeleton, e) and the pruned-curve-skeleton, f).

respect to a given function f in each direction on the surface. Points belonging to the same wave-front are characterized by the same function value by construction, and therefore define a level set of f.

The construction of the *Level Set Diagrams* from triangulated polyhedra proposed in [76] uses Euclidean distances for wave propagation [14]. In practice the wave traversal associates to each vertex of a triangle mesh the Euclidean length of the minimal path from that point and a source point. In particular, at the starting point the value of the wave traversal is zero. Each successive wave is a sub-complex and a subset of the *link* of the previous one. The wave propagation process continues until all vertices of the mesh have been selected using the Dijkstra algorithm for finding the paths of minimum length. The wave traversal may be also defined as a distance function. The seed point to start the wave propagation is automatically selected using a heuristic, which works well on elongate tubular shapes. In this case, skeletal lines obtained with different source points are very similar and the resulting skeleton is invariant under rotation, translation and scaling. Anyway, the choice of only one source point determines a privileged "slicing direction", which can lead to the loss of some features if the object is not tubular shaped (like the horse ears in Figure 18(b)).



(a) (b) (c)

Fig. 18. Isolevels (a) and the centreline (b,c) of the horse as computed as described in [76].

An extension of the approaches in [14] and [76] to non-zero genus surfaces has been presented in [72]. In this case, the evaluation of the measuring function, the mesh characterization

(based on local criteria) and the construction of the graph are performed at the same time using the Djikstra's algorithm. The independence on the object position makes this representation suitable to quadrangulate a surface. A similar extension to volume models with through holes has been presented by Wood et al. [103]; there, the graph is implicitly stored for generating high quality semi-regular and multi-resolution meshes from distance volumes. Also in this case, the object topology is achieved by considering a wavefront-like propagation from a seed point, [13] (see figure 19). The calculation of the isosurfaces is obtained by applying the Dijkstra's algorithm; this makes this approach unavailable for non-uniform scaling.

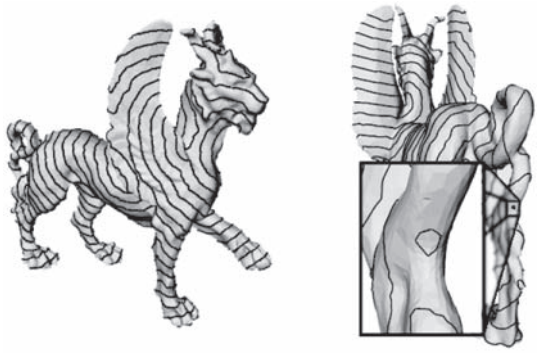
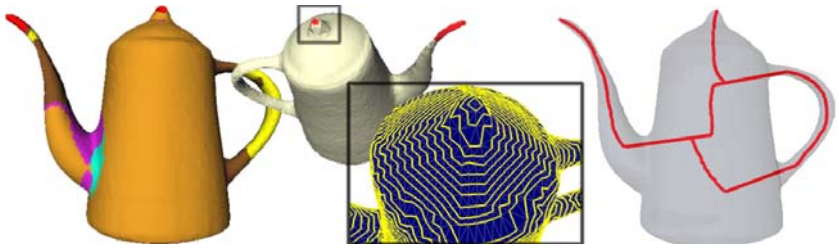


Fig. 19. Simulation of the wave-front propagation in [103].

Finally, a multi-resolution curvature evaluation is introduced in [82] to locate seed points which are sequentially linked by using the natural topological distance on the simplicial complex (see Figure 20(a,b) and also Figure CP-4(a,b) in Appendix E). More precisely, once computed the approximated Gaussian curvature for the mesh vertices, for each high curvature region R_i, $i = 1, \ldots, n$, a *representative vertex* p_i is selected.



(a) (b) (c) (d)

Fig. 20. (a) Vertex classification based on Gaussian curvature, (b) high curvature regions are highlighted; (c) topological rings expanded from centers of high curvature regions (d) resulting skeleton.

Starting at the same time from all representative vertices, waves made of vertices of increasing neighbourhoods are computed in parallel until the whole surface is covered (see Figure 20(c) and also Figure CP-4(c) in Appendix E), in a way similar to the wave-traversal

technique [13]. Waves growing from different seed points will collide and join where two distinct protrusions depart, thus identifying a branching zone; self-intersecting waves can appear expanding near handles and through holes. A skeleton is drawn according to the wave expansion: *terminal nodes* are identified by the *representative vertices*, while union or split of topological rings give *branching nodes*. Arcs are drawn joining the center of mass of all rings (see Figure 20(d) and also Figure CP-4(d) in Appendix E). This *curvature-based graph* graph is invariant to translation, rotation and scaling. On the other hand, if the curvature evaluation process does not recognize at least one feature region, e.g. surfaces with constant curvature value as spheres, this approach is not meaningful for extracting a description of the shape. Finally, this curve-line representation has at least as many cycles as the number of holes of the surface; however, some unforeseen cycles may appear in correspondence of the wavefront collisions.

5.2 Methods based on the Reeb graph

In the general case, the *Reeb graph* [89] of a n-dimensional manifold M under a mapping function f is defined as a quotient space, which identifies the levels sets of f. More formally: let $f : M \to \mathbb{R}$ be a real valued function on a compact manifold M. Then, the Reeb graph of M with respect to f is the quotient space of $M \times \mathbb{R}$ defined by the equivalence relation "$\sim$", which states that $(P, f(P)) \sim (Q, f(Q))$ iff:

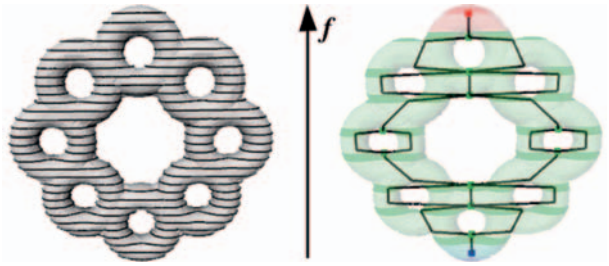
1. $f(P) = f(Q)$;
2. P, Q are in the same connected component of $f^{-1}(f(P))$.

Under the hypotheses that M is smooth and the function f is Morse and simple (i.e., its critical points have different values of f), Reeb demonstrated that the quotient space is a finite and connected simplicial complex of dimension 1, i.e., it is made of a connected collection of vertices and edges. The counter-image of each vertex is a singular connected component of the level sets of f, and the counter-image of an edge is homeomorphic to the topological product of one connected component of the level sets by $\mathbb{R}$ [89]. Under the same hypotheses, the number of cycles of the Reeb graph is an upper bound of the number of loops $\beta_1(M)$ on the manifold [39].

Even if the Reeb graph definition holds in any dimension, in this Chapter we mainly focus our attention to surfaces (bi-dimensional manifolds) embedded in $\mathbb{R}^3$. In the graph representation a node is defined for each creation, merging, split or deletion of a contour, that is, to topological changes affecting the number of connected components in the counter-image of f. Each arc joins two successive critical levels in their own component. If an arc connects two nodes, n_1 and n_2, then the topology of isolevels on M between the levels n_1 and n_2 does not change along the connected component of M joining the corresponding points [69].

From a computational point of view, a centreline representation of the abstract graph may be obtained associating to each contour its centroid; thus providing a geometric embedding of the structure. In this way, the structure roughly sketches the shape, even if some points of the skeletal structure may lye outside the shape. In addition, other geometric entities related to the contour form may be stored in each node so that the original shape may be approximately reconstructed from such a structure.

In Figure 21(a) the points drawn on the manifold represent the equivalence classes of a closed surface with respect to the height function highlighted. In Figure 21(b) the Reeb's quotient space is represented as a *traditional* graph, where the equivalence classes are grouped into arcs.



(a) (b)

Fig. 21. A surface, (a), and its Reeb graph representation with respect to the height function, (b).

From the application point of view, the properties of the Reeb graph strongly depend on those of the function f and the "best" choice for the function f depends on the application context. For a detailed overview on possible choices of the function f and their application in Computer Graphics we refer to [20].

Firstly introduced in Computer Graphics by Shinagawa et al. [94], Reeb graphs have initially been limited to Morse height functions. Methods for extracting Reeb-like graphs have been proposed in [94, 93, 98, 32, 73, 11, 85, 72, 39, 18, 102]. In this Section we focus on methods for constructing the Reeb graph representation of closed surfaces.

A first algorithm, proposed by Shinagawa et al. [93], automatically constructed the graph from surface contours generated by the height function. The extraction algorithm automatically generates the graph arcs relative to a one-to-one correspondence between cross section consisting of only one contour at first. Then the graph is completed using some heuristics based on a weight function and a priori knowledge on the surface genus. Main drawbacks of this algorithm are the need of a priori knowing the genus of the surface and the fact that this procedure is limited to contour levels of the height function [93]. In addition, since information on the shape between two consecutive cross sections is necessarily lost, the frequency of the contours of the surface is critical; therefore, a reasonable computation of the graph requires a high number of surface slices and it is time and space consuming ($O(n^2)$, where n represents the total number of vertices of the scattered contours).

The method proposed by Hilaga et al. in [73] provides a multi-resolution Reeb graph representation of triangle meshes which is independent of the object topology. The construction of the graph begins with the extraction of the graph at the finest resolution desired, then adjacency rules are used to complete the multi-resolution representation in a fine-to-coarse order. First of all, the domain of the mapping function is divided into a set of intervals. Second, triangles whose image under f lies in two intervals are subdivided so that the image of every triangle belongs to only one interval. Third, triangle sets, that is a connected component of triangles whose images belong to the same interval, are calculated. A node of the graph is associated to each triangle set. Then, arcs are detected by checking the region adjacency of triangle sets. The graph extraction is computed in $O(n + m)$ operations, where n and m represent, respectively, the number of triangles of the original mesh and those inserted during the subdivision phase. In Figure 22 an example of the Reeb graph construction method proposed in [73] is shown; in this case the domain of f is subdivided into 4 intervals. The contour insertion in 22(b) determines a set of mesh regions that correspond to the graph nodes 22(c), while their adjacency originates the arcs of the graph 22(d).

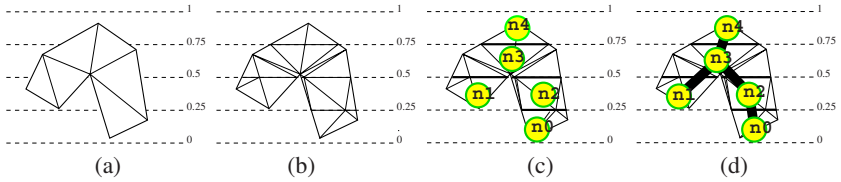
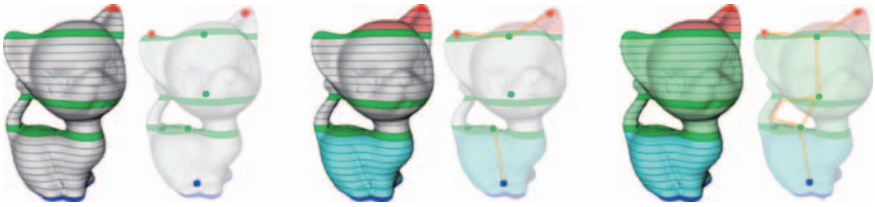


Fig. 22. Pipeline of the Reeb graph extraction in [73]. (b)

In [39] a method that performs also for non-orientable surfaces with or without boundaries, such as the Klein's bottle, has been proposed. Basic assumption of this approach is that the mapping function is Morse, thus critical points have pairwise different function values. Critical points are detected analysing the star of each vertex and non-simple critical points are simplified using the approach proposed in [53]. Once critical points have been identified, all vertices of the model are processed according the increasing value of the function f and the evolution of level sets is tracked. Since operations are done edges the complexity of the algorithm is $O(nlog(n))$, where n is the number of edges of the complex. An extension of this method has been proposed in [52] to analyse the evolution of the Reeb graph when the mapping function varies with time. In this case a point at infinity is added to make the space topologically equivalent to the 3-sphere so that the Reeb graph will be equivalent to a tree. Evolution with time of the graph is coded using a Jacobi curve that collects the birth-death points. Once a Reeb graph is computed, it is updated when an event occurs and stored in a data structure that code the entire evolution. Finally the computational cost of this approach, $O(N + En)$, depends on the number N of simplexes of the triangulation of the space-time data, the upper number n of simplexes at a time t and the amount E of birth-death and interchange events.

The approach proposed in [11, 17, 16] extracts a Reeb graph-like representation, called an *Extended Reeb graph* representation both from a surface with or without boundary through a finite set of contour levels of a given mapping function f. Since the contour levels decompose a surface S into a set of regions, the behavior of their boundaries is used TO detect critical areas and TO classify them as maximum, minimum and saddle areas. The characterization is performed by analysing the number of border components of each region and the values of the function f around them [11]. Critical areas correspond to nodes of the graph. Then arcs between nodes are constructed through an expansion process of the critical areas, in two phases: first arcs from minima/maxima to saddle areas, then the remaining links between saddle areas are inserted. In Figure 23 (see also Figure CP-5 in Appendix E) the main steps of the *ERG* extraction process are depicted; in Figure 23(a) to each critical area is associated a node; Figure 23(b) represents how the maximum (resp. the minimum) is connected to another critical area and the corresponding partial graph representation; finally, Figure 23(c) shows how the expansion process continues until the graph is completed.

On the basis of the ERG representation, a further extension of the domain of the Reeb graph to unorganized point clouds of 3D scan data has been proposed in [100]. The assumption on the point clouds is that they represent a human body. The limitation that the original data are not organized in a polygonal mesh is overcome assuming that the Euclidean distance among a point p and its closest point q, is smaller than a given threshold ϵ, $d(p, q) < \epsilon$. Point sets whose sampling is sufficiently fine are connected in a discrete sense. Therefore, level sets are defined as points that share the same value of the mapping function and are connected in the discrete sense. The resulting graph is called the *Discrete Reeb Graph (DRG)*.



(a) (b) (c)

Fig. 23. The recognition of the critical areas (a), the expansion of maxima and minima (b) and the complete graph. This model comes from the AIM@SHAPE repository: http://shapes.aim-at-shape.net/.

Finally, the method proposed in [102] has been proposed for topologically simplifying and repairing regularly sampled $3D$ grids of scalar values. That is a volumetric model in which each grid cube has 8 neighbor grid points. In this case, the data are swept with a set of parallel planes generating a set of *slices*, which are the sets of grid cubes bounded by two adjacent isosurfaces. Each connected component of a slice is called *ribbon*. The *contours* are given by the intersection of the isosurfaces the slicing plane. In particular, the graph described in this approach is called *augmented* Reeb graph because it codes also geometric information for each contour and each ribbon. Contour nodes in the graph correspond to a distance function traversal of the surface, and cycles, in addition to the geometric information stored in the ribbons, correspond to handles. The traversal is analyzed at discrete z intervals of the volumetric grid along the boundary of a distance function. Therefore, the planar slices are used as an ordered traversal through the slices an may be processed out-of-core of the volume data. Both ribbons and contours correspond to nodes of the Reeb graph while their adjacency is coded in the edges. To avoid that a handle is completely contained within a ribbon, the Euler characteristic of each isosurface component is computed and the sweep is locally refined. In this way the topology of the volume is completely coded and, in each interval, there is the correspondence of the Reeb graph structure with its Euler characteristic.

6 Conclusions and future developments

In this Chapter we have briefly sketched a wide variety of skeletal structures defined in Computer Graphics and Computer Vision. As discussed throughout the Chapter, there is one main structure often referred to as the skeleton, the medial axis transform, and a huge quantity of very similar skeletons that exhibit some (often very small) modifications. Being the MAT unfortunately hard to be computed in the general case and unstable to small perturbations of the shape, a large number of variations of the MAT were introduced: some of them are just approximations of the MAT for facilitating the skeleton computation (e.g. MAT computation through Voronoi diagrams), while others come from different definitions and present different properties. A few descriptors are able to represent the exact medial axis for a small category of input shapes, like the bisectors of parametric curves and surfaces.

In the 3D case the distinction between the MAT and others skeletal structures becomes more evident: in fact, while the MAT in 3D is essentially a medial surface, in many applications a linear skeleton may be preferable. This is the case of path planning for medical applications in which a linear skeleton, as far as possible from the shape boundary, is needed,

maybe to plan the inspection of a human organ [99]. The definitions and properties of these linear 3D skeletons depend mostly on the input data type: for discrete representations like collection of voxels, distance maps and thinning techniques are used; wave-front propagation and level set approaches are preferred in the continuum case.

With reference to the properties that should characterize a descriptor, we highlight that all the skeletal structure described in this Chapter provide a *dimensional reduction* of the original representation. From the storage point of view the MAT (and the shock graph) gives a representation which is also *invertible*, thus paying in terms of spatial and computational costs. Bisectors (that may be seen as an over set of the medial axis) provide the most complete information among the structures considered in this work, but, in practice, their effective computation is limited to a few classes of models. *Linear* structures, like centrelines provided by thinning or distance maps computations, provide a very *compact* and *concise* representation of the shape, even in case of 3D data; unfortunately loosing the *invertibility* property. Nevertheless, as all medial representations, discrete centrelines satisfy the property of being always inside (*centrality*) the shape, which made them popular in applications related to shape animation, deformation and retrieval.

Other structures that are defined on the basis of a real function, like Reeb graphs, capture the topology of the shape. They can always be represented as graphs, eventually enriched with additional geometric attributes, but they are only able to approximate the original dataset. Depending on the application context, the flexibility in the choice of the functions makes these descriptors tunable to different application domains. In particular, there is a growing interest on the definition of functions that do not depend on the (geometric) embedding of the shape, like the so-called Laplacian eigenvalues [50].

In every application context, the choice of the most suitable skeleton must cope with efficiency and expressive power of the representation, and we tried to underline these aspects in the Chapter. Provided that a function, which is independent of object rigid transformations, is suitable for recognition tasks, the Reeb graph it could be anyway preferable to the MAT, which is usually complex (in terms of number of nodes and edges) and is unstable to small perturbations of the shape, thus giving very different skeletons also for similar shapes. Therefore, the relatively simple but topologically effective structure of these descriptions has suggested a large use of them in shape matching and retrieval tasks.

About the *stability* of the representation we observe that bisectors, medial axes and shock graphs intrinsically depend on small shape modifications. Nevertheless, as discussed in Section 4.1, this problem has been partially overcome with pruning strategies that are stable under small shape perturbations. Discrete centrelines derived from thinning or distance maps are usually *robust* to small shape variations.

As far as computational issues are concerned, in table 3 we briefly summarize the complexity of the algorithms described in the Chapter.

In particular, we point out that the complexity of the bisectors may be expressed in terms of degree of the parametric representation. In particular, given two polynomial parametric curves in the plane of degree m, the bisector curve is represented as an implicit function of degree $4m - 2$. For example, for two cubic curves ($m = 3$), the bisector is represented as

[1] For point sets with provable small Voronoi graph, F is reasonably small: $F = O(n \log n)$ [9] or $F = O(n)$ [8].

[2] Once a set of seed points has been recognized, the complexity of the skeleton extraction is linear in the number of mesh vertices but an accurate evaluation of the high curvature points has quadratic cost.

Summary		
Approach	**Description**	**Costs**
[33]	Voronoi graph	$O(n^{\lceil \frac{k}{2} \rceil}) + n \log n$
[34]	Voronoi graph	$O((n+F)\log^2 F)$[1]
[77]	Medial axis of a polygon	$O(n \log n$
	Discrete skeleton 2D images	$O(n^2)$
	Discrete skeleton 3D images	$O(n^3)$
[1]	Straight skeleton	$O(nr \log n)$
[60]	Straight skeleton	$O(n^{1+\epsilon} + n^{8/11+\epsilon} r^{9/11+\epsilon})$
[37]	Straight skeleton	$O(n \log^2 n + r\sqrt{r} \log r)$
[97]	Linear Axis	$O(n)$
[76, 72]	Centerline	$O(n \log(n))$
[103]	Centerline	$O(n \log(n))$
[81][2]	Centerline	$O(n)$
[93, 94]	Reeb graph	$O(n^2)$
[73]	Reeb graph	$O((n+m))$
[11, 16]	Reeb graph	$O(\max(m+n, n \log(n)))$
[39]	Reeb graph	$O(n \log(n))$
[102]	Reeb graph	$O(n \log(n))$

Table 3. Classification of the methods for skeleton extraction. Symbols: n represent the number of vertices or points or pixels (voxels); m the number of vertices inserted in the mesh during an eventual contouring phase; e the number of edges in the neighborhood tree, r is the number of reflex vertices.

an implicit of degree 10. Hence, the Voronoi cells and diagrams of cubic curves could be represented as subregions of degree 10 implicits.

To conclude, we would like to emphasize that skeletal structures will play a fundamental role in the development of specific tools for the (future) semantic annotation of shapes, or shape parts, according to the concepts formalised by a domain ontology. In fact, the computation of a skeleton and the extraction of features automatically provide a way for decomposing a shape into significant parts, which may be further analysed and annotated.

References

1. O. Aichholzer, D. Alberts, F. Aurenhammer, and B. Gartner. A novel type of skeleton for polygons. *Journal of Universal Computer Science*, 1:752–761, 1995.
2. N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, 22:481–504, 1999.
3. N. Amenta, S. Choi, and R.K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19:127–153, 2001.
4. C. Arcelli and G. Sanniti di Baja. Euclidean skeleton via center-of-maximal-disc extraction. *Image and Vision Computing*, 11:163–173, 1993.
5. D. Attali. *Squelettes et graphes de Voronoi 2-D et 3-D*. PhD thesis, University Joseph Fourier, 1995.

6. D. Attali, J.-D. Boissonnat, and H. Edelsbrunner. Stability and computation of medial axes — A state-of-the-art report. In T. Möller, B. Hamann, and B. Russell, editors, *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*. Springer-Verlag, 2005. To appear.
7. D. Attali and J.D. Boissonnat. Complexity of the Delaunay triangulation of points on polyhedral surfaces. *Discrete and Computational Geometry*, 30(3):437–452, 2003.
8. D. Attali and J.D. Boissonnat. A linear bound on the complexity of the Delaunay triangulation of points on polyhedral surfaces. *Discrete and Computational Geometry*, 31:369–384, 2004.
9. D. Attali, J.D. Boissonnat, and A. Lieutier. Complexity of the Delaunay triangulation of points on surfaces: the smooth case. In *SCG '03: Proc. of the 19th Annual Symposium on Computational Geometry 2003*, pages 201–210. ACM Press, 2003.
10. D. Attali and A. Montanvert. Modeling noise for a better simplification of skeletons. In *ICIP '96: Proc. of the International Conference on Image Processing*, volume 3, pages 13–16, 1996.
11. M. Attene, S. Biasotti, and M. Spagnuolo. Shape understanding by contour-driven retiling. *The Visual Computer*, 19(2-3):127–138, 2003.
12. F. Aurenhammer and H. Imai. Geometric relations among Voronoi diagrams. *Geometria Dedicata*, 27:65–75, 1988.
13. U. Axen and H. Edelsbrunner. Auditory Morse analysis of triangulated manifolds. In *Mathematical Visualization*, pages 223–236. Springer-Verlag, 1998.
14. Ulrike Axen. Computing Morse functions on triangulated manifolds. In *SODA '99: Proc. of the 10th ACM-SIAM Symposium on Discrete Algoritms 1999*, pages 850–851. ACM Press, 1999.
15. M. Bern, D. Eppstein, P.K. Agarwal, N. Amenta, P. Chew, T. Dey, D.P. Dobkin, H. Edelsbrunner, C. Grimm, L.J. Guibas, J. Harer, J. Hass, A. Hicks, C.K. Johnson, G. Lerman, D. Letscher, P. Plassmann, E. Sedgwick, J. Snoeyink, J. Weeks, C. Yap, and D. Zorin. Emerging challenges in computational topology. In Report from the NSF-funded Workshop on Computational Topology, 1999.
16. S. Biasotti. *Computational Topology Methods for Shape Modelling Applications*. PhD thesis, Università degli Studi di Genova, May 2004.
17. S. Biasotti. Reeb graph representation of surfaces with boundary. In *SMI '04: Proc. of Shape Modeling Applications 2004*, pages 371–374, Los Alamitos, Jun 2004. IEEE Computer Society.
18. S. Biasotti, B. Falcidieno, and M. Spagnuolo. Surface shape understanding based on extended Reeb graphs. In *Topological Data Structures for Surfaces: An Introduction for Geographical Information Science*, pages 87–103. John Wiley and Sons, 2004.
19. S. Biasotti, L. De Floriani, B. Falcidieno, and L. Papaleo. Morphological representations of scalar fields. In *Shape Analysis and Structuring*. Springer, 2007.
20. S. Biasotti, S. Marini, M. Mortara, and G. Patane. An overview on properties and efficacy of topological skeletons in shape modelling. In M.S. Kim, editor, *SMI '03: Proc. of Shape Modeling International 2003*, pages 245–254, Los Alamitos, May 2003. IEEE Computer Society.
21. H. Blum. A transformation for extracting new descriptors of shape. In Weiant Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form. Proc. of a Symposium*, pages 362–380, Cambridge MA, Nov 1967. MIT Press.
22. J.D. Boissonnat and F. Cazals. Natural neighbor coordinates of points on a surface. *Computational Geometry-Theory and Applications*, 19(2-3):155–173, 2001.

23. J.D. Boissonnat and M. Karavelas. On the combinatorial complexity of Euclidean Voronoi cells and convex hulls of d-dimensional spheres. In *SODA '03: Proc. of the 14th ACM-SIAM Symposium on Discrete Algoritms*, pages 305–312. ACM Press, 2003.
24. G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34:344–371, 1986.
25. G. Borgefors. On digital distance transform in three dimensions. *Computer Vision and Image Understanding*, 64(3):368–376, 1996.
26. G. Borgefors, I. Nyström, G. Sanniti di Baja, and S. Svensson. Simplification of 3D skeletons using distance information. In L. J. Latecki, R. A. Melter, D. M. Mount, and A.Y. Wu, editors, *Proc. of SPIE - Vision Geometry IX*, volume 4117, pages 300–309, San Diego - USA, 2000.
27. J.W. Brandt. Convergence and continuity criteria for discrete approximations of the continuous planar skeletons. *CVGIP: Image Understanding*, 59:116–124, 1994.
28. J.W. Brandt and V.R. Algazi. Continuous skeleton computation by Voronoi diagram. *CVGIP: Image Understanding*, 55:329–337, 1992.
29. C. Burnikel. *Exact Computation of Voronoi Diagrams and Line Segment Intersections*. Ph.D thesis, Universität des Saarlandes, March 1996.
30. C.Arcelli and G. Sanniti di Baja. A thinning algorithm based on prominence detection. *Pattern Recognition*, 13(3):225–235, 1981.
31. C.Arcelli and G. Sanniti di Baja. Skeletons of planar patterns. In T.Y. Kong and A. Rosenfeld, editors, *Topological Algorithms for Digital Image Processing*, pages 99–143. North-Holland, 1996.
32. H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In *SODA '00: Proc. of the 11th ACM-SIAM Symposium on Discrete Algoritms 2000*, pages 918–926. ACM Press, 2000.
33. *The CGAL 3.1 User Manual*.
34. T.M. Chan, J. Snoeyink, and C.K. Yap. Primal dividing and dual pruning: Output-sensitive construction of 4-D polytopes and 3-D Voronoi diagrams. *Discrete and Computational Geometry*, 18:433–454, 1997.
35. F. Chazal and A. Lieutier. Stability and homotopy of a subset of the medial axis. In *SMA '04: Proc. of the 9th ACM Symposium on Solid Modeling and Applications 2004*, pages 243–248. ACM Press, 2004.
36. F. Chazal and R. Soufflet. Stability and finiteness properties of medial axis and skeleton. *Journal on Control Dynamics and Systems*, 10:149–170, 2004.
37. S.W. Cheng and A. Vigneron. Motorcycle graphs and straight skeletons. In *SODA '02: Proc. of the 13th ACM-SIAM Symposium on Discrete Algoritms 2002*, pages 156–165. ACM Press, 2002.
38. S.W. Choi and H.P. Seidel. Linear one-sided stability of MAT for weakly injective 3D domain. In *SMA '02: Proc. of the 7th ACM Symposium on Solid Modeling and Applications 2002*, pages 344–355. ACM Press, 2002.
39. K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in Reeb graphs of 2-manifolds. In *SCG '03: Proc. of the 19th Annual Symposium on Computational Geometry 2003*, pages 344–350. ACM Press, 2003.
40. N. D. Cornea, D. Silver, and P. Min. Curve-skeleton applications. In *Proceedings IEEE Visualization*, pages 95–102, 2005.
41. T. Culver. *Computing the medial axis of a polyhedron reliably and efficiently*. PhD thesis, University North Carolina, Chapel Hill, North Carolina, 2000.
42. T. Culver, J. Keyser, and D. Manocha. Exact computation of the medial axis of a polyhedron. *Computer Aided Geometric Design*, 21(1):65–98, 2004.

43. T. Dey and j. Sun. Defining and computing curve-skeletons with medial geodesic function. In *Proceedings of the Symposium on Geometry Processing*, pages 143–152, 2006.
44. T.K. Dey and W. Zhao. Approximating the medial axis from the Voronoi diagram with a convergence guarantee. *Algorithmica*, 38, 2004.
45. G. Sanniti di Baja. Well-shaped, stable and reversible skeletons from the (3,4)-distance transform. *Visual Communication and Image Representation*, 5:107–115, 1994.
46. G. Sanniti di Baja. Representing shape by line patterns. In P. Wang and A. Rosenfeld, editors, *Advances in Structural and Syntactical Pattern Recognition*, volume 1121 of *Lecture Notes in Computer Science*, pages 230–239. Springer-Verlag, 1996.
47. G. Sanniti di Baja and I. Nyström. Skeletonization in 3D discrete binary images. In C.H. Chen and P.S.P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*, Chapter 2.2, pages 137–156. World Scientific, Singapore, 3rd edition, January 2005.
48. G. Sanniti di Baja and E. Thiel. Skeletonization algorithm running on path-based distance maps. *Image and Vision Computing*, 14:47–57, 1997.
49. G.Sanniti di Baja and S. Svensson. Surface skeletons detected on the D6 distance transform. In F.J. Ferri et al., editor, *Proc. of SSSPR'2000 - Advances in Pattern Recognition*, volume 1121, pages 387–396, Alicante, 2000. LNCS, Springer-Verlag.
50. S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. Hart. Spectral surface quadrangulation. *ACM Transactions on Graphics*, 25(3):1057–1066, August 2006.
51. D. Dutta and C. Hoffmann. On the skeleton of simple CSG objects. *ASME J. of Mechanical Design*, 115:87–94, 1993.
52. H. Edelsbrunner, J. Harer, A. Mascarenhas, and V. Pascucci. Time-varying Reeb graphs for continuous space-time data. In *Proceeding of the 20-th ACM Symposium on Computational Geometry*, pages 366–372, 2004.
53. H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete and Computational Geometry*, 30:87–107, 2003.
54. G. Elber and M. S. Kim. Bisector curves of planar rational curves. *Computer Aided Design*, 30(14):1089–1096, December 1998.
55. G. Elber and M. S. Kim. The bisector surface of freeform rational space curves. *ACM Trans. on Graphics*, 17(1):32–50, January 1998.
56. G. Elber and M. S. Kim. Computing rational bisectors. *Computer Graphics and Applications*, 19(6):76–81, November-December 1999.
57. G. Elber and M. S. Kim. Rational bisectors of CSG primitives. In *The Fifth ACM/IEEE Symposium on Solid Modeling and Applications, Ann Arbor, Michigan*, pages 159–166, June 1999.
58. G. Elber and M. S. Kim. A computational model for nonrational bisector surfaces: Curve-surface and surface-surface bisectors. In *Geometric Modeling and Processing 2000, Hong Kong*, pages 364–372, April 2000.
59. G. Elber and M. S. Kim. Geometric constraint solver using multivariate rational spline functions. In *The Sixth ACM/IEEE Symposium on Solid Modeling and Applications, Ann Arbor, Michigan*, pages 1–10, June 2001.
60. D. Eppstein and J. Erickson. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discrete and Computational Geometry*, 22:569–592, 1999.
61. R. Farouki and J. Johnstone. The bisector of a point and a plane parametric curve. *Computer Aided Geometric Design*, 11(2):117–151, April 1994.
62. R. Farouki and R. Ramamurthy. Specified-precision computation of curve/curve bisectors. *Int. J. of Computational Geometry & Applications*, 8(5-6):599–617, October-December 1998.

63. R. Farouki and R. Ramamurthy. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries I. theoretical foundations. *J. of Computational and Applied Mathematics*, 102:119–141, 1999.
64. R. Farouki and R. Ramamurthy. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries II. detailed algorithm description. *J. of Computational and Applied Mathematics*, 102:119–141, 1999.
65. S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.
66. P. Giblin and B.B. Kimia. A formal classification of 3D medial axis points and their local geometry. In *CVPR 2000: Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2000*, volume 1, pages 566–573, Los Alamitos, 2000. IEEE Computer Society.
67. P.J. Giblin and B.B. Kimia. On the local form and transitions of symmetry sets, medial axes, and shocks. *International Journal of Computer Vision*, 54:143–157, 2003.
68. S. Goswami, T. K. Dey, and C. L. Bajaj. Identifying flat and tubular regions of a shape by unstable manifolds. In *SPM '06: Proceedings of the 2006 ACM symposium on Solid and physical modeling*, pages 27–37, New York, NY, USA, 2006. ACM Press.
69. A. Gramain. *Topologie des surfaces*. Presses Universitaires de France, 1971.
70. I. Hanniel, R. Muthuganapathy, G. Elber, and M. S. Kim. Precise Voronoi cell extraction of free-form rational planar closed curves. In *ACM Symposium on Solid and Physical Modeling*, pages 51–59, June 2005.
71. M. Held. Vroni: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. *Computational Geometry: Theory and Applications*, 18:95–123, 2001.
72. F. Hetroy and D. Attali. Topological quadrangulations of closed triangulated surfaces using the Reeb graph. *Graphical Models*, 65(1-3):131–148, 2003.
73. M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes Los Angeles, CA. *Computer graphics proceedings, annual conference series: SIGGRAPH conference proceedings*, pages 203–212, Aug 2001.
74. D.S. Kim, Y. Cho, and D. Kim. Euclidean Voronoi diagram of 3D balls and its computation via tracing edges. In *Computer Aided Design*, pages 1412–1424, November 2005.
75. B. Kimia, A. Tannenbaum, and S. Zucker. Shapes, shocks, and deformations, I: The components of shape and the reaction-diffusion space. *International Journal of Computer Vision*, 15:189–224, 1995.
76. F. Lazarus and A. Verroust. Level set diagrams of polyhedral objects. In W.F. Bronsvoort and D.C. Anderson, editors, *SMA '99: Proc. of the 5th ACM Symposium on Solid Modeling and Applications 1999*, pages 130–140. ACM Press, 1999.
77. D. T. Lee. Medial axis transformation of a planar shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(4):363–369, 1982.
78. A. Lieutier. Any open bounded subset of $\mathbb{R}^n$ has the same homotopy type as its medial axis. In *Proc. 8th ACM Sympos. Solid Modeling Appl.*, pages 65–75. ACM Press, 2003.
79. G. Matheron. Examples of topological properties of skeletons. In *Image Analysis and Mathematical Morphology, Volume 2: Theoretical Advances*, pages 217–238. Academic Press, 1988.
80. J. Milnor. *Morse Theory*. Princeton University Press, New Jersey, 1963.
81. M. Mortara and G. Patané. Shape-covering for skeleton extraction. *International Journal of Shape Modelling*, 8:245–252, 2002.

82. M. Mortara, G. Patane, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. *Algorithmica*, 38(1):227–248, 2004.
83. R.L. Ogniewicz. Skeleton-space: A multi-scale shape description combining region and boundary information. In *CVPR '94: Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 1994*, pages 746–751, Los Alamitos, 1994. IEEE Computer Society.
84. R.L. Ogniewicz and O. Kubler. Hierarchic Voronoi skeletons. *Pattern Recognition*, 28:343–359, 1995.
85. V. Pascucci and K. Cole-McLaughlin. Parallel computation of the topology of level sets. *Algorithmica*, 38:249–268, 2003.
86. M. Peternell. Geometric properties of bisector surfaces. In *Graphical Models and Image Processing*, 2000.
87. M. Ramanathan and B. Gurumoorthy. Constructing medial axis transform of planar domains with curved boundaries. *Computer Aided Design*, 35:619–632, 2002.
88. M. Ramanathan and B. Gurumoorthy. Constructing medial axis transform of extruded and revolved 3D objects with free-form boundaries. *Computer-Aided Design*, 37(13):1370–1387, 2005.
89. G. Reeb. Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences*, 222:847–849, 1946.
90. A. Rosenfeld. Digital geometry: Introduction and bibliography. In *Advances in Digital and Computational Geometry*, 1998.
91. E. Sherbrooke, N. M. Patrikalakis, and F.-E. Wolter. Differential and topological properties of medial axis transforms. *Graphical Models and Image Processing*, 58:574–592, 1996.
92. E.C. Sherbrooke, N.M. Patrikalakis, and E. Brisson. An algorithm for the medial axis transform of 3D polyhedral solids. *IEEE Trans. on Visualization and Computer Graphics*, 22:44–61, 1996.
93. Y. Shinagawa and T.L. Kunii. Constructing a Reeb graph automatically from cross sections. *IEEE Computer Graphics and Applications*, 11:44–51, 1991.
94. Y. Shinagawa, T.L. Kunii, and Y.L. Kergosien. Surface coding based on Morse theory. *IEEE Computer Graphics and Applications*, 11:66–78, 1991.
95. S. Svensson and G. Sanniti di Baja. Simplifying curve skeletons in volume images. *Computer Vision and Image Understanding*, 90:242–257, 2003.
96. S. Svensson, I. Nystr om, and G. Sanniti di Baja. Curve skeletonization of surface-like objects in 3D images guided by voxel classification. *Pattern Recognition Letters*, 23(12):1419–1426, 2002.
97. M. Tanase and R. C. Veltkamp. A straight skeleton approximating the medial axis. *Lecture Notes in Computer Science*, 3221:809–821, Sep 2004.
98. M. van Kreveld, R. Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface transversal. In *SCG '97: Proc. of the 13th Annual Symposium on Computational Geometry 1997*, pages 212–220. ACM Press, Jun 1997.
99. M. Wan, Z. Liang, Q. Ke, L. Hong, I. Bitter, and A. Kaufman. Automatic centerline extraction for virtual colonoscopy. *IEEE Trans. on Medical Imaging*, 21(12):1450–1460, December 2002.
100. N. Werghi, Y. Xiao, and J. P. Siebert. A functional-based segmentation of human body scans in arbitrary postures. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 36(1):153–165, 2006.

101. F.E. Wolter. Cut locus & medial axis in global shape interrogation & representation. Technical Report Design Laboratory Memorandum 92-2, MIT, 1992.
102. Z. Wood, H. Hoppe, M. Desbrun, and P. Schroeder. Removing excess topology from isosurfaces. *ACM Trans. on Graphics*, 23:190–208, 2004.
103. Z.J. Wood, M. Desbrun, P. Schroder, and D. Breen. Semi-regular mesh extraction from volumes. In *VIS 2000: Proc. of IEEE Conference on Visualization 2000*, pages 275–282, Los Alamitos, 2000. IEEE Computer Society.

Morphological Representations of Scalar Fields

Silvia Biasotti[1], Leila De Floriani[2], Bianca Falcidieno[1], and Laura Papaleo[2]

[1] Istituto di Matematica Applicata e Tecnologie Informatiche, - Italian National Research Council, Genova (Italy) {silvia,bianca}@ge.imati.cnr.it
[2] Department of Computer Science (DISI) - University of Genova, Genova (Italy) {deflo,papaleo}@disi.unige.it

Summary. We consider the problem of representing and extracting morphological information from scalar fields. We focus on the analysis and comparison of algorithms for morphological representation of both 2D and 3D scalar fields. We review algorithms which compute a decomposition of the domain of a scalar field into a *Morse* and *Morse-Smale complex* and algorithms which compute a topological representation of the level sets of a scalar field, called a *contour tree*. Extensions of the morphological representations discussed in the chapter are briefly discussed.

1 Introduction

The problem of representing morphological information extracted from discrete scalar fields is a very relevant issue in several applications, such as terrain modeling and volume data analysis and visualization.

Discrete scalar fields are defined by a finite set of points in a domain D in R^d, at each of which a value of a scalar function f is given. Traditionally, discrete scalar fields are described by decomposing their domain into cells, on which an interpolating function is based on discrete function values given as input. The discretization of the domain is often obtained through a simplicial mesh (such as a triangle, or a tetrahedral mesh), or through a regular grid formed by square cells in 2D, or by hexahedral cells in 3D.

This geometry-based description provides an accurate representation of a scalar field, but it fails in capturing the morphological structure of the field defined by its critical points, and by its integral lines and surfaces. Beside being compact, a morphological description supports also a knowledge-based approach to analyze, visualize and understand the scalar field behavior (in space and time), as required, for instance, in visual data mining applications. Since topology focuses on qualitative properties of spaces (such as their connectedness or the number and types of their holes), it is the best tool to describe the shape of a mathematical model at a high level of abstraction. Specifically, Morse theory deals with the analysis of geometric shapes and on the extraction of synthetic shape abstractions, preserving their topological properties as well as their main morphological characteristics.

In the last decades, there has been a considerable amount of research on extracting critical features from grey-scale images and terrain models, and a more limited amount of work in the case of 3D scalar fields. Also watershed algorithms, originally developed for image

segmentation, can be applied to extract critical features from a scalar field [53, 85, 75, 8]. A survey of watershed techniques can be found in [84, 69]. While most of watershed algorithms are focused on 2D regular grids, there are some algorithms which extend the watershed approach to triangular meshes [50]. More recent work in computational geometry concentrates on representing the morphology of a d-dimensional scalar field through a decomposition of its domain into d-dimensional cells bounded by cells of lower dimensionality, forming a so-called *Morse-Smale decomposition*. [36, 77]. These techniques are rooted in Morse theory and try to simulate in the discrete case the Morse-Smale decomposition defined for C^2-differentiable functions [54, 73]. Another fundamental way for analyzing a scalar field is to extract its level sets. The contour tree is a topological abstraction that encodes the evolution and the arrangement of the contour lines [16, 63, 20, 23]. The contour tree provides a more concise description of the topology of a scalar field, and, unlike a Morse-Smale complex, it does not encode geometric information related to the flow of the gradient of the field.

In this chapter, we focus on the analysis of algorithms for extracting structural information from 2D and 3D scalar fields. Specifically, we will review: (i) algorithms for extracting critical points; (ii) algorithms for computing cellular decompositions of the domain of a scalar field which capture the configuration of the critical points and the integral lines connecting them as well as (iii) algorithms for computing the contour tree. We analyze and classify these methods based on the dimension of the scalar field (2D or 3D) and on the digital model underlying the scalar field. Morphological representations for vector fields are treated in Chapter [81].

The remainder of the chapter is organized as follows. Section 2 reviews some background notions. Section 3 presents theoretical results and algorithms for extracting critical points. Section 4 reviews algorithms for the extraction of a decomposition of the domain of a scalar field into a Morse-Smale complex, while Section 5 reviews algorithms for extracting a contour tree. Section 6 draws some concluding remarks and discusses open problems.

2 Background Notions

In this Section, we introduce some background notions required in the rest of this chapter. First, we briefly review some concepts from algebraic topology on cell and simplicial complexes (see [52] for a complete treatment of this subject). Then, we formalize the notion of digital model of scalar fields in a dimension-independent way (see [33]). We introduce some basic notions of Morse theory in d-dimensions, although our main interest will be in the 2D and 3D cases (see [54, 73], for more details). We then introduce *Morse* and *Morse-Smale* complexes for a C^2-differentiable real-valued function f defined over a domain $D \subseteq \mathbb{R}^d$ [36]. Finally, we discuss a representation of the level sets of a scalar field, called the *contour tree* [20], or, in case of generic manifolds, a *Reeb graph* [68, 72]. In Table 1 we list some of the most important symbols used in the rest of the chapter together with their meaning.

2.1 Cell and Simplicial Complexes

Intuitively, a Euclidean cell complex is a collection of basic elements, called cells, which cover a domain in the Euclidean space [51]. A *k-dimensional cell* (k-cell) γ in the Euclidean space R^d, $1 \leq k \leq d$, is a subset of R^d homeomorphic to an open k-dimensional ball $B^k = \{x \in \mathbb{R}^k : ||x|| < 1\}$[3]. A 0-cell is a point in R^d, k is called the *order*, or *dimension*, of a k-cell γ.

[3] $||x||$ denotes the norm of vector x

Table 1. Symbols used throughout the chapter

Symbol	Description
D	Domain
R^d	d-dimensional Euclidean Space
γ	k-dimensional cell
B^k	k-dimensional ball
Γ	Euclidean cell complex
Σ	k-dimensional simplicial complex
σ	simplex
V	finite set of points in R^d
S	scalar dataset
M	digital model
∇f	gradient vector of f
λ	index of a critical point
$W^s(p)$	stable manifold
$W^u(p)$	unstable manifold
S_N	surface network

A *Euclidean cell complex* is a finite set Γ of cells of dimension at most k in R^d, $0 \leq k \leq d$, such that the interiors of the cells of Γ are disjoint, and if $\gamma, \gamma_1 \in \Gamma$, such that $\gamma \cap \gamma_1 \neq \emptyset$, then $\gamma \cap \gamma_1$ is the disjoint union of cells of Γ.

The maximum k of the dimensions of the cells γ over all cells of a complex Γ is called the *dimension*, or the *order*, of the complex. The *domain* (or *carrier*) $\Delta\Gamma$ of a Euclidean cell complex Γ is the subset of R^d spanned by the cells of Γ. The *relative boundary* $b(\gamma)$ of a k-cell γ, $1 \leq k \leq d$, is the boundary of γ with respect to the topology induced by the usual topology of R^d. Note that the relative boundary of a 0-cell is empty. The *combinatorial boundary* $B(\gamma)$ of a cell γ is the collection of all cells γ' of Γ such that $\gamma' \subseteq b(\gamma)$ (as a point set). An h-cell γ' which belongs to the combinatorial boundary $B(\gamma)$ of a cell γ is called an *h-face* of γ. If $\gamma' \neq \gamma$, then γ' is called a *proper face* of γ. Note that each cell γ is a face of itself. The *star* (or *combinatorial co-boundary*) $St(\gamma)$ of a cell in a Euclidean cell complex Γ is the union of $\{\gamma\}$ with the set of all cells γ' of Γ which contain γ in their combinatorial boundary (see Figure 1). Given a real function f defined on Γ, the upper star $St^+(\gamma)$ and lower star $St^-(\gamma)$ are formed by those simplexes in the star of γ having a function value less or greater than $f(\gamma)$, respectively.

The *link* $Lk(\gamma)$ of a cell γ is the set of cells of Γ forming the combinatorial boundary of the cells in $St(\gamma) - \{\gamma\}$ not containing γ (see Figure 1). Note that $Lk(\gamma)$ is a subcomplex of Γ formed by the cells of $St(\gamma)$ not intersecting γ.

A subset Λ of Γ is called a *sub-complex* of Γ if and only if Λ is a cell complex. In this work, we are interested in a specific kind of sub-complexes, called *skeletons*. The *k-skeleton* of a d-dimensional Euclidean cell complex Γ is the sub-complex of Γ which consists of all the cells of Γ of dimension less than or equal to k, where $0 \leq k \leq d$.

In this chapter, we will consider two classes of complexes that are used as the basis for defining a digital model of a scalar field (see Subsection 2.2), namely *regular grids*, and *simplicial complexes*. A d-dimensional *regular grid* is a Euclidean d-dimensional cell complex in which all k-cells, $0 < k \leq d$, are hypercubes. In 2D, 1-cells are straight-line segments and 2-cells are squares. In 3D, 1- and 2-cells are the same as in the 2D case, and 3-cells are cubes.

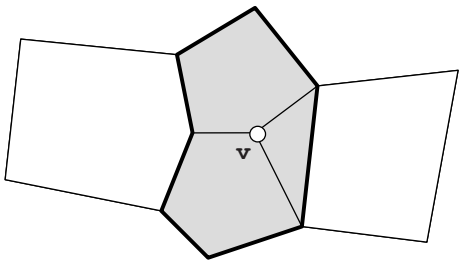


Fig. 1. A complex and the star (the set of cells shaded) and the link (the lines in bold) of a vertex v

Euclidean simplicial complexes are special cases of cell complexes, but their cells, called *simplexes*, are closed and defined by the convex combination of points in the Euclidean space. Let k be a non-negative integer. A *k-simplex* (or a *k-dimensional simplex*) σ is the convex hull of $k+1$ affinely independent points in R^d (with $k \leq d$), called *vertices* of σ. k is called the *dimension* of σ. A *face* σ of a k-simplex γ, $\sigma \subseteq \gamma$, is any h-simplex ($0 \leq h \leq k$) generated by $(h+1)$ vertices of γ. Conversely, γ is said to be a *co-face* of σ.

In the remainder of this chapter, we will consider d-dimensional simplicial complexes with a manifold domain, embedded either in the d-dimensional or in the $(d+1)$-dimensional space (in particular, d will be 2, or 3). Recall that a *d-manifold* is a (separable Hausdorff) topological space in which each point p has a neighborhood which is homeomorphic to the Euclidean space R^d.

2.2 Digital Models of a Scalar Field

Let V be a finite set of points in R^d, and F be a set of scalar values given at the points of V. Then, $S = (V, F)$ is called a *scalar data set*. Given the scalar data set $S = (V, F)$, a *digital model* of a scalar field defined on S is a pair $M = (\Gamma, f)$, where Γ is a d-dimensional cell complex such that the vertex set of Γ is the same as V, and f is a function defined at least at the vertices of Γ, such that $f(p) = F(p)$, for all $p \in V$.

The two most relevant cases of a digital model of a scalar field arise depending on whether Γ is a simplicial complex, or a regular grid. In the former case, we call the field model a *simplicial model*, in the latter case, we call it a *regular model* . In a simplicial model, usually f is chosen as a piecewise-linear interpolating function defined over the simplexes of Γ. Regular models can be encoded in very compact data structures, since only the scalar values have to be stored. Simplicial models, on the other hand, better adapt to the morphology of the field, since their vertices can be irregularly and adaptively sampled over its domain.

In the 2D case, M is called a *terrain model*. A regular terrain model M is often called a *Regular Square Grid* (RSG). A simplicial terrain model M is called a *Triangulated Irregular Network* (TIN), when f is a piecewise-linear interpolating function defined on the triangles and the edges of Γ.

2.3 Morse Theory

We review here the basic notions of Morse theory in the case of d-manifolds (see [54, 73] for more details). Morse theory is a powerful tool to capture the topological structure of a

scalar field. In particular, it states that it is possible to construct topological spaces equivalent to a given differential manifold describing the surface as a decomposition of the manifold into primitive topological cells, through a limited amount of information [54, 73, 43, 42, 15, 38].

Let f be a C^2-differentiable real-valued function defined over a domain $D \subseteq \mathbb{R}^d$. A point $p \in \mathbb{R}^d$ is a *critical point* of f if the gradient ∇f of f vanishes on p, i.e., if and only if $\nabla f(p) = 0$. function f is said to be a *Morse function* if all its critical points are non-degenerate, i.e., the Hessian matrix of the second derivatives of f at p is non-singular (its determinant is $\neq 0$) [54]. The number of negative eigenvalues of the Hessian matrix is called the *index* of a critical point p. In particular, the so-called *Morse Lemma* states that a Morse function f looks extremely simple near non-degenerate critical points: an appropriate local reference system can be always chosen such that f can be expressed in a canonical quadratic form. This result implies that the critical points of a Morse function are isolated.

The set of points belonging to $f^{-1}(h)$ forms a *level set* of function f at value h. The level sets may have several connected components, each of which is called a *contour* or an *isocontour* of f. In case of three-dimensional data, an isocontour is an isosurface. The topology of level sets varies only in correspondence of the critical points of f [54]. Finally, it provides a way of describing a manifold as a CW-decomposition [54, 73].

A two-dimensional manifold can have three types of non-degenerate critical points. A non-degenerate critical point p can be a *minimum* (*pit*), a *saddle*, or a *maximum* (*peak*) if p has index 0, 1 or 2, respectively. A three-dimensional manifold has four types of non-degenerate critical points. A non-degenerate critical point p is a *minimum*, a *1-saddle*, a *2-saddle*, or a *maximum* if p has index 0, 1, 2 or 3, respectively.

The concepts of critical point and Morse function may be extended to d-manifolds with boundary, assuming suitable conditions on the smoothness of the boundary components [24]. In this case, the neighborhood of a point p on the boundary is homeomorphic to the half-space $R^d_+ = \{(x_1, x_2, \ldots, x_d) \in R^d : x_d \geq 0\}$. Most of the results previously stated are still valid and may be adopted to model d-dimensional scalar fields, since they correspond to d-manifolds with boundary embedded in R^{d+1}.

An *integral line* of a function f is a maximal path which is everywhere tangent to the gradient vector field ∇f of f. The classical Taylor formula shows that integral lines follow the gradient directions in which the function has the maximum increasing growth. Integral lines cannot be infinite (in a compact domain D), and they cover the entire domain of f. An integral line is emanating from a critical point, or from the boundary of D, and it reaches another critical point or the boundary of D. If function f is defined on a manifold with boundary D, then an integral line may be open at both ends. The point $q \in D$, $q = \lim_{t \to -\infty} c(t)$ is called the *origin* of c, and the point $r \in D$, $r = \lim_{t \to \infty} c(t)$ is called the *destination* of c. An integral line which connects a critical point p of index $\imath$ to a critical point q of index $\imath + 1$ is called a *separatrix line*. In Geographic Information Systems (GISs), separatrix lines that connect minima to saddles are usually called *ravines*, or *valley lines*, while those that connect saddles to maxima are called *ridge lines*.

2.4 Morse Complexes and Morse-Smale Complexes

Let $f : D \to \mathbb{R}$ be a Morse function (see Subsection 2.3), where D is a d-manifold without boundary. Let $Crit_f$ be the set of critical points of f. Integral lines that converge to a critical point p of index $\imath$ form an $\imath$-cell, called a *stable (or descending) manifold*, which is denoted as $W^s(p)$. Similarly, integral lines that originate from a critical point p of index $\imath$ form a $(d - \imath)$-cell called an *unstable (or ascending) manifold*, which is denoted as $W^u(p)$. Thus,

$$W^s(p) = \{q \in D : \lim_{t \to +\infty} c_q(t) = p\},$$

$$W^u(p) = \{q \in D : \lim_{t \to -\infty} c_q(t) = p\}.$$

The stable manifolds are pairwise disjoint and decompose the domain D of a scalar field f into open cells which form a complex, since the boundary of every cell is the union of lower-dimensional cells. Such complex is called a *stable Morse complex*. The unstable manifolds form a complex as well, called an *unstable Morse complex*, which is dual with respect to the stable complex. Thus,

$$D = \bigcup_{p \in Critf} W^u(p) = \bigcup_{p \in Critf} W^s(p).$$

Figure 2(a) shows an example of a decomposition of the domain of a scalar field into an unstable Morse complex.

In a 2D unstable (stable) Morse complex, the 2-cells correspond to the maxima (minima), the 1-cells to the saddle points, and the 0-cells to the minima (maxima). In a 3D unstable (stable) Morse complex, the 3-cells correspond to the maxima (minima), the 1-cells to the 2-saddles (1-saddles), the 1-cells to the 1-saddles (2-saddles), and the 0-cells to the minima (maxima).

Moreover, a Morse function f is a *Morse-Smale function* when the stable and the unstable manifolds intersect only transversally. This means that the intersection (if it exists) of the stable $(d - \imath)$-dimensional manifold of a critical point p of index $\imath$, and the unstable $\jmath$-dimensional manifold of a critical point q of index $\jmath$, is a $(\jmath - \imath)$-dimensional manifold. The connected components of sets $W^u(p) \cap W^s(q)$, for all critical points $p, q \in Crit_f$, decompose D into a so-called *Morse-Smale complex*. Each cell of the Morse-Smale complex is the union of integral lines that all originate from the same critical point p, with index $\imath$, and converge to the same critical point q, with index $\jmath$. The dimension of the cell is then $\jmath - \imath$. Figure 2(b) shows the Morse-Smale complex for the same function shown in Figure 2(a).

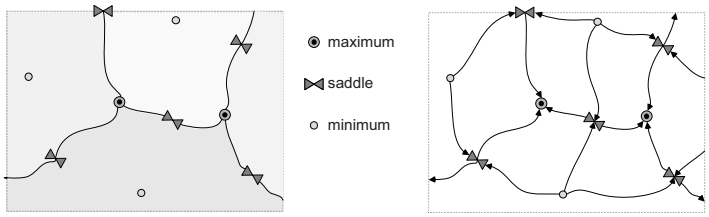


Fig. 2. (a) An example of an unstable Morse complex (the 2-cells correspond to the minima). (b) The Morse-Smale complex. Its 1-skeleton is the critical net.

The 1-skeleton of a Morse-Smale complex consists of critical points and separatrix lines, and it is often called a *critical net* (see Figure 2 (b)). The *surface network* [66, 71], widely used in GISs for morphological terrain modeling, is a combinatorial representation of the critical net in the case of 2D scalar fields. The surface network is a planar graph in which the nodes correspond to the critical points, and the arcs to the integral lines connecting them. Thus, there exists an arc between a pair of nodes in the surface network if the two corresponding critical points are connected by an integral line in the critical net.

The *Critical Point Configuration Graph* (*CPCG*) [56] describes the configuration of the critical points of a C^2-differentiable Morse function f defined on the closure $D = \bar{O}$ of a simply-connected open set O in R^2. Note that f does not need to satisfy the Morse-Smale condition. The nodes of the CPCG represent critical points, while its arcs are in one-to-one correspondence with the integral lines connecting them. A CPCG is a planar graph and its embedding on the domain D of f induces a partition of D into 2D regions, called *slope districts*. Since f is not necessarily a Morse-Smale function, there are configurations of the critical points of f which do not occur for Morse-Smale functions. For instance, an arc may connect a saddle to a maximum, a saddle to a minimum, but also a pair of saddles. Nackman in [56] shows that all the possible configurations for slope districts are equivalent to four basic configurations (up to equivalence, which consists of insertion of saddle points in the arcs), which are illustrated in Figure 3. The first three are quadrangles (which may be glued along the edges) with nodes of index 1,0,1,2 respectively (saddle, minimum, saddle, maximum). These quadrangles correspond to the possible types of 2-cells in a Morse-Smale complex. The first type occurs most frequently in real cases (see Figure 3 (a)), the second and third type correspond to an isolated mountain, or a crater, respectively (see Figure 3 (b) and Figure 3 (c)). The last type of slope district can occur when f is a Morse, but not a Morse-Smale, function. In this case, ascending and descending 1-manifolds do not intersect transversally, but may coincide (see Figure 3 (d)).

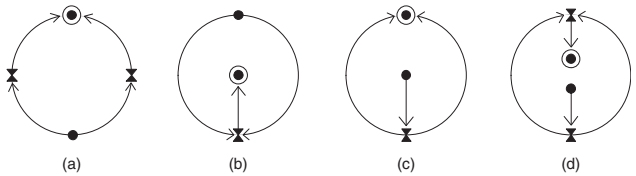


Fig. 3. The four possible configuration for the slope districts in a CPCG.

2.5 Contour Trees

As mentioned before, Morse theory studies the evolution of the level sets of function f that defines the scalar field. The structure that tracks the changes of the contours of the function f is called the *contour tree*. It was originally introduced for terrain models and, more recently, for d-dimensional scalar fields.

Several variations of the contour tree may be found in the literature: the *augmented contour tree* [83, 64, 21, 22]; the *contour topology tree*, [23]; the *criticality tree*, [26]; the *topographic change trees* [41]; or the *component tree*, [25, 46]. In the larger context of manifolds, where the definition of function f may also vary, contour trees are special cases of the more general Reeb graphs [68], as they are also based upon adjacency relationships between contour lines. The contour tree corresponds to the Reeb graph of the digital model M studied according to the evolutions of the level sets as the scalar function f changes. However, since in this chapter we consider only simply connected domains, we will refer to this coding of the contours as contour tree while the presentation of the Reeb graph in a general setting is proposed in chapter [12].

A formal definition of the contour tree in terms of nodes, as critical points of the function f, and edges, which represent the portions of the scalar field where the topology of the contour

does not change, can be found in [20]. Edges of the contour tree are generally assumed to be directed from the higher to the lower values of function f along the specific edge. Informally, the contour tree of a scalar field is the graph obtained by the continuous contraction of each contour of a scalar field to a single point [63, 23].

This representation describes the relations between the connected components of the level sets of a scalar field (see Figure 4). In the case of 2D scalar fields, the level sets of f may be represented as the intersections of the model M with planes orthogonal to the model height. If the function f is Morse, the contour tree may assume only three configurations around a critical point. This fact follows from the Morse Lemma, see Section 2.3. As a consequence, the contour tree of a terrain (also known as paper surface [42]) M encodes the shape of M in terms of its meaningful topographic features, i.e., peaks, pits or passes and structures these features into a topologically consistent framework.

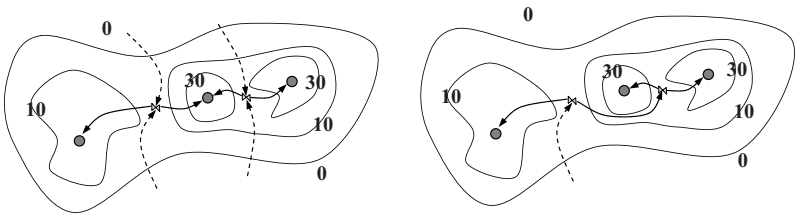
(a) (b) (c)

Fig. 4. A data set (a) with a set of contour levels (b) and the contour tree corresponding to the contours depicted (c). Numbers indicate the elevation of the contours.

The contour tree can be effectively represented as a graph: a node is defined for each critical level of f that corresponds to the creation, merging, split or deletion of a contour, that is, to topological changes affecting the number of connected components of the level sets of f. Each node corresponds to a critical point but, in case of scalar fields of dimension higher or equal to three, there are critical points that do not affect the number of connected components of the contour level. For instance, this happens when in correspondence to a critical point the topological genus of the isosurface changes. Therefore, no node of the contour tree is associated with such critical points. Each arc joins two successive critical levels in their own component. If an arc connects two nodes, n_1 and n_2, then the number of the connected components of the level sets between the critical levels n_1 and n_2 does not change along the connected component of M joining the corresponding critical points.

Since a scalar field f assumes only one value at a given point of the domain D, any given contour divides its complement into disjoint sub-regions, so that every path from a region to another must pass through that contour. This implies that a contour tree has no cycles and can always be represented as a tree.

The definition of a contour tree has been provided also when a mapping function is not formally defined [16]. It can be defined for any set of isocontours that nest inside each other, as shown in [20], where the author proposes to locally take advantage of the property that each contour comes from a scalar field and has “increasing” and “decreasing” directions.



(a) (b)

Fig. 5. The comparison of the surface network (a) and the contour tree representation (b) of a terrain model.

Considering simple[4] Morse functions, i.e., functions whose critical points are non-degenerate, contour trees and surface networks can be compared: the contour tree can be obtained from the surface network of the same scalar field, as in the algorithms proposed in [78, 77]. Both the contour tree and the surface network encode the topological structure of a surface, but surface networks provide a surface-oriented decomposition, while contour trees provide a contour-oriented description. In Figure 5, the surface network of a terrain represented by contours is compared with the corresponding contour tree. All arcs of the surface network coming from the outside of the surface boundary originate from the isocontour having minimum height that encloses the others.

3 Extracting Critical Points

Almost all the algorithms that compute a Morse-Smale decomposition, or a contour tree first extract the critical points from the digital model of the scalar field. In this Section, we focus on the problem of extracting critical points from digital models of 2D and 3D scalar fields.

In the literature, several algorithms have been proposed for simplicial models and for regular ones. All algorithms apply a local approach that simulates the definition of critical point in the continuum. In this context, one of the most representative approaches is the one presented by Banchoff in [6] where the vertices of a polyhedral surface are locally characterized analyzing their star. Note that in a simplicial model, critical points can only be at the vertices, while they may be inside the square or hexahedral cells in regular models when a C^0-continuous (at least) interpolating function is selected. Homology-oriented methods have also been developed which detect the regions where the topology changes instead of the points.

3.1 Extracting Critical Points from a Piecewise Linear Field

The algorithms for extracting critical points from a simplicial model of a scalar field apply a result by Banchoff [6], who has generalized the notion of critical point for a Morse function to piece-wise linear functions defined over a d-dimensional simplicial complex. Let Σ denote a k-dimensional simplicial complex in R^d. Let $f : R^d \rightarrow R$ be a function. f is called *general*

[4] A function is called simple if all its critical points have different values, i.e., any pair x, y of distinct critical points is such that $f(x) \neq f(y)$. Notice that a Morse and simple function is sometimes referred as Morse [37].

for Σ if $f(v) \neq f(w)$ whenever v and w are the vertices of a 2-simplex of Σ. Then, for each simplex $\sigma \in \Sigma$ a value A_σ is defined as follows [2]:

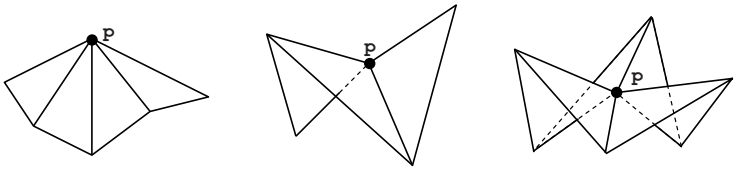
$$A_\sigma(v) = \begin{cases} 1 \text{ if } v \in \sigma \text{ and } f(v) \geq f(w)\ \forall w \in \sigma \\ 0 \text{ otherwise} \end{cases}$$

Let σ be an r-dimensional simplex of Σ and f a general function defined on Σ, the *index* of the vertex v with respect to f is given by:

$$i(v) = \sum_{r=0}^{k} (-1)^r \sum_{\sigma \in S, dim(\sigma)=r} A_\sigma(v). \tag{1}$$

For regular points both the indices λ and $i(v)$ are 0. If v is a critical point for f, having index λ, the relation $i(v) = (-1)^\lambda$ is verified.

The assumption on the function being general is the discrete counterpart of the hypothesis a C^2-differentiable function is Morse. Nevertheless, it may happen that a function f may have m-fold saddles even if it is general for Σ. However, m-fold saddles may be unfolded by considering a barycentric subdivision of all the triangles and performing a small perturbation of the function values [3], or splitting the wedges of the lower and upper stars of a critical point p [37]. In particular, each m-fold saddle may be unfolded in m simple saddles. According to this fact, the value $i(v)$ in correspondence of a multiple saddle may be written as: $i(v) = -m$. Note that the result of the unfolding may be ambiguous since there are several splitting choices. Examples of critical points for a triangular mesh are given in Figure 6.



(a) (b) (c)

Fig. 6. A maximum (a), a simple saddle (b) and a 2-fold (monkey) saddle (c).

Formula (1) implicitly defines an algorithm for detecting the index of a point p in a simplicial model, when the interpolating function f satisfies the discrete Morse condition, which takes into consideration all the simplexes in the star of p. Several authors have developed simplified versions of the algorithm, thus considering only edges or vertices in the link of p [36, 35, 58, 78]. In [5], Bajaj et al. apply a different approach for identifying and classifying the critical points in a 2D simplicial model. They study the normal vectors of the triangles in the star of p and they consider a range of values based on these vectors. A vertex p is considered a critical point when this range of values includes a vector $(0, 0, 1)$. A critical point p is classified as minimum, maximum, or saddle point, depending on whether the gradient flow is away from, towards, or towards-and-away from p, respectively.

Several algorithms have been proposed for extracting critical points from a 3D simplicial model [40, 79, 35, 23]. All of them detect minima, maxima and saddle points, but sometimes they do not distinguish between 1- and 2-saddles [40], or they fail in recognizing multiple saddles [23]. As in the 2D case, the classification of a vertex p is performed by considering the link of p. Given a vertex p, we denote as $N^+(p)$, $N^-(p)$, the number of vertices q in $Lk(p)$

such $f(p) \leq f(q)$, $f(p) \geq f(q)$, respectively. We denote as $C^+(p)$ and $C^-(p)$ the numbers of connected components in $N^+(p)$ and in $N^-(p)$, respectively. Thus, if $C^+(p) = 0$ (and, thus. $C^-(p) = 1$), then p is maximum, while if $C^-(p) = 0$, (and thus $C^+(p) = 1$), then p is minimum. Chiang et al. [23] identify saddle points as those points for which $C^-(p) \geq 1$, and $C^+(p) \geq 1$. In particular, they identify the saddle points as points at which an iso-surface splits into two, or two iso-surfaces merge. Gerstner et al. [40] identify a saddle point p as a point such that $C^+(p) + C^-(p) = 2$, without distinguishing between 1- and 2-saddles. In order to identify multiple saddles, Takahashi et al. [79] count the multiplicity of multiple saddles as follows. If $C^+(p) = k + 1$, then p is a 1-saddle with multiplicity k. If $C^-(p) = k + 1$, then p is a 2-saddle with multiplicity k.

The algorithm proposed by Edelsbrunner et al. [35] uses the reduced Betti numbers of the lower link of a vertex in order to classify saddle points. The *lower link* $Lk^-(p)$ of a vertex p is defined as the collection of all simplexes in the link of p such that the value of the function f at such simplexes is less than $f(p)$. Note that the values of the function f are given at the vertices of the complex, and are defined on any other higher-dimensional simplex by linear interpolation. Informally, the Betti numbers β_0, β_1, and β_2 of a simplicial complex indicate the number of connected components, the number of independent 'tunnels', and the number of holes of its carrier, respectively. The reduced Betti numbers $\tilde{\beta}_{-1}$, $\tilde{\beta}_0$, $\tilde{\beta}_1$, and $\tilde{\beta}_2$ are the same as Betti numbers, except that $\tilde{\beta}_0 = \beta_0 - 1$ for non-empty complexes, and $\tilde{\beta}_{-1} = 1$ for empty complexes. The reduced Betti numbers of the lower link of a regular point are all equal to zero, while simple critical points have exactly one non-zero Betti number (which is equal to one). A multiple saddle p satisfies $\tilde{\beta}_{-1} = \tilde{\beta}_2 = 0$, and $\tilde{\beta}_0 + \tilde{\beta}_1 >= 0$. It has been shown that p can be unfolded into simple $\tilde{\beta}_0$ 1-saddles, and $\tilde{\beta}_1$ 2-saddles.

It can be easily seen that the worst-case time complexity of all the algorithms for extracting critical points is linear in the number of maximal simplexes[5] in the simplicial complex in both the 2D and 3D case. This time is provided if the underlying simplicial complexes are encoded in data structures which allow retrieving the maximal simplexes in the star of a vertex in time linear in the number of such simplexes. Thus, the worst-case complexity of critical points extraction algorithm is linear in the number of vertices of the complex in the 2D case, but it may be quadratic in the number of vertices of the complex in 3D case [35].

3.2 Extracting Critical Points from a Regular Grid

Several algorithms have been developed over the years to extract critical points from regular grids, mainly within the field of image processing. Since a grey-scale image can be seen as an integer-valued scalar field, the grid vertices are in this case the pixels, or voxels, in the image. The general idea is to consider a vertex p of a regular grid and compare its field value $f(p)$ with the field values of some suitably-defined neighbors on the grid. These algorithms, rooted in digital geometry, consider a discontinuous approximation of the field, which is just a step function defined at the vertices of grid (i.e., the pixels or voxels of the image). Most of such algorithms do not extract only critical points, but also label vertices of the grid belonging to critical lines of the scalar field, such as *crest* and *course lines*. These latter are characterized locally by considering principal normal curvature k_1 and k_2 of f at p. A line is considered a *crest (course) line* if one of the principal curvatures k_1 and k_2 has an extremum along its curvature line, which is an integral line of the vector field of principal directions. Note that the set of separatrix lines is a subset of the set of crest and course lines, while the reverse is not true.

[5] Recall that a *maximal* simplex is any k-simplex in a k-dimensional simplicial complex.

Two classical algorithms [65, 82] produce a set of uniformly labeled regions of points of a regular grid, identifying maxima, minima, saddles, and also points belonging to crest and course lines. Both can be applied by considering the 4-adjacent, or the 8-adjacent vertex neighbors of a vertex p of the grid, where the 4-adjacent vertex neighbors are those connected to p by an edge, and the 8-adjacent neighbors of p are the vertices in the link of p. These techniques have been extended to 3D grids by Papaleo in [59] considering the 6-adjacent neighbors of p, i.e., the vertices in the link of p connected to p through an edge.

Other algorithms extract critical points by fitting some local, sometimes globally discontinuous, approximating function on grid data. Watson et al. [86] present a classification algorithm for grey-scale images in which a surface on the square patch centered at the vertex p is defined by considering the field values of its 8-adjacent vertices. The approximation is a generalized quadratic B-spline, and it is C^0- differentiable inside the square patch, C^2-differentiable at p, but globally discontinuous. First and second partial derivatives of the approximating functions are computed at p analytically, and critical points as well as points belonging to course or crest lines are classified as in the C^2-differentiable case. Schneider et al. in [71] fit a biquadratic polynomial by considering the 8-adjacent neighbors for each vertex p of the regular grid. The method produces a globally discontinuous approximation, formed by local surface patches (see [71] for details).

The algorithm proposed by Schneider et al. in [70] uses a bilinear C^0-differentiable interpolating function on each 2-cell of the grid. Minima and maxima can occur only at grid vertices, but additional saddles may be introduced by the interpolation inside the cell. Bajaj et al. [4] use a globally C^1-differentiable Bernstein-Bézier bi-cubic interpolant, locally defined on each square cell. This interpolating function does not remove any critical point of the initial input data and may add a small number of additional critical points. The classification of the points is done analytically.

Extraction methods based on grid data interpolation have been also developed for 3D regular models. Bajaj et al. [4] extend their 2D algorithm to the 3D case by considering a Bernstein-Bézier tri-cubic function as interpolant in each cubic grid cell. Weber et al. [87] use a tri-linear interpolating function in each cubic cell. In this case also, there may be saddles inside the cubic cells and on their boundaries. The algorithm does not distinguish between 1-saddles and 2-saddles. Weber et al. in [88] relax the assumption that edge-adjacent vertices must have different field value, and thus connected components of grid points at the same elevation, which are critical for an function f, are extracted.

3.3 Extracting Critical points from Contours

Analyzing the evolution of the contour levels of the field function provides an alternative way of characterizing the critical points. Let f be a scalar piecewise-linear function defined on a simplicial decomposition of a two- or three-dimensional manifold. If the interval $[a, b] \subseteq \mathbb{R}$ does not contain any critical value, all level sets $f^{-1}(h)$, $h \in [a, b]$ are homeomorphic. On the contrary, if $[a, b]$ contains a critical value, the topology of the level sets varies at this value, [23] (see Figure 7).

In the case of surfaces, critical points are located where the number of contour levels varies. Based on these considerations, several approaches have considered the evolution of the contour levels for characterizing a surface [47, 13, 14, 1, 11], while methods for three-dimensional data are still missing. This approach is independent of the underlying digital model. For example, the method in [47] works on a parametric surface with boundary, while the method in [13, 1] has been developed for simplicial models.

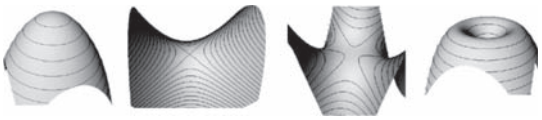


Fig. 7. Possible configurations of contour levels around a critical point.

In the presence of noise, this approach is more stable (i.e., less sensitive to noise) than those methods that compute the critical points through analytical techniques, and it is also able to detect degenerate configurations, like critical regions.

4 Extracting Approximations of a Morse-Smale Complex

In this Section, we review algorithms for decomposing the manifold domain of a scalar field f into an approximation of a Morse-Smale complex. Such an approximation is obtained either by fitting a C^1-, or C^2-differentiable surface on a discrete dataset, or by simulating a Morse-Smale complex, or a Morse complex, on a piecewise-linear interpolation of the input data.

Most of the algorithms proposed in the literature, with the exception of the one proposed in [35], work on 2D scalar fields. Most of them use a *boundary-based* approach, in the sense that they extract an approximation of the critical net, by computing the critical points and then tracing the integral lines, starting from saddle points. Some of these algorithms are based on regular models [4, 71, 70], others are based on simplicial models, such as the ones in [78, 36, 5, 17, 60, 35].

Boundary-based methods which operate on simplicial models can be viewed as techniques for computing the discrete component of a Morse-Smale complex, called a *Quasi Morse-Smale complex* (QMS) introduced in [36, 35]. In Subsection 4.1, we present algorithms for extracting a Morse-Smale complex from a simplicial model. In Subsection 4.2, we review boundary-based algorithms for 2D regular models.

A different approach to compute a discrete approximation of Morse complexes on a simplicial model is based on a region-growing technique which starts from the minima (maxima) and compute a stable (unstable) 2-cell by adding one triangle at time [27, 28, 49]. In Subsection 4.3, we present such algorithms. Finally, in Subsection 4.4, we discuss methods for simplifying Morse-Smale complexes and generating hierarchical representations of such complexes.

4.1 Boundary-based Algorithms on a Simplicial Model

In [36, 35], the definition of a complex is introduced which has the same combinatorial structure of a Morse-Smale complex for 2D and 3D simplicial models defined over a 2-manifold, or a 3-manifold without boundary, respectively. Such complex is called a *Quasi Morse-Smale complex (QMS)*. The 0-cells (vertices) of a QMS complex are the critical points of function f, the 1-cells connect minima to saddles (1-saddle in 3D), maxima to saddles (2-saddles in 3D) and, only in the 3D case, 1-saddles to 2-saddles. There are no critical points inside 1-cells, 2-cells (or 3-cells in 3D). In the 2D case, each saddle point p has four incident 1-cells, two joining p to maxima, and two joining p to minima. Such 1-cells alternate in a cyclic order around p. Also, the 2-cells are quadrangles whose vertices are critical points of f of index 1,0,1,2 (saddle, minimum, saddle, maximum) in this order.

In the 3D case, all 2-cells are quadrangles whose vertices are a minimum, 1-saddle, 2-saddle, 1-saddle in this order (quadrangles of type 1) , or a 1-saddle, a 2-saddle, a maximum, a 1-saddle in this order (quadrangles of type 2). A 1-cell connecting a 1-saddle and a 2-saddle is on the boundary of four quadrangles which alternates between quadrangles of type 1 and type 2. The 3-cells are called *crystals* and are bounded by quadrangles. A QMS complex differs from the Morse-Smale complex defined in the C^2 differentiable case since the 1-cells in 2D, and the 1-cells and 2-cells in 3D are not necessarily those of maximal ascent, or descent. The 1-skeleton of the QMS complex is a discrete version of the critical net.

In the 2D case, most algorithms [78, 5, 36, 17, 60] extract a QMS complex from a 2D simplicial model by computing its 1-skeleton (the critical net) in three steps:

1. *extract* the critical points (see Section 3)
2. *unfold* multiple saddles (see Section 3)
3. *compute* the 1-cells of the QMS complex by *starting* from the saddle points, and by *tracing* two paths of steepest descent and two paths of steepest ascent on the underlying triangle mesh which stop at minima and maxima, respectively.

The algorithms in [78, 36, 5] extract the 1-cells of the QMS complex by computing paths along the edges of the triangle mesh. The algorithms in [5, 78] trace the paths by selecting the vertex of highest (or lowest) elevation at each step, while the algorithm in [36] selects the steepest ascending or descending edge at each step. The algorithms in [17, 60] estimate the gradient along 1-simplexes and 2-simplexes, and compute the ascending and descending paths not only moving along the edges, but possibly cutting triangles in order to follow the actual paths of steepest ascent, or descent.

All such algorithms have a worst-case time complexity which is linear in the number of vertices of the simplicial model. This happens provided that suitable data structures are used for encoding the triangle mesh, which allow retrieving the star of a vertex in time linear in the number of triangles in the star.

An algorithm has been proposed by Edelsbrunner et al. in [35] for computing a QMS complex for a 3D simplicial model. An implementation is described in [57]. The algorithm computes the QMS complex for a 3D simplicial model by extracting first the critical points, then the stable Morse complex and, finally, the unstable manifolds in pieces inside the cells formed by the stable manifolds. In other words, the structure of the stable manifolds is used while computing the unstable ones in order to maintain the structural integrity of the QMS complex. Note that it is not guaranteed that the same complex would be obtained if first the unstable, and then the stable manifolds were computed.

4.2 Boundary-based Algorithms on a Regular Model

No discrete definition has been introduced of discrete a Morse or of a Morse-Smale complex when we consider a regular model. The three algorithms in [4, 71, 70] compute the boundary of the Morse-Smale complex, i.e., the critical net from a 2D regular model, through a technique conceptually very similar to the one used for 2D simplicial models. They classify critical points as maxima, minima, saddle or regular points, as discussed in Section 3. For each saddle point, they trace the four separatrix lines emanating from it as lines of steepest ascent or descent. All three algorithms try to fit a surface of a certain degree of continuity to the input data in order to extract the critical points.

The algorithm by Bajaj et al. [4] uses a C^1-differentiable Bernstein-Bézier bicubic interpolant and computes integral lines through a Runge-Kutta integration technique [67]. Four

separatrix lines are traced, from each saddle point, in the direction of the appropriate eigenvectors. The computation stops when the line reaches a neighborhood of another critical point, or the boundary of the domain.

The algorithm proposed by Schneider et al. in [70] traces separatrix lines point by point. Separatrix lines can follow grid edges, or can cross 2-cells. When a separatrix line crosses a grid cell, it can be approximated with small (linear) steps, or computed exactly, by solving a linear system of differential equations. The exact integral line is a hyperbolic function inside a grid cell. The algorithm by Schneider et al. in [71] computes the the first and second derivatives analytically (see also [70]) and uses this information to trace the separatrix lines starting from the saddles.

4.3 Region-based Algorithms for Approximating Morse complexes

Region-based algorithms [27, 28, 49] compute approximation of the stable and unstable Morse complexes, by simulating in the discrete case the behavior of the gradient of the field function f. The only assumption is that the function f defining the simplicial model is a Morse function, i.e., no adjacent vertices have the same height. Recall that the unstable (stable) manifold of a point p for a Morse function f is the set of points q such all that the descending (ascending) integral lines from q reach p. The algorithms simulate this definition in the discrete case and they consist of two major steps:

1. *Extract* minima and maxima.
2. *Compute* the stable (unstable) Morse complex by applying a region-growing procedure. This procedure add triangles to a 2-cell of the complex incrementally.

The three algorithms differ in the way they select the next triangle to add to a 2-cell at each step. We consider, for clarity, the computation of the stable complex. In [28], a triangle t is added to the current 2-cell C in the stable complex if the vertex of t which is not on the boundary of C has higher elevation than the other vertices of t. If a local minimum lies on the boundary of a 2-cell generated, this latter is merged with the adjacent 2-cell. In [55], it has been proven that the discrete gradient field generated by this approach can be interpreted as a discrete Forman gradient field within the discrete Morse Theory framework proposed by Forman [39].

The algorithm in [27] computes the *gradient* for each triangle t in the model M, and the angles between the normal vector at each edge of t and the gradient. The edge e of t corresponding to the largest angle is marked as *exit*, the one corresponding to the smallest angle is marked as *entrance*. At a generic step, a 2-cell of the stable complex is extended by adding a new triangle t sharing an edge e with the cell provided that e is an entrance for t and an exit for the triangle t' in the cell sharing edge e with t.

The algorithm in [49] initially classifies the three vertices of each triangle t in the model. The highest, middle, amd lowest vertex of a triangle are labelled as S (source), T (through), and D (drain), respectively, corresponding to the intuitive idea that water flows from S to D through T. As the previous algorithms, it starts from each local minimum m of the model and grows its 2-cell in the stable complex (initially formed by the star of m). At each step, a triangle t externally adjacent to a boundary edge e of the current 2-cell is considered, and the algorithm decides whether to include t or not as follows. If the opposite vertex of t to e is labelled D, then t is not included; if it is labelled S, then t will be included. If it is labelled T, then the fan of triangles having their lowest vertex in the D-labelled vertex of t are examined (such edge is an endpoint of e). The fan is splitted into two parts at its radial edge of maximum slope, and the part adjacent to e (if not empty) is included in the basin of m.

All the algorithms exhibit a worst-case time complexity which is linear in the number of vertices in the simplicial model. Note that the algorithms in [55, 49] do not require any floating point computation. The algorithm in [55] has been extended to arbitrary dimensional simplicial models. All of them compute the unstable complex in a completely simmetric way.

The overlay of the two Morse complexes provides the QMS complex, if the two complexes intersect transversally. Saddle points are extracted as the intersection of the two complexes. Computing the overlay also is a linear process since each triangle in Σ belongs to one 2-cell in the stable and to one 2-cell in the unstable complex. Figure 8 (a) and (b) (see also Figure CP-1 in Appendix F) show the stable and unstable Morse-complexes, respectively, computed on Mont Marcy dataset by using the algorithm in [27]. Figure 8 (c) shows the intersection of the two (see also Figure CP-1 (c) in Appendix F).

The region-growing approach used in [27, 28, 49] is similar the approach used by watershed algorithms which are region-growing in nature, as for example the algorithms presented in [53, 85, 50]. This makes them suitable to be extended to 3D simplicial models.

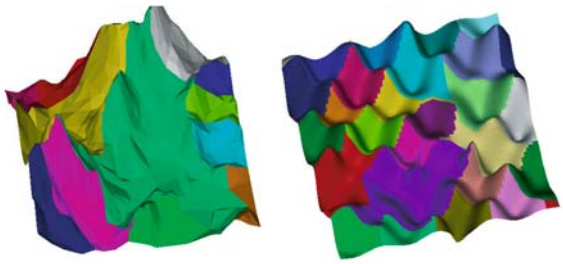


Fig. 8.

4.4 Generalization of Morse-Smale Complexes

Two major issues arise when computing a representation of a scalar field as a Morse, or a Morse-Smale, complex. The first issue which is a common problem in image and mesh segmentation algorithms, is the over-segmentation due to the presence of noise in the data sets. To this aim, *generalization algorithms* have been developed by several authors to locally simplify the structure of a Morse-Smale complex [89, 36, 18, 78, 77, 44]. The second issue is related to the large size and complexity of available scientific data sets. Thus, a multi-resolution representation is crucial for an interactive exploration of such data sets. There exist just a few proposals in the literature for multi-resolution representation for 2D scalar fields based on morphology [27, 18, 19].

The generalization of a Morse-Smale complex for a two-dimensional scalar field consists of collapsing a maximum-saddle pair into a maximum, or a minimum-saddle pair into a minimum, so as to maintain the consistency of the underlying complex. Usually, this operation is

viewed as the *cancellation* of a pair of critical points, namely, a maximum and a saddle, or a minimum and a saddle. A cancellation simulates the smoothing of the scalar field by modifying the gradient flows around two critical points. A generalization can be formalized in terms of the combinatorial representation of the critical net, defined by the surface network [29].

Suitable feasibility conditions need to be satisfied for the cancellation of a pair of critical points p and s to be feasible. If p is a minimum (maximum), all relevant saddles must be at a higher (lower) elevation than p. Moreover, a cancellation cannot involve isolated minima or maxima, as in the two configurations described in Figures 3(b)-(c). The various generalization techniques proposed in the literature differ in the criterion used to pair the critical points for cancellation, and in the order in which the cancellations are performed. In [89], a minimum (maximum) p is chosen for cancellation together with its lowest (highest) adjacent saddle s. In [36, 18], a saddle s is chosen together with its adjacent maximum at lower elevation, or its adjacent minimum at higher elevation. The order in which the pairs of points are canceled is determined based on a technique, called *persistence*, which grows a space incrementally and analyzes the topological changes that occur during this growth [36].

In [77], a pair of critical points p and s, which are adjacent in the contour tree, is chosen in a way such that the difference in elevation between p and s is minimal among all (unsigned) differences in elevation between a saddle and an adjacent minimum, or a saddle and an adjacent maximum. The order is according to increasing values of the differences in elevation.

The problem of generalizing 3D Morse-Smale complexes has been recently investigated [44]. This method extends the technique discussed in [36, 18] to functions defined over 3-manifolds. The extension is not trivial, since in 3D there are three possible types of legal cancellations: minimum and 1-saddle, 1-saddle and 2-saddle, and 2-saddle and maximum. The cancellations are performed in order of persistence. Conditions on the elevations of the critical points involved in the generalization operation similar as in the 2D case have to be imposed. While the two cancellations involving a minimum or a maximum are similar to the ones performed in the 2D case, the saddle-saddle cancellation does not have an analog in lower dimensions. In order to ensure the separation of the minima and maxima originally separated by the two saddles, additional cells has to be introduced in the Morse-Smale complex, and thus this operation may not produce a reduction of the number of its cells. However, the authors show in [44] that all these cells are removed by subsequent saddle-extremum cancellations.

In [27], a multi-resolution representation for a triangulated terrain is proposed which includes the critical net at different levels of resolution. This is achieved through an algorithm for building and simplifying a constrained Delaunay triangulation of a set of points, where the constraints are represented by the polygonal edges forming the critical net at different resolutions. In [18, 19, 29], multi-resolution representations of a triangulated terrain have been proposed. The method in [29] is based on the framework for multi-resolution modeling of simplicial and cell complexes proposed in [32, 31]. According to this framework, a multi-resolution representation can be defined on the basis of a sequence of legal generalizations applied to the a complex at the maximum resolution. This representation encodes the complex at the coarsest resolution, obtained as result of the generalization sequence, plus the a collection of atomic operations, called *anti-cancellations*, and a dependency partial order relation among them. An *anti-cancellation* is the inverse operation with respect to a cancellation of a pair of critical points. Intuitively, two anti-cancellations are considered to be independent if they do not affect the same region in the Morse-Smale complex, called *the region of influence* of the operation. The dependency relation is then encoded as a Directed Acyclic Graph (DAG). The *region of influence* of an anti-cancellation defined in [18] is larger than the ones defined in the other approaches. This gives less flexibility in extracting variable resolution representations.

5 Algorithms for Extracting a Contour Tree

The contour tree was originally introduced in topography for encoding the contours of a map in an ordered manner. A first overview of algorithms for extracting the contour tree representation can be found in [20]. As discussed in Subsection 2.5, the contour tree is a special case of the Reeb graph [68]; therefore, differently from [20], in this Section we describe algorithms that refer to both concepts. Several algorithms proposed for the computation of the Reeb graph can be easily adapted to terrains [78, 14], or to scalar fields in higher dimensions [79] through the addition of a *minimum* to the set of critical points that virtually closes the scalar field and makes it homeomorphic to a d-dimensional sphere, as discussed in [13, 78, 79].

The algorithm proposed in [16] encodes the contour tree of a terrain considering the nesting relationship of a set of polygonal contours manually extracted from a topographical map. The whole surface is enclosed by an "outside region", so that each contour has an inside and an outside region. Nodes representing contours are added to the tree one at time. Methods that perform the automatic contour tree extraction usually take as input a mesh, often a triangle mesh.

A systematic approach to encode geographical data organized in a triangle mesh has been proposed by de Berg and van Kreveld in [30]. This method is specialized for two-dimensional scalar fields and runs in $O(Nlog(N))$ operations, where N is the number of simplexes of the mesh. A simplification of the algorithm for two-dimensional scalar fields and the extension to higher dimensions has been proposed in van Kreveld et al. [83]. There, the authors suggest to extract the iso-lines (also called contours) and keep track of their evolution by sweeping the data set twice: first, from the highest to the lowest height value and then sweeping again in the reverse direction (from low to high). Moreover, the notion of *augmented contour tree* has been introduced to encode the vertices along an arc of the contour tree. In [83], the vertices and the edges of the contour tree are called super-nodes and super-arcs. In addition to super-nodes, nodes are introduced to represent the regular points in the augmented contour tree, that is, the nodes represent points which belong to a super-arc. Such nodes have always degree two. Super-nodes, indeed, have variable degrees: one, if they correspond to minima, or maxima (i.e., the leaves of the tree), and at least three if they correspond to passes. Moreover, contour trees of terrain surfaces may also have super-nodes with degree two [83]. During the computation, all super-nodes of the contour tree, which point out where topological changes happen, are assumed to be simple (i.e., no more than two contours meet in correspondence of them). Critical points in the mesh correspond to super-nodes of the contour tree, while super-arcs determine the connections between super-nodes.

The proposed algorithm constructs the contours tree of a given mesh coding the junctions and the branches of the surface in two separate trees that are successfully merged [83]. Each level set is explicitly stored as a set of contours, each one represented as an ordered list of simplexes. However, each list of simplexes may correspond to more than one contour in the first sweep, as this sweep does not detect when contours separate. Since it is assumed that all vertices have different elevation, contours may appear and merge only in correspondence of a vertex. In the first sweep, as each local maximum is passed, a new contour is constructed in the level set. Since contours are separately labeled during the sweeping, when two contours meet in a vertex, the smaller of the two contours is merged to the largest one and the vertex is classified as a *join*. In the second phase, the algorithm detects both local minima and splits and it stores the information necessary to know how components break apart splits. It finally constructs the complete contour tree. The computational cost of this last algorithm is reasonable, ($O(NlogN)$ for terrain surfaces, $O(N^2)$ for higher dimensions, where N is the

number of simplexes) but, similarly to the approach in [78], the critical point definition suffers of instability.

Tarasov at al. [80] improves the time complexity of this approach for $3D$ scalar fields, by showing that the relabeling process could also be done efficiently in three dimensions (it requires $O(nlogn)$ operations, where n denotes the number of vertices), and by extending the pre-processing to multiple saddles, while Carr at al. [21, 22, 20] consider also scalar fields of higher dimensions so their algorithm can be applied for X-ray analysis. In particular, in the method proposed by Carr et al. [21, 22] contours are not explicitly maintained and no pre-processing for multiple saddles and the surface boundary is required. Analogously to [83], two sweeps are performed to compute a *join tree* and a *split tree*, which represent the connectivity of the sets $\{x : f(x) \geq h\}$ and $\{x : f(x) \leq h\}$, respectively (see Figure 9). Finally, similarly to the algorithm proposed in [78], the contour tree is assembled by picking local extrema from the join and split trees and transferring them to the augmented contour tree. As an optional step, nodes belonging to super-arcs may be removed so that the contour tree is explicitly coded. This method performs in four main steps:

1. *computation* of the join tree;
2. *computation* of the split tree;
3. *merge* of the join tree and the split tree to obtain the augmented contour tree;
4. eventual *extraction* of the contour tree from the augmented contour tree.

The merge step has been shown to be equivalent to the sweep through the data [83], sweeping one contour at a time through the height values represented by a single arc (or super-arc) of the join and split tree. Height values are associated to each node of both the joint and the split trees. Moreover, to speed up the merging phase, it is observed that the up-degree of a node in the join tree is the same as the up-degree of the corresponding node in the contour tree. Analogously, the down-degree of a node in the split tree and its equivalent in the contour tree are the same. Taking advantage of this observation, leaf nodes of the contour tree are easily identified as nodes having up-degree 0 in the joint tree and down-degree 1 in the split tree, or nodes having up- degree 1 in the joint tree and down degree 0 in the split tree. Moreover, once a node has been inserted in the contour tree, it is removed from both the split and joint tree so that it is possible to recursively continue until the contour tree is computed. The following algorithm has been proposed in [20]:

1. *choose* a leaf node l by checking the up-degree and the down-degree of both joint and split trees (denoted J and S);
2. *remove* the edge e in J incident to l;
3. *remove* the node l from S. If l is not the global maximum in S, the nodes u and w adjacent to l in S are added to S by splicing (u, l) and (l, w);
4. *Add* the edge e to the contour tree T.

By recursively invoking this procedure, the smaller T is computed and the contour tree may be successively obtained. Moreover, the same authors propose an iterative version of the algorithm in which the queue of leaf nodes is maintained. The most efficient implementation of this method requires $O(tlog(t) + N\alpha(N))$ operations, where t is the number of nodes of the tree, N is the number of simplexes and α is the inverse of the Ackermann number. Details may be found in [21, 22, 20].

An algorithm that optimally (both in space and time) computes the contour tree for two- and three-dimensional scalar fields has been proposed in [23]. Here, the algorithm of Carr et al. [21, 22] has been slightly modified, by differing the way in which the join and the split tree are obtained. Instead of ordering all mesh vertices, the authors propose of firstly characterizing the critical points and, then, sorting and connecting them through monotone paths [23].

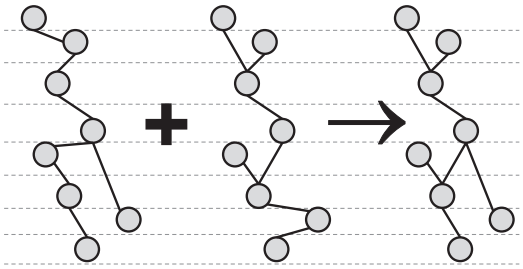


Fig. 9. The join (a) and the split tree (b) of a terrain model are merged in the contour tree (c). Image taken from [20].

Only the critical points where the number of contours of the level sets varies are needed for constructing the contour tree,. These critical points are denoted as *component-critical points* and may be identified through an initial (unordered) scan of the vertices analyzing each vertex neighborhood. The main advantage of this method is that only the component-critical points are ordered. This improves the computational complexity to $O(N + clog(c))$, where N is the number of simplexes and c the number of component-critical points of the mesh.

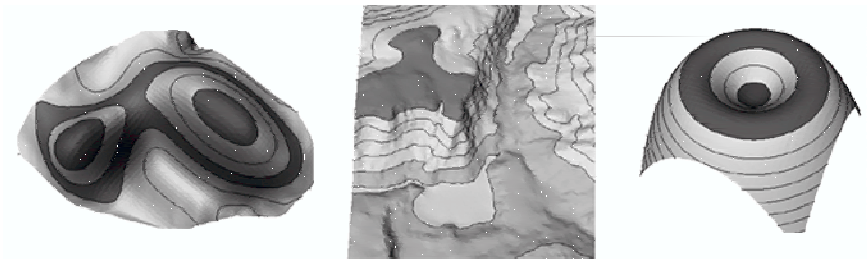
Finally, an extension of the algorithm in [83] to the three-dimensional domain has been proposed in [62] and further extended in [63], where the topology of the contour levels is detected with a parallel approach, which is based on a divide-and-conquer paradigm. This paradigm is used to compute both the joint and the split tree. In addition, a more detailed characterization of a contour is achieved by encoding the Betti numbers associated to each arc of a tree. The Betti numbers are computed by augmenting all reduced trees with all Morse critical points, and by making local revisions to the Betti numbers of each component during the merge phase. Also in this case, the tree (obtained by adding all nodes that correspond to Morse critical nodes) is called *augmented contour tree*. The meaning is different with respect to [21, 22] where it denotes the subdivision of the contour tree by all vertices of the input mesh. To avoid this ambiguous notation, Chiang et al. in [23] denote the augmented contour tree in [62] as *contour topology tree*. The complexity of the algorithm in [62] for a three-dimensional structured mesh, that is triangle mesh obtained from a regular grid, depends on the output size since it is $O(n + clog(n))$ where n is number of vertices of the mesh and c is the number of critical points [23]. However, in case of arbitrary simplicial meshes, even if the method is available, the need of splitting the mesh into sub-parts for the parallel computation does not improves the complexity which still is $O(nlog(n))$.

The approach in [45] extracts a tree-like structure for a volumetric dataset that may be considered equivalent to the contour tree (see [20] for details). The approach is based on the removal of simplexes (i.e., vertices, edges, faces and tetrahedra) in a mesh without removing the local maxima of the given function f. The process is heuristic: simplexes are removed as long as critical points of the height function are not disconnected (i.e., for any two critical points, there is a continuous chain of simplexes connecting them). The simplexes of the tree structure, provided by the algorithm, are used as seeds for the isosurface extraction procedure. Since all critical points are included and all are connected to each others, this structure intersects all possible contours. Moreover, cycles can be broken by removing a single cell. Thus, this structure will be a tree that, heuristically, corresponds to the contour tree. The authors remove simplexes (i.e., vertices, edges, faces and tetrahedra), preserving the connection be-

tween the mesh critical points. Even if the quality of the resulting tree is quite poor, the time complexity is linear in the number of vertices of the mesh.

Takahashi et al. [78] use an approach based on surface-networks to reconstruct the Reeb graph of a terrain surface. The terrain model is represented by a grid and the output structure encodes only the topological relationship among the critical points, neglecting intermediate points. The nodes of the two structures (surface network and Reeb graph) are identified by the critical points on a triangulation associated to the grid, which are detected through a classification criterion similar to that proposed by Banchoff [7]. Moreover, a global virtual minimum is introduced to give a unique interpretation of the surface behavior along its boundary [78, 13]. Then, the surface network of the model is obtained by connecting minima with passes and passes with maxima. This is done by analyzing the star of each vertex and following the steepest ascend direction. Finally, the contour tree extraction easily follows from the remark that, in case of generic height functions, the Reeb graph is a sub-graph of the surface network and it may be obtained with a finite number of pruning operations. The computational complexity of this algorithm is not directly given by the authors but an analysis is provided in [23], where it is claimed that the method requires $O(N)$ operations for reading the data, $O(nc)$ operation for finding the paths and $O(c^2)$ time for constructing the tree, where N, n and c are the number of simplexes, vertices and critical points of the mesh, respectively. Finally, this approach has been extended to volume data [79] represented by tetrahedral cells. In this case, the contour tree extraction is performed with the support of a so-called *voxel flow network* (instead of a surface network).

Finally, the algorithm for extracting the *extended Reeb graph* representation (ERG) of a two-dimensional scalar field proposed in [13] and extended in [14] is a special case of the more general approach to Reeb graph representation proposed in [1, 10]. The algorithm classifies a surface according to the a model decomposition induced by the insertion of a number of contour levels. When a scalar field is considered, each contour is either a simple closed line or an open line with the end points on the surface boundary B_S. If a region R intersects the surface boundary B_S, its external component is a closed sequence of open contours connected among them through open contours. In particular, the existence of the virtual minimum implies that, during the classification process, each border component has to be considered as a descending direction. In Figure 10(b), the dark regions represent critical areas, which belong to the boundary surface. Due to the assumption that all outgoing directions across the surface boundary B_S are descending, minima cannot be adjacent to B_S, and in this sense the classification of minima and maxima is not fully symmetrical.



(a) (b) (c)

Fig. 10. Maximum and saddle characterization for regions of a terrain surface (a), (b). In (c) a minimum and a non-simply connected maximum are presented.

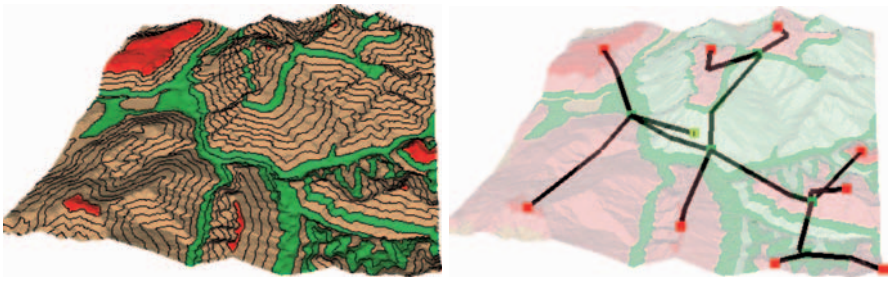
Since the critical areas have been recognized, the *ERG* is initialized by creating the node corresponding to the virtual minimum, V_M. The V_M is connected to the saddle having the minimum elevation and being external to each macro-node. If such a saddle does not exist, the V_M is connected to the nearest (in terms of geodesic distance) complex maximum area. Otherwise, if there are not complex maxima, the *ERG* is a trivial graph connecting the V_M to the only simple maximum existing and the surface is topologically equivalent to a sphere [54]. To detect arcs of the graph, contours and regions are topologically growned until a critical area, or the surface boundary B_S, is reached. The algorithm can be summarized as proposed in the following:

1. *recognition* of the critical areas;
2. *ordering* of the critical areas by elevation;
3. *expansion* of maxima and minima (leaf arc extraction);
4. *completion* of the set of arcs.

The set of arcs of the *ERG* is completed by iteratively running the following instructions:

1. *for each* non visited growing direction of a node N *expand* the region until another critical area or the surface boundary are reached;
2. *if* another critical area R has been reached, *connect* the node N with R.

An example of the *ERG* extraction in [14] for a real model is provided in Figure 11 (see also Figure CP-2 in Appendix F). The nodes of the *ERG* representation correspond to critical areas of the model.



(a) (b)

Fig. 11.

The computational cost of the whole algorithm for the *ERG* extraction is given by the sum of the costs of its single sub-parts (i.e., the insertion of contour levels into the mesh, the extraction of the critical areas and the final expansion process). In particular, both the extraction of the critical areas and the expansion process are linear in the number of triangles of the constrained mesh. Thus, the main contribution to the complexity of the algorithm are given by the insertion of the contours in the triangle mesh, which requires $O(nlog(n))$ operations in average, the extraction of the critical areas that costs $O(n+m)$, where m is the number of vertices inserted in the mesh during the slicing phase, and the ordering of the critical areas, which costs $O(Alog(A))$ operations at most, where A is the number of critical areas and generally small with respect to the number n of vertices of the mesh. Therefore, the computational cost of the overall graph construction is $O(\max((n+m), nlog(n)))$.

Cox et al. [26] propose a variant of the contour tree named *criticality tree* for volume data sets. This approach is based on the analysis of the isosurfaces without considering the classical Morse theory. In particular, the authors take advantage of the criteria provided by the *digital Morse theory* [48] to disambiguate the isosurfaces characterization of cubic grids and include also functions whose values are no unique. In practice, the criticality tree is a join tree with the insertion of the component-critical points rather than a contour tree. In fact, only maxima are leaf nodes of the criticality tree. Moreover, the criticality tree stores the evolution of the volumes that, starting from maxima, are bounded by the isosurfaces. These volumes are denoted topological zones and they are locally nested. Due to the need of extracting the topological zones, the computational complexity of this method is $O(kNlog(kN))$, where N is the of simplexes and k is the length of the longest path traversed in the tree.

More recently, the interest in time-dependent datasets has increased and methods for extracting the graph have been proposed for time-varying models: [76, 74]. The method in [76] describes how contours of a $2D$ and $3D$ dataset join and split during a time interval. In particular, the author investigates also the relationship between the contour tree in a domain and those restricted to its sub-domains. The contour tree of the whole domain and those of its sub-parts are computed in a pre-processing stage, using the sweep algorithm proposed in [23, 21, 22] and extracting the iso-surfaces from regularly sampled scalar fields, according to the method [9]. In particular, the domain is divided in time slices ($D \times \{t\}$, where $t \in [T_0, T_1]$ is an time integer), thick time slices ($D \times [t, t+1]$) and thick boundary slices ($(\delta D) \times [t, t+1]$). Therefore the novelty of this approach is in how contour trees relate among them rather than in the contour tree extraction algorithm. Similarly, the method proposed in [74] faces the problem of defining and computing the temporal correspondence of contours. Also in this case, contour trees are computed for each time in a pre-processing phase. The method chosen for the contour tree computation is the one proposed in [21, 22]. Then, the evolution in time of contours is computed through the analysis of the *significant* overlapping area between the contours of two successive time values. Once the contour correspondence has been identified, it is stored in a graph called a *topology change graph* that supports the visualization of the contour evolution. Computationally, for the detection of the contour correspondence, the labeling phase is the most expensive one: $O(nlog(n) + N + (c^t)^2)c^{t+1}$, where n and N are respectively the number vertices and tetrahedra of the mesh and c^t is the number of critical points of f at the time t.

6 Concluding Remarks

We have provided an overview and an analysis of algorithms which extract structural information from 2D and 3D scalar fields. We have analyzed algorithms for the extraction of critical points as well as algorithms which compute a cellular decomposition of the domain of the scalar field capturing the configuration of the critical points and the integral lines. We have also reviewed and analysed algorithms for computing the contour tree of a scalar field. The analysis and the classification have been performed on the basis of the dimension of the scalar field and on the underlying digital model of the scalar field.

From the application point of view, Morse and Morse-Smale complexes have proven to be useful tools in analyzing the morphology of terrains. Moreover they naturally provide a shape segmentation, which is suitable both for cutting a surface into a single flattenable piece, and for simplifying the model representation through the extraction of a combinatorial base domain. This is fundamental for several geometry processing tasks, such as parameterization, remeshing, surface texturing and deformation.

Structural problems, like over-segmentation in the presence of noise, or efficiency issues arising because of the very large size of existing data sets have been faced and solved by using generalization techniques and hierarchical representations. Beside the visual inspection skills of these descriptions, recent works use the Morse-Smale complex defined by the eigenfunctions of the Laplacian of the simplicial model as bases for the extraction of surface quadrangulations that are stable and intrinsic to the model [34].

Contour trees are mainly exploited in the visualization context. They have become popular in image processing and topography for their properties that allow a real time navigation of the data. In particular, the recent developments on this topic have highlighted their potential for analysing high-dimensional and time dependent data, like the visualization of the hemoglobin dynamic and the simulation of galaxy formation in the universe (see for example [74]).

Challenging research issues rely in developing techniques for computing the Morse-Smale decomposition for $3D$ and $4D$ scalar fields. The algorithm proposed in [35] is theoretically correct and its implementation [61] is efficient only for small size datasets. Region-based techniques seem to be more promising for extensions to higher dimensions, and the algorithm in [27] is already working in arbitrary dimension, and they are specific for computing the stable and unstable Morse complexes.

Acknowledgments

The authors want to thank Paola Magillo and Emanuele Danovaro from the University of Genova (Italy), Lidija Comic from the University of Novi Sad (Serbia) and Michela Spagnuolo from the IMATI-CNR (Italy) for many helpful discussions. This work has been partially supported by the MIUR-FIRB project SHALOM under contract number RBIN04HWR8, by the MIUR-PRIN project on *Multi-resolution modeling of scalar fields and digital shapes*, and by the National Science Foundation under grant CCF-0541032.

References

1. M. Attene, S. Biasotti, and M. Spagnuolo. Shape understanding by contour-driven retiling. *The Visual Computer*, 19(2-3):127–138, 2003.
2. U. Axen. *Topological Analysis using Morse Theory and Auditory Display*. PhD thesis, University of Illinois at Urbana-Champaign, 1998.
3. U. Axen. Computing Morse functions on triangulated manifolds. In *SODA '99: Proceedings of the 10th ACM-SIAM Symposium on Discrete Algoritms 1999*, pages 850–851. ACM Press, 1999.
4. C. L. Bajaj, V. Pascucci, and D. R. Shikore. Visualization of scalar topology for structural enhancement. In *Proceedings IEEE Visualization'98*, pages 51–58. IEEE Computer Society, 1998.
5. C. L. Bajaj and D. R. Shikore. Topology preserving data simplification with error bounds. *Computers and Graphics*, 22(1):3–12, 1998.
6. T. F. Banchoff. Critical points and curvature for embedded polyhedra. *Journal of Differential Geometry*, 1:245–256, 1967.
7. T. F. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *American Mathematical Monthly*, 77:475–485, 1970.

8. S. Beucher. Watershed, hierarchical segmentation and waterfall algorithm. In Jean Serra and Pierre Soille, editors, *Proc. Mathematical Morphology and its Applications to Image Processing*, pages 69–76, Fontainebleau, September 1994. Kluwer Ac. Publ.
9. P. Bhahaniramka, R. Wenger, and R. Crawfi. Iso-contouring in higher dimensions. In *IEEE Visualization 2000*, pages 267–273, 2000.
10. S. Biasotti. *Computational topology methods for shape modelling applications*. PhD thesis, University of Genova, May 2004.
11. S. Biasotti. Reeb graph representation of surfaces with boundary. In *SMI '04: Proceedings of Shape Modeling Applications 2004*, pages 371–374, Los Alamitos, Jun 2004. IEEE Computer Society.
12. S. Biasotti, D. Attali, J.-D. Boissonnat, H. Edelsbrunner, G. Elber, M. Mortasa, G. Sanniti di Baja, M. Spagnuolo, and M. Tanase. Skeletal structures. In *Shape Analysis and Structuring*. Springer, 2007.
13. S. Biasotti, B. Falcidieno, and M. Spagnuolo. Extended Reeb graphs for surface understanding and description. *Lecture Notes in Computer Science*, 1953:185–197, 2000.
14. S. Biasotti, B. Falcidieno, and M. Spagnuolo. *Surface Shape Understanding based on Extended Reeb Graphs*, pages 87–103. John Wiley & Sons, 2004.
15. R. Bott. Morse Theory Indomitable. *Publ Math.I.H.E.S.*, 68:99–117, 1998.
16. R.L. Boyell and H. Ruston. Hybrid techniques for real-time radar simulation. In *Proceedings of the 1963 Fall Joint Computer Conference*, Nov 1963.
17. P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A multi-resolution data structure for two-dimensional Morse functions. In G. Turk, J. van Wijk, and R. Moorhead, editors, *Proceedings IEEE Visualization 2003*, pages 139–146. IEEE Computer Society, October 2003.
18. P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):385–396, July/August 2004.
19. P.-T. Bremer, V. Pascucci, and B. Hamann. Maximizing adaptivity in hierarchical topological models. In *International Conference on Shape Modeling and Applications*, pages 300–309. IEEE Computer Society, 2005.
20. H. Carr. *Topological Manipulation of isosurfaces*. PhD thesis, The University of British Columbia, 2004.
21. H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In *SODA '00: Proceedings of the 11th ACM-SIAM Symposium on Discrete Algoritms 2000*, pages 918–926. ACM Press, 2000.
22. H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry : Theory and Applications*, 24:75–94, 2003.
23. Y.-J. Chiang, T. Lenz, X. Lu, and G. Rote. Simple and optimal output-sensitive construction of contour trees using monotone paths. *Computational Geometry: Theory and Applications*, 30:165–195, 2005.
24. K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in Reeb graphs of 2-manifolds. In *SCG '03: Proceedings of the 19th Annual Symposium on Computational Geometry 2003*, pages 344–350. ACM Press, 2003.
25. M. Couprie and G. Bertrand. Topological grayscale watershed transformation. In *Vision Geometry V. Proceedings SPIE 3168*, pages 136–146. SPIE, 1997.
26. J. Cox, D. B. Karron, and N. Ferdous. Topological zone organization of scalar volume data. *Journal of Mathematical Imaging and Vision*, 18:95–117, 2003.
27. E. Danovaro, L. De Floriani, P. Magillo, M. M. Mesmoudi, and E. Puppo. Morphology-driven simplification and multi-resolution modeling of terrains. In E. Hoel and P. Rigaux,

editors, *Proceedings ACM-GIS 2003 - The 11th International Symposium on Advances in Geographic Information Systems*, pages 63–70. ACM Press, November 2003.
28. E. Danovaro, L. De Floriani, and M. M. Mesmoudi. Topological analysis and characterization of discrete scalar fields. In T. Asano, R. Klette, and C. Ronse, editors, *Theoretical Foundations of Computer Vision, Geometry, Morphology, and Computational Imaging*, volume 2616 of *Lecture Notes on Computer Science*, pages 386–402. Springer Verlag, 2003.
29. E. Danovaro, L. De Floriani, L. Papaleo, and M. Vitali. A multi-resolution representation for terrain morphology. In M. Raubal, H.J. Miller, A.U. Frank, and M.F. Goodchild, editors, *Geographic Information Science, 4th International Conference, GIScience 2006, Münster, Germany, September 20-23, 2006, Proceedings*, volume 4197 of *Lecture Notes in Computer Science*, pages 33–46. Springer, 2006.
30. M. de Berg and M. van Kreveld. Trekking in the Alps without freezing or getting tired. *Algorithmica*, 19:306–323, 1997.
31. L. De Floriani, P. Magillo, and E. Puppo. Data structures for simplicial multi-complexes. In R. H. Guting, D. Papadias, and F. Lochovsky, editors, *Advances in Spatial Databases*, volume 1651 of *Lecture Notes in Computer Science*, pages 33–51. 1999.
32. L. De Floriani, E. Puppo, and P. Magillo. A formal approach to multi-resolution modeling. In W. Strasser, R. Klein, and R. Rau, editors, *Geometric Modeling: Theory and Practice*, pages 302–323. Springer-Verlag, 1997.
33. L. De Floriani, E. Puppo, and P. Magillo. Applications of computational geometry to Geographic Information Systems. In J. R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, chapter 7, pages 333–388. Elsevier Science, 1999.
34. Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. Spectral surface quadrangulation. *ACM Transactions on Graphics*, 25(3):1057–1066, July 2006.
35. H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale complexes for piecewise linear 3-manifolds. In *Proceedings 19th ACM Symposium on Computational Geometry*, pages 361–370, 2003.
36. H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse complexes for piecewise linear 2-manifolds. In *Proceedings 17th ACM Symposium on Computational Geometry*, pages 70–79. ACM Press, 2001.
37. H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete and Computational Geometry*, 30:87–107, 2003.
38. R. Forman. Morse theory for cell complexes. *Advances.in Mathematics*, 134:90–145, 1998.
39. R. Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134:90–145, 1998.
40. T. Gerstner and R. Pajarola. Topology-preserving and controlled topology simplifying multi-resolution isosurface extraction. In *Proceedings IEEE Visualization 2000*, pages 259–266, 2000.
41. C. Giertsen, A. Halvorsen, and P.R. Flood. Graph-directed modelling from serial sections. *The Visual Computer*, 6:284–290, 1990.
42. H. B. Griffiths. *Surfaces*. Cambridge University Press, 1976.
43. V. Guillemin and A. Pollack. *Differential Topology*. Englewood Cliffs, New Jersey, 1974.
44. Attila Gyulassy, Vijay Natarajan, Valerio Pascucci, Peer-Timo Bremer, and Bernd Hamann. Topology-based simplification for feature extraction from 3d scalar fields. In *Proceedings of IEEE Conference on Visualization*, 2005.

45. T. Itoh and K. Koyamada. Automatic isosurface propagation using extrema graph and sorted boundary cell lists. *IEEE Transactions on Visualization and Computer Graphics*, 1:319–327, 1995.
46. R. Jones. Connected filtering and segmentation using component trees. *Computer Vision and Image Understanding*, 75:215–228, 1999.
47. C. Jun, D. Kim, D. Kim, H. Lee, J. Hwang, and T. Chang. Surface slicing algorithm based on topology transition. *Computer-Aided Design*, 33(11):825–838, 2001.
48. D. B. Karron and J. Cox. Extracting 3D objects from volume data using digital morse theory. In *CVRMed '95: Proc. of the First Int. Conf. on Computer Vision, Virtual Reality and Robotics in Medicine*, volume 905 of *Lecture Notes in Computer Science*, pages 481–486, London, UK, 1995. Springer-Verlag.
49. P. Magillo, E. Danovaro, L. De Floriani, L. Papaleo, and M. Vitali. Extracting terrain morphology: A new algorithm and a comparative evaluation. In *2nd International Conference on Computer Graphics Theory and Applications*, March 2007.
50. A. Mangan and R. Whitaker. Partitioning 3D surface meshes using watershed segmentation. *IEEE Transaction on Visualization and Computer Graphics*, 5(4):308–321, 1999.
51. M. Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, 1987.
52. W. S. Massey. *A basic course in algebraic topology*. Springer-Verlag, New York, NY, USA, 1991.
53. F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 38:113–125, 1994.
54. J. Milnor. *Morse Theory*. Princeton University Press, New Jersey, 1963.
55. Mesmoudi Mohammed Mostefa and De Floriani Leila. Morphology-based representations of discrete scalar fields. In *In proceedings of the 2nd International Conference on Computer Graphics Theory*. ISI Proceedings, March 2007.
56. L. R. Nackman. Two-dimensional critical point configuration graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(4):442–450, 1984.
57. V. J. Natarajan and N. Pascucci. Volumetric data analysis using Morse-Smale complexes. In *Proceedings Shape Modeling International 2005*, 2005.
58. X. Ni, M. Garland, and J. C. Hart. Fair morse functions for extracting the topo-logical structure of a surface mesh. *ACM Transaction on Graphics*, 23(3):613–622, 2004.
59. L. Papaleo. *Surface Reconstruction: online mosaicing and Modeling with uncertnaity*. PhD thesis, Department of Computer Science - University of Genova, May 2004.
60. V. Pascucci. Topology diagrams of scalar fields in scientific visualization. In S. Rana, editor, *Topological Data Structures for Surfaces*, pages 121–129. John Wiley and Sons Ltd, 2004.
61. V. Pascucci. Generalization 3D Morse-Smale complexes. In *Dagstuhl Seminar on Scientific Visualization*, 2005 - to appear.
62. V. Pascucci and K. Cole-McLaughin. Efficient computation of the topology of the level sets. In *Proceedings of Visualization 2002*, pages 187–194. IEEE Press, 2002.
63. V. Pascucci and K. Cole-McLaughlin. Parallel computation of the topology of level sets. *Algorithmica*, 38:249–268, 2003.
64. Valerio Pascucci. On the topology of the level sets of a scalar field. In *Proceedings of the 13th Canadian Conference on Computational Geometry*, pages 141–144, University of Waterloo, Ontario, Canada, 2001.
65. T. K. Peucker and D. H. Douglas. Detection of Surface-Specific Points by Local Parallel Processing of Discrete Terrain Elevation Data. *Computer Graphics and Image Processing*, 4:375–387, 1975.
66. J. L. Pfaltz. Surface networks. *Geographical Analysis*, 8:77–93, 1976.

67. W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical recipes in c - second edition*. Cambridge University Press, 1992.
68. G. Reeb. Sur les points singulièrs d'une forme de Pfaff complètement intégrable ou d'une fonction numèrique. *Comptes Rendu de l'Academie des Sciences*, 222:847–849, 1946.
69. J. Roerdink and A. Meijster. The watershed transform: definitions, algorithms, and parallelization strategies. *Fundamenta Informaticae*, 41:187–228, 2000.
70. B. Schneider. Extraction of hierarchical surface networks from bilinear surface patches. *Geographical Analysis*, 37:244–263, 2005.
71. B. Schneider and J. Wood. Construction of metric surface networks from raster-based DEMs. In S. Rana, editor, *Topological Data Structures for Surfaces*, pages 53–70. John Wiley and Sons Ltd, 2004.
72. Y. Shinagawa, T. L. Kunii, and Y. L. Kergosien. Surface coding based on Morse theory. *Ieee Computer Graphics and Applications*, 11:66–78, 1991.
73. S. Smale. Morse inequalities for a dynamical system. *Bulletin of American Mathematical Society*, 66:43–49, 1960.
74. B.-S. Sohn and C. L. Bajaj. Time-varying contour topology. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):14–25, 2006.
75. P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag, Berlin and New York, 2004.
76. A. Szymczak. Subdomain aware contour trees and contour evolution in time-dependent scalar fields. In *SMI '05: Proceedings of Shape Modeling Applications 2005*. IEEE Press, 2005.
77. S. Takahashi. *Algorithms for Extracting Surface Topology from Digital Elevation Models*, pages 31–51. John Wiley & Sons Ltd, 2004.
78. S. Takahashi, T. Ikeda, Y. Shinagawa, T. L. Kunii, and M. Ueda. Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data. *Computer Graphics Forum*, 14:181–192, 1995.
79. S. Takahashi, Y. Takeshima, and I. Fujishiro. Topological volume skeletonization and its application to transfer function design. *Graphical Models*, 66(1):24–49, 2004.
80. S. P. Tarasov and M. N. Vyalyi. Construction of contour trees in 3D in O(nlogn) steps. In *SCG '98: Proceedings of the 14th Annual Symposium on Computational Geometry 1998*, pages 68–75. ACM Press, 1998.
81. H. Theisel and C. Roessl. Morphological representations of vector fields. In *Shape Analysis and Structuring*. Springer, 2007.
82. J. Toriwaki and T. Fukumura. Extraction of structural information from gray pictures. *Computer Graphics and Image Processing*, 7:30–51, 1978.
83. M. van Kreveld, R.van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *SCG '97: Proceedings of the 13th Annual Symposium on Computational Geometry 1997*, pages 212–220. ACM Press, 1997.
84. L. Vincent and S. Beucher. Introduction to the morphological tools for segmentation. *Traitement d'images en microscopie balayage et en microanalyse par sonde lectronique*, pages F1–F43, March 1990.
85. L. Vincent and P. Soille. Watershed in digital spaces: an efficient algorithm based on immersion simulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.
86. L. T. Watson, T. J. Laffey, and R. M. Haralick. Topographic classification of digital image intensity surfaces using generalized splines and the discrete cosine transformation. *Computer Vision, Graphics, and Image Processing*, 29:143–167, 1985.

87. G. H. Weber, G. Schueuermann, H. Hagen, and B. Hamann. Exploring scalar fields using critical isovalues. In *Proceedings IEEE Visualization 2002*, pages 171–178. IEEE Computer Society, 2002.
88. G. H. Weber, G. Schueuermann, and B. Hamann. Detecting critical regions in scalar fields. In G.-P. Bonneau, S. Hahmann, and C. D. Hansen, editors, *Proceedings Data Visualization Symposium*, pages 85–94. ACM Press, New York, 2003.
89. G. W. Wolf. Topographic surfaces and surface networks. In S. Rana, editor, *Topological Data Structures for Surfaces*, pages 15–29. John Wiley and Sons Ltd, 2004.

Topological Representations of Vector Fields

Holger Theisel[1], Christian Rössl[2], and Tino Weinkauf[3]

[1] Bielefeld University theisel@techfak.uni-bielefeld.de
[2] INRIA Sophia-Antipolis christian.roessl@sophia.inria.fr
[3] Zuse Institute Berlin (ZIB) weinkauf@zib.de

Summary. This chapter gives an overview on topological methods for vector field processing. After introducing topological features for 2D and 3D vector fields, we discuss how to extract and use them as visualization tools for complex flow phenomena. We do so both for static and dynamic fields. Finally, we introduce further applications of topological methods for compressing, simplifying, comparing, and constructing vector fields.

1 Introduction

Vector fields appear in many areas of science, engineering, and industry. In recent years, a variety of methods to process, model, analyze and visualize vector fields have been developed. Similar to other areas of Computer Graphics, a common challenge is the dramatically increasing size and complexity of the vector fields. One common approach to processing vector fields is *feature extraction* [24]. Features represent certain interesting objects or structures in the vector field like topological features, vortex core lines, or shock waves. The idea of feature extraction is to detect, extract and track these features and use them instead of the whole data set for further processing.

Among the feature extraction techniques, topological methods have gained a rather high popularity because they offer to describe even complex flow behaviors by only a limited number of graphical primitives. The main idea of them is to segment the vector field into areas of different flow behavior.

Topological structures are well-studied in the context of dynamical systems and partial differential equations [1, 3, 10]. However, in recent years they attracted the Visualization community, leading to a quite intensive research on how to use them as visualization tools.

In this chapter, we give an overview of topological methods for vector field processing. The main class of applications we have in mind is the visualization of flow structures (sections 2–4). In addition, we discuss further applications of topological methods for vector fields (section 5).

2 Topological features of 2D vector fields

2.1 Concepts

To describe topological features of 2D vector fields in detail, we start with a steady 2D vector field

$$\mathbf{v}(x,y) = \begin{pmatrix} u(x,y) \\ v(x,y) \end{pmatrix} \tag{1}$$

and assume $\mathbf{v}$ to be continuous and differentiable. Then the *Jacobian matrix* $\mathbf{J_v}$ is a 2×2 matrix which is defined in every point of the domain of the vector field by

$$\mathbf{J_v}(x,y) = (\mathbf{v}_x, \mathbf{v}_y) = \begin{pmatrix} u_x(x,y) & u_y(x,y) \\ v_x(x,y) & v_y(x,y) \end{pmatrix}. \tag{2}$$

The determinant of $\mathbf{J_v}$ is called *Jacobian* of $\mathbf{v}$.

A point $\mathbf{x}_0 \in E_2$ is called a *critical point* if $\mathbf{v}(\mathbf{x}_0) = (0,0)^T = \mathbf{0}$ and $\mathbf{v}(\mathbf{x}) \neq \mathbf{0}$ for any $\mathbf{x} \neq \mathbf{x}_0$ in a certain neighborhood of $\mathbf{x}_0$.

A *stream line* $\mathbf{s}(t)$ of the vector field $\mathbf{v}$ is a curve in the domain of $\mathbf{v}$ with

$$\dot{\mathbf{s}}(t) = \mathbf{v}(\mathbf{s}(t)) \tag{3}$$

for any t of the domain of $\mathbf{s}$. In (3), $\dot{\mathbf{s}}$ denotes the tangent vector of $\mathbf{s}$. Considering the vector field $\mathbf{v}$ as the velocity field of a steady flow, a stream line describes the path of a massless particle set out at a certain location in the flow.

Stream lines do not intersect each other (except for critical points of $\mathbf{v}$). Given a point in the flow, there is one and only one stream line through it (except for critical points of $\mathbf{v}$).

Classification of critical points

To classify a critical point in a 2D steady vector field, sectors of different flow behavior around it have to be considered. Three kinds of sectors can be distinguished ([7]):

- In a *parabolic sector* either all stream lines end, or all stream lines originate, in the critical point. Figure 1a shows an example.
- In a *hyperbolic sector* all stream lines pass by the critical point, except for two stream lines being the boundaries of the sector. One of these two stream lines ends in the critical point while the other one originates in it. Figure 1b shows an example.
- In an *elliptic sector* all stream lines originate and end in the critical point. Figure 1c shows an example.

A critical point in a 2D vector field is completely classified by specifying number and order of all sectors around it. Consider figure 2a for an example. This critical point consists of 7 sectors in the following order: hyperbolic, elliptic, hyperbolic, elliptic, parabolic, hyperbolic, hyperbolic.

The different sectors are delimited by stream lines originating or ending in the critical point. Figure 2b shows such a stream line delimiting two hyperbolic sectors.

Each critical point can be assigned an *index*:

$$\text{index} = 1 + \frac{n_e - n_h}{2} \tag{4}$$

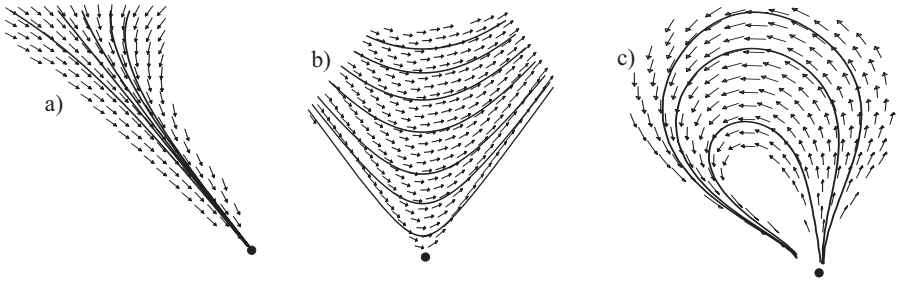


Fig. 1. Sectors of a critical point; a) parabolic sector; b) hyperbolic sector; c) elliptic sector (from [42]).

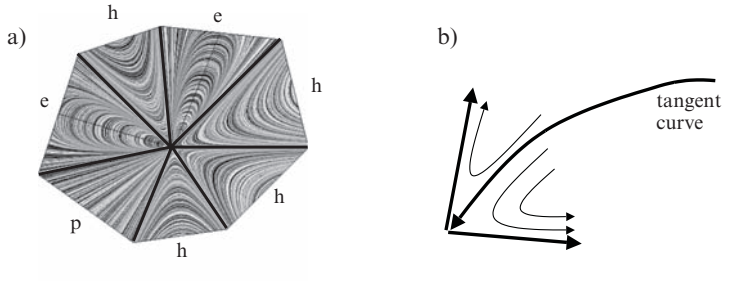


Fig. 2. a) general critical point; b) stream line separating two hyperbolic sectors.

where n_e is the number of elliptic sectors and n_h is the number of hyperbolic sectors. The index can also be interpreted as the number of counterclockwise revolutions made by the vectors of $\mathbf{v}$ while travelling counterclockwise on a closed curve around the critical point (the closed curve must be so tight to the critical point that no other critical point is inside it).

The index can be considered as an overview of the complexity of a critical point but does not cover the complete classification: there are critical points with different sectors but the same index. An further introduction to the classification of 2D critical points and their indices can be found in [7].

A critical point $\mathbf{x}_0$ in the vector field $\mathbf{v}$ is called a *first-order critical point* if the Jacobian does not vanish in $\mathbf{x}_0$; otherwise the critical point is called *high-order critical point*. As shown in [13] and [14], the classification of critical points $\mathbf{x}_0 = (x_0, y_0)$ in the vector field $\mathbf{v}$ simplifies if $\mathbf{x}_0$ is a first order critical point. In this case a first order Taylor expansion

$$\mathbf{v}_{T1,\mathbf{x}_0} = \begin{pmatrix} u_x(\mathbf{x}_0) & u_y(\mathbf{x}_0) \\ v_x(\mathbf{x}_0) & v_y(\mathbf{x}_0) \end{pmatrix} \cdot \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} \tag{5}$$

of the flow around $\mathbf{x}_0$ is sufficient to obtain the complete classification of it. (5) ensures that

$$\mathbf{J}_{\mathbf{v}}(\mathbf{x}_0) = \mathbf{J}_{\mathbf{v}_{T1,\mathbf{x}_0}}(\mathbf{x}_0). \tag{6}$$

It turns out that for $\det(\mathbf{J}_{\mathbf{v}}(\mathbf{x}_0)) < 0$, the critical point $\mathbf{x}_0$ consists of 4 hyperbolic sectors and therefore has an index of -1. A critical point of this classification is called a *saddle point*. In this case the eigenvectors of $\mathbf{J}_{\mathbf{v}}(\mathbf{x}_0)$ denote the delimiters of the hyperbolic areas around $\mathbf{x}_0$. For $\det(\mathbf{J}_{\mathbf{v}}(\mathbf{x}_0)) > 0$, the critical point $\mathbf{x}_0$ consists of one parabolic sector and therefore has an index of +1.

This classification of a first order critical point $\mathbf{x}_0$ with an index of +1 can be refined by considering the eigenvalues of $\mathbf{J_v}(\mathbf{x}_0)$. Let R_1, R_2 be the real parts of the eigenvalues of $\mathbf{J_v}(\mathbf{x}_0)$, and let I_1, I_2 be the imaginary parts of the eigenvalues of $\mathbf{J_v}(\mathbf{x}_0)$. Then the refined classification following [13] is shown in figure 3. Note that positive real parts denote a repelling behavior of the flow while negative real parts indicate an attracting behavior. Non-zero imaginary parts denote a circulating behavior of the flow. [23] detects and classifies critical

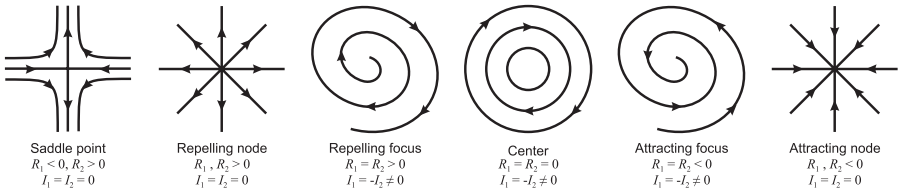


Fig. 3. Classification of first order critical points; R_1, R_2 denote the real parts of the eigenvalues of the Jacobian matrix while I_1, I_2 denotes its imaginary parts (from [13]).

points using a discrete Hodge decomposition.

Boundary switch points

Vector fields are usually defined over a limited domain. Along its boundary curves, the vector field has either an inflow or an outflow behavior. *Boundary switch points* separate these areas. A boundary switch point is a point on the boundary curve with the property that the tangent of the boundary curve is parallel to the vector of the field there. Two kinds of boundary switch points can be distinguished: inbound or outbound points. At an inbound point, the stream line starting there in forward and backward direction goes into the domain of $\mathbf{v}$, while for an outbound point it leaves the domain immediately. Figure 4 (a) and (b) give an illustration.

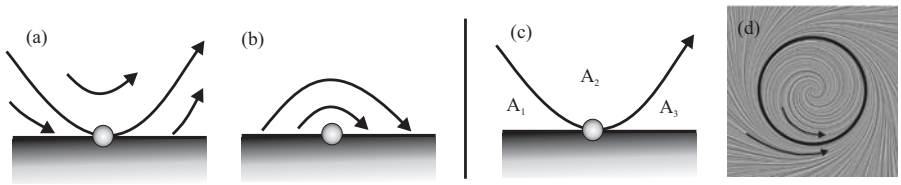


Fig. 4. (a) inbound boundary switch point; (b) outbound boundary switch point; (c) separatrix from inbound boundary switch points divides the domain into 3 sectors A_1, A_2, A_3; (d) an isolated closed stream line divides the domain into 2 sectors.

Separatrices

Separatrices are stream lines which divide the domain of $\mathbf{v}$ into areas of different flow behavior. Different types of separatrices are possible:

- Each tangent curve originating/ ending in the critical point and separating two sectors there is a separatrix. Figure 2b illustrates a separatrix which separates two hyperbolic sectors of a critical point.
- Stream lines from inbound boundary switch points divide the domain into 3 different areas. Figure 4(c) gives an illustration.
- Isolated closed stream lines are separatrices. Figure 4(d) gives an illustration.

2.2 Visualizing 2D topology

After the introduction of topological methods as a visualization tool for 2D vector fields in [13], an intensive research has been done in this field. [26] treats higher order critical points. In [5], separatrices starting from boundary switch points are discussed. [46] and [37] give methods to detect closed separatrices. To visualize the topology of a 2D vector field, critical points, boundary switch points, and separatrices have to be extracted. Critical points can be extracted directly (in case of a piecewise (bi-)linear vector field) or numerically. Also, boundary switch points can be found by a closed solution.

Most visualization approaches consider only first order critical points. Then the starting directions of the separatrices are the eigendirections of the Jacobian matrices at the saddle points.

For integrating stream lines (for instance separatrices), usually numerical methods are applied[4]. Standard is a fourth order Runge-Kutta integration [27]. Figure 5 shows an example of

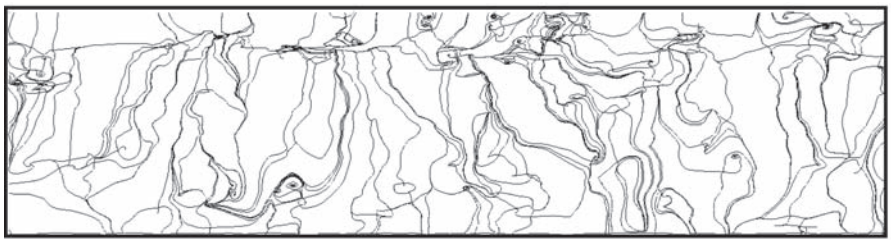


Fig. 5. Topological skeleton of the skin friction data set.

a topological skeleton of a 2D vector field describing the skin friction on a face of a cylinder[5].

Isolated closed stream lines can only be extracted and visualized by a global analysis of the vector field. [46] uses the underlying grid structure of a piecewise linear vector field: each grid cell is analyzed concerning the re-entering behavior of the stream lines starting at its boundaries. [37] presents an approach which uses the fact that searching isolated stream lines in 2D vector fields corresponds to intersecting stream surfaces in certain 3D vector fields. Figure 6 gives an illustration (see also Figure CP-1 in Appendix G).

[4] Only for piecewise linear vector fields, a closed solution exists [21].

[5] The data set was generated by R.W.C.P. Verstappen and A.E.P. Veldman of the University of Groningen (the Netherlands).

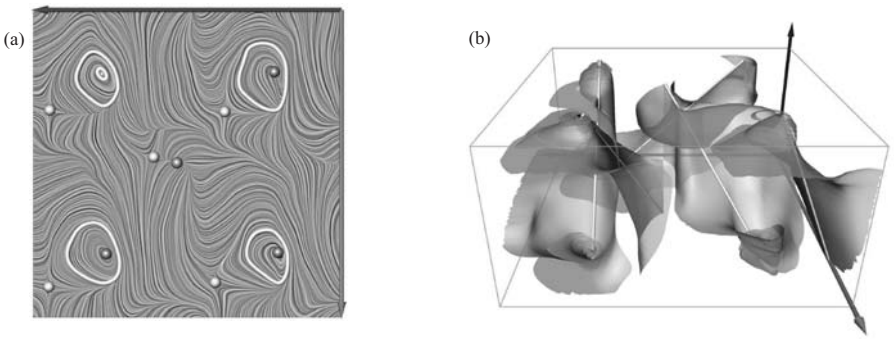


Fig. 6. (a) detected closed stream lines in a 2D vector field; (b) to get them, certain stream surfaces of a 3D vector field are integrated and intersected (from [37]).

3 Topological Features of 3D Vector Fields

3.1 Concepts

Topological structures of 3D vector fields are well-understood in the visualization community for many years [14, 2, 4, 22]. In this section, we collect the most important concepts and properties.

Critical points

Given a 3D vector field $\mathbf{v} : \mathbb{E}^3 \rightarrow \mathbb{R}^3$, a first order critical point $\mathbf{x}_0$ (i.e., a point $\mathbf{x}_0$ with $\mathbf{v}(\mathbf{x}_0) = \mathbf{0}$ and $\det(\mathbf{J}_\mathbf{v}(\mathbf{x}_0)) \neq 0$, where $\mathbf{J}_\mathbf{v}(\mathbf{x}) = \nabla\mathbf{v}(\mathbf{x})$ is the Jacobian matrix of $\mathbf{v}$, can be classified by an eigenvalue/eigenvector analysis of $\mathbf{J}_\mathbf{v}(\mathbf{x}_0)$. Let $\lambda_1, \lambda_2, \lambda_3$ be the eigenvalues of $\mathbf{J}_\mathbf{v}(\mathbf{x}_0)$ ordered according to their real parts, i.e. $Re(\lambda_1) \leq Re(\lambda_2) \leq Re(\lambda_3)$. Furthermore, let $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ be the corresponding eigenvectors, and let $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3$ be the eigenvectors of the transposed Jacobian $(\mathbf{J}_\mathbf{v}(\mathbf{x}_0))^T$ corresponding to $\lambda_1, \lambda_2, \lambda_3$. (Note that $\mathbf{J}$ and $\mathbf{J}^T$ have the same eigenvalues but not necessarily the same eigenvectors.) Concerning the real parts of the eigenvalues, the following classification of critical points is possible:

- sources: $0 < Re(\lambda_1) \leq Re(\lambda_2) \leq Re(\lambda_3)$
- repelling saddles: $Re(\lambda_1) < 0 < Re(\lambda_2) \leq Re(\lambda_3)$
- attracting saddles: $Re(\lambda_1) \leq Re(\lambda_2) < 0 < Re(\lambda_3)$
- sinks: $Re(\lambda_1) \leq Re(\lambda_2) \leq Re(\lambda_3) < 0$

Each of these classes can be further divided into two stable[6] subclasses by deciding if imaginary parts in the eigenvalues are present. Since vector fields usually consist of a finite number of critical points, an iconic representation is the appropriate visualization approach. Several icons have been proposed in the literature, see [14, 9, 19, 11]. In the following we describe the different classes of critical points as well as the icons which were used in [36, 43] for their visual representation. These icons were colored depending on the flow behavior: Attracting parts (inflow) are colored blue, while repelling parts (outflow) are colored red.

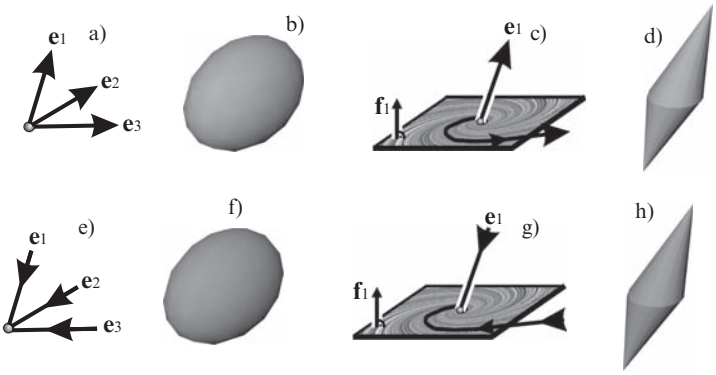


Fig. 7. Sources and sinks; (a) repelling node and (b) its icon; (c) repelling focus and (d) its icon; (e) attracting node and (f) its icon; (g) attracting focus and (h) its icon (from [43]).

Sources and Sinks

A source $\mathbf{x}_{So}$ is characterized by the fact that in its neighborhood all stream lines diverge from $\mathbf{x}_{So}$. The two stable subclasses are repelling nodes and repelling foci.

A *repelling node* is characterized by the absence of imaginary parts in $\lambda_1, \lambda_2, \lambda_3$, and $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ are linearly independent (Figure 7a). To visualize a repelling node, we use a red ellipsoid with a shape determined by the eigenvectors and eigenvalues of the Jacobian (Figure 7b).

A *repelling focus* is characterized by the presence of two eigenvalues with imaginary parts, say λ_2, λ_3. In this case, the only real eigenvector $\mathbf{e}_1$ of $\mathbf{J}$ describes the direction of straight outflow. In addition, there is a plane in which a 2D repelling focus behavior appears. This plane is perpendicular to the only real eigenvector $\mathbf{f}_1$ of $\mathbf{J}^T$ (Figure 7c). As an icon, we used a red double cone representing the outflow plane and the outflow direction by its shape (Figure 7d).

A sink $\mathbf{x}_{Si}$ can be considered as an inverse source: in its neighborhood all stream lines converge to $\mathbf{x}_{Si}$. The two subcases are *attracting nodes* (Figures 7e-f) and *attracting foci* (Figures 7g-h). See also Figure CP-2 in Appendix G.

Repelling Saddles and Attracting Saddles

A repelling saddle $\mathbf{x}_R$ has one direction of inflow behavior (called *inflow direction*) and a plane in which a 2D outflow behavior occurs (called *outflow plane* through $\mathbf{x}_R$). For all other directions around $\mathbf{x}_R$, the stream lines do not touch $\mathbf{x}_R$. The two stable subclasses are repelling node saddles and repelling focus saddles.

A *repelling node saddle* has no imaginary parts in $\lambda_1, \lambda_2, \lambda_3$, and $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ are linearly independent (Figure 8a). Its icon includes a red ellipse denoting the outflow plane defined by $\mathbf{e}_2, \mathbf{e}_3$ and λ_2, λ_3, while a blue arrow pointing to the center of the ellipse represents the inflow direction (Figure 8b).

[6] A critical point in $\mathbf{v}$ is called stable if a small perturbation of $\mathbf{v}$ does not change the classification of the critical point.

A *repelling focus saddle* is characterized by $Im(\lambda_2) = -Im(\lambda_3) \neq 0$. Here, the only real eigenvector $\mathbf{e}_1$ of $\mathbf{J}$ describes the inflow direction. The only real eigenvector $\mathbf{f}_1$ of $\mathbf{J}^T$ describes the plane with the 2D repelling focus behavior (Figures 8c-d).

An attracting saddle $\mathbf{x}_A$ can be interpreted as an inverse version of a repelling saddle. It has one direction of outflow behavior (*outflow direction*) and a plane in which a 2D inflow behavior appears (*inflow plane* through $\mathbf{x}_A$). The two stable subclasses are *attracting node saddles* without imaginary parts of the eigenvalues (Figures 8e-f) and *attracting focus saddles* (Figures 8g-h).

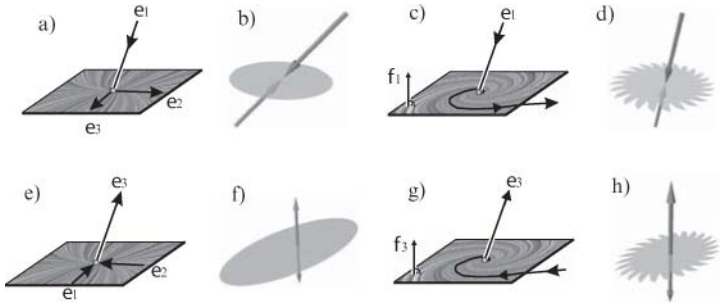


Fig. 8. Repelling and attracting saddles; (a) repelling node saddle and (b) its icon; (c) repelling focus saddle and (d) its icon; (e) attracting node saddle and (f) its icon; (g) attracting focus saddle and (h) its icon (from [43]).

See also Figure CP-3 in Appendix G.

Unstable Critical Points

In addition to the kinds of critical points described above, a number of unstable versions of sources, sinks and repelling/attracting saddles exist. Also, two further classes of unstable critical points exist which do not belong to any of the above-mentioned classes: attracting centers and repelling centers. A repelling center is characterized by $Re(\lambda_1) = Re(\lambda_2) = 0 < Re(\lambda_3)$ and $Im(\lambda_1) = -Im(\lambda_2) \neq 0$. It consists of one direction $\mathbf{e}_3$ of outflow behavior and one plane perpendicular to $\mathbf{f}_3$ with a 2D circulating behavior. An attracting center has $Re(\lambda_1) < 0 = Re(\lambda_2) = Re(\lambda_3)$ and $Im(\lambda_2) = -Im(\lambda_3) \neq 0$. The inflow direction is defined by $\mathbf{e}_1$ and the 2D circulating behavior can be found in the plane perpendicular to $\mathbf{f}_1$.

Boundary switch curves

Consider the 3D vector field $\mathbf{v}$ in the domain

$$D = (x_{min}, x_{max}) \times (y_{min}, y_{max}) \times (z_{min}, z_{max}) \tag{7}$$

with $x_{min} < x_{max}$, $y_{min} < y_{max}$, $z_{min} < z_{max}$. The boundary surfaces of D (which are the 6 faces of the bounding box) consist of outflow and inflow areas which are separated by *boundary switch curves*. Boundary switch curves consist of all points on the boundary where the flow direction is tangential to the boundary surface. Figure 9(a) illustrates an example of the boundary plane $z = z_{min}$ consisting of one inflow and one outflow area. (In the following

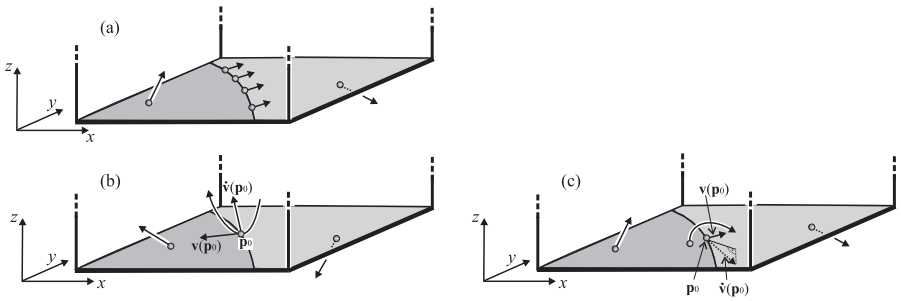


Fig. 9. (a) boundary plane $z = z_{min}$ consisting of an inflow area (blue), an outflow area (red), and their separating boundary switch curve; shown are 4 vectors of $\mathbf{v}$ on the boundary switch curve, and one each in the inflow and outflow area; (b) inbound point $\mathbf{p}_0$ on a boundary switch curve: $\mathbf{v}(\mathbf{p}_0)$ points into the inflow area, $\dot{\mathbf{v}}(\mathbf{p}_0)$ points inside D; shown is a part of the stream line starting in $\mathbf{p}_0$ both in forward and backward integration; (c) outbound point $\mathbf{p}_0$ on a boundary switch curve; $\mathbf{v}(\mathbf{p}_0)$ points into the outflow area, $\dot{\mathbf{v}}(\mathbf{p}_0)$ points outside D; shown is a stream line close to $\mathbf{p}_0$ starting in the inflow area and leaving D in the outflow area.

we illustrate the concept of boundary switch curves only on the boundary plane $z = z_{min}$. Similar statements hold for the 5 remaining boundary planes of D.)

Given a point $\mathbf{p}_0$ on a boundary switch curve, two cases are possible concerning the stream line starting at $\mathbf{p}_0$:

- Starting from $\mathbf{p}_0$, the stream line integration moves inside D for both backward and forward integration. We call this point an *inbound point* on the boundary switch curve (Figure 9(b)).
- Starting from $\mathbf{p}_0$, the stream line integration moves outside D for both backward and forward integration. Therefore, this stream line in D consists only of $\mathbf{p}_0$ itself. We call this point an *outbound point* (Figure 9(c)).

Separatrices

Separatrices are curves or surfaces which separate regions of different flow behavior. Since around sources and sinks a homogeneous flow behavior is present (either a complete outflow or inflow), sources and sinks do not contribute to separatrices. A repelling saddle $\mathbf{x}_R$ creates two separatrices: one separation curve (which is a stream line starting in $\mathbf{x}_R$ in the inflow direction by backward integration) and a separation surface (which is a stream surface starting in the outflow plane by forward integration). Separatrices are also emanating from inbound segments of boundary switch curves.

Figure 10 shows the topological visualization of a data set describing the electrostatic field of a Benzene molecule[7]. Figure 10(a) shows the iconic representation of all 184 critical points. Figure 10(b) shows the critical points and the separation surfaces starting from the saddles, while figure 10(c) additionally shows the separatrices emanating from boundary switch curves. See also CP-4 in Appendix G.

[7] This data set was calculated on a 101^3 regular grid using the fractional charges method described in [28].

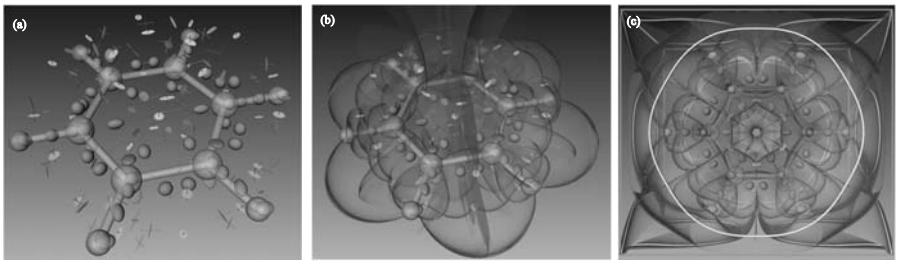


Fig. 10. Topological representations of the benzene data set with 184 critical points; (a) iconic representation; (b) separation surfaces starting from saddles; (c) separation surfaces starting from saddles and boundary switch curves (from [36, 43]).

Saddle- and boundary switch connectors

A saddle connector is the intersection curve between a separation surface starting from an attracting saddle and a separation surface starting from a repelling saddle. Figure 11(a),(b) give an illustration.

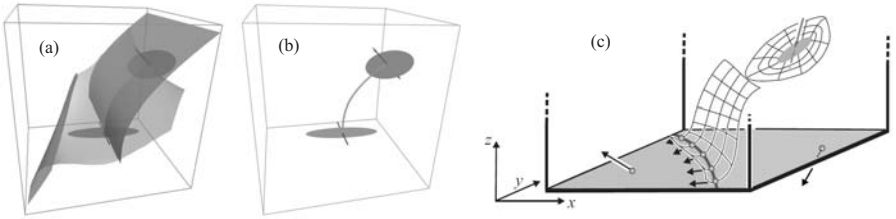


Fig. 11. (a) separation surfaces of two saddles; (b) the intersection of the separation surfaces is the saddle connector; (c) finding the intersection of two separation surfaces one comes from a saddle, while the other one comes from a boundary switch curve (from [36, 43]).

Boundary switch connectors are the intersection curves of separation surfaces starting from saddles or boundary switch curves. Figure 11(c) illustrates this. Concerning the different kinds of separation surfaces, 4 kinds of boundary switch connectors are possible. They are shown in figure 12

3.2 Visualizing 3D topology

Given a 3D vector field $\mathbf{v}$, the critical points can be extracted directly (if $\mathbf{v}$ is piecewise linear) or numerically. Then the classification is done by an eigenvalue/eigenvector analysis of the Jacobian matrix.

Stream surfaces are obtained by a numerical stream surface integration [15, 25]. A standard approach is a 4th order Runge-Kutta integration. The result is a triangular mesh representing the stream surface which can be represented in a semitransparent way (figures 10(b),(c)).

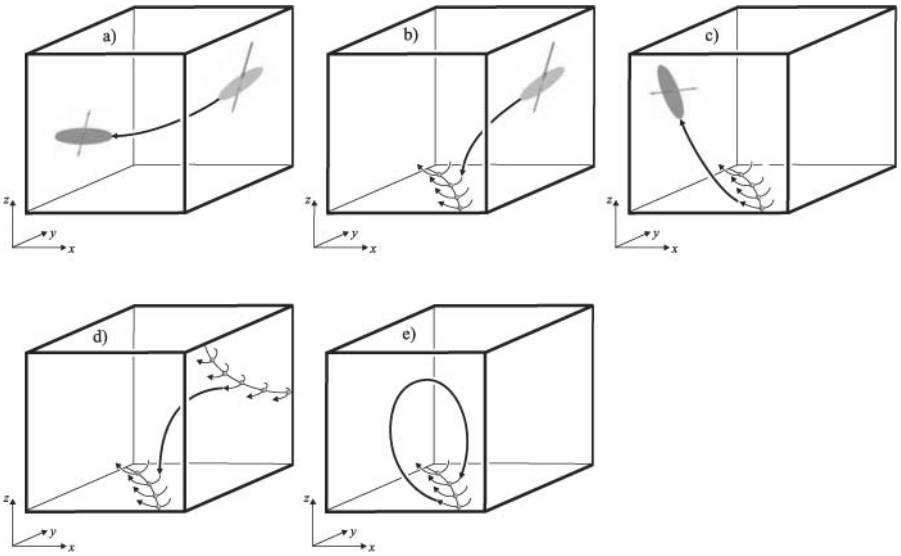


Fig. 12. Cases of intersection curves of separation surfaces: a) saddle connectors; b)-e) boundary switch connectors (from [43]).

Saddle connectors can serve as a visual alternative to visualizing separation surfaces since they tend to hide themselves and other features and thus produce cluttered visual representations (figures 10(b),(c)). In order to extract saddle connectors, stream surfaces have to be intersected. To do so, the front of the evolving stream surfaces are observed for intersection. Figure 13 gives an illustration. Boundary switch connectors are extracted in a similar way.

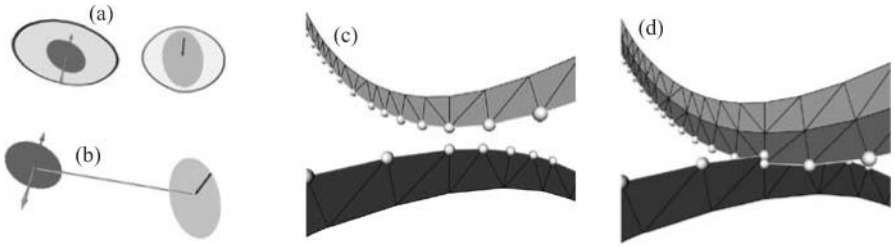


Fig. 13. Extracting a saddle connector: (a) simultaneously observe fronts of evolving stream surfaces; (b) stream line integration from intersection point gives saddle connector; (c) closeup shortly before intersection is found; (d) intersection is found (from [36]).

Example:

Figures 14a–d visualize a snapshot of a transitional wake behind a circular cylinder [48]. See also Figure CP-5 in Appendix G. This flow exhibits periodic vortex shedding leading to the well known von Kármán vortex street. This phenomenon plays an important role in many industrial applications, like mixing in heat exchangers or mass flow measurements with vortex counters. However, this vortex shedding can lead to undesirable periodic forces on obstacles, like chimneys, buildings, bridges and submarine towers.

This data set was derived from a direct numerical simulation of the Navier-Stokes equation by Gerd Mutschke [20]. The data resolves the so-called 'mode A' of the 3D transition at a Reynolds number of 200 and at a spanwise wavelength of 4 diameters. The figures display a small near-wake region of a large computational domain. All 13 critical points are contained in the shown domain and on its boundaries 13 boundary switch curves are observed. Together they span the topological skeleton of the incompressible velocity field.

The inspection of figure 14a suggests a high amount of circulating flow behavior in the data set, but due to the occlusion effects introduced by the separation surfaces neither the flow behavior on the boundaries nor the critical points can be seen easily. This complicates further examinations to a high degree.

The simplified topological skeletons shown in Figures 14b–d enable to reduce this high-dimensional data set to a simple conceptual flow representation from which qualitative conclusions can be drawn. Using connectors, the skeleton elucidates the symmetry of the mode A with respect to a plane which is perpendicular to the cylinder axis. The high number of spanwise and transverse running connectors of a single snapshot already indicate the experimentally observed good mixing properties of vortex shedding.

4 Topological features of time-dependent vector fields

Up to now we treated the topology of steady (time-independent) vector fields by segmenting areas of similar behavior of the stream lines. For time-dependent vector fields, there are two important classes of characteristic curves: stream lines and path lines. Hence, two different kinds of topologies can be obtained: a stream line and a path line oriented topology. We explain both concepts for 2D time-dependent vector fields and mention that – except for some special configurations [8] – the topology of 3D time-dependent vector field is rather unsolved.

Given a 2D time-dependent vector field

$$\mathbf{v}(x,y,t) = \begin{pmatrix} u(x,y,t) \\ v(x,y,t) \end{pmatrix}, \tag{8}$$

where (x,y) describe the 2D domain and t is the temporal component, stream and path lines are generally different classes of curves. Stream lines are the tangent curves of $\mathbf{v}$ for a fixed time t, while path lines describe the paths of massless particles in $\mathbf{v}$ over time. To capture both types of lines, we define two 3D vector fields and consider their topological behavior.

To treat stream lines and path lines of $\mathbf{v}$, we consider

$$\mathbf{s}(x,y,t) = \begin{pmatrix} u(x,y,t) \\ v(x,y,t) \\ 0 \end{pmatrix} \quad , \quad \mathbf{p}(x,y,t) = \begin{pmatrix} u(x,y,t) \\ v(x,y,t) \\ 1 \end{pmatrix}. \tag{9}$$

Both $\mathbf{s}$ and $\mathbf{p}$ can be seen as a steady 3D vector field. The stream lines of $\mathbf{s}$ coincide with the stream lines of $\mathbf{v}$, since any integration step in $\mathbf{s}$ keeps the time component unchanged. Any

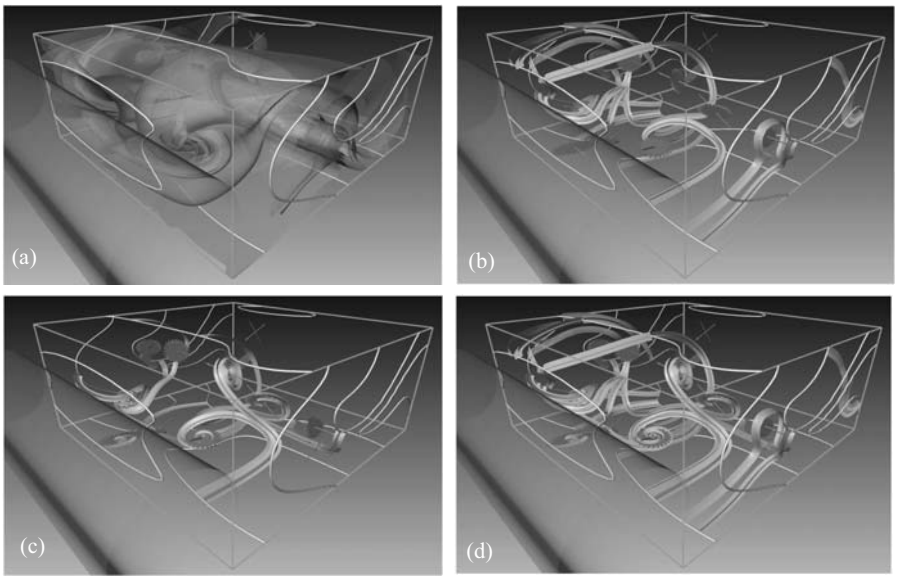


Fig. 14. Flow behind a circular cylinder: (a) separation surfaces emanating from boundary switch curves and saddles; (b) boundary switch connectors between boundary switch curves; (c) boundary switch connectors between saddle points and boundary switch curves; (d) saddle connectors and both types of boundary switch connectors (from [43]).

(x, y)-slice through $\mathbf{s}$ represents $\mathbf{v}$ at a constant time. The stream lines of $\mathbf{p}$ coincide with the path lines of $\mathbf{v}$: given a starting point $(\mathbf{x}_0, t_0)$, one step of a simple Euler approximation of $\mathbf{p}$ would be

$$\begin{pmatrix} \mathbf{x}_1 \\ t_1 \end{pmatrix} = \begin{pmatrix} \mathbf{x}_0 \\ t_0 \end{pmatrix} + d\ \mathbf{p}(\mathbf{x}_0, t_0) = \begin{pmatrix} \mathbf{x}_0 + d\ \mathbf{v}(\mathbf{x}_0, t_0) \\ t_0 + d \end{pmatrix} \tag{10}$$

which does not only change the location but also goes forward in time.[8] Figure 15 illustrates $\mathbf{s}$ and $\mathbf{p}$ for a simple example vector field $\mathbf{v}$. Note that in all figures throughout this section the coordinate system is shown as follows: red/green coordinate axes denote the (x, y)-domain, the blue axis shows the time component.

Now the problem of finding a stream line and path line oriented topology is simply reduced to finding the topological skeletons of $\mathbf{s}$ and $\mathbf{p}$ respectively. Unfortunately, neither for $\mathbf{s}$ nor for $\mathbf{p}$ the classical vector field topology extraction techniques for 3D vector fields are applicable: $\mathbf{s}$ consists of critical lines while $\mathbf{p}$ does not have any critical points at all.

4.1 Stream line oriented topology

Stream line oriented topology is well-understood in the visualization community ([14], [1], [3]). In addition to track the topological features over time, bifurcations have to be extracted. Bifurcations are the events of structural changes of the flow behavior at a certain time. We can distinguish between local and global bifurcations depending on whether a local or a global analysis is necessary to extract them.

[8] For the actual integration one may use a fourth-order Runge-Kutta method.

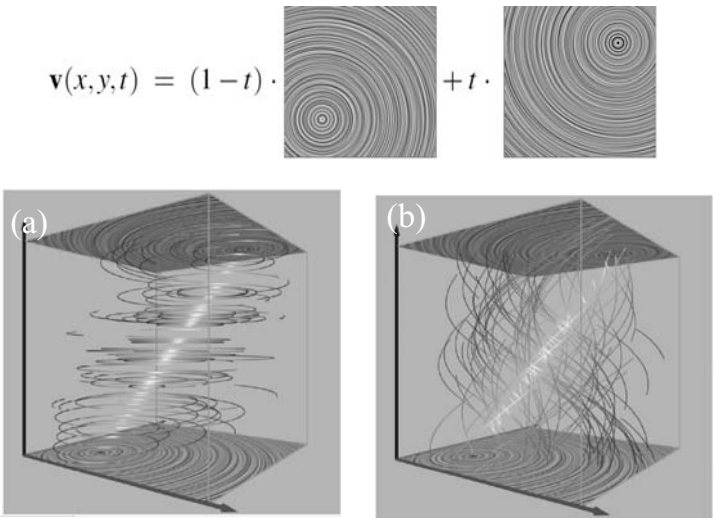


Fig. 15. Characteristic curves of a simple 2D time-dependent vector field shown as illuminated field lines: Stream lines of $\mathbf{s}$ correspond to the stream lines in $\mathbf{v}$; (b) stream lines of $\mathbf{p}$ correspond to the path lines in $\mathbf{v}$.

Tracking critical points

Critical points are important topological features of steady vector fields. Tracking their location over time is necessary for capturing the topological behavior of $\mathbf{v}$. This is equivalent to extracting the zero lines of $\mathbf{s}$, where all points on these lines are zero points of $\mathbf{v}$ at a certain time. To do so, one can either extract and connect the zeros on the faces of an underlying prism cell grid ([41]), or a feature flow field integration from a start zero point of $\mathbf{s}$ is applied. The feature flow field for tracking critical points is a 3D vector field $\mathbf{f}$ which points into the direction where all components of $\mathbf{s}$ remain unchanged. [34] shows that

$$\mathbf{f}(x,y,t) = \begin{pmatrix} \det(\mathbf{v}_y, \mathbf{v}_t) \\ \det(\mathbf{v}_t, \mathbf{v}_x) \\ \det(\mathbf{v}_x, \mathbf{v}_y) \end{pmatrix}. \tag{11}$$

Starting a stream line integration of $\mathbf{f}$ from a point $\mathbf{x}_0$ with $\mathbf{s}(\mathbf{x}_0) = (0,0,0)^T$, all points $\mathbf{x}$ on this stream line fulfill $\mathbf{s}(\mathbf{x}) = (0,0,0)^T$ as well.

To extract all critical lines of $\mathbf{s}$, an appropriate number of starting points is needed. We get them by considering all critical points at the boundaries of the domain of $\mathbf{s}$ (which can easily be obtained as critical points of 2D vector fields) and by additionally considering all *fold bifurcations* of $\mathbf{v}$. A fold bifurcation appears if at a certain time t a critical point appears, and in the same moment splits up to a saddle and source/sink/center.[9] Fold bifurcations can be found as the zeros of the following system of equations: $[u = 0,\ v = 0,\ \det(\mathbf{v}_x, \mathbf{v}_y) = 0]$ which can be solved numerically.

Another important class of local bifurcations are *Hopf bifurcations* denoting locations where a sink becomes a source or vice versa. Thus, this denotes the location of a center, i.e.

[9] Or the other way around: a saddle and a source/sink/center collapse and disappear.

a critical point with a vanishing divergence and a positive Jacobian. Hopf bifurcations can be extracted similar to fold bifurcations by numerically solving the system $[u = 0,\ v = 0,\ \mathrm{div}(\mathbf{v}) = u_x + v_y = 0]$ for (x, y, t) and selecting all isolated solutions with positive Jacobian.

Another part of the topological skeleton of $\mathbf{v}$ are the separation curves starting from saddle points. It is a well-known fact that a saddle of a 2D vector field creates 4 separation curves by starting the integration into the directions of the eigenvectors of the Jacobian matrix. While the saddle moves over time in $\mathbf{v}$, their swept surfaces form 4 stream surfaces dividing $\mathbf{s}$ into areas of different flow behavior. Figure 16(a),(b) gives an illustration of a simple vector field containing all topological feature mentioned above. In this figure (as well as in the following

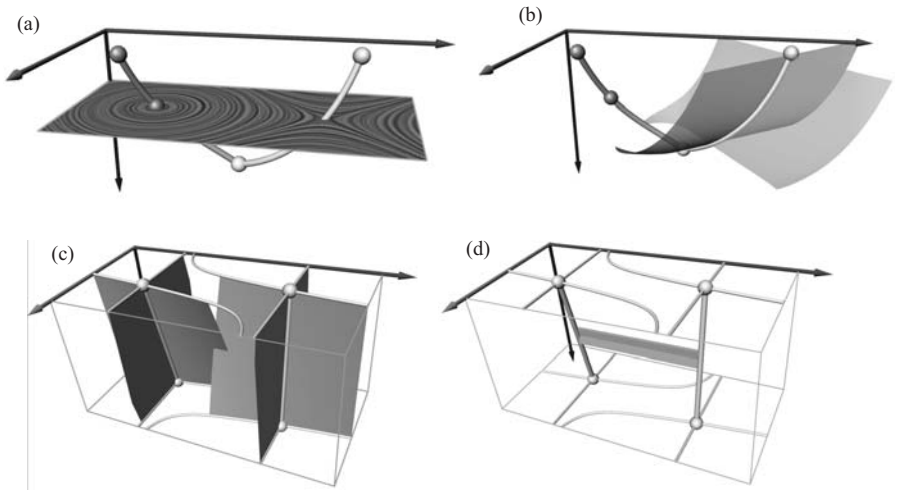


Fig. 16. (a),(b) topological visualization of a simple 2D time-dependent vector field consisting of sink, source, saddle, fold and Hopf bifurcation - one of each type: (a) critical lines of $\mathbf{s}$, LIC plane through Hopf bifurcation; (b) separation surfaces created by the moving saddle. (c),(d) Extracting saddle connections: (c) separation surfaces starting from critical lines of $\mathbf{s}$; (d) saddle connection as the intersection of these surfaces (from [38]).

figures) we use the following color coding: the critical lines of $\mathbf{s}$ are color coded according to the inflow/outflow behavior of the represented critical points in $\mathbf{v}$: a red/blue/green/yellow line segment represents a source/sink/center/saddle critical point respectively. The same color coding is used for particular critical points which are visualized as small spheres. This means that a Hopf bifurcation is shown as a small green sphere. Furthermore, fold bifurcations are shown as gray sphere, while particular stream lines of $\mathbf{s}$ are shown as gray lines. For integrated separation surfaces we color code according to the integration direction as red (forward integration) or blue (backward integration) surfaces.

Saddle connections

Saddle connections are global bifurcations which appear when two separatrices starting from saddle points collapse, i.e. when a separatrix of one saddle ends in another saddle. To extract

them, we modify the idea of saddle connectors of 3D vector fields [36]: instead of starting the integration of one separation surface at each saddle of a 3D vector field, we start in the critical lines of $\mathbf{s}$ which represent a moving saddle. In fact, we start four stream surface integrations[10] at the critical lines of $\mathbf{s}$ into the directions of the eigenvectors of the Jacobian matrix. The rest of the algorithm is similar to saddle connectors [36] and yields all saddle connections in $\mathbf{v}$. Figure 16(c),(d) give an illustration.

A special case of saddle connections is the so-called *periodic blue sky bifurcation* ([1]) where two separatrices of the same saddle collapse. The algorithm described above to extract saddle connections automatically extracts these bifurcations as well. Figure 17 illustrates this.

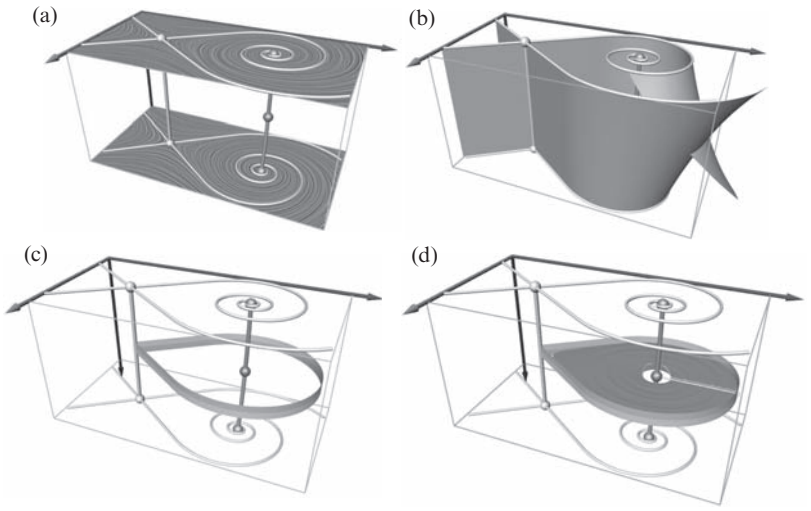


Fig. 17. Periodic blue sky bifurcation: (a) critical lines of s and two LIC planes; (b) separation surfaces shortly after their intersection; (c) two separation curves of the same saddle collapse; (d) tracked closed stream line starting from Hopf bifurcation. (from [38])

Tracking closed stream lines

Closed stream lines are global topological features which evolve over time in $\mathbf{v}$. Doing so, several bifurcations can occur: a closed stream line may appear or disappear, or two closed stream lines may collapse and disappear. The last case is called *cyclic fold bifurcation*.

To track isolated closed stream lines, an extraction in different time slices and subsequent linking was demonstrated in [47]. [38] presents a solution based on feature flow fields which works on space-time and can detect cyclic fold bifurcations as well. Figure 17(d) given an illustration.

[10] Two forwards and two backwards.

4.2 Path line oriented topology

Considering a path line oriented topology for visualization purposes is a relatively new research area. Constructing a path line oriented topology means to consider the stream lines of $\mathbf{p}$ and segment $\mathbf{p}$ into regions of different flow behavior of them. [38] introduces an approach which does the segmentation based on local path line properties. This way the domain segmented into areas where the path lines have attracting, repelling, or saddle like behavior.

4.3 An Example:

Figure 18 shows the visualization of a vector field describing the flow over a 2D cavity[11]. 1000 time steps have been simulated using the compressible Navier- Stokes equations; it exhibits a non-zero divergence inside the cavity, while outside the cavity the flow tends to have a quasi-divergence-free behavior. See also Figure CP-6 in Appendix G. Figure 18 shows approximately one period, 100 time steps, of the full data set. Figures 18 both reveal the overall movement of the topological structures, the most dominating ones originating in or near the boundaries of the cavity itself. The quasi-divergence-free behavior outside the cavity is confirmed by the fact that a high number of Hopf bifurcations has been found in this area.

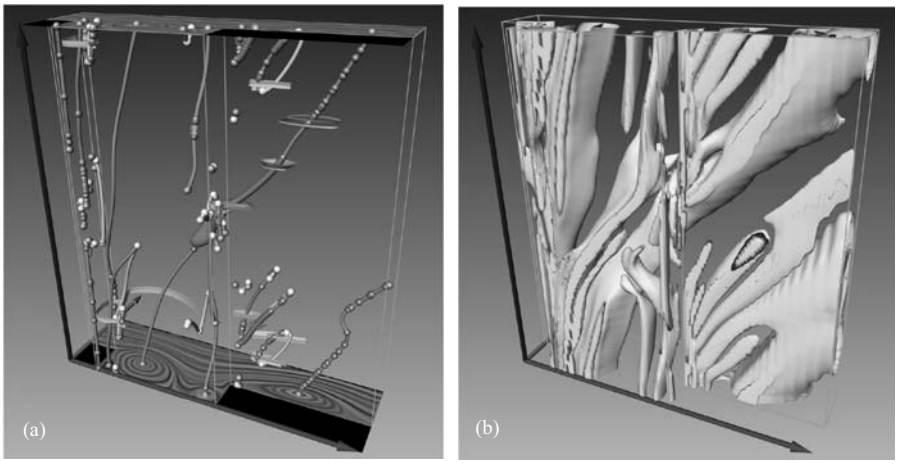


Fig. 18. 2D time-dependent flow at a cavity: (a) stream line oriented topology of the first 100 time steps; (b) path line oriented topology of the first 100 time steps (from [38]).

5 Further applications of topological features

Topological features of vector fields have not only proved to be a valuable visualization tool, they can also be used for other task in processing vector fields. Here we introduce approaches to compress, simplify, compare, and construct vector fields based on topological methods.

[11] This data set was kindly provided by Mo Samimy and Edgar Caraballo (both Ohio State University) [30] as well as Bernd R. Noack and Ivanka Pelivan (both TU Berlin).

5.1 Compressing vector fields

Flow data sets (i.e., vector fields) tend to be large and complex. This fact has motivated an intensive research in simplifying and compressing vector fields. For both challenges, topological concepts have been applied. Compression techniques for vector fields are motivated by the necessity of transmitting large flow data sets over networks with low bandwidth, or by the goal to produce visualizations of the data in low-end machines with a small main memory. For these cases the consideration of compressed vector fields makes the process of visual analysis of the flow data more efficient and is sometimes the only way to process the data in reasonable time rates at all.

The main idea of a (lossy) data compression is to reduce the amount of data while keeping the important structures. Since generally the topological skeleton is known to give a compact description of the global flow behavior, topology preserving compression techniques are an obvious approach. Lodha et al. [18, 17] introduce a compression technique for 2D vector fields which prohibits strong changes of location and Jacobian matrix of the critical points.

Theisel et al. [32] introduce an approach which guarantees that the topology of original and compressed vector field coincides both for critical points and for the connectivity of the separatrices. It is shown that even under these strong conditions high compression ratios for vector fields with complex topologies are achieved. The method works on a piecewise linear vector field over a triangulation. The vector field is interpreted as a piecewise triangular mesh. Then a standard mesh reduction algorithm can be adapted to this specific problem, i.e. the compression is achieved by iteratively applying half-edge collapses. Before a half-edge collapse is carried out, it is checked that it does not change the global topology of the vector field. As the theoretical foundation of the algorithm in [32], it is shown that for local modifications of a vector field, it is possible to decide entirely by a local analysis whether or not the global topology is preserved.

Figure 19 shows the application of the compression algorithm to a data set of a complex topology. Figure 19(a) shows the underlying triangular grid of the data set consisting of 12,726 triangles. Figure 19(c) shows the topological skeleton consisting of 338 critical points, 34 boundary switch points, and 714 separatrices. Figure 19(b) shows the underlying triangular grid after applying the compression algorithm. This grid contains of 2,153 triangles. The topological skeleton of the compressed vector field is shown in figure 19(d).

5.2 Topological simplification of vector fields

The topological skeleton of a vector field may be very complex due to the presence of noise. In this case, unimportant topological features have to be removed. This is done by a topological simplification. The simplest way to do so is to apply a smoothing of the vector field before extracting the topology ([6]). More involved techniques start with the original topological skeleton and repeatedly apply local modifications of the skeleton and/or the underlying vector field in order to remove unimportant critical points. They are based on the index theorem for vector fields which ensures that the sum of the indices of the critical points remains constant in the modified area. (See [7] or another textbook on vector analysis for an introduction of the index of critical points and the index theorem.)

De Leeuw and van Liere [5] denote the importance of a critical point (source or sink) by computing the area from which the flow ends in forward or backward integration. Based on this area metric, the unimportant critical points are repeatedly collapsed to more important critical points in the neighborhood. [6] finds couples of first order critical points by considering distance and connectivity of them. Then the unimportant critical points are pairwise removed.

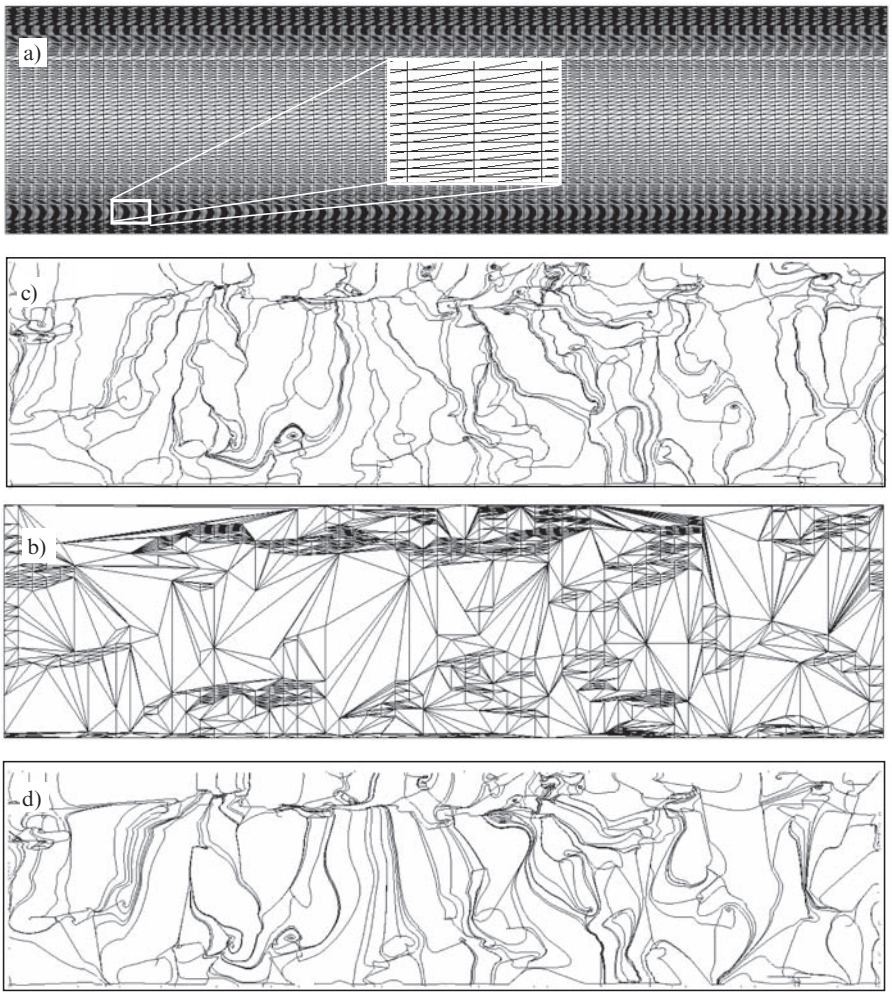


Fig. 19. (a) piecewise triangular domain of the original data set; (b) piecewise triangular domain of the compressed data set; (c) topological skeleton of original data set; (d) topological skeleton of compressed data set (from [32]).

Tricoche et al. [40] use a similar approach but provide a way of consistently updating the underlying vector field.

Theisel et al. [31] solve the coupling problem of critical points by a feature flow field approach. This gives not only the couples of critical points but also provides them and the separatrices with an importance weight. Then topological features with an importance below a certain threshold can be removed. Figure 20 gives an illustration.

Tricoche et al. [39] present another approach to simplifying the topology of 2D vector fields by replacing clusters of first order critical points with a higher order critical point. Weinkauf et al. [45] extend this to 3D vector fields. Figure 21 illustrates this.

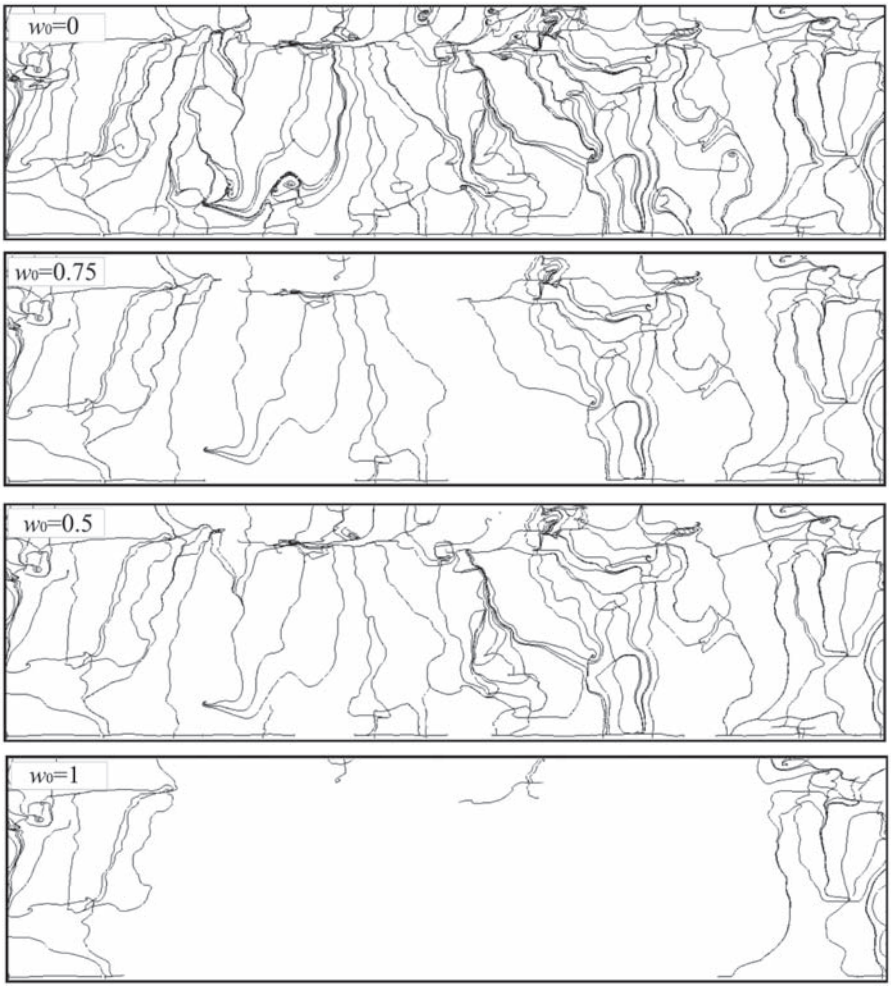


Fig. 20. Important topological features for different thresholds w_0; the image upper left ($w_0 = 0$) shows the complete topological skeleton. (from [31]).

5.3 Topological comparison of vector fields

To deal with the increasing size and complexity of the vector fields, a number of reconstruction, compression and simplification techniques have been introduced. All these techniques rely on certain distance measures between vector fields: the original and the derived vector field have to be compared to guarantee a sufficient similarity between them. Hence the definition of useful metrics on vector fields plays a crucial role in the applications above. The first approaches on metrics (distance measures) of vector fields consider local deviations of direction and magnitude of the flow vectors in a certain number of sample points ([12], [29]). These distance functions give a fast comparison of the vector field but do not take any structural information of the vector fields into consideration.

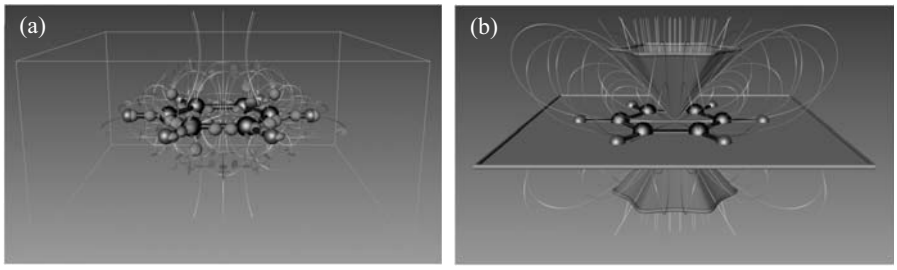


Fig. 21. Topological representations of the electrostatic field of the Benzene molecule: (a) 184 first order critical points. The box around the molecule represents the chosen area for topological simplification. (b) Topologically simplified representation with one higher order critical point elucidates the far field behavior of the benzene (from [45]).

A first approach to define a topology based distance function was given in [16]. Given two vector fields $\mathbf{v}_1$ and $\mathbf{v}_2$, all critical points are extracted and coupled. Then the distance of the vector fields is obtained as sum of the distances of the corresponding critical points in $\mathbf{v}_1$ and $\mathbf{v}_2$. To compute the distance between two critical points, a number approaches exist [16, 35]. To couple the points, [33] proposes to use feature flow fields: a time-dependent vector field $\mathbf{v} = (1-t)\,\mathbf{v}_1 + t\,\mathbf{v}_2$ is constructed in which the critical points are tracked by a stream line integration of (11).

We demonstrate the application of topological comparison on a real data set. Figure 22(a) shows the visualization of a 2D flow in a bay area of the Baltic Sea near Greifswald, Germany (Greifswalder Bodden) at two different time steps[12]. The data set can be considered as a collection of 25 vector fields $\mathbf{v}_0, ..., \mathbf{v}_{24}$. To evaluate the temporal behavior of the topology, the topological distance of each time step with all other time steps is computed. As an example, figure 22(a) illustrates the computation of the distance of the vector fields $\mathbf{v}_5$ and $\mathbf{v}_{10}$. Shown are the topological skeletons of $\mathbf{v}_5$ and $\mathbf{v}_{10}$ as well as the integration of the stream lines of the feature flow field starting in the critical point. We can see that most of the points find their partners in the other vector field. Figures 22(b),(c) show magnifications of figure 22(a). Figure 22(d) shows the color coded distance matrix of all vector fields $\mathbf{v}_0, ..., \mathbf{v}_{24}$. The distance varied between 0 and a maximal value of 104.5 (which was detected between $\mathbf{v}_3$ and $\mathbf{v}_{24}$). The distance was linearly color coded in such a way that a zero distance corresponds to black while the maximal distance corresponds to white. Figure 22(d) shows that the distance matrix is symmetric and with a zero main diagonal. The most important observation which can be made from figure 22(d) is that the distance of two vector fields $\mathbf{v}_i$ and $\mathbf{v}_j$ is approximately proportional to the distance $\|i-j\|$ of the time indices. This means that the rate of change of the topology is approximately linear over time. This result is particularly interesting if the number of critical points in the vector fields $\mathbf{v}_0, ..., \mathbf{v}_{24}$ is considered. They are (in this order) 65, 71, 71, 68, 65, 71, 63, 62, 66, 64, 65, 63, 70, 70, 51, 61, 52, 50, 56, 52, 63, 62, 72, 65. This shows that there is no correlation between the number of critical points and the topological distance: both $\mathbf{v}_0$ and $\mathbf{v}_{24}$ have the same number 65 of critical points but a maximal topological distance.

[12] This data set was obtained by a numerical simulation on a regular 115×103 grid at 25 time steps. It was created by the Department of Mathematics, University of Rostock (Germany).

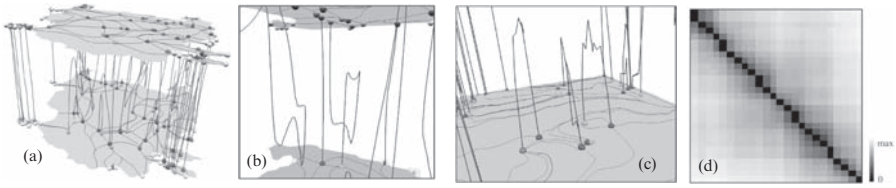


Fig. 22. (a) coupling the critical points of the $\mathbf{v}_5$ and $\mathbf{v}_{10}$ of the bay data set by integrating the stream lines of $\mathbf{f}$; (b),(c): magnifications of (a); (d) distance matrix between $\mathbf{v}_0, ..., \mathbf{v}_{24}$ (from [33]).

5.4 Constructing vector fields

The vector fields considered in flow visualization are usually obtained by a simulation or measurement process. Nevertheless they can also be obtained by construction. Applications of this approach are vector fields used for pattern matching, optimizing flow, education and testing new visualization techniques.

The approach of constructing vector fields is strongly related to the ideas of constructing curves and surfaces in the context of CAGD (Computer Aided Geometric Design). There the curves/surfaces are designed by creating a skeleton of control polygons (for instance Bezier- or B-spline polygons). This skeleton contains the relevant information of the curve/surface in an intuitive way. [30] presents an approach to transform the CAGD methods to the construction of 2D vector fields. To do so, first the topological skeleton of a vector field is constructed by a number of control polygons. As a second step, a piecewise linear vector field of exactly the specified topology is automatically created. Figure 23 gives an example.

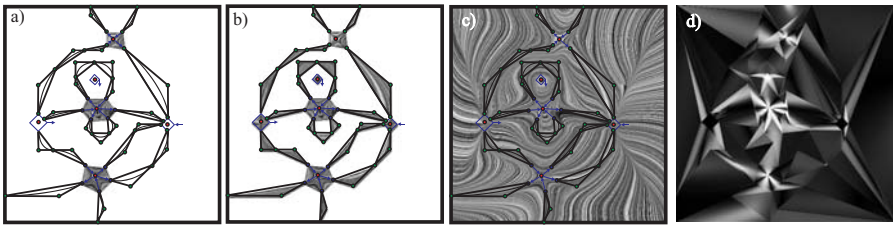


Fig. 23. Constructing a 2D vector field; a) topological skeleton of a vector field containing a number of higher order critical points; b) piecewise linear vector field describing the constructed topological features, i.e. the critical points and separatrices; c) complete piecewise linear vector field; d) curvature plot (from [30]).

An approach to constructing 3D vector fields is presented in [44]. There, the skeleton is modeled by interactively moving a number of control polygons determining location and characterization of the (first or higher order) critical points and the saddle connectors. Then a piecewise linear vector field is automatically constructed which has the same topological skeleton as modeled before. This approach is based on a complete segmentation of the areas around critical points into sectors of different flow behavior. Based on this, an approach to visualizing higher order critical points of 3D vector fields is presented.

Figure 24(a) shows a modeled topological skeleton consisting of 6 critical points and 8 connectors. Each of the critical points consists of two hyperbolic sectors and is actually a first order saddle point. Each of the connectors was defined by specifying start and end point and omitting any intermediate points. Thus, each connector consists of one cubic segment. Figure 24(b) shows the result of the tetrahedrization of the critical points and the connectors. In this figure we can clearly see that each connector is constructed in one tetrahedron. Figure 24(c) shows the complete tetrahedrization of the piecewise linear vector field consisting of 256 tetrahedra. Figures 24(d) and 24(e) show different visualizations of the newly constructed vector field. Figure 24(d) shows a stream surface integration of the separation surfaces. They are color coded in red (outflow surface) and blue (inflow surface). Figure 24(e) shows the extraction of saddle connectors [36] revealing that they coincide with the modeled connectors of figure 24(a). In addition, figure 24(e) shows a number of illuminated stream lines [49]. See also Figure CP-7 in Appendix G.

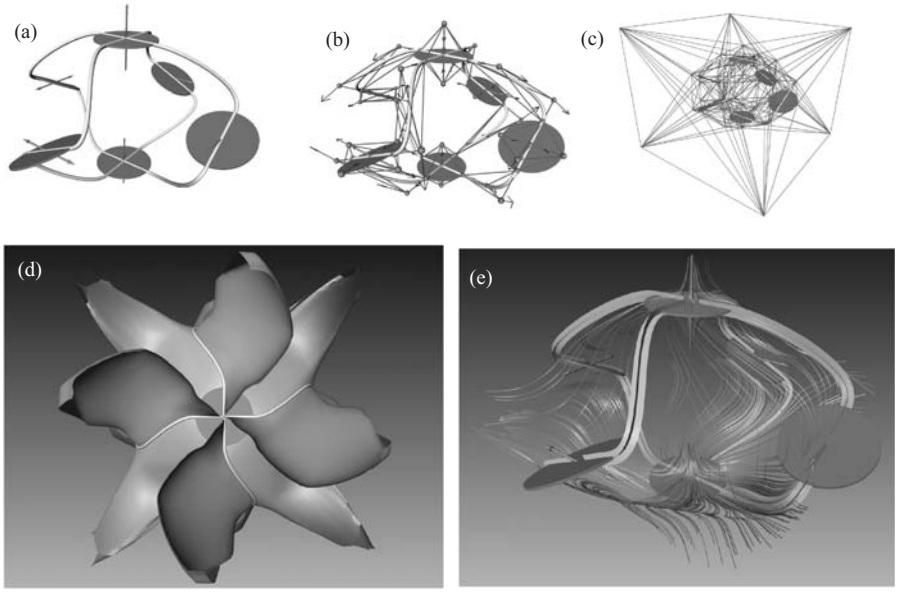


Fig. 24. Constructed 3D vector field: (a) Modeled topological skeleton; (b) Tetrahedrization of critical points and connectors; (c) Complete tetrahedrization; (d) Separation surfaces of constructed vector field, view from top; (e) Saddle connectors and stream lines of constructed vector field (from [44]).

6 Conclusions

In this chapter we have shown that topological methods provide a useful framework for the visual analysis of vector fields. However, there is a number of open problems which are still rather unsolved. Firstly, an appropriate visual representation of the topological skeleton of 3D

time-dependent vector fields is still a challenge. Secondly, the path line oriented topological representation of time-dependent vector fields remains an open problems. Because of this we expect an active ongoing research in the field in the next years.

References

1. L. Abraham and K. Shaw. *Dynamics, The Geometry of Behaviour*. Addison-Wesley, 1992.
2. D. Asimov. Notes on the topology of vector fields and flows. Technical report, NASA Ames Research Center, 1993. RNR-93-003.
3. P. G. Bakker. *Bifurcations in Flow Patterns (Theory and Applications of Transport in Porous Media)*. Kluwer Academic Publishers, 1991.
4. M. S. Chong, A. E. Perry, and B. J. Cantwell. A general classification of three-dimensional flow fields. *Physics of Fluids A*, 2(5):765–777, 1990.
5. W. de Leeuw and R. van Liere. Collapsing flow topology using area metrics. In *Proc. IEEE Visualization '99*, pages 149–354, 1999.
6. W. de Leeuw and R. van Liere. Visualization of global flow structures using multiple levels of topology. In *Data Visualization 1999. Proc. VisSym 99*, pages 45–52, 1999.
7. P.A. Firby and C.F. Gardiner. *Surface Topology*, chapter 7, pages 115–135. Ellis Horwood Ltd., 1982. Vector Fields on Surfaces.
8. C. Garth, X. Tricoche, and G. Scheuermann. Tracking of vector field singularities in unstructured 3D time-dependent datasets. In *Proc. IEEE Visualization 2004*, pages 329–336, 2004.
9. A. Globus, C. Levit, and T. Lasinski. A tool for visualizing the topology of three-dimensional vector fields. In *Proc. IEEE Visualization '91*, pages 33–40, 1991.
10. J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer, 2nd edition, 1986.
11. H. Hauser and E. Gröller. Thorough insights by enhanced visualization of flow topology. In *9th international symposium on flow visualization*, 2000.
12. B. Heckel, G.H. Weber, B. Hamann, and K.I.Joy. Construction of vector field hierarchies. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 19–26, Los Alamitos, 1999.
13. J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *IEEE Computer*, 22(8):27–36, 1989.
14. J. Helman and L. Hesselink. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, 11:36–46, May 1991.
15. J. Hultquist. Constructing stream surfaces in steady 3D vector fields. In *Proc. IEEE Visualization '92*, pages 171–177, 1992.
16. Y. Lavin, R.K. Batra, and L. Hesselink. Feature comparisons of vector fields using earth mover's distance. In *Proc. IEEE Visualization '98*, pages 103–109, 1998.
17. S. Lodha, N. Faaland, and J. Renteria. Topology preserving top-down compression of 2d vector fields using bintree and triangular quadtrees. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):433–442, 2003.
18. S.K. Lodha, J.C. Renteria, and K.M. Roskin. Topology preserving compression of 2D vector fields. In *Proc. IEEE Visualization 2000*, pages 343–350, 2000.
19. H. Löffelmann, H. Doleisch, and E. Gröller. Visualizing dynamical systems near critical points. In *Spring Conference on Computer Graphics and its Applications*, pages 175–184, Budmerice, Slovakia, 1998.

20. G. Mutschke, 2003. private communication.
21. G.M. Nielson. Tools for computing tangent curves and topological graphs for visualizing piecewise linearly varying vector fields over triangulated domains. In G.M. Nielson, H. Hagen, and H. Müller, editors, *Scientific Visualization*, pages 527–562. IEEE Computer Society, 1997.
22. P. A. Philippou and R. N. Strickland. Vector field analysis and synthesis using three dimensional phase portraits. *Graphical Models and Image Processing*, 59:446–462, November 1997.
23. K. Polthier and E. Preuss. Identifying vector fields singularities using a discrete hodge decomposition. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 135–150. Springer Verlag, Heidelberg, 2002.
24. F.H. Post, B. Vrolijk, H. Hauser, R.S. Laramee, and H. Doleisch. Feature extraction and visualisation of flow fields. In *Proc. Eurographics 2002, State of the Art Reports*, pages 69–100, 2002.
25. G. Scheuermann, T. Bobach, H. Hagen K. Mahrous, B. Hamann, K. Joy, and W. Kollmann. A tetrahedra-based stream surface algorithm. In *Proc. IEEE Visualization 01*, pages 151 – 158, 2001.
26. G. Scheuermann, H. Krüger, M. Menzel, and A. Rockwood. Visualizing non-linear vector field topology. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):109–116, 1998.
27. H. Schumann and W. Müller. *Visualisierung - Grundlagen und allgemeine Methoden*. Springer-Verlag, 2000. (in German).
28. D. Stalling and T. Steinke. Visualization of vector fields in quantum chemistry. Technical report, ZIB Preprint SC-96-01, 1996. ftp://ftp.zib.de/pub/zib-publications/reports/SC-96-01.ps.
29. A. Telea and J.J. van Wijk. Simplified representation of vector fields. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 35–42, Los Alamitos, 1999.
30. H. Theisel. Designing 2D vector fields of arbitrary topology. *Computer Graphics Forum (Eurographics 2002)*, 21(3):595–604, 2002.
31. H. Theisel, Ch. Rössl, and H.-P. Seidel. Combining topological simplification and topology preserving compression for 2d vector fields. In *Proc. Pacific Graphics*, pages 419 – 423, 2003.
32. H. Theisel, Ch. Rössl, and H.-P. Seidel. Compression of 2D vector fields under guaranteed topology preservation. *Computer Graphics Forum (Eurographics 2003)*, 22(3):333–342, 2003.
33. H. Theisel, Ch. Rössl, and H.-P. Seidel. Using feature flow fields for topological comparison of vector fields. In *Proc. Vision, Modeling and Visualization 2003*, pages 521 – 528, Berlin, 2003. Aka.
34. H. Theisel and H.-P. Seidel. Feature flow fields. In *Data Visualization 2003. Proc. VisSym 03*, pages 141–148, 2003.
35. H. Theisel and T. Weinkauf. Vector field metrics based on distance measures of first order critical points. In *Journal of WSCG*, volume 10:3, pages 121–128, 2002.
36. H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Saddle connectors - an approach to visualizing the topological skeleton of complex 3D vector fields. In *Proc. IEEE Visualization 2003*, pages 225–232, 2003.
37. H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Grid-independent detection of closed stream lines in 2D vector fields. In *Proc. Vision, Modeling and Visualization 2004*, 2004.

38. H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Stream line and path line oriented topology for 2D time-dependent vector fields. In *Proc. IEEE Visualization 2004*, pages 321–328, 2004.
39. X. Tricoche, G. Scheuermann, and H. Hagen. A topology simplification method for 2D vector fields. In *Proc. IEEE Visualization 2000*, pages 359–366, 2000.
40. X. Tricoche, G. Scheuermann, and H. Hagen. Continuous topology simplification of planar vector fields. In *Proc. Visualization 01*, pages 159 – 166, 2001.
41. X. Tricoche, G. Scheuermann, and H. Hagen. Topology-based visualization of time-dependent 2D vector fields. In *Data Visualization 2001. Proc. VisSym 01*, pages 117–126, 2001.
42. T. Weinkauf. Krümmungsvisualisierung für 3D-Vektorfelder. Diplomarbeit, University of Rostock, Computer Science Department, 2000. (in German).
43. T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Boundary switch connectors for topological visualization of complex 3D vector fields. In *Proc. VisSym 04*, pages 183–192, 2004.
44. T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Topological construction and visualization of higher order 3D vector fields. *Computer Graphics Forum (Eurographics 2004)*, 23(3):469–478, 2004.
45. T. Weinkauf, H. Theisel, K. Shi, H.-C. Hege, and H.-P. Seidel. Extracting higher order critical points and topological simplification of 3D vector fields. In *Proc. IEEE Visualization 2005*, pages 95–102, 2005.
46. T. Wischgoll and G. Scheuermann. Detection and visualization of closed streamlines in planar flows. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):165–172, 2001.
47. T. Wischgoll, G. Scheuermann, and H. Hagen. Tracking closed stream lines in time-dependent planar flows. In *Proc. Vision, Modeling and Visualization 2001*, pages 447–454, 2001.
48. H.-Q. Zhang, U. Fey, B.R. Noack, M. König, and H. Eckelmann. On the transition of the cylinder wake. *Phys. Fluids*, 7(4):779–795, 1995.
49. M. Zöckler, D. Stalling, and H.C. Hege. Interactive visualization of 3D-vector fields using illuminated stream lines. In *Proc. IEEE Visualization '96*, pages 107–113, 1996.

Control Structure and Multi-Resolution Techniques for Virtual Human Representation

Thomas Di Giacomo[1], HyungSeok Kim[2], Laurent Moccozet[3], and Nadia Magnenat-Thalmann[4]

[1] MIRAlab, University of Geneva, giacomo@miralab.unige.ch
[2] MIRAlab, University of Geneva, hyung.kim@acm.org
[3] MIRAlab, University of Geneva, moccozet@miralab.unige.ch
[4] MIRAlab, University of Geneva, thalmann@miralab.unige.ch

Summary. A virtual human is a typical instance of articulated physical objects: it does not have only one shape but many, corresponding to all the possible postures that the underlying articulated skeleton can reach. For realistic rendering results, a high-quality texture is usually associated to the shape and skeleton structure. Controlling and animating a virtual human model requires simultaneously many graphics and computational resources.

The first part of this chapter discusses the control articulated skeleton structure and different approaches to build skeletons [10] and bind it to the geometry. The second part addresses the production of LoDs for virtual humans, both for the 3D shape (geometry) and the articulated skeleton (motion and animation).

1 Definitions and Background

A virtual human is a subset of a larger family that can be defined as articulated deformable characters. To mimic the flexible and dynamic behaviour of the human body shape, the traditional approach uses skeleton-driven deformations; a classical method for basic skin deformation that is the most widely used technique in 3D character animation. It binds a 3D shape representing the body shape to an articulated control skeleton. Binding information is then used to deform the body shape according to the skeleton motion. The control skeleton is the main structuring information for articulated deformable characters. Constructing such a character requires assembling a static 3D shape with a control skeleton so that skeleton-driven deformations accurately reproduce the body shape in any possible skeleton posture.

Textured virtual humans are used in a wide range of applications, from movies to 3D games, and platforms, from high-end graphics workstation to PDA. Adapting a virtual human model to the application and platform context therefore requires defining Level of Details (LoDs) without penalising the Degrees of Freedom (DoF) and visual results. To that end, the production of LoDs must consider the virtual human as a whole including the 3D body shape, the articulated skeleton and if necessary the texture, which implies to consider at the same time geometric and animation LoDs. Moreover, geometry simplification must not alter the DoF of the underlying skeleton and the visual quality of the associated texture.

1.1 Control skeleton definition

A control skeleton is a connected set of segments, corresponding to limbs, and joints. A joint is the intersection of two segments, which means it is a skeleton point where the limb that is linked to this point may move. The angle between two segments connected by a joint is called the joint angle.

A joint may have at most three kinds of position angles: flexion, pivot and twisting. The flexion is a rotation of the limb influenced by the joint, and which causes the motion of all limbs linked to this joint. This flexion is carried out relatively to the joint point and a flexion axis has to be defined. The pivot makes the flexion axis rotate around the limb influenced by the joint. Twisting causes a torsion of the limb influenced by the joint. The direction of the twisting axis is found similarly to the direction of the pivot. Static posture and motions can be produced by modifying the joint angles of the skeleton. Various approaches are available to control the skeleton posture and produce new postures or motions [23]. Control skeletons are particularly applied to simulate and control anthropomorphic characters where the segments of the control skeleton correspond to the limbs of the body. Control skeleton are usually used to drive deformation of an attached geometric skin surface. The issues to solve are organized into two main categories: producing realistic and believable motion of the skeleton and adjusting the geometric skin in order to constrain it to "follow" the motion of the skeleton. An articulated skeleton can be seen as an extension to the traditional transformation hierarchy where a single rotational Degree of Freedom (DoF), *i.e.* flexion, twist or pivot, is attached to each node.

The main structure of transformation hierarchy in computer graphics is a tree data structure. It is described as a n-ary tree of nodes with an homogeneous transform at each node. The transformation represents the node's local transformation with respect to its parent. Each node can store its local information in its own local coordinate system. In addition to the joint angle value, each joint data structure requires maintaining information about the joint limits and a default or rest position. Joint limits are important to prevent the skeleton from achieving unrealistic postures.

A standard way of representing virtual humans has been defined in VRML/X3D: H-ANIM [1]. No assumptions are made about the types of applications that will use virtual humans. An H-ANIM model is made of different types of nodes. The main ones are the Joint and Segment nodes. Joint nodes represent skeleton joints and are arranged to form a hierarchy similar to the ones described previously. Each Joint node can contain other Joint nodes and may also contain a Segment node. A Segment node describes the body part associated with a joint.

Various approaches have been proposed to control articulated skeletons and generate postures and motions. These approaches are reviewed in Section 2 and an overview of motion control methods for articulated figures is also proposed in [49].

1.2 Historical background

Historically, the first characters animated with a skeleton were only represented using stick figures, *i.e.* a graphical representation of the control skeleton structure, such as for example "'Hipi"', the hero of the film "'Dream Flight"' [97]. Rapidly, volume and surface figures have then been proposed. The volume figure representation is an approach that consists in decomposing the body into several rigid primitive volumes such as cylinders or ellipsoids. These primitives are matched to the limbs of the control skeleton, and they are also used for the visualization of the character.

The NUDES (Numerical Utility Displaying Ellipsoid Solids) illustrates this kind of animation system. This approach has been surpassed by surface figures [40]. They are still interesting as many articulated character animation systems are based on a volume approach to produce body motion and deformation. In these approaches, the volume primitives are parameterized according to the joint angles. Once they are adjusted to match the skeleton posture, they are used to produce a continuous skin surface. A usual approach is based on fusing ellipsoids together as metaballs [98] in order to produce a single continuous skin surface. In the surface figures approach, skeletons are surrounded by surfaces. In these approaches, the main issue consists in accurately mapping the skin surface to the control skeleton. Early examples are the human skin model proposed by Komatsu in [51] and the Joint-dependent Local Deformation (JLD) operators, introduced by Magnenat-Thalmann in [66] to smoothly deform the skin surface. This latter approach has been applied in the film Rendez-vous in Montreal.

Skeleton techniques for animation have a long history. They have initially been proposed for 2D computer animation as a tool to simplify and automate the production of in-between images from 2D key-frames. Although the methods to map the skin to the skeleton have progressed, the basic approach is still the same. The idea between the skeleton technique [19] is that skeletons of 2D figures, rather than the figures themselves, can be used as a basis for inbetweening. A skeleton is defined as a network of 4-sided polygons. Relative coordinates can then be associated to each vertex. Once the skeleton has been assigned a new posture, the figure can be reconstructed using the relative coordinates of the figure vertices with respect to the skeleton. The film "'The Hunger"' has been produced with this technique in 1974.

A survey of the most recent approaches for handling skeleton-driven deformations is proposed in the next Section, and a review of common methods for the construction and deformation of articulated character models is also proposed in [28].

2 Skeleton control methods

In real life, human motions are basically activated by muscles, *i.e.* rigid and inert bones are moving by contractions of muscles. The general approach adopted in computer graphics, coming first from robotics and extended over the years, is to define an underlying articulated structure which joints control the displacements. These articulated structures, or skeletons, are topologically defined by a set of bones, or limbs, and a set of joints, and can be mathematically expressed by directed acyclic graphs.

Generally, skeleton-based animation consists in computing the variation of positions and orientations of the joints to drive the motion of bones and of the attached geometric representations, independently from their topology and complexity. One of the main advantages is the possibility to animate complex representations with high resolution meshes using a simple structure for motion computations, the skeleton. Bones, with constant lengths in the case of rigid bodies, are connected by joints, which have a certain DoF, to form a hierarchy. Furthermore, skeletons feature a reference point, named the root, that handles their global positioning, thus the global translations of the structure. The root node also provides an initial frame for the cascade of transformations.

Traditionally, two main approaches have coexisted to control skeletons, namely kinematics and dynamics. Kinematics are dealing with pre-recorded and stored geometrical parameters, and they have been extended by parameters processing with various interpolation techniques in keyframing methods. It is a fast approach, relatively easy to manage, but it lacks interactivity, genericity and automaticity. On the other hand, dynamics generates geometric

parameters to simulate motion according to forces and physical constraints. Though animations are automatically produced and are reactive to input, they might look un-natural, especially for human, and they are computationally expensive. Those two approaches are now detailed, as well as the intermediate kinetics and some combined methods.

2.1 Control Techniques

Kinematics methods animate a skeleton by dealing with linear or angular positions and velocities of the joints. The fewer DoF or number of joints, the less a kinematically animated skeleton costs in terms of transformation. A structure consists in a hierarchy of joints. Let $\theta = (\theta_0, ..., \theta_n)$, referred as the state-vector, or generalized coordinates, denotes the different DoF of the structure in the joint space, where n is the number of joints. Let $X = (X_0, ..., X_n)$ denotes the position in the Cartesian space, or world coordinates. θ_i are one to three dimensional, and both X_i and θ_i are time dependent for animation.

Forward, or direct, kinematics, usually expressed in the joint parameter space consists in fixing the different $\Delta\theta$ over time, to compute the ΔX. In other words, the orientations of all joints are specified by the animator, computed or pre-recorded and stored for runtime uses, and the motion of bones are processed as the results of previous transformations in the hierarchy. It is actually relatively simple to implement and gives direct control over the joints' DoF, though it is not really suitable to explicitly control the position of end-effectors, *i.e.* specific nodes corresponding to the leaves in the hierarchy. One of the main limitations of forward kinematics, apart from difficulties in control and user-interaction, is the prohibitive quantity of manual input required to produce animation. For instance, Kalra *et al.* [44] are using a 32 joints skeleton with 74 DoF, including 25 DoF for each hand, and complex human models can even consist of more than 100 DoF. Thus even by using keyframing, forward kinematics, where each joints parameters are user-defined, are not straightforwardly usable to animate complex skeletons.

With user-defined positions of the end-effectors as input, *e.g.* hands and feet in grasping or walking tasks, inverse kinematics compute automatically the previous joint angles in the hierarchy. It is equivalent to setting X_j, where $j \in E$ with E being the set of end-effectors, and then computing θ_i, where $i \notin E$. From Watt *et al.* [103], a general solution to inverse kinematics is $\Delta\theta = J^+\Delta X + \alpha(I - J^+J)\Delta z$, where J is the Jacobian matrix of the system of equations, J^+ its pseudo-inverse, α a penalty constant and Δz a constraint to be minimized, *e.g.* on one angle range. Δz is called the secondary task, since it is independent from the first term $J^+\Delta X$. For any unknown parameters, some bottlenecks arise due to the non-linearity of the systems to solve. Furthermore, these systems are possibly ill-conditioned. Some numerical optimizations, aiming at convergence to a single solution, and algebrical methods can be used to solve these systems [76]. To further accelerate solving, one can use heuristics or constrained angles, or can specify intermediate orientations, additional end-effectors, or also limit cones associated to bones. Actually, human body is a highly redundant system, and the limits of joints are both implicit due to biomechanical properties and explicit with selecting specific task or families of motion to achieve. Manocha *et al.* [67] propose fast and robust algorithms to solve systems for skeletons of six or fewer joints, *i.e.* robot-like structures. Recently, Tolani *et al.* [99] propose some analytical and numerical algorithms to compute real-time inverse kinematics for a human arm or leg. The sample 7 DoF chain consists of two spherical joints, one for the shoulder and one for the wrist, and a single revolute joint for the elbow. The combination of inverse and direct kinematics is explored by Boulic *et al.* [16]. They control joints by both techniques, and they apply the method to walking with half-spaces constraints

to correct the positions of end-effectors. To reduce the computations of the correction process, when meeting an enforced Cartesian constraint the articulated structure is duplicated into a coach and a trainee structures.

Kinetics are a combination of kinematics with the effect of a mass distribution concentrated on the centre of mass of the skeleton. Direct kinetics consist in computing the sequential positions of the centre of mass according to the configurations of the structure, while inverse kinetics consist in computing the skeleton configurations according to the positions of the centre of mass. At the same order of complexity, this approach extends previous control methods of skeleton by providing a physical behaviour to the animation through intuitive parameters such as mass. Furthermore, inverse kinetics resolve some additional constraints to inverse kinematics as stated by Boulic *et al.* [15], who investigate the design of static human postures. Their approach allows for single or multiple supports of the body mass distribution using a compatible flow model of the supporting influence. A mass fraction is associated to each support with the sum of all fractions being the total body mass. Zhao [110] also proposes a technique based on kinetics to control human postures for performing specific tasks. The postures are first simplified by manually decoupling the DoF of the human body, *e.g.* the upper and lower body movements, in order to reduce the configuration space.

Skeletons are also animatable with the use of dynamics properties such as mass, inertia, linear or angular acceleration. These dynamics-based methods have several benefits: external world influences such as gravity and collisions are more easily handled compared to pure kinematics, they allow motion interactions between objects, *etc.* As for kinematics, we can distinguish between direct and inverse dynamics. Direct dynamics compute new states, *i.e.* positions, velocities, accelerations, of the system when all external and internal forces are known. Inverse dynamics, computing forces according to new states, are very expensive. Additionally, similar artefacts as inverse kinematics can occur, such as un-natural motions if used globally on the skeletons. Direct dynamics compute and update motions thanks to forces and dynamics. The Newton laws are the fundamental principles of dynamics that bound force and acceleration, *i.e.* the second derivative of motions as follows: $\Sigma F = M\frac{d^2x}{dt^2}$, where F is a sum of forces applied on an object, M its mass and x its position. Though these equations hold for a single object, they have been validated for articulated figures, considered as hierarchies of rigid solids connected by articular joints. To effectively compute position updates, one must integrate accelerations over time. Due to the computational complexity of this task, various schemes have been proposed: explicit or implicit Euler methods, Verlet or Runge-Kutta integration schemes *etc.*, refer to Volino *et al.* [101], and many work are still dealing with this topic. To compute the update of accelerations, one must sum all external and internal forces applied to an object. Torque is an example of an external momentum due to forces occurring on angular joints. A torque τ is computed as the cross product between the bone segment r and the force applied on the segment F: $\overrightarrow{\tau} = \overrightarrow{r} \times \overrightarrow{F}$. For joints, the dynamics of rotation are: $\Sigma\tau_i = I\frac{d^2\theta}{dt^2} + \theta \times i\frac{d\theta}{dt}$, where I is the inertia matrix, θ the orientation. τ can then be summed to other forces or projected on the axis of the bone to update its position. For instance, Hodgins *et al.* [41] uses joints torque as an external force of a leg on a bicycle. Kokkevis *et al.* [50] combine kinematics and forward dynamics. Kinematics enable a direct and intuitive user-control on the main trajectories and goals, while dynamics compute all the other joints with physical correctness.

2.2 Space-time Constraints and Controllers

Introduced by Witkin and Kass [105], space-time constraints methods compute motions and time-varying forces over the whole animation sequence. The large vector of unknowns consisting of forces, velocities and positions, is computed iteratively through a constrained optimization process. The constraints are the Newton laws, the limit of the muscular forces, some intermediate positions and velocities, initial and final positions and velocities. Though extending and optimizing the use of dynamics, this method has a few limitations due to the high number of unknowns, the non-linearity of constraints and goals, and the a-priori specifications of constraints. To decrease the computational costs and to allow user interactions and system convergence, some improvements have been proposed such as the use of subsets of DoF, sub-intervals of time, hierarchical wavelets representations, *etc.* For instance, Brogan *et al.* [17] propose to use space-time constraints to generate biomechanical motions. Rose *et al.* [80] take advantage of the space-time constraints, combined with inverse kinematics on the root and supports of the body, by using them for transitions during motion blending between segments of human bodies with 44 DoF.

The finest and most efficient control on skeletons is a combination of the different methods previously discussed. Combining them is however not straightforward, since they are often based on different sets of parameters. With Newton laws and time integration, forces can lead to positions, therefore the definition of forces is a possible bound between the different methods. Controllers are basically processing forces for required goals, and in most work controllers are combined with kinematics for the description of desired-motions and direct dynamics for generating motions out of the controllers forces. Proportional-Derivative (PD) controllers are expressed as: $f = -k_p(q - q_d) - k_v(\frac{dq}{dt} - \frac{dq_d}{dt})$, where k_p and k_v are respectively the proportional and derivative gains, q is the current state of the system, mostly the positions, and q_d its desired state. PD controllers can be used as transitions between two adjacent states, each with an associated shape of the figure, of a finite state machine. As stated by Multon *et al.* [72], who survey different human walk models from purely kinematics technique to dynamic simulations, some approaches use finite state machines since they are particularly well-suited for cyclic motions. When controllers are tuned by hand, such as proposed by Hodgins *et al.* [41], the process is long and the resulting controllers are specific to a certain figure. On the other hand, when controllers are generated automatically, it is difficult to predict the resulting motions. Additionally to walking, controllers are widely investigated for general animations. Hodgins *et al.* [41] apply dynamics to bodies, of approximately 30 DoF, performing athletic behaviours. The algorithm is based on dynamics controlled by state machines. The mass and moment of inertia are computed for each part of the human body, allowing the simulated human to maintain balance and to add secondary motions. To efficiently manage dynamics, Brogan *et al.*[18] propose behavioural controllers to simulate a herd of one-legged robots and virtual cyclists who consist of 17 DoF including 4 DoF for the bicycle. A damped spring is linked between the hands of the cyclists and the handlebar, between the feet and the pedals for the interactions. Wooten *et al.* [106] present a work on smooth transition between dynamically simulated motions. As the authors state, most approaches are based on a library of motions in which a user can select a desired motion, then the current motion and the newly selected one are blended together by interpolation. Here, Wooten *et al.* [106] use parameterized basis controllers such as landing or leaping, where the parameters define the height of the jump. The controllers are designed so that all their exit states leave the simulation in a valid initial state for the next controllers. Though this enables smooth transitions, it also constraint them to occur at the end of a motion. Faloutsos *et al.* [32] compose controllers to simulate realistic motions of 37 DoF humans. This framework needs controllers to meet

manual or automatic pre-conditions, involving the initial state and the balance of the figure, some environmental parameters and a target state, to be selected as candidates.

3 Skeleton skinning and skin mapping

Various approaches have been proposed to connect a deformable skin to a control skeleton. They can be subdivided into two main categories: skin mapping and skeleton skinning. A third alternative approach, where the skeleton is directly built from the 3D shape to control, is described in Section 4. All approaches share the same multi-layers organization, with at least two layers: the skeleton and the skin. Intermediate structures are inserted in-between to tune and control more precisely the behaviour of the skin shape. Many models insert a third layer usually called musculature by analogy with anatomy.

3.1 Skeleton skinning

The typical example of this approach is proposed by Thalmann *et al.* [98] and [14]. The authors proposed to interactively attach ellipsoids to the control skeleton in order to define a simplified musculature. The main parameters of the ellipsoids are parameterized with joints angles. The shape of the ellipsoid muscles are controlled by the values of the joint angles of the skeleton. To produce the final skin for rendering and visualization, the ellipsoid muscles are fused together as metaballs. This approach has been investigated to simulate the behaviour of the real anatomy and reproduce as close as possible the anatomical behaviour of the body [84] (see Figure 1). It can be defined as anatomy-based modelling. The anatomy based modelling approach considers that simply using flexible surfaces at joints and along the limbs is an oversimplification producing poor results. It follows the same approach as the one applied in artistic anatomy and assume that the skin surface shape changes may be more accurately represented as the result of the behaviour of the underlying anatomical structures such as muscles, fat and connective tissues.

Turner *et al.* [100] propose a more sophisticated multi-layers structure, where the skin is modelled as an elastic surface, muscles as ellipsoids, connective tissues as springs connecting the skin surface to the muscles.

Complex volumes are used to approximate the real shape of muscles with tendons' attachments to the bones. Various approaches have been investigated:

1. Passive muscles: in this category, geometric [104] or physics-based deformations are used to adjust the shape of the muscle to the current posture of the skeleton.
2. Active muscles: in this category, biomechanical physics-based deformation models [26] are used to simulate the dynamic behaviour of muscles. Muscles contraction and elongation are used to activate the skeleton joints and produce motions.

Cani *et al.* [20] propose to use an automatic skinning algorithm based on swept surfaces. The skeleton is used as a path for extrusion of interpolating spline surfaces. This approach is called automatic fleshing of skeletons and wrap a regular skin around the skeleton of an object. This skeleton coating process is based on various surface operators proposed to create complex shapes from simple surfaces such as welding and pinching. Skin binding is achieved by linking the skeleton to the skin with stiff springs.

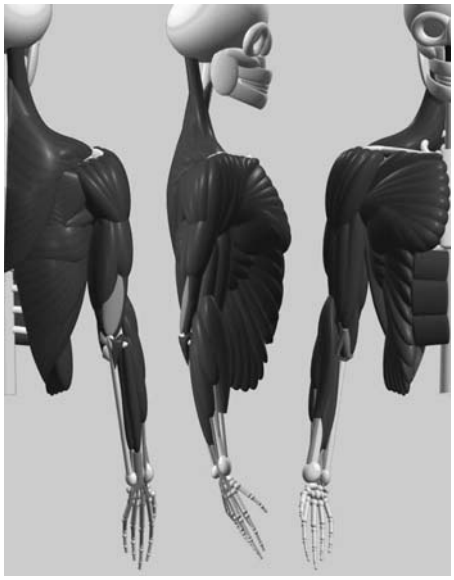


Fig. 1. Realistic musculature structure attached to the skeleton, [84], *(courtesy of F. Sheepers and R. Parent)*

3.2 Skin mapping

In this family of approaches, the skin is produced separately as a geometric surface and is then attached to the limbs of the skeleton. Intermediate data structures, or layers, can be used to connect the skin to the skeleton. Their role consists in keeping the skin tightly attached to the skeleton. Two main constraints have to be considered:

1. The skin has to follow the motion of the skeleton and to keep globally the same relative position with respect to the skeleton at any time of the animation. The main constraint here is to keep globally the skin wrapped around the control skeleton.
2. The skin has to mimic the real behaviour of the physiology and anatomy of the character simulated. For example for a virtual human, the shape of the muscles is changing according to the posture and therefore local deformations have to be controlled and adjusted along the limbs and at the joints. The main constraint here is to add a level of refinement to the skin deformation process in order to adjust it to mimic the anatomy of the simulated character.

The first issue consists in segmenting the skin and in binding each skin segment to the bone segment(s) that is(are) expected to control. The next issue is to define a local representation system so that each skin segment can be represented locally with respect to the influencing bone segments. Once the local representation is defined between the skin and the skeleton, skeleton-driven deformation can be applied to deform the skin when the skeleton is in motion.

In most of the skeleton-driven deformation models, the articulated character is made of two layers: the skeleton and the skin. More complex models include intermediate layers to improve and refine the control of the skin. For example, a complex layered elastic framework

for animated characters is proposed in the CRITTER system by Chadwick *et al.* [25]. An intermediate muscle layer is introduced between the flexible skin and the rigid skeleton. This intermediate structure is made of Free-Form Deformation (FFD) control boxes attached to the skeleton. These boxes define a space deformation around the skeleton. The control boxes shapes are parameterized with the joint angles of the skeleton. Two main deformation operators are applied to the control boxes, the first one controls the bending of the mesh around the joint and the second one mimics the muscle behavior by inflating the skin along the bones to replicate muscle contraction effect. Similar extended and optimized approaches have been proposed in [64, 69, 70] where the deformable skin is embedded inside a control lattice. Singh *et al.* [88] propose an alternative intermediate structure that consists of a low resolution mesh instead of a control lattice.

Most of the existing modeling approaches attempt to capture and simulate the deformations due to skeleton motion. Very few approaches are investigating secondary deformations such as the one due to biological functions such as breathing [111]. In [111], Zordan *et al.* propose a sophisticated extended anatomy-based model of the trunk based on a simple spring system to represent the muscles that control the motion of the ribs and diaphragm combined with a volume conserving deformable model that integrates the motion of the abdomen moving in reaction to the simulated diaphragm. The volume of the abdomen's deformable body is preserved and is quasi-incompressible.

4 Skeleton-driven deformation

Skeleton-driven deformation, a classical method for basic skin deformation, is probably the most widely used technique in 3D character animation. In research literature, an early version was presented by Magnenat-Thalmann *et al.* [66], who introduced JLD (Joint-dependent Local Deformation) operators to smoothly deform the skin surface.

This technique has been given various names such as Sub-Space Deformation (SSD), linear blend skinning, or smooth skinning. This method works first by assigning a set of joints with weights to each vertex in the character. The location of a vertex is then calculated by a weighted combination of the transformation of the influencing joints. The skeletal deformation makes use of an initial character pose, namely dress pose, where the transformation matrix of the i^{th} influencing joint and the position of the vertex are defined. While this method provides fast results and is compact in memory, its drawbacks are the undesirable deformation artefacts in case of important variation of joint angles. Several attempts have been made to overcome the limitation of geometric skin deformation by using examples of varying postures and blending them during animation. Aimed mostly at real-time applications, these example-based methods essentially seek for solutions to efficiently leverage realistic shapes that come either from captured skin shape of real people, physically based simulation results, or sculpted by skilled designers. Mohr *et al.* [71] have presented an extension to SDD by introducing pseudo-joints. The skeleton hierarchy is completed with extra joints inserted between existing ones to reduce the dissimilarity between two consecutive joints. These extra joints can also be used to simulate some nonlinear body deformation effects such as muscle bulges. Once all the extra joints have been defined, they use a fitting procedure to set the skinning parameters of these joints. The weights and the dress position of the vertices are defined by a linear regression so that the resulting skin surface fits to example body shape designed by artists. Having weights well defined, those examples could be discarded during runtime. Guo *et al.* [38] propose a pseudo-anatomical skinning method. Their purpose is to provide a simple intermediate layer between the skeleton and the skin, namely the chunks, to avoid the tedious design of the anatomical

layers. Their hypothesis is that internal anatomical structures do not need to mimic the real ones. A chunk is a deformable structure modeled as a set of nodes connected by links in a multi-sliced structure, automatically extracted from a few patches designed on the skin by the user. All chunks are connected and constitute a continuous layer between the skeleton and the skin. When large volume of space needs a huge amount of nodes to fill it up, it is possible to reduce computational costs by only taking a thinner layer of space between the skin and a base shell attached to the skeleton. Deformations of chunks, controlled with a finite element method (FEM), are adjusted according to the skeleton posture. The authors demonstrate that their approach is able to depict most of the visual aspects of skin deformations generated by real internal anatomical structure and particularly by muscle contractions. However, the final visual results and the computational costs greatly rely on the designed chunks architecture and therefore on the designer's skills.

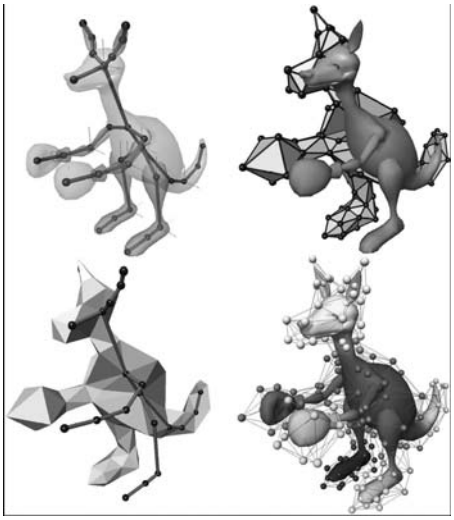


Fig. 2. Interactive construction process for skeleton-driven animation, [22], *(courtesy of S. Capell, B. Curless and Z. Popovic)*

As stated by Capell *et al.* [21], "elastic simulation has proved to be a powerful method both for automatically creating plausible skeleton-dependent deformations and for introducing secondary motions". Since Terzopoulos *et al.* [94, 96, 95], various approaches have been proposed to introduce more realism by introducing dynamic-based deformations such as proposed by Faloutsos *et al.* [33]. Among the most recent ones, Capell *et al.* [22] introduce a framework for skeleton-driven animation of elastically deformable characters. The proposed framework is somehow similar to FFD-based (Free Form Deformation) animation approaches as it embeds the object in a control lattice. However, the authors use continuum elasticity and FEM to compute the dynamics of the object being deformed. Bones of the control skeleton are restricted to lie along the edges of the control lattice, so that the skeleton can be considered as a constraint in the dynamic system. The animator specifies the skeleton and the control lattice interactively (see Figure 2). A joint is created by clicking on the object and it is placed midway between the two mouse ray/surface intersections to ensure that joints are centrally

located inside the object. By selecting a root joint and defining bones, a complete control skeleton can be defined. The control lattice is also interactively constructed by the user by adding cells incrementally and updating the control vertices as needed. Although the authors stated that existing volumetric meshing schemes should allow the automatic construction of the control lattice, they acknowledged that several hours are required even for an experienced user to create a moderately complex control lattice. This approach has been recently extended by Capell *et al.* [21] to provide more control and flexibility at interactive rate by introducing force-based rigging. Force-based rigs provide more flexibility to the animator for controlling the shape, other than by only moving the skeleton (see Figure 3). The realism of the deformations is improved by introducing a collision scheme for managing collisions near creases. Rig forces can be computed from sculpted or measured surface deformations. Moreover, rigs can be transferred between characters.

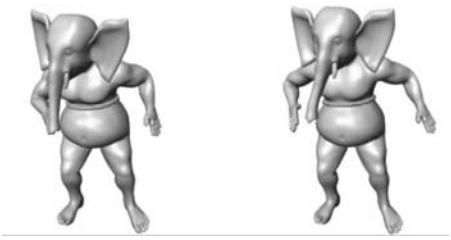


Fig. 3. Constrained deformations on a Ganesh-like character, [21], *(courtesy of S. Capell, B. Curless and Z. Popovic)*

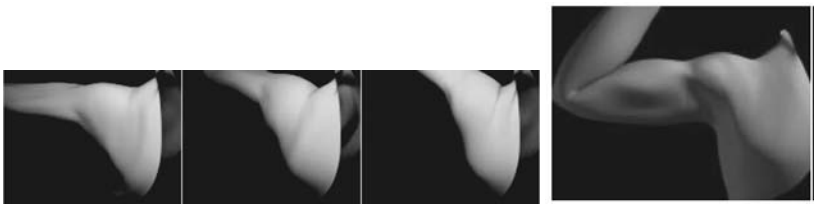


Fig. 4. Pose space deformation, [59], *(courtesy of J. P. Lewis)*

Pose space deformation [59] approaches the problem by using artistically sculpted skin surfaces of varying posture and blending them during animation. Each vertex on the skin surface is associated with a linear combination of Radial Basis Functions (RBFs) that compute its position given the pose of the moving character (see Figure 4). These functions are formed by using the example pairs - the poses of the character, and the vertex positions that comprise skin surface. More recently, Kry et al [53] proposed an extension of that technique by using Principal Component Analysis (PCA), allowing for optimal reduction of the data and thus faster deformation. Sloan et al [90] have shown similar results using RBFs for blending the arm models. Their contribution lies in that they make use of equivalent of cardinal basis function. The blending functions are obtained by solving the linear system per example rather than per DoF, which potentially is of a large number, thus resulting in an improved performance.

The method proposed by Allen *et al.* [4] is also based on skin pose examples (see Figure 5). Similarly to animation by target-morph, they use a set of poses as targets or examples in order to produce the skinning information, whereas traditional approaches rely on much less information, namely a skeleton and a template skin in a neutral posture. If the set of target poses correctly covers the pose space, then the estimated skinning can be applied to any motion. The resulting advantages over other approaches are to be able to introduce the targets as constraints in the skinning and to rely on real or manually designed data, such as the ones obtained from 3D scanners. Moreover, it alleviates the designer's involvement so that the results are less dependent on their skills but still remain user controllable. The approach recently proposed by James *et al.* [42], named Skinning Mesh Animations (SMAs), presents an original example-based skinning scheme. The main difference is that it does not require any pre-defined skeleton. The method takes as inputs a set of deformed meshes representing the pseudo-articulated deformable shape in different poses. It then automatically estimates statistically relevant bones based on the hypothesis that clustering triangles with analogous rotation sequences indicates the near-rigid structure of the mesh animation. It further determines bone transformations, bone-vertex influence sets, and vertex weight values for producing skinned animations that approximate the original deformable animation. The skinning approximation is particularly suited for shapes with a sufficient near-rigid structure and does not apply well for highly deformable shapes. SMAs support hardware rendering and pose editing as well.

Fig. 5. Example-based skin deformation, [4], *(courtesy of B. Allen, B. Curless and Z. Popovic)*

Recent attempts extend the usual input data to other capture and acquisition modalities in order to achieve more reliable and accurate models. For example in [55], Kurihara and Miyata propose an example-based deformable human hand model derived from medical images. Multiple CT scans of the hand are taken in several poses. The link structure, joint rotation centers and joint angles are estimated for each scan using bone shapes. A polygonal mesh of one pose, the reference mesh, is deformed and fitted to other poses. The resulting hand model is deformed by using pose space deformation. In [75], the authors present a technique for capturing and animating human body motions using a commercial motion capture system and approximately 350 markers. Their goal is to obtain not only the motion of the skeleton but also the motion of the surface of the skin. The motion of the body surface is reconstructed by applying the three-dimensional trajectories for this dense marker set to a subject-specific polygonal model. The polygonal model is first adapted to fit the three-dimensional locations of the markers from a static pose. The rigid body motion is then extracted from the marker set whereas the remaining motion of the markers is used to estimate local deformations of the polygonal model. The deformations of the markers set allow dynamically simulating the fleshy areas under the influence of muscle shape variations.

5 Generation of Control Skeleton

In most cases, the articulated character is made of some geometric structure, and an articulated skeleton is bound to the model by the user, typically by manual interactions defining a correspondence between the primitives of each structure. Some binding must then be made to couple skin surface motions to those of the skeleton. This can be done for example by generating spring networks or spatial deformation fields. These two processes are particularly tedious, especially when the model to be articulated is given only as a boundary representation. Few attempts have been done in order to automatically generate the articulated skeleton from a 3D surface or shape. The main expected advantage of this approach is to have an immediate mapping of the skin to the skeleton.

The main difficulty of the automatic generation of control skeleton is to map the articulated skeleton to the extracted geometric skeleton, particularly when the topology of the articulated skeleton is predefined such as for virtual humans. There is no guarantee that the topology of the extracted geometric skeleton will match the predefined topology of the corresponding articulated one. Moreover, the geometric skeleton only defines the location of the joints, but does not contain the rotation axis.

5.1 Medial axis-based methods

Some of the existing approaches are based on the generation of the medial axis, which is further simplified in order to catch the appropriate articulated structure of the 3D shape. Bloomenthal [12] proposes a method to derive a geometric skeleton from the medial axis of a static object. An articulated skeleton is further attached to the geometric and is used to control and alter the shape of the object for animation. The shape of the object is reconstructed from the updated geometric skeleton once it has been updated by the articulated one. Teichmann *et al.* [93] create an articulated control skeleton and bind it to the surface by first computing an approximate medial axis of the 3D mesh using 3D Voronoi diagram. The medial axis is then simplified, resulting in a tree structure made of chains of edges and nodes. Selected nodes are interpreted as joints of a skeleton, and the chains connecting them as its limbs (see Figure 6). A spring network is then produced to bind the skeleton to the boundary, so that skeletal motions will update the surface boundary as specified by the animator (see Figure 7).

Wade *et al.* [102] describe an algorithm for automatically generating a control skeleton for use in polygonal data model animation. The main process consists in discretizing the 3D shape by voxelization, computing the corresponding discrete medial surface, and then using it both to create the skeletal structure and to attach the vertices of the model to that structure (see Figure 8). Unlike previous methods, the algorithm is almost fully automatic, requiring very little user input.

Lazarus *et al.* [56] provide an approach that is expected to overcome the limitations of medial axis for boundary-based representations. The proposed paradigm constructs one-dimensional axial structures associated with a polyhedral surface. These structures are called the level set diagrams (LSDs). They are associated with scalar functions defined over the set of vertices of a polyhedron. They catch the overall shape and the topology of an object and can be used for deforming or animating an object. The skeletons for a man, a dolphin and a horse are demonstrated in the paper (see Figure 9). However, the skeleton proposed for the man shape is not accurate for animation as the location of joints and segments do not correspond to anatomical joints and limbs with enough accuracy. This is particularly obvious for the hand. A similar diagram based on the Reeb graph has been investigated in [45].

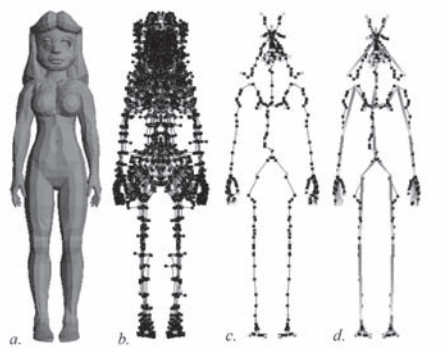


Fig. 6. Generation of control skeleton from approximate medial axis of the 3D mesh, [93], *(courtesy of M. Teichmann and S. Teller)*

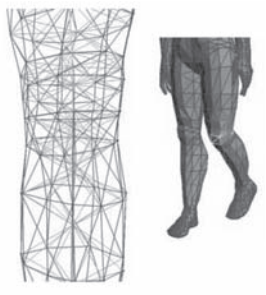


Fig. 7. Skin binding and update according to skeleton motion, [93], *(courtesy of M. Teichmann and S. Teller)*

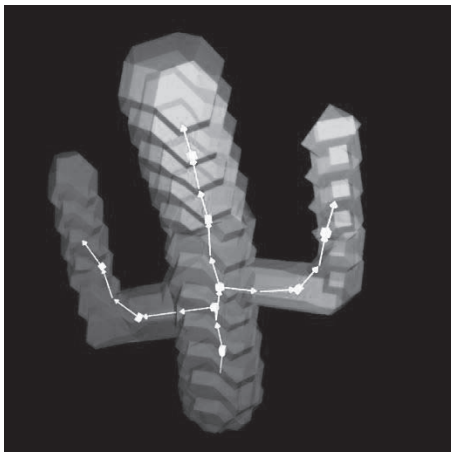


Fig. 8. Resulting articulated cactus character, [102], *(courtesy of L. Wade and R. Parent)*

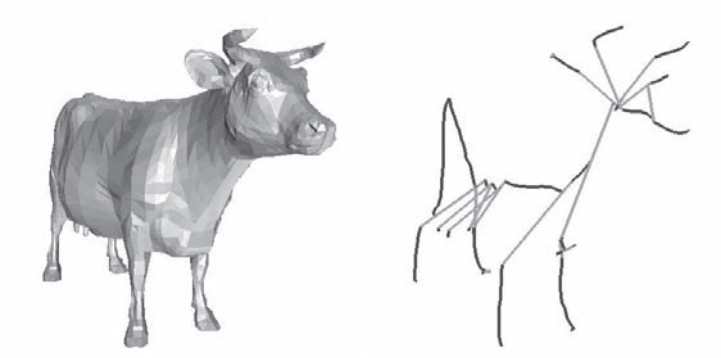


Fig. 9. Skeleton extraction using LSD, [56], *(courtesy of F. Lazarus)*

An alternate method has been suggested by Liu *et al.* [62] based on the computation of the repulsive force field with ray-casting. Local minimal points are selected as joint candidates. A modified thinning algorithm is then used to identify the final joints. Skin vertices are then bound to the resulting skeleton for animation using an SSD solution. Du *et al.* [31] propose to use diffusion equations to approximate medial axis of arbitrary 3D objects. The skeletonization method allows user interactions in order to build user-controlled skeleton. Yoshizawa *et al.* [109] propose an automatic method where a control animation structure and the associated skinning information are automatically obtained from a 3D surface. A skeletal mesh is first extracted from a given 3D surface using a Voronoi-based medial axis approximation. This skeletal mesh is then deformed with an interactive FFD scheme, detailed in [108]. During this skeletal deformation process, control points are first associated to the skeletal mesh and FFDs are applied to the skeletal mesh so that it follows the control points' displacements. The 3D surface is further reconstructed according to the skeletal mesh deformation. A two-step post-process is finally applied in order to remove artefacts such as folds, protrusions and global and local self-intersections. Gagvani *et al.* [34] propose a voxel-skeleton based method to animate a virtual human model made of volume data. They compute the skeleton of the Visible Human data set. The volume data set is skeletonised, then a skeleton tree is defined by connecting voxels, prior to animate it using motion capture data. Finally, the volume date set is regenerated to match to the skeleton tree and the volume animation is produced.

Most of the medial axis [10] approaches are sensitive to noise and deformation, which is particularly critical for extracting articulated skeletons from human scan data sets. Moreover, there is no guarantee on the topology of the extracted skeleton. This may not be important for most of articulated characters, but it is mandatory for virtual humans, as the articulated skeleton topology is fixed and predefined.

5.2 Template-based methods

In [85], the authors propose a template-based method to synthesise automatically human body, including the body shape and the control skeleton, from body measurements. The main idea consists in fitting a pre-defined template model to match a set of constraints corresponding to body measurements. The result is immediately ready for animation. The skeleton of the template model is adjusted to match the body measurements. The reconstruction process maintains the coherence between the resulting skin and skeleton. The skeleton of the template model can

be produced with any method. A previous template-based approach has also been proposed to reconstruct body shape from two photographs [58] (see Figure 10).

[79] and [3] propose deformation techniques to create individual hand models from photographs. In [79], an individual model is built from the surface anatomy visible in a single photograph of the palm. Image analysis allows extracting surface anatomy features, hand geometry and creases. The joint structure is then estimated from the surface anatomy features and the skin geometry is deformed using Radial Basis Functions (RBFs). In [3] Albrecht *et al.* developed a human hand anatomy-based model with its three layers structure: skin, muscles, and bones. Hand animation employs a hybrid muscle model. Pseudo-muscles control the motion of joints based on anatomical data and mechanical laws, while geometric muscles deform the skin tissue using a standard mass-spring system. Based on this reference model, they propose a deformation technique based on feature points to create individual hand models from photographs where the whole structure of the reference model is warped according to feature points assignments. The resulting deformed hand model is instantly animatable.

In [91], the authors aim at animating high-resolution human surface data captured from commercially available 3D active sensing technology. They apply a model-based approach, by matching a generic control model to the acquired surface data. The generic model is registered with the surface data using a set of interactively defined feature points, *i.e.* landmarks, and joint locations to recover the model posture. The generic model is further automatically fitted to the surface data as a shape constrained deformable surface. A similar approach is proposed by Moccozet *et al.* [68]. A full reconstruction pipeline produces a close to animatable approximation of the scanned data of a human body (see Figure 11). It is based on fitting a human template model defined in [86]. An initial location of landmarks is automatically defined from a multi-scale morphological analysis of the 3D data surface.

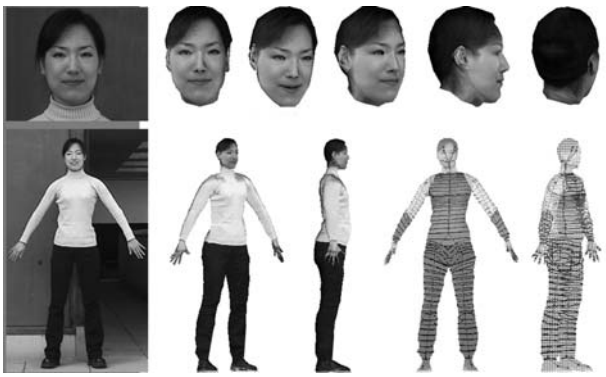


Fig. 10. Human body shape cloning from photos, [58]

Blanz and Vetter [11] derive a morphable face model from a data set of 3D face models by transforming the shape and texture of the examples into a vector space representation using statistical analysis (Principal Components Analysis - PCA). New faces and expressions can be produced by linear combinations of the examples. Shape and texture constraints derived from the reference faces can be used to control manual modeling or automated matching algorithms from individual photographs. Reveret *et al.* [78] propose a method based on PCA for the automatic generation of control skeletons for four footed animals. They use a set of skeletons

built by a skilled animator as the learning database. The resulting morphable skeleton model can be adjusted to any 3D quadruped model by taking three measurements on the side view on the quadruped mesh. Animation is further controlled with smooth skinning for geometry binding.

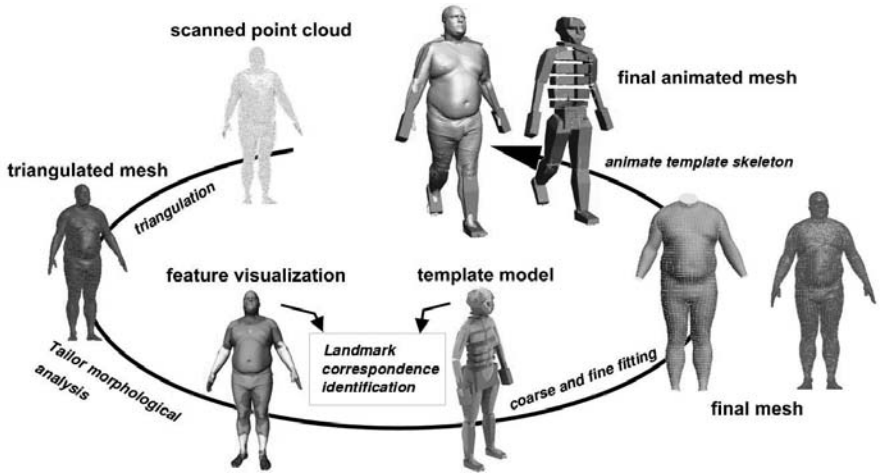


Fig. 11. Template-based reconstruction pipeline, [68]

The reconstruction of human body shapes from skeletal remains of particular interest in forensic medecine (postmortem identification [43]) or archeology and ethnology (reconstruction from ancient skeletal remains [8]). The point is to predict and model the layers of tissue from the skull. Such prediction is a well-known approach in forensic medecine, usually achieved by physical sculpting with clay. These techniques are very close to surface fitting and can be easily automated based on the forensic knowledge. The input data sets can be either a surface mesh built from 3D scan data or volumetric model made from medical images such as CT. In [43], anatomical landmarks are attached to a skull model generated from 3D scan data of a skull. The landmarks are correlated with statistical tissue depth measurements in order to provide reference points to generate muscles and skin by fitting a reference anatomy-based virtual head model, incorporating skin and muscles. The resulting anatomical model can be animated in order to mimic various expressions of the reconstructed face. A similar approach is proposed in [8] where a volumetric model of a Egyptian mummy head is processed in order to fit a reference model. The model can also be textured in order to improve the visual aspect of the result.

Template-based methods rely on prior knowledge of the articulated structure of the articulated 3D shape to build. A drawback of this approach is that it is not general and requires to produce a template for each family of articulated shape to handle. In addition, these methods usually require landmarks extraction for template matching that are difficult to control in a fully automatic way.

5.3 Mesh decomposition based methods

Although in [42] an underlying skeleton is implicitly extracted by clustering triangles with similar rotation sequences, indicating the near-rigid structure of the mesh animation, work with a specific focus on mesh decomposition to extract skeleton have been proposed. Katz *et al.* [46] apply a hierarchical decomposition algorithm to define a method for generating and attaching an articulated control skeleton to a given polyhedral surface for animation (see Figure 13). Once the main components of the objects have been segmented and identified by the decomposition algorithm, joints are hierarchically positioned and attached to the skin surface (see Figure 12). A similar technique is proposed in [60]. The authors use approximate convex decomposition, which partitions a model into nearly convex components. A skeleton of the model is then extracted from the convex hulls of the nearly convex components. This process is iterated until the quality of the skeleton becomes satisfactory. They also demonstrate that this can be used to generate natural skeletal deformations. They compare their results with [46] and [107] and show that the skeleton extraction remains stable and robust under perturbation and deformation.

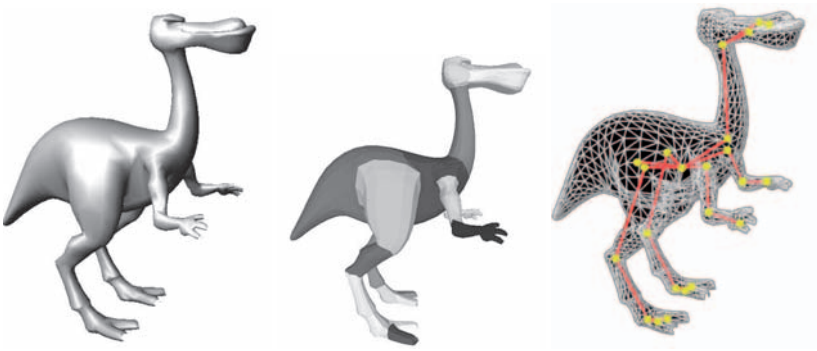


Fig. 12. Polyhedral surface and associated reconstructed skeleton, [46], *(courtesy of S. Katz and A. Tal)*

The method proposed by Anguelov *et al.* [5] starts from a set of 3D meshes corresponding to different configurations of an articulated object. The algorithm automatically recovers a decomposition of the object into approximately rigid segments, the location of the segments in the different object instances, and the articulated object skeleton corresponding to the segments (see Figure 14). The algorithm registers the input meshes with the correlated correspondence algorithm. It then iteratively evaluates the segment assignment for each point and the rigid transformation of each segment. Finally, the joints are estimated with articulation constraints.

Similarly to the previous family of approaches, segmentation methods do not guarantee the topology of the extracted skeleton, although some of them show that they are quite robust and stable under noise, perturbation and deformation. More precise and stable results can be obtained with methods that use a set of data scans from the same subject in different postures.

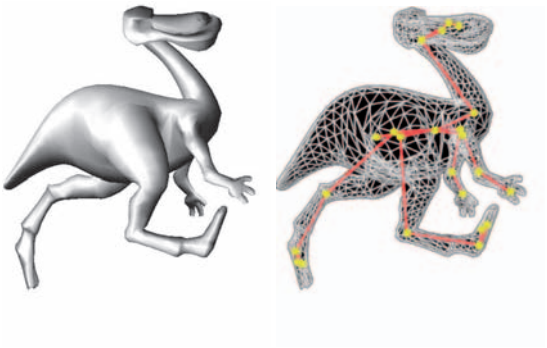


Fig. 13. Animated surface driven by the articulated skeleton, [46], *(courtesy of S. Katz and A. Tal)*

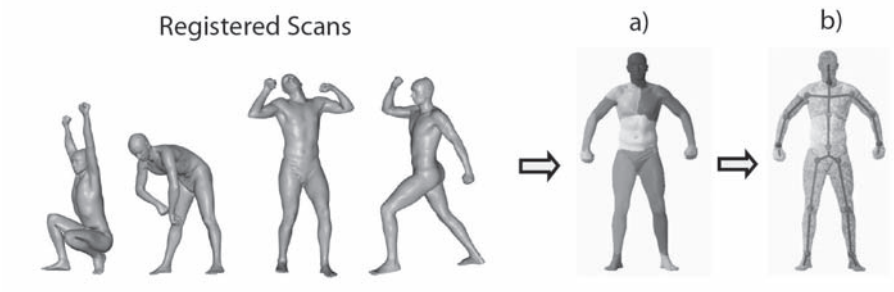


Fig. 14. Automatic decomposition of articulated object into rigid parts, [5], *(courtesy of D. Anguelov)*

6 Discussion on skeleton for Virtual Humans

Based on this survey, we suggest the following taxonomy regarding articulated skeleton generation for virtual humans:

1. Direct binding approaches
   In this family of approaches, a skin shape and a pre-defined skeleton are first produced separately and the skin is then bound to the skeleton. Traditional approaches consist in interactively or semi-automatically placing the skeleton inside the skin, and in associating skin vertices to skeleton limbs to bind the skin deformations to the skeleton motions. These methods are usually time-consuming and greatly rely on the skills of the designers. The accuracy of the skin deformations is closely related to the correct placement of the skeleton with respect to the skin shape. It usually requires a lot of interactive tuning and refinements in order to achieve a correct and appropriate binding.
2. Pseudo-anatomical approaches
   In pseudo-anatomical approaches, intermediate layers are attached to the skeleton. They usually mimic the behaviour of a muscular layer and reproduce the real or visual behaviour. The geometric primitives associated to the layers can be deformed according to the postures of the skeleton. Once these geometric primitives have been adjusted to match

the current skeleton posture, a geometric skin is produced to reflect the posture. The geometric skin can be either generated from the muscle layers at each frame or deformed to match to the shapes of the muscle geometric primitives. The main advantage of this family of approaches is that the articulated character is iteratively built from the skeleton layer to the skin layer. In addition, these approaches allow modelling a wide range of articulated characters and are not limited to a specific type. The main drawback is related to the building of the character, which is usually tedious and time consuming. Additionally, existing methods are mainly focused towards the deformations during motion and do not address the issue of modeling new characters. Some methods provide example-based features and propose to adjust some parameters of the multi-layer structure in order to reflect possible modification of anatomy if the underlying skeleton structure is the same.

3. Skeleton extraction-based approaches
   In these approaches, the articulated skeleton is directly extracted from the skin shape. The main advantages are that the resulting skeleton catches the structure of the shape and that the binding between the skeleton and the skin is immediately available from the extraction process. The drawback is that extracted skeletons are usually too complex to reflect the anatomical skeleton structure and that it is not immediately usable, moreover it does not provide the DoF of the articulated structure.
4. Example-based approaches
   In example-based approaches, a pre-defined articulated template model is fitted to match some data extracted from the instances to model. The main limitation is that these methods are based on pre-existing know-how and are therefore limited to a given range of articulated models. A new template has to be built for each kind of character. On the other side, the binding between the skin and the skeleton is immediate, and depending on the methods used to fit the template to the input data, it is possible to get a multi-modal reconstruction scheme able to handle low and high level data input for the reconstruction.

New approaches are currently investigated that do not limit capturing techniques to catch the static shape or the skeleton motion but instead consider at the same time dynamic shapes and skeleton motions to automate the building and deformation of articulated character. In [81], the authors propose a method for the acquisition of deformable human geometry from silhouettes. The technique involves a tracking system to capture the motion of the skeleton, and estimates skin shape for each bone using constraints provided by the silhouettes from one or more cameras. The proposed algorithm provides a simple mechanism for solving the problems of view aggregation, occlusion handling, hole-filling, noise removal, and deformation modelling. The resulting model can be parameterized to synthetize geometry for new poses of the skeleton. The quality of this kind of approach is limited by the amount of details captured, the accuracy of the skeleton estimation from motion capture, and the range of motion in the training data set as usual for example-based approaches. In [6], the SCAPE method (Shape Completion and Animation for PEople) is a data-driven method for building a human shape model. It simultaneously catches the variation in both subject shape and pose. The method is based on a representation that incorporates both skeletal and body deformations. A pose deformation model is trained to derive the body surface deformation as a function of the pose of the articulated skeleton. A second model is trained to acquire the variation of the body shape. The two models are combined to produce 3D surface models with realistic body deformations for different people in different postures.

The investigation of hybrid methods combining example-based and skeleton extraction-based approaches should lead to the development of robust and accurate frameworks for reconstruction of deformable articulated characters.

7 Multi-Resolution Techniques

A virtual human is represented through a set of surfaces for describing the shape of a body and through its animation information, which usually consist of facial and body animation. In general, a surface is described by a triangle mesh enhanced with texture information. A high-quality texture gives realistic rendering results with relatively lower computational costs. With respect to regular solid objects in a virtual environment, the most important feature of the virtual human is animation. The surface of a virtual human is deformed by its animation. In this Section, we discuss several approaches to Level-of-Detail (LoD) representations for virtual humans, which are independant of the four families of approaches presented in Section 6.

7.1 Simplification of Shape and Control Structure

Simplification of textured body surfaces

An LoD model of a free-form surface is generated by a repeated sequence of surface simplification operations. One important surface feature is the texture information associated with the surface (see Figure 15). Several approaches have been proposed in the literature to deal with texture information while simplifying a surface [35, 27, 82, 48]. A LoD model for the virtual human should preserve information on deformations that would be changed by animation along with traditional surface features. Here, we briefly review approaches that consider texture information in surface simplification and possible extensions to deformable surfaces.

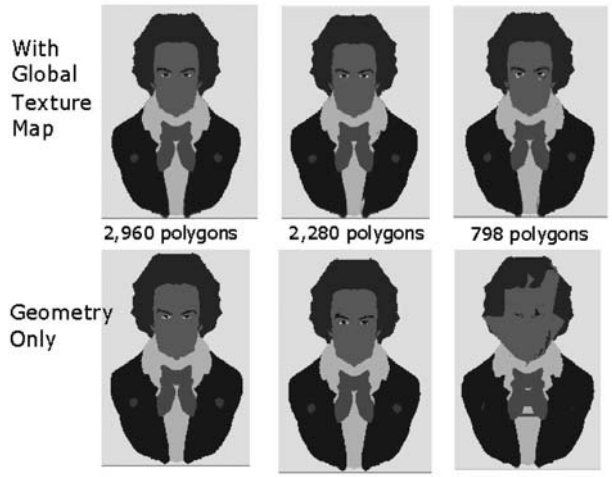


Fig. 15. Example of texture generating simplification, [48]

A common approach to consider texture is texture preserving simplification methods. These methods preserve texture coordinates during simplification. Garland *et al.* [36] extend energy space into combined space of position, color and normal vectors. Cohen *et al.* [27] use

transformed deviation of the normal and color in the three-dimensional geometric space. Lindstrom [61] proposes to use deviation criteria based on image differences. Although texture preserving simplification methods give good results, applying these methods to deformable object still has limitations. As a deformable surface has different mapping onto the surface caused by the changing in surface areas, polygon distributions and shapes, most of the current measurements for texture distortion do not correctly fit into the deformable surface. Simplification approaches based on texture generation have been proposed with the purpose of replacing complex geometry with texture information.

An impostor is an example of this kind of texture generating simplification [65]. It is an image that represents a part of a complex object, and it is usually generated by capturing a rendered image from predefined camera positions. It is mapped on the polygon that is best suited for other camera positions. Sillion *et al.* [87] suggest re-meshing the area of an impostor to improve rendering results. Schaufler [83] proposes a regeneration method of impostors using image warping techniques while Oliveira *et al.* [74] uses more general image-based rendering techniques. An impostor approach to reproduce character animation with animated textures has been proposed by Aubel *et al.* [9], and by Tecchia *et al.* [92]. With the pre-sampling of 32 positions at 8 elevations, Tecchia *et al.* [92] select at runtime a texture to map with the animation current frame and the orientation of the character to the camera as inputs. By taking advantage of the temporal coherence, Aubel *et al.* [9] use geometric LoDs and dynamically generated animated impostors with multi-planes. The textures are regenerated when needed, according to a threshold value on the animation. Dobbyn *et al.* [30] also use impostors on top of a full geometry-based crowd rendering engine (see Figure 16). Their hybrid engine is able to seamlessly switch from geometry to impostor, computed on the GPU, according to a pixel-to-texel ratio computed as:

$$d_{switch} = \frac{d_{nearplane}.Texel_{size}}{Pixel_{size}}$$

with d_{switch} the distance where the ratio is equal to 1, $d_{nearplane}$ the distance from the camera to the near plane, and:

$$Texel_{size} = \frac{2.d_{nearplane}.tan^{-1}(\frac{\theta}{2})}{x}$$

$$Pixel_{size} = \frac{2.d_{cam}.tan^{-1}(\frac{\theta}{2})}{x}$$

where x is the resolution of the impostor image, θ the camera's field of view, and d_{cam} the distance of the character to the camera. Hamill *et al.* [39] study the perceptual impact of both image-based and hybrid geometry-impostor methods on characters and on virtual buildings with different experiments: character discrimination, transition detection, impostor updates, *etc.* Their experiments also provide thresholds on impostors efficiency. Although an impostor offers the advantage of preserving details with a small number of triangles, it cannot be easily applied to general detailed triangle meshes, especially those used for deformable surfaces. In addition, the possible variety of different character animations is an important issue for such approaches.

There have been other approaches to generate texture images to be used in combination with an LoD model. Certain *et al.* [24] propose a model for generating a texture map for a simplified surface, which keep geometry resolution under control. Cohen *et al.* [27] propose an energy-based metric for mapping texture images to different geometries. This error measurement can be used for a simplification method combined with texture generation to ensure the quality of the simplified mesh in terms of texture mapping error. The most significant accom-

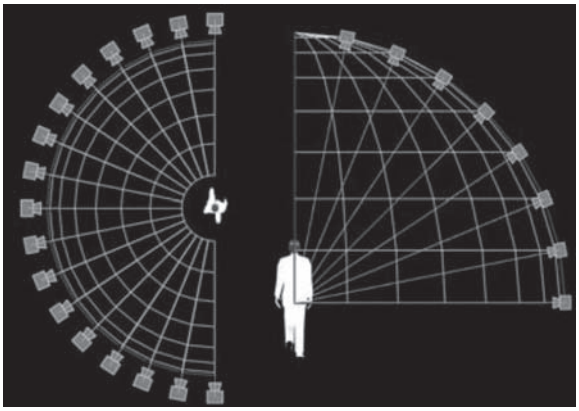


Fig. 16. Generation of impostors from different viewpoints, [30], *(courtesy of C. O'Sullivan)*

Fig. 17. The left two images are generate without proper handling of texture information. The right two images with minimized texture stretching, [82], *(courtesy of P. Sander)*

plishment in simplification methods based on texture generation is given by texture generation methods for LoD meshes as [82, 48] (see Figure 17).

Preservation of Features in Simplification

Considering animation parameters is an important issue in surface simplification for virtual humans. The animation of the virtual human consists of facial and body animation. The simplification process to generate an LoD model should consider these animation parameters so that the low complexity model can be animated as close as possible to the original model.

Facial animation is usually represented by the movement of feature points on the surface of the face. Feature points also control deformations of neighbouring surfaces thus resulting in the animation of facial expressions [54]. A method has been developed to preserve facial animation parameters in simplification [47] (see Figure 18). Facial animation has a well-defined surface deformation which follows facial action points and follows deformation parameters. One fundamental issue is preserving facial action points over the simplification process as long as possible. Whether action points are preserved or not, it is still required to regenerate deformation parameters. A method for regenerating these deformation parameters from the action points and deformation bounds, both in direction and magnitude, is proposed in [89].

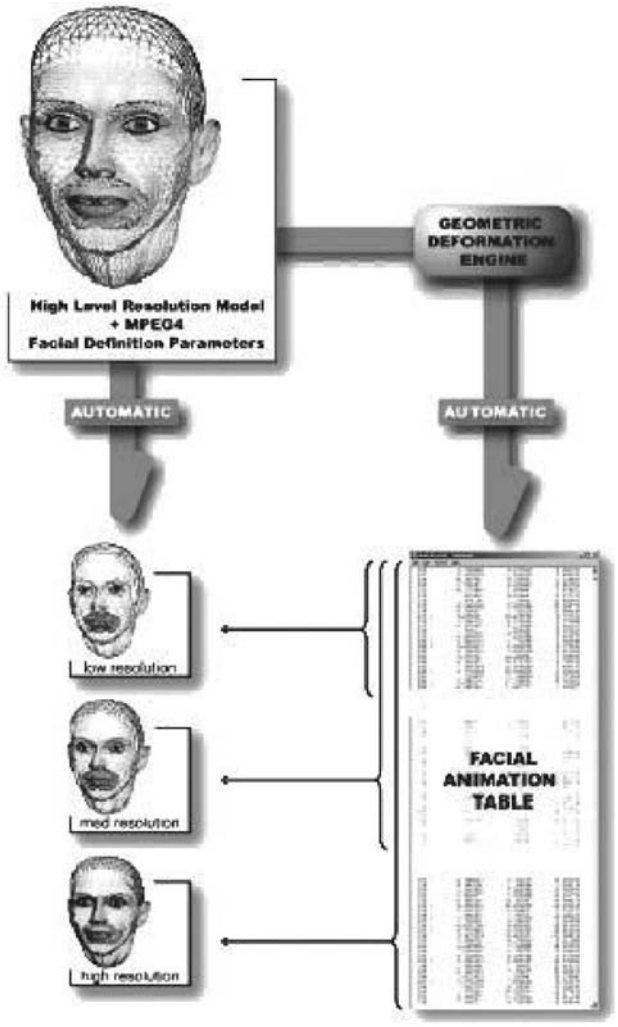


Fig. 18. Handling facial animation parameters in multiresolution model, [47]

The surface deformation parameter is calculated by its geometrical properties such as surface distance, or curvature.

For body animation, it is important to preserve surface details around deforming parts such as surface parts near joints. For example, a surface part which is close to a joint could have more detail, to represent enough surface detail even when the part is highly deformed by joint rotation. Overlapping regions around joints could be simplified according to animation parameters as well. Preserving more details for the dynamically deformable surface is still an open research problem as research that relates LoD representations with skeleton-based deformation and animation is relatively recent.

7.2 Multi-Resolution Modeling for Key Parameters

Although the complexity and representation of 3D humans are predominant factors to consider for optimization, motion data (see Figure 19) is also a very important step in character animation that researchers investigate for improvements.

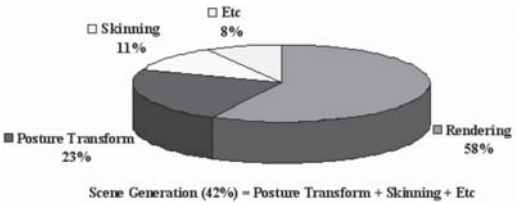


Fig. 19. Illustration of sample costs per step for a 1000 animated characters crowd, [2]

Motion LoD

As for geometry, animation models and parameters can be adapted and simplified in certain cases, *i.e.* when motions are too fast, too distant, or too numerous for human sight, or when motions are of low interest in a given context. LoD-like approaches are thus applicable for animation to manage the computational and memory costs of transformations and deformations, and to provide a controlled trade-off between performances and quality.

Early work by Granieri *et al.* [37] proposes the pre-definition of three different static resolutions for character animation. These resolutions are depending on the geometric model complexity, but also on the motion sampling frequency and on the skeleton complexity of characters, *i.e.* DoF, number of bones and joints. The pre-processed resolutions are stored in a graph and selected during the real-time rendering based on characters distance to the camera. To reduce the animation complexity, Ahn *et al.* [2] present a method that simplifies motion data to be used later on in real-time applications. The simplification phase consists in a posture clustering, generated according to the frame-based computations of joint motion distances and a given error threshold. Cozot *et al.* [29] propose an architecture to adapt the complexity of character animation based on a pipeline of different animation models. For each step in the pipeline, namely the body motion, the angular trajectories and the remainder of the body, a set of different animation models with different complexity are usable, *i.e.* none, direct and inverse kinematics and dynamics. The possible resolutions for runtime are therefore pre-defined by the architecture construction, which facilitates smooth transitions between the different levels. They apply their method on walking characters, where models are selected dynamically according to the global complexity of the scene and the distance of the character to the camera.

Recently, Redon *et al.* [77] presented an adaptive forward dynamics model. Based on dynamic bodies, they simulate hybrid skeletons with active joints, with complete accelerations, velocity and position updates, passive joints, with bias acceleration and inverse inertia updates, and rigid joints, with only bias acceleration. In areas concerned, and according to a defined total desired number of DoF, joints change states according to error metrics on acceleration and velocity updates detailed below.

MULTIRES	Dynamic	Character	Data
Granieri[37]	Low	Medium	Low
Cozot[29]	Medium	High	Low
Ahn[2]	Low	High	High
Redon[77]	High	Medium	Medium

Table 1. Adaptation features from multi-resolution methods for character animation.

RESOLUTION	Grain	Control
Granieri[37]	Low	High
Cozot[29]	Medium	Medium
Ahn[2]	High	High
Redon[77]	High	Medium

Table 2. Resolution features from multi-resolution methods for character animation.

We propose an overview of the features of methods in multi-resolution character animation in Tables 1, 2 and 3. The first one is on the multi-resolution mechanism itself, its dynamicity, its application for character animation, and if it provides data adaptation. The second presents the granularity of the multi-resolution methods, as well as the control on resolution in terms of generation. The last table illustrates the features for runtime selection of appropriate resolutions based on a cost for resolution changes, on view dependency and on motion error metrics.

DECISION-MAKING	Cost	View	Error
Granieri[37]	Low	Yes	No
Cozot[29]	Medium	Yes	No
Ahn[2]	Low	No	Low
Redon[77]	High	No	Medium

Table 3. Decision features from multi-resolution methods for character animation.

Motion Error Metrics

Motion error metrics allow the quantification of loss when simplifying skeletons with joint or DoF reduction. This provides a useful control to select appropriate resolution according to different animations. Although some of the metrics presented here might have not been initially designed for multi-resolution animation methods, such as the error metrics from motion graph techniques, we believe they are relevant for the optimization of character animation as used by Redon *et al.* [77] and Ahn *et al.* [2]. In the later work, the posture clusters are computed according to an error cost on joint motion depending on orientation (first term in the equation below), and on position (second term) errors:

$$E_j(t, t_{ref}) = \alpha.2log(q_j(t)^{-1}.q_j(t_{ref})) + r_j(t).\theta_j(t, t_{ref}))$$

with t the current frame, t_{ref} the estimated key frame, and q_j the orientation of joint j. θ_j, respectively r_j, is an angle difference, respectively a weighting factor, defined as:

$$\theta_j(t, t_{ref}) = arccos \frac{v_j^{'}(t).v_j^{'}(t_{ref}).}{||l_{j,c}(t)||^2}$$

$$r_j(t) = \sum_{c}^{leaf} l_{j,c}(t)$$

where $l_{j,c}$ is the segment length between j and c, and $v_j^{'}$ the trajectory of j. Key-postures per cluster are then computed in the resulting cost matrix as $KP = min_{t_{ref}}(\sum_t^m E_j(t, t_{ref}))$.

In the adaptive forward dynamics model proposed by Redon *et al.* [77], joints are activated, rigidified or set as passive according to error metrics on acceleration $a(C)$ and velocity $v(C)$ updates as follows:

$$a(C) = \sum_{i \in C} \ddot{q}_i^T.A_i.\ddot{q}_i$$

$$v(C) = \sum_{i \in C} \dot{q}_i^T.V_i.\dot{q}_i$$

with C is the articulated structure, A_i and V_i two $d_i x d_i$ symmetric positive definite weight matrices equal to the identity matrix in their work, and q_i the position of joint i.

To define and select potential transitions in motion graph, error metrics have also been defined in such work. Kovar *et al.* [52] propose a 2D similarity metric with point clouds based on vertices positions of polygonal meshes and not on the orientation and/or position of the skeleton joints. The distance D between two frames A_i and B_i is:

$$\min_{\theta, x_0, z_0} \sum_i w_i ||p_i - T_{\theta, x_0, z_0} p_i'||^2$$

where T_{θ, x_0, z_0} is a rigid 2D transformation that rotates p on the y axis of θ degrees and translates it by (x_0, z_0), w_i weights, p_i and p_i' corresponding points in the point clouds. A closed-form solution to his optimization is:

$$\theta = \arctan \frac{\sum_i w_i(x_i z_i' - x_i' z - i) - \frac{1}{\sum_i w_i}(\bar{x}\bar{z'} - \bar{x'}\bar{z})}{\sum_i w_i(x_i x_i' + z_i z_i') - \frac{1}{\sum_i w_i}(\bar{x}\bar{x'} - \bar{z}\bar{z'})}$$

$$x_0 = \frac{1}{\sum_i w_i}(\bar{x} - \bar{x'}\cos(\theta) - \bar{z'}\sin(\theta))$$

$$z_0 = \frac{1}{\sum_i w_i}(\bar{z} + \bar{x'}\sin(\theta) - \bar{z'}\cos(\theta))$$

where $\bar{x} = \sum_i w_i x_i$. Also for motion graphs, Lee *et al.* [57] propose a different metric, which is more similar to [2] and [77]. It is based on joint angles and velocities with which a probability of transitioning from frame i to frame j is mapped and estimated. The distance between frame i and frame j is computed as:

$$D_{ij} = d(p_i, p_j) + \nu d(v_i, v_j)$$

with $d(v_i, v_j)$ the difference of joint velocities, ν its weight, and $d(p_i, p_j)$ the following joint angles difference:

$$d(p_i, p_j) = ||p_{i,0} - p_{j,0}||^2 + \sum_{k=1}^{m} w_k ||\log(q_{j,k}^{-1}, q_{i,k})||^2$$

where $p_{i,0} \in \Re^3$ the root position at frame i, $q_{i,k} \in S^3$ the relative orientation of joint k, w_k joint weights manually determined, and m the number of joints. To express an animation sequence with selected key poses, Assa *et al.* [7] compute affinity matrices, representing the dissimilarities between frames, as:

$$d_a(f_1, f_2) = \sum_{j \in joints} b_j \frac{(x_j^{f_1} - x_j^{f_2})^2}{\sigma_j^2}$$

where d_a is the dissimilarity aspect between frame f_1 and f_2, x_a^f the a aspect at frame f and σ_j^2 the variance of joint j coordinates. With the same goals, Loy *et al.* [63] compute distance matrices with shape matching between each pair of frames of an animation.

7.3 Discussion on LoD for Virtual Humans

Virtual humans are represented by their shape and animation structures. The animation control structures modify shape through segment transformation or skin deformation. In parallel, for real-time rendering and animation, multiresolution methods for shape have been investigated and widely used [13]. There have been a few approaches to remap the animation structures for the simplified shapes but those are partly limited to the specific cases such as facial animation parameters. For efficient control and rendering, further research focus should be on generating animation structures according to the simplified shapes. To do this, it is necessary to make a mapping between the animation semantics (control structure) and syntax (shape). Furthermore, using this mapping, simplification on the animation structures could be conceivable in a more flexible way.

The animation of virtual humans is not only related to the body itself but also its close surroundings, especially clothes. For efficient animation, relationships between all component should be conceived in simulation methods. There have been a few works on modeling and simplification of these structures [73]. A control hierarchy starting from the environment to the base of body control, e.g. skeletons, would bring a full control of semantics over shapes in virtual human animation and rendering.

References

1. ISO/IEC JTC1/SC24 FCD 19774:200x. Humanoid animation (h-anim), 2004. `http://www.h-anim.org/`.
2. J. Ahn and K. Wohn. Motion level-of-detail: A simplification method on crowd scene. In *Proc. Computer Animation and Social Agent, CASA'04*, pages 129–137, 2004.
3. I. Albrecht, J. Haber, and H.-P. Seidel. Construction and animation of anatomically based human hand models. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 98–109, 2003.
4. B. Allen, B. Curless, and Z. Popović. Articulated body deformation from range scan data. *ACM Transactions on Graphics*, 21(3):612–619, July 2002.
5. D. Anguelov, D. Koller, H.-C. Pang, P. Srinivasan, and S. Thrun. Recovering articulated object models from 3d range data. In *Proceedings of the Uncertainty in Artificial Intelligence Conference (UAI2004)*, 2004.
6. D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: Shape completion and animation of people. In *Proceedings of the SIGGRAPH Conference 2005*, 2005.

7. J. Assa, Y. Caspi, and D. Cohen-Or. Action synopsis: pose selection and illustration. *ACM Transactions on Graphics*, 24(3):667–676, 2005.
8. G. Attardi, M. Betro, M. Forte, R. Gori, A. Guidazzoli, S. Imboden, and F. Mallegni. 3d facial re-construction and visualization of ancient egyptian mummies using spiral CT data. In *SIGGRAPH99 Abstracts and Applications*, pages 223–239, 1999.
9. A. Aubel, R. Boulic, and D. Thalmann. Real-time display of virtual humans: Levels of detail and impostors. *IEEE Transactions on Circuits and Systems for Video Technology*, 2:207–217, 2000.
10. S. Biasotti, D. Attali, J.-D.l Boissonnat, H. Edelsbrunner, G. Elber, M. Mortara, G. Sanniti di Baja, M. Spagnuolo, and M. Tanase. Skeletal structures. In L. De Floriani and M. Spagnuolo, editors, *Shape Analysis and Structuring*. Springer, 2007.
11. V. Blanz and T. Vetter. Construction and animation of anatomically based human hand models. In *Proc. of ACM SIGGRAPH 99*, pages 187–194, 1999.
12. J. Bloomenthal. Skeletal methods of shape manipulation. In Bob Werner, editor, *Proceedings of the International Conference on Shape Modeling and Applications (SMI-99)*, pages 44–49, Los Alamitos, CA, March 1–4 1999. IEEE Computer Society.
13. G.-P. Bonneau, G. Elber, S. Hahmann, and B. Sauvage. Multiresolution analysis. In L. De Floriani and M. Spagnuolo, editors, *Shape Analysis and Structuring*. Springer, 2007.
14. R. Boulic, T. Capin, Z. Huang, P. Kalra, B. Linterrnann, N. Magnenat-Thalmann, L. Moccozet, T. Molet, 1. Pandzic, K. Saar, A. Schmitt, J. Shen, and D. Thalmann. The humanoid environment for interactive animation of multiple deformable human characters. *Computer Graphics Forum*, 14(3):337–348, August 1995.
15. R. Boulic, R. Mas, and D. Thalmann. Complex character positioning based on a compatible flow model of multiple supports. In *IEEE Transactions on Visualization and Computer Graphics*, volume 3, 1997.
16. R. Boulic and D. Thalmann. Combined direct and inverse kinematic control for articulated figure motion editing. *Computer Graphics Forum*, 2, 1992.
17. D. Brogan, K. Granata, and P. Sheth. Space-time constraints for biomechanical movements. In *IASTED International Conference on Applied Modeling and Simulation (AMS)*, 2002.
18. D. Brogan, R. Metoyer, and J. Hodgins. Dynamically simulated characters in virtual environments. In *IEEE Computer Graphics and Applications*, pages 58–69, 1998.
19. N. Burtnyk and M. Wein. Interactive skeleton techniques for enhancing motion dynamics in key frame animation. *Commun. ACM*, 19(10):564–569, 1976.
20. M. P. Cani-Gascuel and M. Desbrun. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):39–50, jan - mar 1997.
21. S. Capell, M. Burkhart, B. Curless, T. Duchamp, and Z. Popović. Physically based rigging for deformable characters. In *Proc. Symposium on Computer Animation, SCA'05*, pages 301–310, 2005.
22. S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović. Interactive skeleton-driven dynamic deformations. In *Proc. SIGGRAPH'02*, pages 41–47, 2002.
23. M. Cavazza, R. Earnshaw, N. Magnenat-Thalmann, and D. Thalmann. Survey: Motion control of virtual humans. *IEEE Computer Graphics & Applications*, 18(5):24–31, sep - oct 1998.
24. A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle. Interactive multiresolution surface viewing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 91–98. ACM Press, 1996.

25. J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 243–252. ACM Press, 1989.
26. D. Chen and D. Zeltzer. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, pages 89–98, July 1992.
27. J. Cohen, M. Olano, and D. Manocha. Appearance-perserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 115–122. ACM Press, 1998.
28. G. Collins and A. Hilton. Modelling for character animation. *Software Focus*, 2(2):44–51, 2001.
29. B. Cozot, F. Multon, B. Valton, and B. Arnaldi. Animation levels of detail design for real-time virtual human. In *Proc. Eurographics Workshop on Computer Animation and Simulation, EGCAS'99*, pages 35–44, 1999.
30. S. Dobbyn, J. Hamill, K. O'Conor, and C. O'Sullivan. Geopostors: A real-time geometry/impostor crowd rendering system. In *Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games*, pages 95–102, 2005.
31. H. Du and H. Qin. Medial axis extraction and shape manipulation of solid objects using parabolic pdes. In *Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications 2004*, pages 25–35, 2004.
32. P. Faloutsos, M. Van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. In *SIGGRAPH'01*, pages 251–260, 2001.
33. P. Faloutsos, M. VanDePanne, and D. Terzopoulos. Dynamic freeform deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):201–214, 1997.
34. N. Gagvani and D. Silver. Animating volumetric models. *Graphical models*, 63(6):443–458, nov 2001.
35. M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.
36. M. Garland and P. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings of the conference on Visualization '98*, pages 263–269. IEEE Computer Society Press, 1998.
37. J. Granieri, J. Crabtree, and N. Badler. Production and playback of human figure motion for visual simulation. *ACM Transactions on Modeling and Computer Simulation*, 5(3), 1995.
38. Z. Guo and K. C. Wong. Skinning with deformable chunks. *Computer Graphics Forum*, 24(3):373–382, 2005.
39. J. Hamill, R. McDonnell, S. Dobbyn, and C. O'Sullivan. Perceptual evaluation of impostor representations for virtual humans and buildings. *Computer Graphics Forum*, 24(3), 2005.
40. D. Herbison-Evans. Real-time animation of human figure drawings with hidden-lines omitted. *IEEE Computer Graphics & Applications*, 2(9):27–33, 1982.
41. J. Hodgins, W. Wooten, D. Brogan, and J. O'Brien. Animating human athletics. In *SIGGRAPH'95*, pages 71–78, 1995.
42. D. James and C. Twigg. Skinning mesh animations. *ACM Transactions on Graphics*, 24(3), 2005.
43. K. Kahler, J. Haber, and H.-P. Seidel. Reanimating the dead: Reconstruction of expressive faces from skull data. *ACM Transactions on Graphics*, 22(3):554–561, 2003.

44. P. Kalra, N. Magnenat-Thalmann, L. Moccozet, G. Sannier, A. Aubel, and D. Thalmann. Real-time animation of realistic virtual humans. In *IEEE Computer Graphics and Applications*, volume 18, 1998.
45. P. Kanongchaiyos and Y. Shinagawa. Articulated reeb graphs for interactive skeleton animation. In *Proceeding Modeling Multimedia Information and System*, pages 451–467, october 2000.
46. S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954–961, July 2003.
47. H. Kim, C. Joslin, T. Di Giacomo, S. Garchery, and N. Magnenat-Thalmann. Adaptation mechanism for three dimensional content within the mpeg-21 framework. In *Computer Graphics International 2004*, June 2004.
48. H. Kim and K. Wohn. Multiresolution model generation with geometry and texture. *Proceedings of Seventh International Conference on Virtual Systems and Multimedia*, pages 780–789, 2001.
49. S. Kiss. Computer animation for articulated 3d characters. Technical Report 45, Twente University, 2002. `http://purl.org/utwente/38232`.
50. E. Kokkevis, D. Metaxas, and N. Badler. User-controlled physics-based animation for articulated figures. In *Computer Animation*, 1996.
51. K. Komatsu. Human skin model capable of natural shape variation. *The Visual Computer*, 3(5):265–271, 1988.
52. L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proc. SIGGRAPH'02*, pages 473–482, 2002.
53. P. Kry, D. James, and D. Pai. Eigenskin: Real time large deformation character skinning in hardware. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 153–160, July 2002.
54. S. Kshirsagar, S. Garchery, G. Sannier, and N. Magnenat-Thalmann. Synthetic faces : Analysis and applications. *International Journal of Imaging Systems and Technology*, 13(1):65–73, June 2003.
55. T. Kurihara and N. Miyata. Modeling deformable human hands from medical images. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2004.
56. F. Lazarus and A. Verroust. Level set diagrams of polyhedral objects. In *ACM Solid Modeling'99*, Ann Arbor, Michigan, USA, June 1999.
57. J. Lee, J. Chai, P. Reitsma, J. Hodgins, and N. Pollard. Interactive control of avatars animated with human motion data. In *Proc. SIGGRAPH'02*, pages 491–500, 2002.
58. W. Lee, J. Gu, and N. Magnenat-Thalmann. Generating animatable 3d virtual humans from photographs. *Computer Graphics Forum*, 19(3), August 2000.
59. J. P. Lewis, M. Cordner, and N. Fong. Pose space deformations: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 165–172, July 2000.
60. J.-M. Lien and N. M. Amato. Simultaneous shape decomposition and skeletonization using approximate convex decomposition. Technical report, Texas A&M University, 2005. `http://parasol-www.cs.tamu.edu/publications/download.php?file_id=461`.
61. P. Lindstrom and G. Turk. Image-driven simplification. *ACM Trans. Graph.*, 19(3):204–241, 2000.
62. P. Liu, F. Wu, W. Ma, R. Liang, and M. Ouhyoung. Automatic animation skeleton construction using repulsive force field. In *Pacific Graphics 2003*, page 409, october 2003.

63. G. Loy, J. Sullivan, and S. Carlsson. Pose-based clustering in action sequences. In *Proc. Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis, HLK'03*, page 66, 2003.
64. R. MacCracken and K. Joy. Free-form deformations with lattices of arbitrary topology. In *Proc. SIGGRAPH'96*, pages 181–188, 1996.
65. P. Maciel and P. Shirley. Visual navigation of large environments using textured clusters. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 95–ff. ACM Press, 1995.
66. N. Magnenat-Thalmann, R. Laperriere, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Graphics Interface '88*, pages 26–33, June 1988.
67. D. Manocha and Y. Zhu. A fast algorithm and system for inverse kinematics of general serial manipulators. In *IEEE Conference on Robotics and Automation*, 1994.
68. L. Moccozet, F. Dellas, N. Magnenat-Thalmann, S. Biasotti, M. Mortara, B. Falcidieno, P. Min, and R. Veltkamp. Animatable human body model reconstruction from 3d scan data using templates. In *Proc. CapTech Workshop on Modelling and Motion Capture Techniques for Virtual Environments, CAPTECH2004*, 2004.
69. L. Moccozet and N. Magnenat-Thalmann. Dirichlet free-form deformations and their application to hand simulation. In *Proc. Computer Animation, CA'97*, pages 93–102, 1997.
70. L. Moccozet and N. Magnenat-Thalmann. Multilevel deformation model applied to hand simulation. In *Proc. Virtual Systems and MultiMedia, VSMM'97*, pages 119–128, 1997.
71. A. Mohr and M. Gleicher. Building efficient, accurate character skins from examples. *ACM Transactions on Graphics*, 22(3):562–568, July 2003.
72. F. Multon, L. France, M.-P. Cani, and G. Debunne. Computer animation of human walking: a survey. *The Journal of Visualization and Computer Animation*, 10:39–54, 1999.
73. S. Oh, H. Kim, N. Magnenat-Thalmann, and K. Wohn. Generating unified model for dressed virtual humans. *The Visual Computer*, 21(8):522–531, 2005.
74. M. Oliveira, G. Bishop, and D. McAllister. Relief texture mapping. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 359–368. ACM Press/Addison-Wesley Publishing Co., 2000.
75. S. Park and J. K. Hodgins. Capturing and animating skin deformation in human motion. *ACM Trans. Graph.*, 25(3):881–889, 2006.
76. W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.
77. S. Redon, N. Galoppo, and M. Lin. Adaptive dynamics of articulated bodies. In *Proc. SIGGRAPH'05*, pages 936–945, 2005.
78. L. Reveret, L. Favreau, C. Depraz, and M.-P. Cani. Morphable model of quadrupeds skeletons for animating 3d animals. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2005)*, 2005.
79. T. Rhee, U. Neumann, and J. P. Lewis. Human hand modeling from surface anatomy. In *Proc. of the 2006 Symposium on Interactive 3D graphics and games*, 2006.
80. C. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen. Efficient generation of motion transitions using space-time constraints. *Computer Graphics*, 30(Annual Conference Series):147–154, 1996.
81. P. Sand, L. McMillan, and J. Popović. Continuous capture of skin deformation. *ACM Transactions on Graphics*, 22(3):578–586, July 2003.

82. P. Sander, J. Snyder, S. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 409–416. ACM Press, 2001.
83. G. Schaufler. Per-object image warping with layered impostors. In *Proceedings of the 9th Eurographics Workshop on Rendering '98*, pages 145–156, June 1998.
84. F. Scheepers, R. E. Parent, W. E. Carlson, and S. F. May. Anatomy-based modeling of the human musculature. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 163–172, August 1997.
85. H. Seo, F. Cordier, and N. Magnenat-Thalmann. Synthesizing animatable body models with parameterized shape modifications. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Diego, CA, USA, 2003.
86. H. Seo and N. Magnenat-Thalmann. An automatic modeling of human bodies from sizing parameters. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics*, pages 19–26, 2003.
87. F. Sillion, G. Drettakis, and B. Bodelet. Efficient impostor manipulaiton for real-time visualization of urban scenery. In *Proceedings of Eurographics '97*, pages 207–218, September 1997.
88. K. Singh and E. Kokkevis. Skinning characters using surface oriented free-form deformations. In *Proc. Graphics Interface, GI'00*, pages 35–42, 2000.
89. S. Kshirsagar, S. Garchery, and N. Magnenat-Thalmann. Feature point based mesh deformation applied to mpeg-4 facial animation. In *Proceedings Deform'2000, Workshop on Virtual Humans by IFIP Working Group 5.10 (Computer Graphics and Virtual Worlds)*, pages 23–34. Kluwer Academic Publishers, November 2000.
90. P. P. Sloan, C. Rose, and M. Cohen. Shape by example. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics*, 2001.
91. J. Starck, G. Collins, R. Smith, A. Hilton, and J. Illingworth. Animated statues. *Journal of Machine Vision Applications*, 2002.
92. F. Tecchia, C. Loscos, and Y. Chrysanthou. Image-based crowd rendering. *IEEE Computer Graphics & Applications*, 22(2):36–43, 2002.
93. M. Teichmann and S. Teller. Assisted articulation of closed polygonal models. In *Proc. 9th Eurographics Workshop on Animation and Simulation*, pages 87–102, Lisbon, Portugal, August 31 - September 1 1998.
94. D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *Computer Graphics*, 22(4):269–278, 1988.
95. D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, 1987.
96. D. Terzopoulos and A. Witkin. Physically based models with rigid and deformable components. *IEEE Computer Graphics and Applications*, 8(6):41–51, 1988.
97. D. Thalmann, N. Magnenat-Thalmann, and P. Bergeron. Dream flight: a fictional film produced by 3d computer animation. In *Proceedings Computer Graphics'82*, pages 353–368, 1982.
98. D. Thalmann, J. Shen, and E. Chauvineau. Fast realistic human body deformations for animation and vr applications. In *Computer Graphics International 1996*, 1996.
99. D. Tolani, A. Goswami, and N. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models*, 62(5):353–388, 2000.
100. R. Turner and E. Gobbetti. Interactive construction and animation of layered elastically deformable characters. *Computer Graphics Forum*, 17(2):135–152, 1998.
101. P. Volino and N. Magnenat-Thalmann. Comparing efficiency of integration methods for cloth simulation. In *Computer Graphics International, CGI'01*, pages 265–274, 2001.

102. L. Wade and R. E. Parent. Automated generation of control skeletons for use in animation. *The Visual Computer*, 18(2):97–110, March 2002.
103. A. Watt and M. Watt. *Advanced animation and rendering techniques*. Addison-Wesley, 1992.
104. J. Wilhelms. Animals with anatomy. *IEEE Computer Graphics and Applications*, 17(3):22–30, /1997.
105. A. Witkin and M. Kass. Space-time constraints. In *SIGGRAPH'88*, pages 159–168, 1988.
106. W. Wooten and J. Hodgins. Transitions between dynamically simulated motions: Leaping, tumbling, landing, and balancing, 1997. Animation Sketch, Siggraph'97.
107. F.-C. Wu, W.-C. Ma, P.-C. Liou, R.-H Laing, and M. Ouhyoung. Skeleton extraction of 3d objects with visible repulsive force. In *Computer Graphics Workshop 2003, Taiwan*, 2003.
108. S. Yoshizawa, A. G. Belyaev, and H.-P. Seidel. A simple approach to interactive free-form shape deformations. In *Proc. Pacific Graphics, PG'02*, pages 471–474, 2002.
109. S. Yoshizawa, A. G. Belyaev, and H.-P. Seidel. Free-form skeleton-driven mesh deformations. In *Proc. ACM Solid Modeling*, pages 247–253, 2003.
110. X. Zhao. *Kinematic Control of Human Postures for Task Simulation*. PhD thesis, University of Pennsylvania, 1996.
111. V. B. Zordan, B. Celly, B. Chiuand, and P. C. Dilorenzo. Breathe easy: Model and control of human respiration for computer animation. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 29–38, 2004.

A

Colour Plates: Shape Interrogation

(a) (b) (c) (d)

Fig. CP-1.

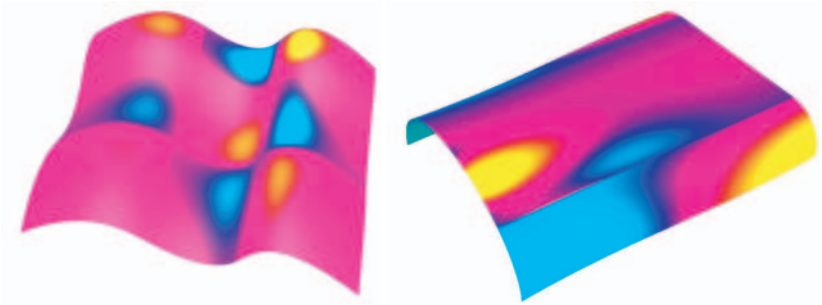


Fig. CP-2.

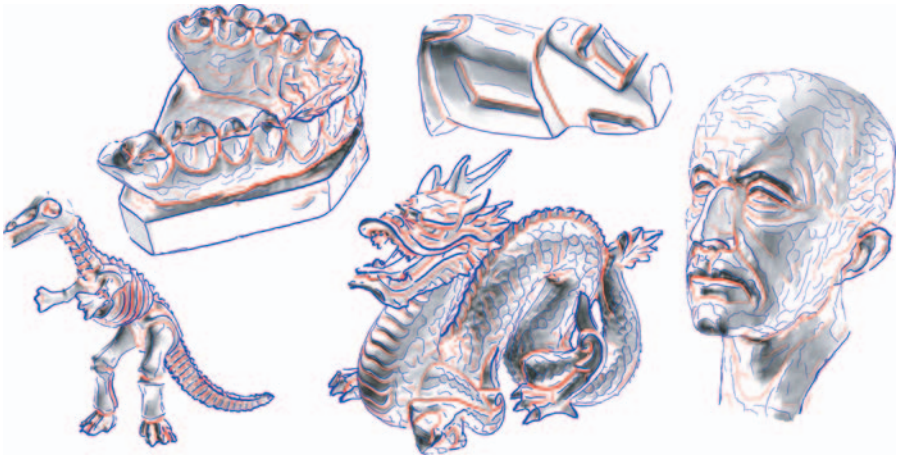


Fig. CP-3.

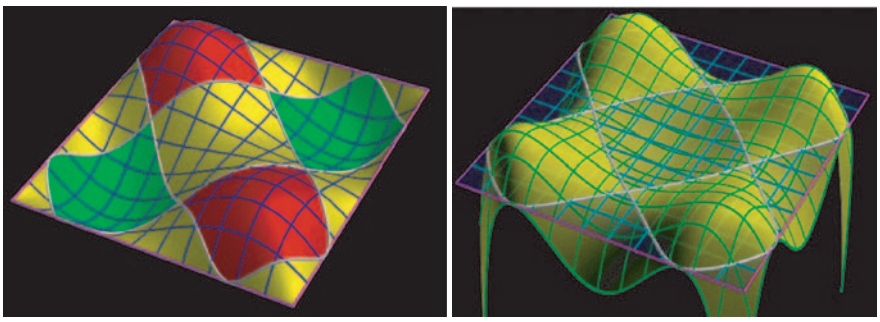


Fig. CP-4.

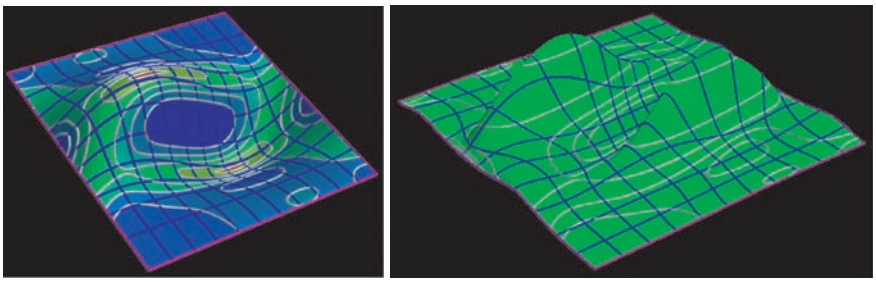


Fig. CP-5.

B

Colour Plates: Recent Advances in Remeshing of Surfaces

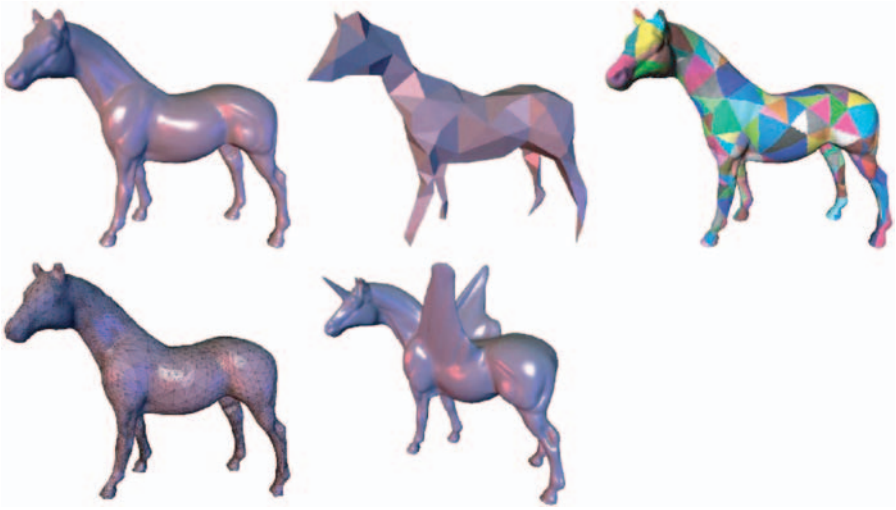


Fig. CP-1.

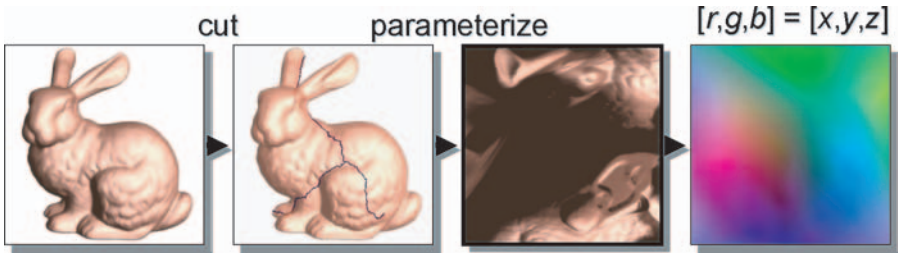


Fig. CP-2.

Fig. CP-3.

C

Colour Plates of the Chapter: Multiresolution Analysis

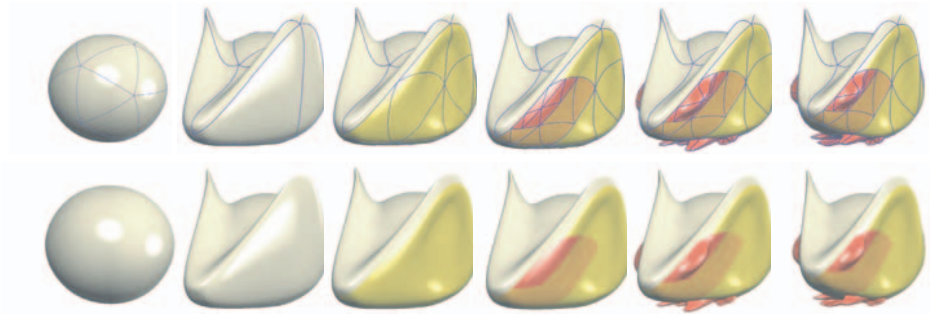


Fig. CP-1.

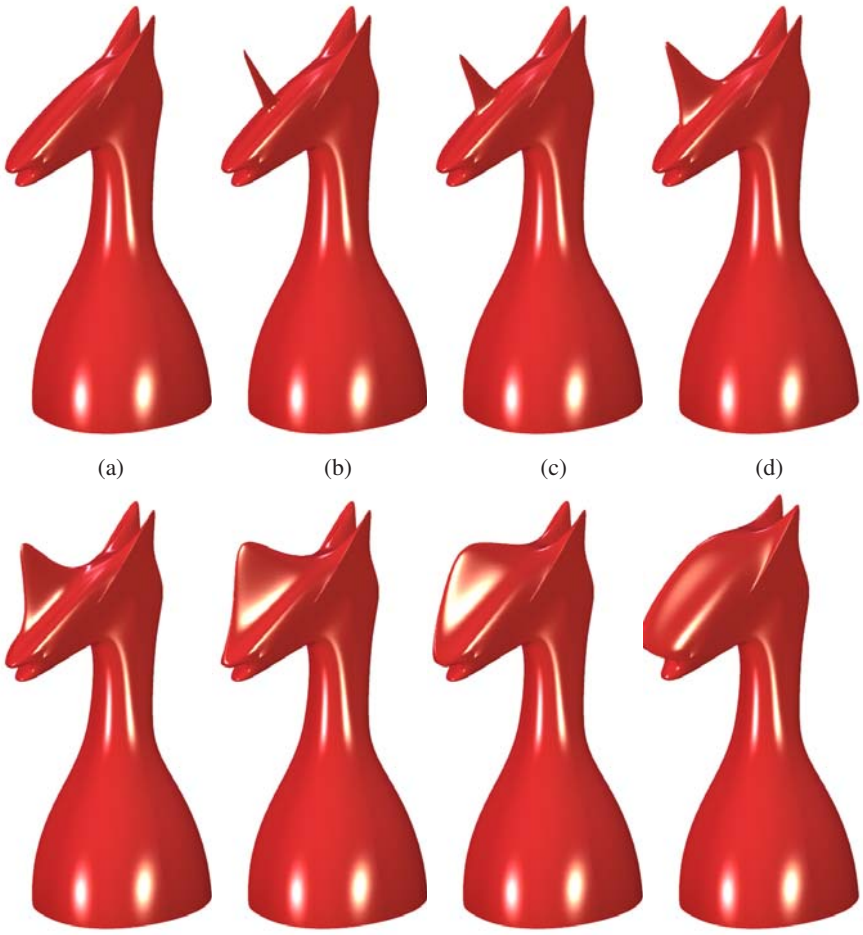


Fig. CP-2.

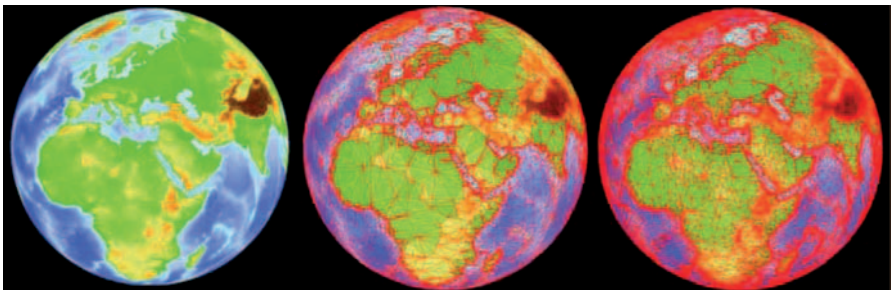


Fig. CP-3.

D

Colour Plates: Subdivision Surfaces and Applications

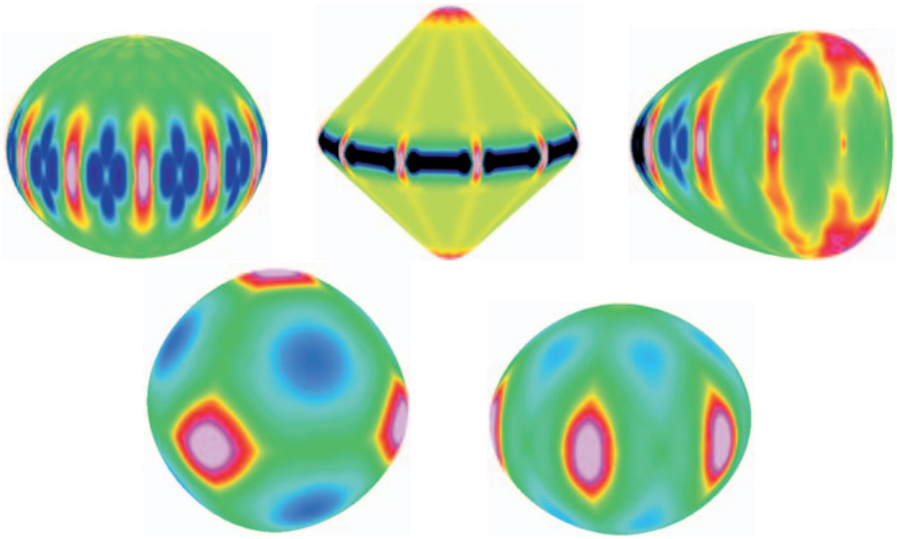
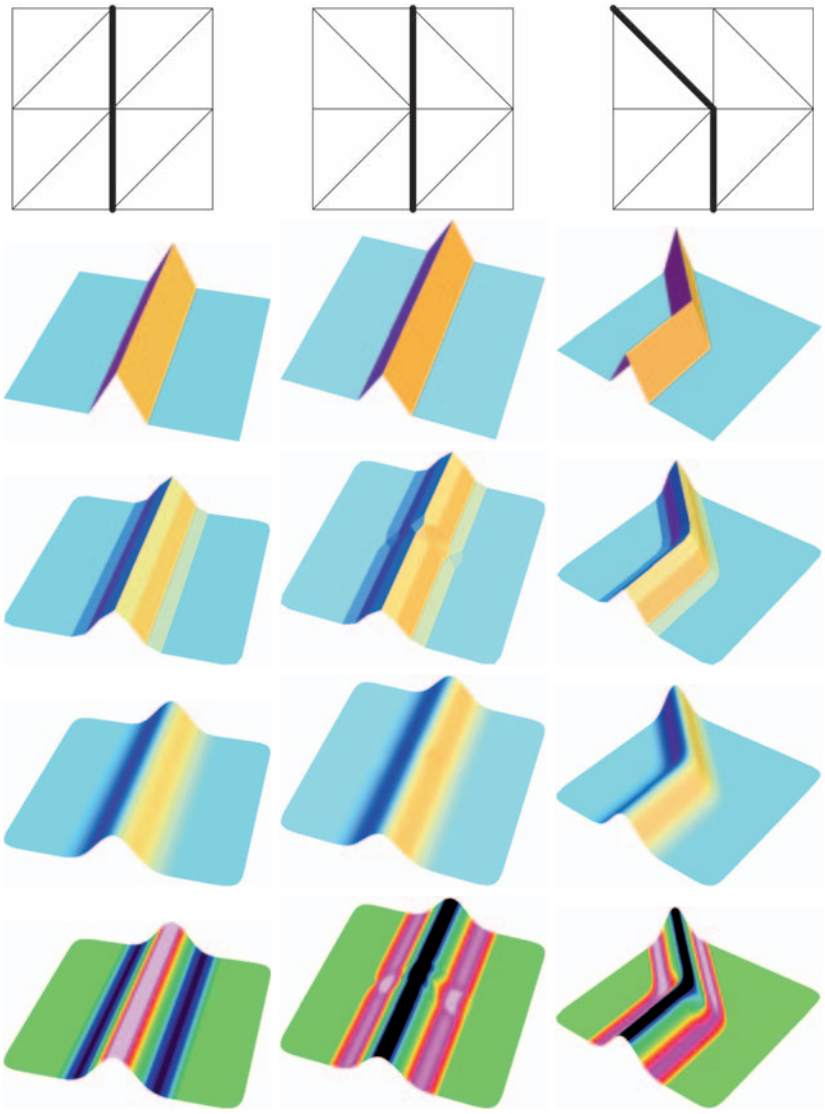


Fig. CP-1.



(a) (b) (c)

Fig. CP-2.

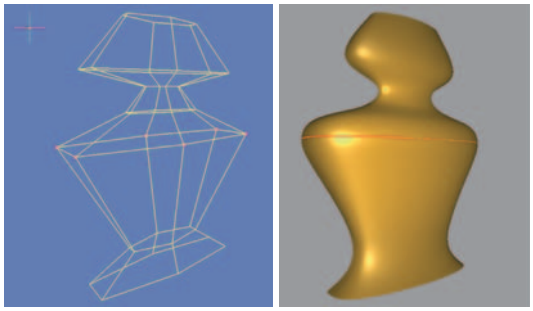


Fig. CP-3.

Fig. CP-4.

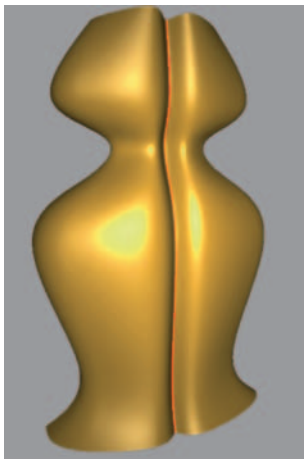


Fig. CP-5.

E

Colour Plates: Skeletal Structures

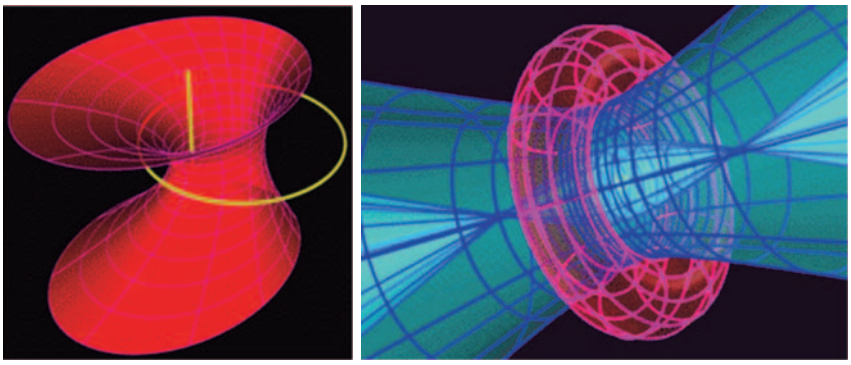


Fig. CP-1.

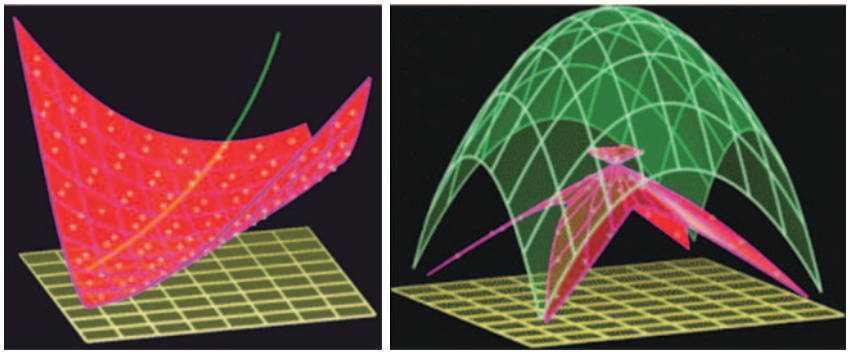
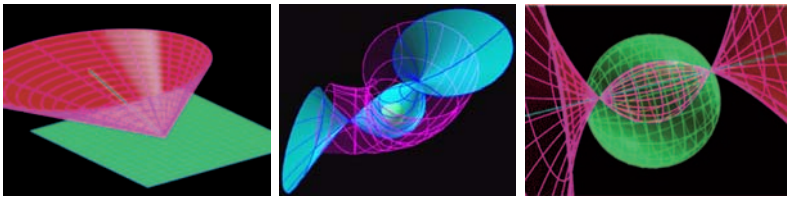
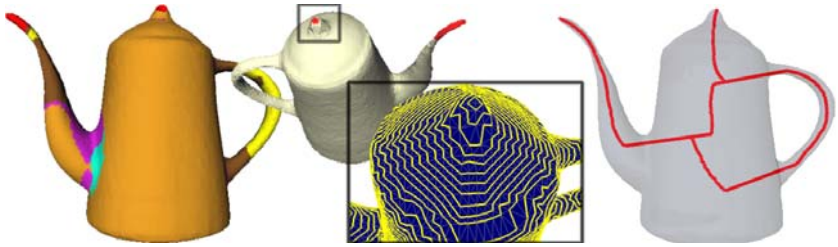


Fig. CP-2.



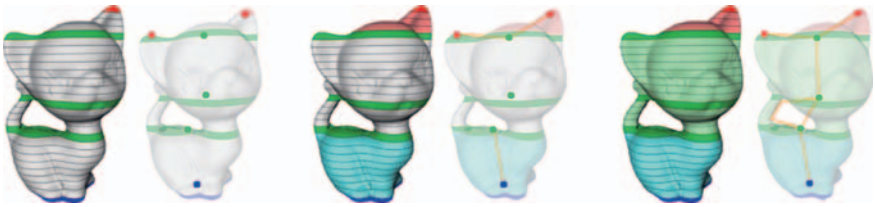
(a) (b) (c)

Fig. CP-3.



(a) (b) (c) (d)

Fig. CP-4.



(a) (b) (c)

Fig. CP-5.

F

Colour Plates: Morphological Representations of Scalar Fields

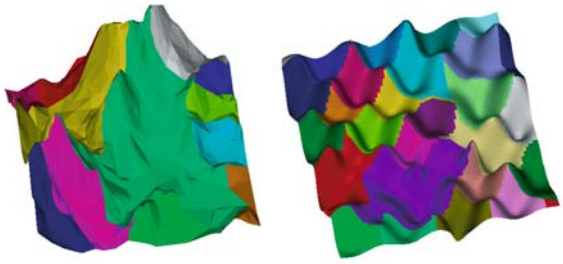
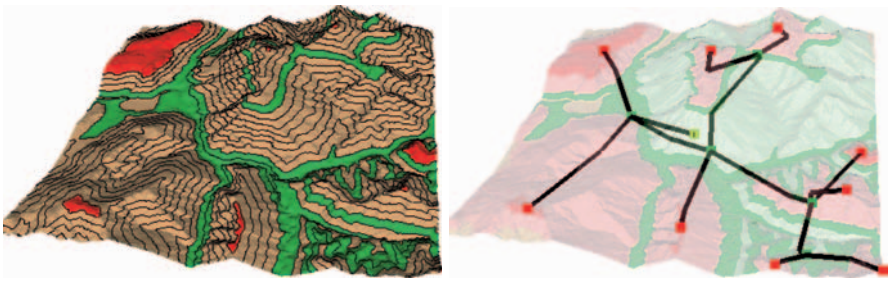


Fig. CP-1.



(a) (b)

Fig. CP-2.

G

Colour Plates: Topological Representations of Vector Fields

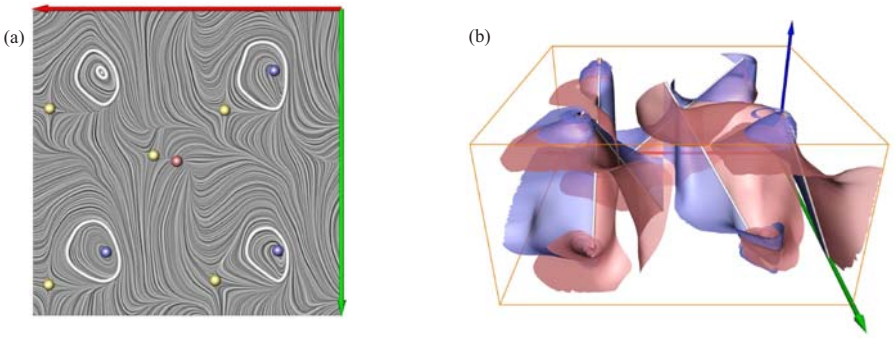


Fig. CP-1.

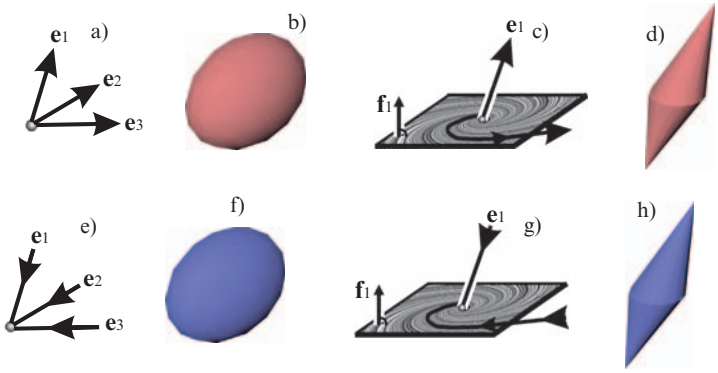


Fig. CP-2.

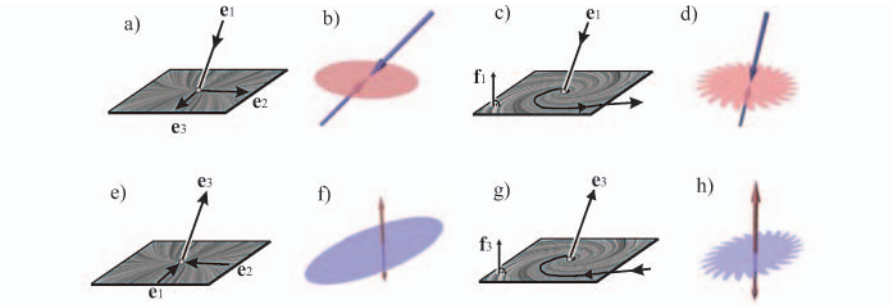


Fig. CP-3.

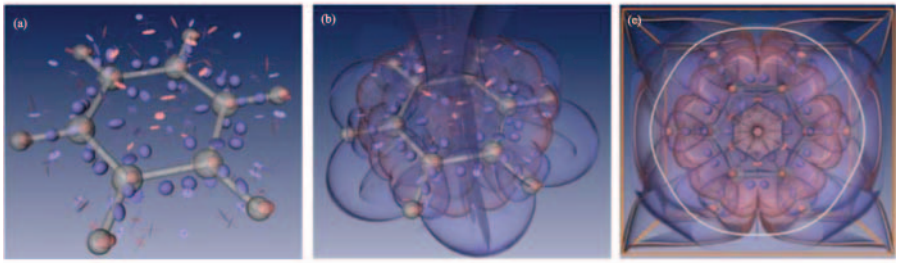


Fig. CP-4.

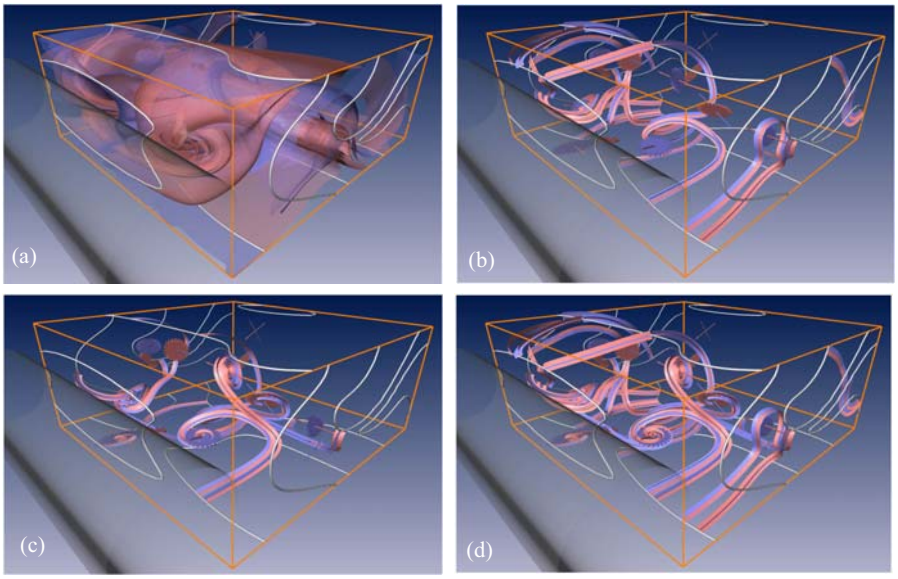


Fig. CP-5.

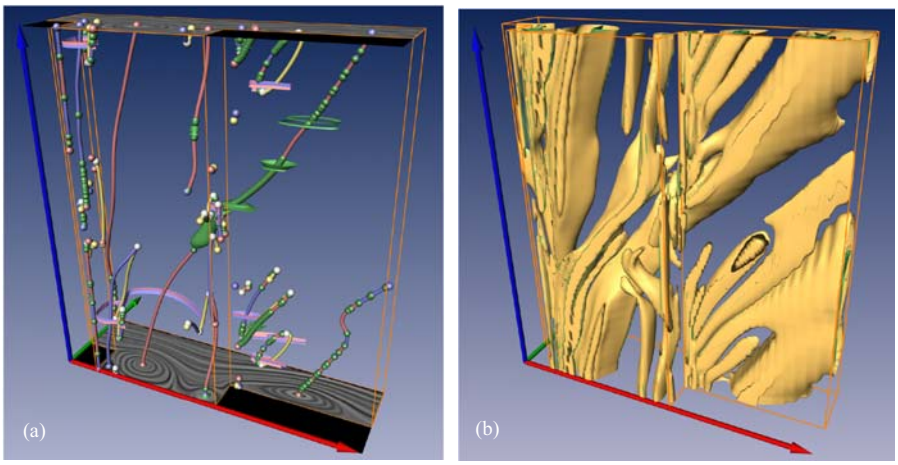


Fig. CP-6.

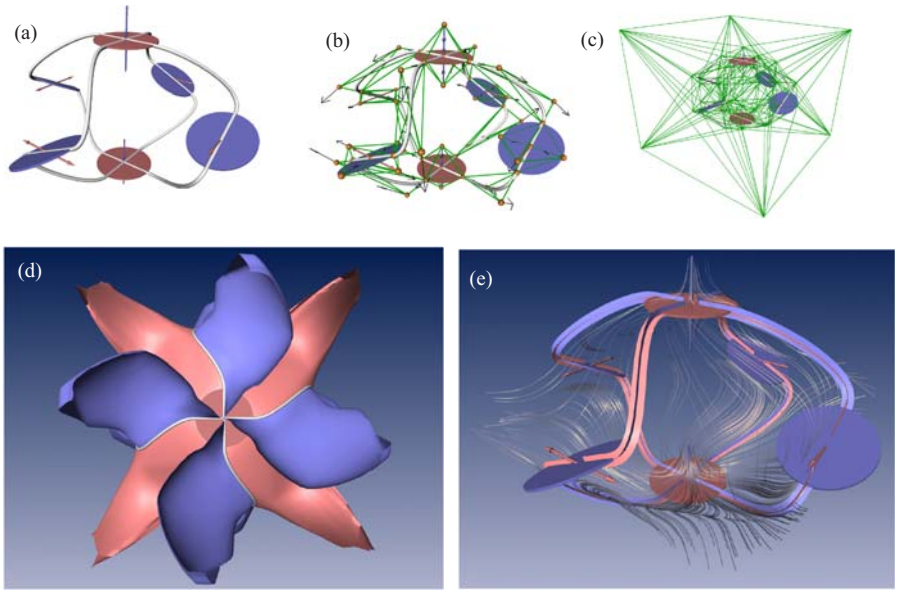


Fig. CP-7.

Index

k-simplex, 174
k-skeleton, 173
boundary-based approach, 183
1-skeleton, 176

advancing front, 57
algorithm
  eigenvalue, eigenvector, 40, 41
aliasing, 45
analysis
  multiresolution, 69
anatomy-based, 231–233, 241, 243
angle-length multiresolution, 86
area constraint, 78
Artifact, 106
  First order artifact, 106
  Second order artifact, 107
attracting saddle, 207
augmented contour tree, 178, 188

Betti numbers, 181
bi-linear constraints
  area, 78
Bisector, 136, 137
  curve-curve bisector, 138
  curve-surface bisector, 136
  point-curve bisector, 138
  point-point bisector, 138
  point-surface bisector, 138
  rational bisectors, 140
  surface-surface bisector, 136
Boundary sample, 146
  ϵ-sample, 146
  κ-light ϵ-sample, 146
  noisy ϵ-sample, 146
boundary switch connector, 211
boundary switch curve, 208
boundary switch point, 204
boundary swtich connector, 210
Butterfly scheme, 103

cancellation, 187
Catmull-Clark scheme, 103
cell complex, 172
Centreline, 132
chamfer triangle, 63
Characteristic map, 105
combinatorial boundary, 173
component tree, 178
component-critical points, 190
contour, 175
contour topology tree, 191
contour tree, 178
critical net, 176
critical point, 175, 180, 202
  first-order, 203
  high-order, 203
  index, 202
  saddle, 207
  sectors, 202
  sink, 207
  source, 207
  tracking, 214
  unstable, 208
Critical Point Configuration Graph, 177
criticality tree, 193
Curvature-based graph, 158
Curve-skeleton, 143

d-manifold, 174
Delaunay tree, 89
detail coefficients, 81
digital model, 174
digital Morse theory, 193
direct manipulation, 74
Discrete Reeb Graph, 160
Distance map, 152
  chessboard distance, 154
  Manhattan distance, 154
  weighted distance, 154
DOF, 226–228, 235, 244, 249, 250
DoF, 225
Doo-Sabin scheme, 103
dynamics, 227, 229, 230, 234, 249

edge sharpener, 63
eigenvalue, 40, 41
eigenvector, 40, 41
elliptic sector, 202
Euclidean cell complex, 173
extended Reeb graph, 191
extraordinary vertex, 47

fairing, 81
fast marching method, 57
filter bank, 80
form
  normal, 40

generalization algorithms, 187
generalization of a Morse-Smale complex, 187
geometry image, 51

Hausdorff distance
  one-sided Hausdorff distance, 144
Hausdorff distance, 144
hierarchical Delaunay triangulations, 89
hierarchical representations, 70
  arbitrary topology, 70
  hierarchical B-splines, 70
  hierarchical triangular splines, 70
hyperbolic sector, 202

index, 202
interpolation, 70
irregular mesh representations
  filter bank algorithm, 91
  multiresolution, 88
  non-nested multiresolution analysis, 89
isocontour, 175

join tree, 189
joint parameterization, 53

kinematics, 227–230, 249

Laurent polynomial, 104
level of detail, 69
Level set, 154
level set, 175
Level set diagram, 156
Linear axis, 149
linear constraints, 77
  normal, 77
  position, 77
  tangency, 77
link, 173
LoD, 225, 245, 247, 249
Loop scheme, 103

Mask, 102
Maximal ball, 152, 153
maximum, 175
Medial Axis Transform, 133, 134
Medial structures, 134
Medial surface, 135
mesh
  finite element, 45
  parameterization, 48
  piecewise regular, 52
  regular, 47
  semi-regular, 47
  valid, 44
metamorphosis, 84
minimum, 175
morphing, 84
Morse complex, 176
Morse function, 175
Morse Lemma, 175
Morse theory, 174
Morse-Smale complex, 176
Morse-Smale function, 176
multiplication map, 39
multiresolution analysis, 69
multiresolution curve, 81

non-linear constraints, 81

Offset, 139, 144

parabolic sector, 202
path line, 212, 217
PN triangle, 46
pose space deformation, 237, 240, 243

Quasi Morse-Smale complex, 183, 184

Reeb graph, 133, 158
  augmented Reeb graph, 161
  extended Reeb graph, 160
refinement, 70
regular grid, 174
regular model, 174
Regular Square Grid, 174
remeshing
  anisotropic, 65
  applications, 44
  definition, 44
  techniques, 46
repelling saddle, 207
root
  multiple, 41
  simple, 40

saddle, 175, 207
saddle connector, 210, 211
Semi-algebraic sets, 140
separatrix, 204, 209
separatrix lines, 182
Shape descriptors, 131
Shock graph, 135
shrink wrap, 50
simplicial complexes, 174
simplicial model, 174
sink, 207
skeleton, 225–228, 231–233, 237, 242, 249, 250
skeleton generation, 237
skeleton-driven deformation, 225, 227, 232–234
Skeletonisation, 142, 152
skinning, 231, 233, 236, 239
slope districts, 177
solver
  algebraic, 40, 41
source, 207
spline
  knot sequence, 72
  wavelet decomposition, 72
split tree, 189
stable Morse complex, 176
star, 173
steam line, 212
Stencil, 102
Straight skeleton, 147
stream line, 202
  separatrix, 204
  tracking, 216
sub-complex, 173
surface mesh representations
  multiresolution, 94
surface network, 176
Surface-skeleton, 143
Sweep-like feature, 121
Symmetry set, 141

tangent plane continuity, 70
Taylor formula, 175
terrain model, 174
Thinning, 142, 150
topology change graph, 193
Triangulated Irregular Network, 174

unstable Morse complex, 176

variational approximation, 66
variational modeling, 86
vector field, 202
  compression, 218
  construction, 222
  simplification, 218
  time dependent, 212
  topological comparison, 220
Vertex
  Extraordinary vertex, 103
  Regular vertex, 103
virtual human, 225, 226, 232, 237, 239, 243, 245, 247, 252
volume constraint, 81
Voronoi cell, 135
Voronoi diagram, 135
Voronoi graph, 145
  λ-Voronoi graph, 147

wavelet
  B-spline, 72

wavelets, 81
  analysis, 81
  multiresolution analysis, 81
  scaling functions, 81
  synthesis, 81
well-shaped triangle, 56

X-configuration, 123