

Mathematics and Visualization

Series Editors

Gerald Farin

Hans-Christian Hege

David Hoffman

Christopher R. Johnson

Konrad Polthier

Martin Rumpf

Neil A. Dodgson
Michael S. Floater
Malcolm A. Sabin

Editors

Advances in Multiresolution for Geometric Modelling

With 163 Figures



Neil A. Dodgson
Computer Laboratory
University of Cambridge
William Gates Building
15 J. J. Thomson Avenue
Cambridge CB3 0FD, UK
e-mail: nad@cl.cam.ac.uk

Michael S. Floater
Computer Science Department
Oslo University
P. O. Box 1053, Blindern
0316 Oslo, Norway
e-mail: michaelf@ifi.uio.no

Malcolm A. Sabin
Numerical Geometry Ltd.
26 Abbey Lane, Lode
Cambridge CB5 9EP, UK
e-mail: malcolm@geometry.demon.co.uk

Library of Congress Control Number: 2004094681

Mathematics Subject Classification (2000): 51M30, 53A25, 51J15, 70B10, 65Y25, 68U05, 14Qxx

ISBN 3-540-21462-3 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2005
Printed in Germany

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typeset by the authors using a Springer \TeX macro package
Production: LE- \TeX Jelonek, Schmidt & Vöckler GbR, Leipzig
Cover design: *design & production* GmbH, Heidelberg

SPIN 10987803 46/3142YL - 5 4 3 2 1 0 - Printed on acid-free paper

Preface

This book marks the culmination of the four-year EU-funded research project, Multiresolution in Geometric Modelling (MINGLE). The book contains seven survey papers, providing a detailed overview of recent advances in the various fields within MINGLE, and sixteen research papers. Each of the seven parts of the book starts with a survey paper, followed by associated research papers in that area.

All papers were originally presented at the MINGLE 2003 workshop held in Cambridge, UK, 9–11 September 2003. Over the course of the three day workshop there were thirty-one presentations covering the whole range of topics within the MINGLE project. From those presentations, this book contains twenty-three papers, one by an invited speaker and the rest by members of the project. All papers have been refereed by an international panel.

Our thanks go to the authors and referees for their hard work, which has made this an excellent collection of recent work in the field. We would also like to thank Martin Peters, Ute McCrory, and Leonie Kunz at Springer-Verlag in Heidelberg and Peggy Glauch at Le-TEx in Leipzig for guiding this book from initial concept to finished product.

Cambridge
March 2004

*Neil Dodgson
Michael Floater
Malcolm Sabin*

The MINGLE project

MINGLE was a European Union Research Training Network with nine partners from six countries. The project ran from January 2000 to January 2004. Its main objectives were: (1) to train young European researchers in various aspects of multiresolution in geometric modelling, and (2) to accelerate the research effort in this area with regard to both theoretical advances and industrial and commercial applications. The first objective was undoubtedly achieved: thirty-four young researchers were funded by the project and over a hundred researchers attended the project's Summer School in August 2001 in Munich¹. There have also been a large number of academic publications arising from the project, showing that the second objective has been met; this book bears just some of the fruit of the research undertaken over the course of the project.

The research topics of the MINGLE project included: thinning of triangle meshes; coding of triangle meshes; remeshing; multiresolution deformation; hierarchical meshes; data structures for hierarchical and nested triangulations; subdivision surfaces; wavelets in geometric modelling; and surface parameterization.

The nine MINGLE partners were:

- SINTEF Applied Mathematics, Oslo, Norway
- Tel-Aviv University, Israel
- Munich University of Technology, Germany
- Israel Institute of Technology (TECHNION), Haifa, Israel
- Max-Planck-Institut für Informatik, Saarbrücken, Germany
- Laboratoire de Modélisation et Calcul, Université Joseph Fourier, Grenoble, France
- Computer Laboratory, University of Cambridge, Cambridge, UK
- Department of Computer and Information Sciences (DISI), University of Genova, Italy
- Systems in Motion AS, Oslo, Norway

¹ The tutorial lectures given at the Summer School formed the basis for the companion volume *Tutorials on Multiresolution in Geometric Modelling*, A. Iske, E. Quak, and M. S. Floater (eds.), Springer-Verlag, 2002, ISBN 3-540-43639-1.

Sponsors

We would like to thank our sponsors for their support. The European Union funded the MINGLE project as a Fifth Framework Research Training Network (Contract Number HPRN-CT-1999-00117). Emmanuel College, Cambridge, UK was the venue for the MINGLE 2003 workshop, 9–11 September 2003, and it provided extra funding to allow PhD students from outside the project to attend the workshop.

Referees

We would like to thank our panel of referees, including those who chose to remain anonymous, for their efforts and their attention to detail. Each paper was refereed by at least one senior member of the MINGLE project and at least one expert external to the project. We would like to extend special thanks to Carsten Mönnig for his help in administering the refereeing process.

Internal referees

Pierre Alliez
 Georges-Pierre Bonneau
 Neil Dodgson
 Nira Dyn
 Michael Floater
 Craig Gotsman
 Stefanie Hahmann
 Armin Iske
 Ioannis Ivrissimtzis
 Leif Kobbelt
 David Levin
 Jürgen Prestin
 Enrico Puppo
 Ewald Quak
 Malcolm Sabin

External referees

Lyuba Alboul
 Richard Bartels
 Rick Beatson
 Peer-Timo Bremer
 Stephan Dahlke
 Wolfgang Dahmen
 Gershon Elber
 Stefan Gumhold
 Igor Guskov
 Øyvind Hjelle
 Hugues Hoppe
 Kai Hormann
 Alain Le Méhauté
 Burkhard Lenze
 Charles Loop
 Tom Lyche
 Alan Middleditch
 Knut Mørken
 Jörg Peters
 Gerlind Plonka-Hoch
 Ulrich Reif
 Alla Sheffer
 Joachim Stöckler
 Gabriel Taubin
 Joe Warren
 Zoë Wood

Contents

Part I — Compression

| | |
|---|----|
| Recent Advances in Compression of 3D Meshes <i>Pierre Alliez, Craig Gotsman</i> | 3 |
| Shape Compression using Spherical Geometry Images <i>Hugues Hoppe, Emil Praun</i> | 27 |

Part II — Data Structures

| | |
|---|-----|
| A Survey on Data Structures for Level-of-Detail Models <i>Leila De Floriani, Leif Kobbelt, Enrico Puppo</i> | 49 |
| An Algorithm for Decomposing Multi-dimensional Non-manifold Objects into Nearly Manifold Components <i>M. Mostefa Mesmoudi, Leila De Floriani, Franco Morando, Enrico Puppo</i> | 75 |
| Encoding Level-of-Detail Tetrahedral Meshes <i>Neta Sokolovsky, Emanuele Danovaro, Leila De Floriani, Paola Magillo.</i> .. | 89 |
| Multi-Scale Geographic Maps <i>Raquel Viaña, Paola Magillo, Enrico Puppo</i> | 101 |

Part III — Modelling

| | |
|--|-----|
| Constrained Multiresolution Geometric Modelling <i>Stefanie Hahmann, Gershon Elber</i> | 119 |
|--|-----|

| | |
|---|-----|
| Multi-scale and Adaptive CS-RBFs for Shape Reconstruction from Clouds of Points | |
| <i>Yutaka Ohtake, Alexander Belyaev, Hans-Peter Seidel</i> | 143 |
| <hr/> | |
| Part IV — Parameterization | |
| Surface Parameterization: a Tutorial and Survey | |
| <i>Michael S. Floater, Kai Hormann</i> | 157 |
| Variations on Angle Based Flattening | |
| <i>Rhaleb Zayer, Christian Rössl, Hans-Peter Seidel</i> | 187 |
| <hr/> | |
| Part V — Subdivision | |
| Recent Progress in Subdivision: a Survey | |
| <i>Malcolm Sabin</i> | 203 |
| Optimising 3D Triangulations: Improving the Initial Triangulation for the Butterfly Subdivision Scheme | |
| <i>Nurit Alkalai, Nira Dyn</i> | 231 |
| Simple Computation of the Eigencomponents of a Subdivision Matrix in the Fourier Domain | |
| <i>Loïc Barthe, Cédric Gérot, Malcolm Sabin, Leif Kobbelt</i> | 245 |
| Subdivision as a Sequence of Sampled C^p Surfaces | |
| <i>Cédric Gérot, Loïc Barthe, Neil A. Dodgson, Malcolm Sabin</i> | 259 |
| Reverse Subdivision | |
| <i>Mohamed F. Hassan, Neil A. Dodgson</i> | 271 |
| $\sqrt{5}$-subdivision | |
| <i>Ioannis P. Ivrissimtzis, Neil A. Dodgson, Malcolm Sabin</i> | 285 |
| Geometrically Controlled 4-Point Interpolatory Schemes | |
| <i>Martin Marinov, Nira Dyn, David Levin</i> | 301 |
| <hr/> | |
| Part VI — Thinning | |
| Adaptive Thinning for Terrain Modelling and Image Compression | |
| <i>Laurent Demaret, Nira Dyn, Michael S. Floater, Armin Iske</i> | 319 |
| Simplification of Topologically Complex Assemblies | |
| <i>Carlos Andújar, Marta Fairén, Pere Brunet, Víctor Cebollada</i> | 339 |

| | |
|---|-----|
| Topology Preserving Thinning of Vector Fields on Triangular Meshes | |
| <i>Holger Theisel, Christian Rössl, Hans-Peter Seidel</i> | 353 |

Part VII — Wavelets

| | |
|--|-----|
| Periodic and Spline Multiresolution Analysis and the Lifting Scheme | |
| <i>Jürgen Prestin, Ewald Quak</i> | 369 |
| Nonstationary Sibling Wavelet Frames on Bounded Intervals: the Duality Relation | |
| <i>Laura Beutel</i> | 391 |
| Haar Wavelets on Spherical Triangulations | |
| <i>Daniela Roşca</i> | 405 |

Part I

— Compression

Recent Advances in Compression of 3D Meshes

Pierre Alliez¹ and Craig Gotsman²

¹ INRIA, Sophia-Antipolis, France

pierre.alliez@sophia.inria.fr

² Technion, Haifa, Israel

gotsman@cs.technion.ac.il

Summary. 3D meshes are widely used in graphical and simulation applications for approximating 3D objects. When representing complex shapes in raw data format, meshes consume a large amount of space. Applications calling for compact storage and fast transmission of 3D meshes have motivated the multitude of algorithms developed to compress these datasets efficiently. In this paper we survey recent developments in compression of 3D surface meshes. We survey the main ideas and intuition behind techniques for single-rate and progressive mesh coding. Where possible, we discuss the theoretical results obtained for asymptotic behaviour or optimality of the approach. We also list some open questions and directions for future research.

1 Introduction

The emerging demand for visualising and simulating 3D geometric data in networked environments has motivated research on representations for such data. Slow networks require data compression to reduce the latency, and progressive representations to transform 3D objects into streams manageable by the networks. We distinguish between single-rate and progressive compression techniques depending on whether the model is decoded during, or only after, the transmission. In the case of single-rate lossless coding, the goal is to remove the redundancy present in the original description of the data. In the case of progressive compression the problem is more challenging, aiming for the best trade-off between data size and approximation accuracy (the so-called rate-distortion trade-off). Lossy single-rate coding may also be achieved by modifying the data set, making it more amenable to coding, without losing too much information. These techniques are called *remeshing*.

Sect. 2 gives some basic definitions for surface meshes. Sect. 3 surveys recent algorithms for single-rate compression, and Sect. 4 surveys recent techniques for progressive compression.

2 Basic Definitions

The specification of a polygon surface mesh consists of combinatorial entities: vertices, edges, and faces, and numerical quantities: attributes such as vertex positions, normals, texture coordinates, colours, etc. The *connectivity* describes the incidences between elements and is implied by the topology of the mesh. For example, two vertices or two faces are adjacent if there exists an edge incident to both. The *valence* of a vertex is the number of edges incident to it, and the *degree* of a face is the number of edges incident to it (see Fig. 1). The *ring* of a vertex is the ordered list of all its incident faces. The total number of vertices, edges, and faces of a mesh will be denoted V , E , and F respectively.

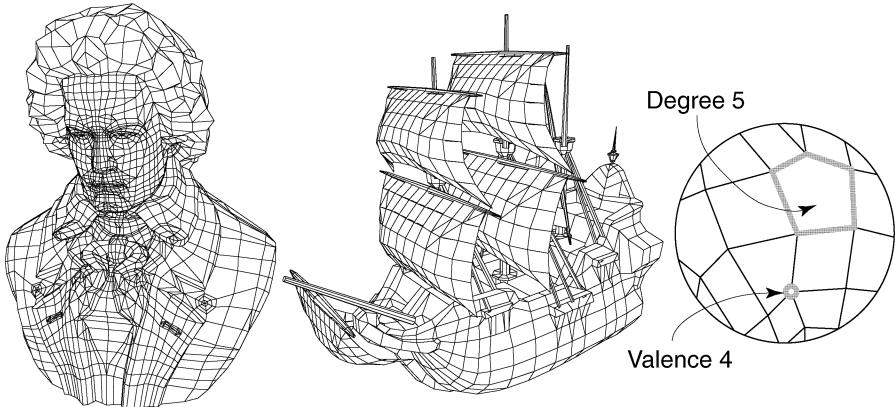


Fig. 1. Examples of polygon meshes. Left: Beethoven mesh (2812 polygons, 2655 vertices). Middle: Galleon mesh (2384 polygons, 2372 vertices). Right: close-up of a polygon mesh – the *valence* of a vertex is the number of edges incident to this vertex, while the *degree* of a face is the number of edges enclosing it.

3 Single-rate Compression

We classify the techniques into two classes:

- Techniques aiming at coding the original mesh without making any assumption about its complexity, regularity or uniformity. This also includes techniques specialised for massive datasets, which cannot fit entirely into main memory. Here we aim at restoring the original model after decoding (for carefully designed models or applications where lossy compression is intolerable).
- Techniques which remesh the model before compression. The original mesh is considered as just one instance of the shape geometry.

3.1 Triangle Meshes

The triangle is the basic geometric primitive for standard graphics rendering hardware and for many simulation algorithms. This partially explains why much of the work in the area of mesh compression prior to 2000 has been concerned with triangle meshes only. The *Edgebreaker* coder [52] gives a worst-case bound on the connectivity compression bit rate of 4 bits per vertex. Besides the popular *Edgebreaker* and its derivatives [39, 17, 59, 30], two techniques transform the connectivity of a triangle mesh into a sequence of valence codes [62, 28], which can automatically benefit from the low statistical dispersion around the average valency of 6 when using entropy encoding. This is achieved either through a deterministic conquest [62] or by a sequence of half edge collapses [28]. In [62], Touma and Gotsman proposed the conquest approach and compress the connectivity down to less than 0.2 bit per vertex (b/v) for very regular meshes, and between 2 and 3.5 b/v otherwise, in practice. The so-called conquest consists of conquering the edges of successive pivot vertices in an orientation-consistent manner and generating valence codes for traversed vertices. Three additional codes: *dummy*, *merge* and *split* are required in order to encode boundaries, handles and conquest incidents respectively. The *dummy* code occurs each time a boundary is encountered during the conquest; the number of *merge* codes is equal to the genus of the mesh being encoded. The *split* code frequency is linked mainly to the mesh irregularity. Intuitively, if one looks at the coding process as a conquest along a spiralling vertex tree, the *split* codes thus indicate the presence of its branching nodes. The *Mesh Collapse Compression* scheme by Isenburg and Snoeyink [28] performs a sequence of edge contractions until a single vertex remains in order to obtain bit rates of 1 to 4 b/v. For a complete survey of these approaches, we refer the reader to [14].

One interesting variant on the Edgebreaker technique is the Cut-Border Machine (CBM) of Gumhold and Strasser [18]. The main difference is that the CBM encodes the split values as a parameter like the valence based schemes. This makes an upper bound on the resulting code more difficult to establish (although there is a bound of 5 b/v in [17]), but on the other hand allows for single pass coding and decoding. This difference is significant for coding massive data sets.

3.2 Non-triangle Meshes

Compared with triangle meshes, little work has been dedicated to the harder problem of connectivity coding of 2-manifold graphs with arbitrary face degrees and vertex valences. There are a significant number of non-triangular meshes in use, in particular those carefully designed, e.g. the high-quality 3D models of the Viewpoint library [65] contain a surprisingly small proportion of triangles. Likewise, few triangles are generated by tessellation routines in existing modelling software. The dominant element in these meshes is the

quadrilateral, but pentagons, hexagons and higher degree faces are also common.

The performance of compression algorithms is traditionally measured in bits per vertex (b/v) or bits per edge (b/e). Some early attempts to code general graphs [63, 34], which are the connectivity component of a geometric mesh, led to rates of around 9 b/v. These methods are based on building interlocking spanning trees for vertices and faces. Chuang et al. [7] later described a more compact code using canonical ordering and multiple parentheses. They state that any simple 3-connected planar graph can be encoded using at most $1.5 \log_2(3)E + 3 \simeq 2.377$ bits per edge. Li and Kuo [48] proposed a so-called “dual” approach that traverses the edges of the dual mesh³ and outputs a variable length sequence of symbols based on the type of a visited edge. The final sequence is then coded using a context based entropy coder. Isenburg and Snoeyink coded the connectivity of polygon meshes along with their properties in a method called *Face Fixer* [29]. This algorithm is gate-based, the gate designating an oriented edge incident to a facet that is about to be traversed. A complete traversal of the mesh is organised through successive gate labelling along an active boundary loop. As in [62, 52], both the encoder and decoder need a stack of boundary loops. Seven distinct labels F_n , R, L, S, E, H_n and $M_{i,k,l}$ are used in order to describe the way to *fix* faces or holes together while traversing the current active gate. King et al. [40], Kronrod and Gotsman [42] and Lee et al. [46] also generalised existing methods to quad, arbitrary polygon and hybrid triangle-quad meshes respectively. However, none of these polygon mesh coders come close to the bit rates of any of the best, specialised coders [62, 3] when applied to the special case of a triangle mesh. At the intuitive level, given that a polygon mesh with the same number of vertices contains fewer edges than a triangle mesh, it should be possible to encode it with fewer bits. These observations motivated the design of a better approach to code the connectivity of polygonal meshes.

The Degree/Vалence Approach

Since the Touma-Gotsman (TG) valence coder [62] is generally considered to have the best performance, it seems natural to try to generalise it to arbitrary polygon meshes. This was done independently by Khodakovsky et al. [35] and Isenburg [23]. The generalisation relies on the key concept of *duality*. Consider an arbitrary 2-manifold triangle graph \mathcal{M} . Its dual graph $\tilde{\mathcal{M}}$, in which faces are represented as dual vertices and vertices become dual faces (see Fig. 2), should have the same connectivity information since dualisation neither adds nor removes information. The valences of $\tilde{\mathcal{M}}$ are now all equal to 3, while the face degrees take on the same values as the vertex valences of \mathcal{M} . Since a list of all 3s has zero entropy, coding just the list of degrees of $\tilde{\mathcal{M}}$ would lead to the same bit rate as found for the valences of \mathcal{M} . Conversely, if a polygon mesh has only valence-3 vertices, then its dual would be a triangle mesh.

³ See Fig. 2 for an illustration of a dual mesh.

Hence, its entropy should be equal to the entropy of the list of its degrees. This observation leads to the key idea of the degree/valence approach: the compression algorithm should be *self-dual*, in the sense that both a mesh and its dual are coded with the same number of bits. A direct consequence of this is that the coding process should be symmetric in the coding of valences and degrees. A second direct consequence is that the bit rate of a mesh should be measured in *bits per edge* (b/e), since the number of edges is the only variable not changing during a graph dualisation. This contrasts with the former practice of measuring the coding efficiency for triangle meshes in bits/vertex.

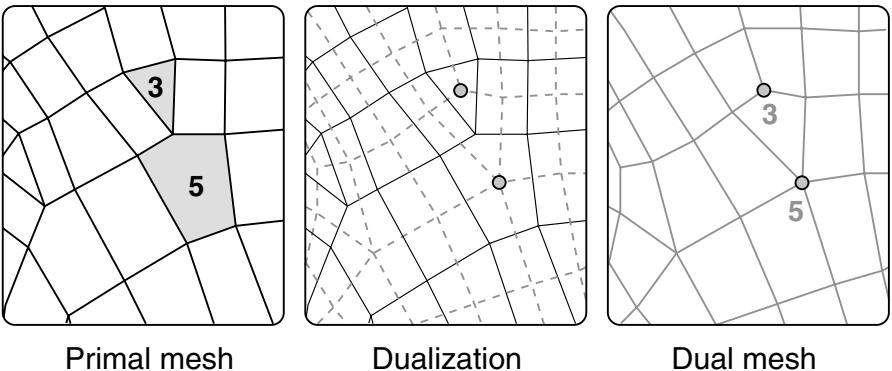


Fig. 2. Left: a polygon mesh with highlighted faces of degree 3 and 5. Middle: the dual mesh is built by placing one node in each original face and connecting them through each edge incident to two original faces. Right: the dual mesh now contains corresponding vertices of valence 3 and 5.

The core technique underlying the algorithm described in [23, 35] is similar to most connectivity compression methods: a seed element is chosen and all its neighbours are traversed recursively until all elements of the corresponding connected component are “conquered”. A new seed element of the next connected component is then chosen and the process continues. Every time the encoder conquers the next element of the mesh, it outputs some symbol which uniquely identifies a new state. From this stream of symbols, the decoder can reconstruct the mesh. Various coding algorithms differ in the way they traverse the mesh and in the sets of symbols used for identifying the encoder state. During the mesh traversal of [23, 35], two sets of symbols are generated to code vertex valences and face degrees using an entropy encoder. At any given moment in time, both encoder and decoder know with which type of symbol (face or vertex) they are dealing.

While the valence and degree sequences of a mesh dominate the mesh code, they are not sufficient to uniquely characterise it. As in [62], some extra “split”, and possibly other symbols may be required during the mesh conquest. To minimise the occurrence of such symbols – hence improve the

compression ratios – both techniques [23, 35] drive the traversal by various heuristics inspired from the valence-driven approach [3]. To better exploit correlation between streams and between symbols within each stream, it is possible to use a *context-based* arithmetic coder.

3.3 Connectivity: Entropy and Optimality

The entropy is a measure of the information content of a set of symbols, equipped with a probability distribution. It is thus the minimal average *number of bits per symbol* required for lossless encoding of a sequence of symbols from the set, each appearing with frequency given by its probability:

$$\text{entropy} = \sum_{i=1}^N p_i \log_2 \frac{1}{p_i}. \quad (1)$$

When the probability is not specified, this means that all symbols are equiprobable. When coding the connectivity of a mesh using entropy coding of its valences as introduced by Touma and Gotsman [62] for the case of triangular meshes, the bit-rates obtained are mostly dictated by the distribution of the valences. This automatically benefits from the *regularity* in the mesh. A triangle mesh is perfectly regular when the valence of all vertices is 6. The vertex valence distribution then has an entropy of zero. Later work, mainly generalisations of the Edgebreaker technique, developed methods to take explicit advantage of mesh regularity, and their performance has been shown to scale with this measure [39, 16, 59]. In [35], Khodakovsky et al. discuss the optimality of their valence/degree approach and show that the entropy of both the valence and degree sequences is no more than the entropy of the class of planar graphs as established by Tutte in the sixties [64]. Gotsman [13] later showed that the precise entropy of the valence and degree sequences is actually strictly less, but not by much, than the Tutte entropy, and the difference is made up by the split commands. Hence the number of split commands, albeit very small, is not negligible.

Entropy and constructive enumeration. Given a finite class of discrete elements, all equally probable, the entropy e of the class is the logarithm of the number of elements in the class. Obviously, the best possible performance of any algorithm coding this class of elements is to use at most e bits to encode one arbitrary element in the class. Hence, the issue of optimal coding of a class is equivalent to the one of constructive enumeration [41]. Poulalhon and Schaeffer [51] have described a provably optimal coder for connectivity of meshes homeomorphic to a sphere, using a bijection between a triangulation and a Schnyder tree decomposition (i.e. a constructive enumeration of the connectivity graph). Although effective bounds are obtained, the code lengths do not adapt to the mesh regularity (every mesh consumes the same number of bits, whatever the distribution of valences). An objective of theoretical interest is

to add flexibility to these methods in order to benefit from mesh regularity. Another obvious extension is to obtain similar results for high genus and non-triangular graphs.

3.4 Geometry Compression

Although the geometry data is often given in precise floating point representation for representing vertex positions, some applications may tolerate the reduction of this precision in order to achieve higher compression rates. The reduction of the precision involves *quantisation*. The resulting values are then typically compressed by entropy coding after *prediction* relying on some data smoothness assumptions.

Quantisation. The early works usually quantise the vertex positions uniformly for each coordinate separately in Cartesian space [10, 61, 62], and a more sophisticated vector quantisation has also been proposed by Lee and Ko [45]. Karni and Gotsman [33] have also demonstrated the relevance of applying quantisation in the space of spectral coefficients (see [14] for more details on this approach). In their elegant work, Sorkine et al. [57] address the issue of reducing the visual effect of quantisation errors. Building on the fact that the human visual system is more sensitive to normal than to geometric distortion, they propose to apply quantisation not in the coordinate space as usual, but rather in a transformed coordinate space obtained by applying a so-called “ k -anchor invertible Laplacian transformation” over the original vertex coordinates. This concentrates the quantisation error at the low-frequency end of the spectrum, thus preserving the fine normal variations over the surface, even after aggressive quantisation (see Fig. 3). To avoid significant low-frequency errors, a set of anchor vertex positions are also selected to “nail down” the geometry at a select number of vertex locations.

Prediction. The early work employed simple delta coding [10] or linear prediction along a vertex ordering dictated by the coding of the connectivity [61, 62]. The approach proposed by Lee et al. [46] consists of quantising *in the angle space* after prediction. By applying different levels of precision while quantising the dihedral or the internal angles between or inside each facet, this method achieves better visual appearance by allocating more precision to the dihedral angles since they are more related to the geometry and normals. Inspired by the Touma-Gotsman (TG) parallelogram prediction scheme [62], Isenburg and Alliez [25] complete the techniques described in [23, 35] by generalising it to polygon mesh geometry compression. The polygon information dictates where to apply the parallelogram rule used to predict the vertex positions. Since polygons tend to be fairly planar and fairly convex, it is more appropriate to predict within polygons rather than across them. Intuitively, this idea avoids poor predictions resulting from a crease angle between polygons.

Despite the effectiveness of the published predictive geometry schemes, they are not optimal because the mesh traversal is dictated by the connec-

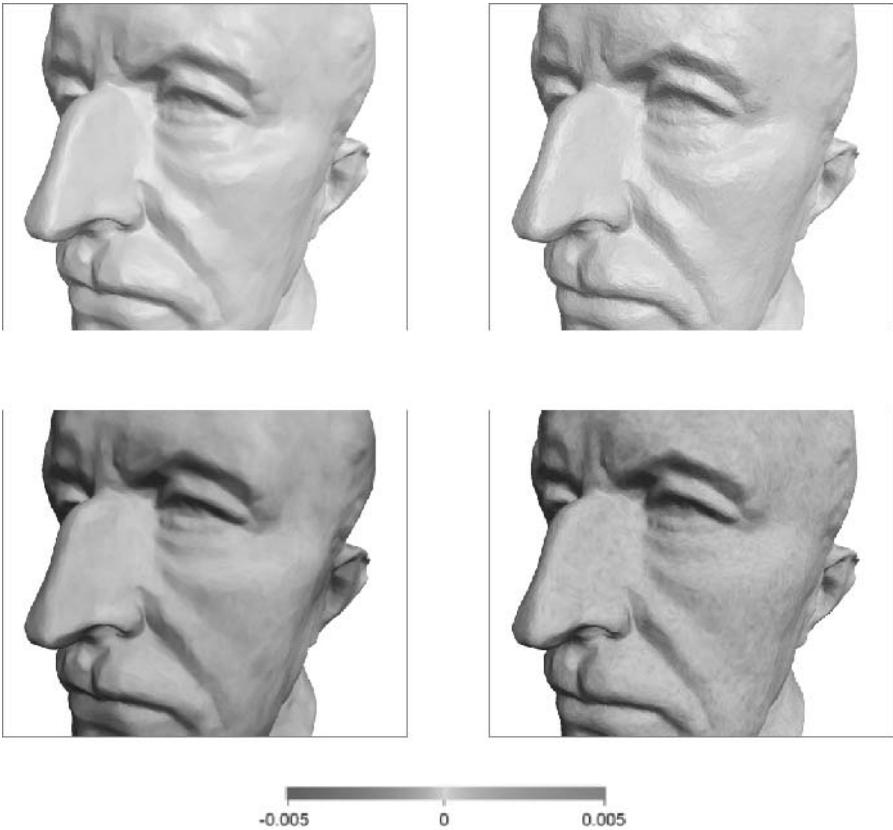


Fig. 3. [Reproduced in colour in Plate 1.] The delta-coordinate quantisation to 5 bits/coordinate (left) introduces low-frequency errors to the geometry, whereas Cartesian coordinate quantisation to 11 bits/coordinate (right) introduces noticeable high-frequency errors. The upper rows shows the quantised model and the bottom figures use colour to visualise corresponding quantisation errors. Data courtesy O. Sorkine.

tivity scheme. Since this traversal order is independent of the geometry, and prediction from one polygon to the next is performed along this, it cannot be expected to do the best job possible. A first approach to improve the prediction was the prediction trees [43], where the geometry drives the traversal instead of the connectivity as before. This is based on the solution of an optimisation problem. In some case it results in a decrease of up to 50% in the geometry code entropy, in particular in meshes with significant creases and corners, such as CAD models. Cohen-Or et al. [9] suggest a multi-way prediction technique, where each vertex position is predicted from all its neighbouring vertices, as opposed to the one-way parallelogram prediction. An extreme approach to

prediction is the feature discovery approach by Shikhare et al. [56], which removes the redundancy by detecting similar geometric patterns. However, this technique works well only for a certain class of models and involves expensive matching computations.

3.5 Optimality of Spectral Coding

Karni and Gotsman [33] showed that the eigenvectors of the Laplacian matrix derived from the mesh connectivity graph may be used to transform code the three Cartesian geometry n -vectors (x, y, z). The eigenvectors are ranked according to their respective eigenvalues, which are analogous to the notion of frequency in Fourier analysis. Smaller eigenvalues correspond to lower frequencies. Karni and Gotsman showed empirically that when projected on these basis vectors, the resulting projection coefficients decrease rapidly as the frequency increases. Hence, similarly to traditional transform coding, a good approximation to the geometry vectors may be obtained by using just a linear combination of a small number of basis vectors. The code for the geometry is then just this small number of coefficients (quantised appropriately). While this method seems to work quite well, and intuitively it seems that the Laplacian is a linear operator which captures well the smoothness of the mesh geometry relative to the mesh neighbour structure, there was no proof that this is the optimal basis for this purpose. The only indication that this might be the case is that in the case of a regular mesh, the eigenvectors of the Laplacian are the well-known 2D Fourier basis, which is known to be optimal for common classes of signals [20].

Ben-Chen and Gotsman [5] have imposed a very natural probability distribution on the class of meshes with a given connectivity, and then used principal component analysis (also known as the Karhunen-Loeve transform) to derive the optimal basis for that class. A series of statistical derivations then shows that this basis is identical to the eigenvectors of the symmetric Laplacian of the given connectivity (the sparse matrix whose diagonal is the vertex valence sequence, a negative unit entry for an edge, and zero otherwise). While this is a very interesting result, it remains theoretical, since computation of the Laplacian eigenvectors is still considered too expensive to be practical.

3.6 Coding Massive Data Sets

Due to their size and complexity, massive datasets [47] require dedicated algorithms since existing mesh compression are effective only if the representation of the mesh connectivity and geometry is small enough to fit “in-core”. For large polygonal models that do not fit into main memory, Ho et al. [21] propose cutting meshes into smaller pieces that can be encoded in-core. They process each piece separately, coding the connectivity using the Edgebreaker coder, and the vertex positions using the TG parallelogram linear predictor. Additional information required to stitch the pieces back together after decoding

is also recorded, leading to bit-rates 25% higher than the in-core version of the same compression algorithm. A recent out-of-core technique introduced by Isenburg and Gumhold [27] makes several improvements upon [21] by (i) avoiding the need to break the model *explicitly* into several pieces, (ii) decoding the entire model in a single pass without any restarts, and (iii) streaming the entire mesh through main memory with a small memory footprint. The key technique underlying this compression method consists of building a new external memory data structure – the *out-of-core mesh* – in several stages, all of them being restricted to clusters and active traversal fronts which fit in-core. The latter traversal order, consisting of a reordering of the mesh primitives, is computed in order to minimise the number of memory cache misses, similar in spirit to the notion of a “rendering sequence” [6] developed for improving performance of modern graphics cards, but at a much larger scale. The resulting compressed mesh format can stream very large meshes through the main memory by providing the compressor transparent access to a so-called *processing sequence* that represents a mesh as a fixed, yet seamless interleaved ordering of indexed triangles and vertices. At any point in time, the remaining part of the mesh data is kept on disk.

3.7 Remeshing for Single-rate Geometry Compression

The majority of mesh coders adapt to the regularity and the uniformity of the meshes (with the noticeable exception of [12] that adapts to the non-uniformity). Therefore, if the application allows lossy compression, it is prudent to exploit the existing degrees of freedom in the meshing process to transform the input into a mesh with high regularity and uniformity. Recent work produces either (i) piecewise regular meshes by using the subdivision paradigm, or (ii) highly regular remeshing by local mesh adaptation, or (iii) perfectly regular remeshing by surface cutting and global parameterization.

Szymczak et al. [60] first split the mesh into relatively flat patches with smooth boundaries. Six axis-aligned vectors (so-called defining vectors) first determine some reference directions. From these vectors a partition of the mesh is built with a set of patches whose normals do not deviate more than a prescribed threshold. An approximation of the geodesic distance using Dijkstra’s algorithm is then used in combination with a variant of the farthest point Voronoi diagram to smooth the patch boundaries. Each patch is then resampled by mutual tessellation over a regular hexagonal grid, and all the original vertices, but the boundary ones, are removed by half edge collapses (see Fig. 4). The connectivity of the resulting mesh is encoded using a version of Edgebreaker optimised for regular meshes, and vertex positions are compressed using differential coding and separation of tangential and normal components.

Attene et al. [4] tile the input mesh using isosceles “triangloids”. From each boundary edge of the tiling process, they compute a circle centred on

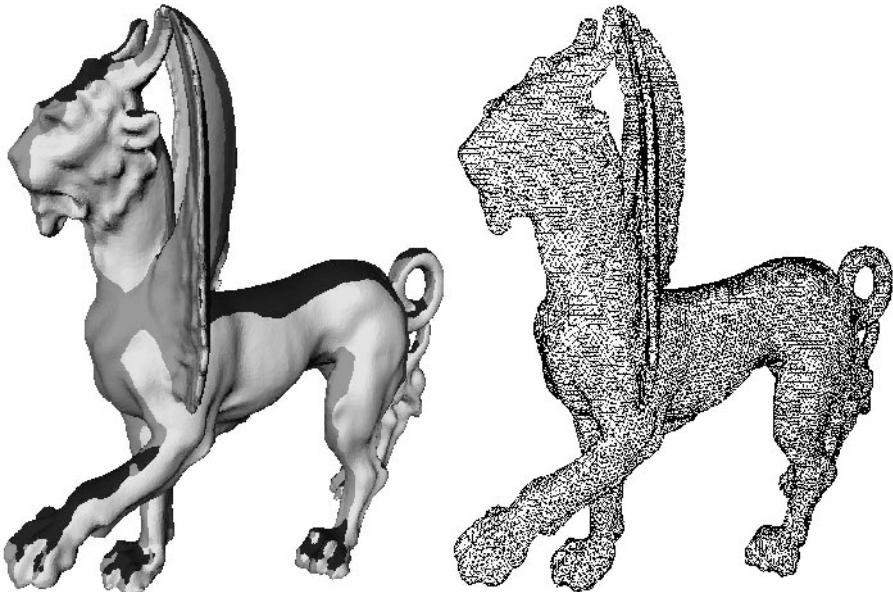


Fig. 4. [Reproduced in colour in Plate 2.] Piecewise regular remeshing (data courtesy A. Szymczak).

the edge mid-point and lying in the bisecting plane between the two edge vertices. The location where the circle pierces the original mesh determines the tip vertex of the newly created triangloid tile. The original mesh is this way wrapped, the regions already discovered being identified as the triangles lying inside the regions bounded by geodesic paths between the three vertices of the new tile. Connectivity of the new mesh is compressed by Edgebreaker, while geometry is compressed by entropy coding one dihedral angle per vertex, after quantisation.

Surazhsky and Gotsman [58] generate a triangle mesh with user-controlled sample distribution and high regularity through a series of atomic Euler operators and vertex relocations applied locally. A density function is first specified by the user as a function of the curvature onto the original mesh. This mesh is kept for later reference to the original surface geometry, and the mesh adaptation process starts on a second mesh, initialised to a copy of the original mesh. The vertex density approaches the prescribed ideal density by local decimation or refinement. A new area-based smoothing technique is then performed to isotropically repartition the density function among the mesh vertices. A novel component of the remeshing scheme is a surprisingly efficient algorithm to improve the mesh regularity. The high level of regularity is obtained by performing a series of local edge-flip operations as well as some edge-collapses and edge-splits. The vertices are first classified as black, regular or white according to their valence deficit or excess (respectively < 6 , $= 6$ and > 6). The

edges are then classified as regular, long, short, or drifting according to their vertex colours (regular if both vertices are regular, long if both are white, short if both are black and drifting if bi-coloured). Long edges are refined by edge-split, and short edges are removed by edge-collapse until only drifting edges remain. The drifting edges have the nice property that they can migrate through regular regions of the mesh by edge-flips without changing the repartition of the vertex valences. Improving the mesh regularity thus amounts to applying a sequence of drifting-edge migrations until they meet irregular vertices, and then have a chance to generate short or long edges whose removal becomes trivial. As a result the models are better compressed using the TG coder which benefits from the regularity in mesh connectivity and geometry.

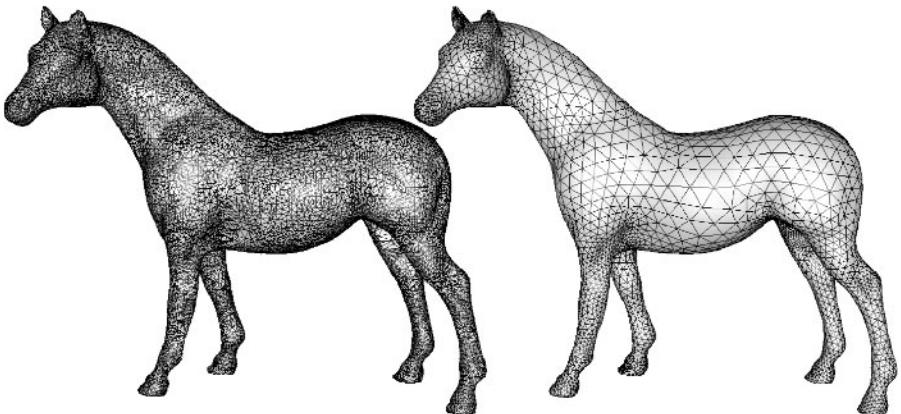


Fig. 5. Highly regular remeshing (data courtesy V. Surazhsky and C. Gotsman).

Gu et al. [15] propose a technique for completely regular remeshing of surface meshes using a rectangular grid. Surfaces of arbitrary genus must be cut to reduce them to a surface which is homeomorphic to a disc, then parameterized by minimising a geometric-stretch measure [53], and finally represented as a so-called *geometry image* that stores the geometry, the normals and any attributes required for visualisation purposes. Such a regular grid structure is compact and drastically simplifies the rendering pipeline since all cache indirections found in usual irregular mesh rendering are eliminated. Besides its appealing properties for efficient rendering, the regular structure allows direct application of “pixel-based” image-compression methods. The authors apply wavelet-based coding techniques and compress separately the topological sideband due to the cutting. After decoding, the topological sideband is used to fuse the cut and ensure a proper welding of the surface throughout the cuts. Despite its obvious importance for efficient rendering, this technique reveals a few drawbacks due to the inevitable surface cutting: each geometry image has to be homeomorphic to a disk, therefore closed or $\text{genus} > 0$ models

have to be cut along a cut graph to extract either a polygonal schema [15] or an atlas [54]. Finding a “smart” cut graph (i.e. minimising a notion of distortion) is a delicate issue and introduces a set of artificial boundary curves, associated pairwise. These boundaries are later sampled as a set of curves (i.e. 1-manifolds) and therefore generate a visually displeasing seam tree. Another drawback comes from the fact that the triangle or the quad primitives of the newly generated meshes have neither orientation nor shape consistent with approximation theory, which makes this representation not fully optimised for efficient geometry compression as reflected in the rate-distortion trade-off.

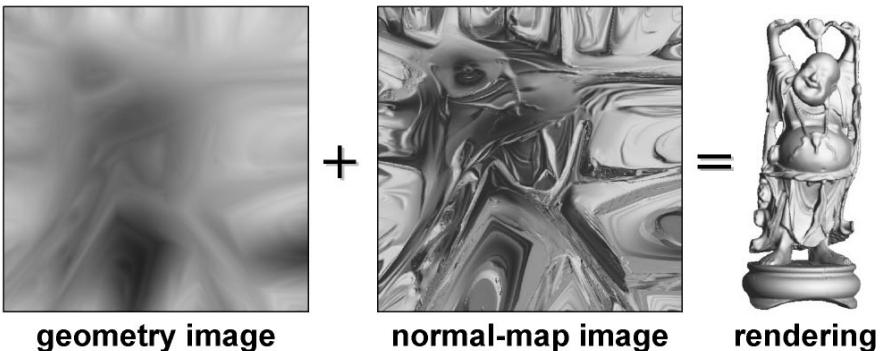


Fig. 6. [Reproduced in colour in Plate 3.] Geometry image (data courtesy X. Gu).

3.8 Comparison and Discussion

A recent trend in mesh connectivity compression is generalisation from triangle meshes to general polygon meshes, with arbitrary genus and boundaries. Adapting to the regularity of the mesh, i.e. the dispersion in the distribution of valences or degrees, is usually reflected in the coding schemes. Semi-regularity being a common property of “real-world” meshes, this is a very convenient feature.

On the theoretical side, the bit-rates achievable by degree/valence connectivity coders have been shown to approach the Tutte entropy lower bound. Because of some remaining “split” symbols, whose number has not been bounded, some additional work has to be done in order to design truly optimal polygon mesh coders which also adapt to regularity. In particular, the connectivity coder of Poulalhon and Schaeffer [51] for triangle meshes offers some promise for extension to polygonal models. As for volume meshes, although some recent work has demonstrated a generalisation of the valence coder to hexahedral meshes [24], nothing has been proven concerning the optimality of this approach.

Most of the previous work has studied the coding of geometry as dictated by the connectivity code, the vertex positions being predicted in an order dictated by the connectivity coder. This happens even though the geometry component dominates the code sizes, so the result will tend to be suboptimal. One attempt to change this was to make the coding of the geometry cooperate with the coding of the connectivity, using prediction trees [43] or multi-way prediction techniques [9]. Other work [57] compresses the geometry globally, showing that applying quantisation in the space of Laplacian transformed coefficients, instead of in the usual space of Cartesian coordinates, is very useful. In a way, the latter is an extension of the multi-way approach since it amounts to predicting each vertex as the barycentre of its neighbours. More recent work [5] aims to find an optimal basis best suited to decorrelate the geometric signal.

Isenburg et al. provide an on-line implementation of the degree/valence coder for bench marking purposes [26]. Isenburg also demonstrates an ASCII-based compression format for web applications able to achieve bit-rates within 1 to 2 percent of those of the binary benchmark code [31].

In order to benefit most from the adaptation of a coding scheme to regularity or uniformity in the input mesh, recent work advocates highly (or even completely) regular remeshing without distorting the geometry too much. In particular, the geometry images [15] technique demonstrates the efficiency of modern image compression techniques when applied to geometry which has been remeshed in a completely regular manner.

A more recent trend takes the remeshing paradigm further, with the design of efficient meshes for approximation of surfaces [1]. This leads to anisotropic polygon meshes, that “look like” carefully designed meshes. The efficiency of such a scheme is expressed in terms of error per number of geometric primitives. The question that now naturally arises is whether the remeshing process should be influenced by the mesh compression scheme used, namely, should it remesh in a manner that suits the coder best. Since rapid progress in the direction of efficient surface meshing is emerging, it seems that it will certainly motivate new approaches for dedicated single-rate mesh compression schemes.

4 Progressive Compression

Progressive compression of 3D meshes uses the notion of refinement: the original mesh is transformed into a sequence (or a hierarchy) of refinements applied to a simple, coarse mesh. During decoding the connectivity and the geometry are reconstructed incrementally from this stream. The main advantage of progressive compression is that it provides access to intermediate states of the object during its transmission through the network (see Fig. 7). The challenge then consists of rebuilding a least distorted object at all points in time during the transmission (i.e. optimisation of rate-distortion trade-off).

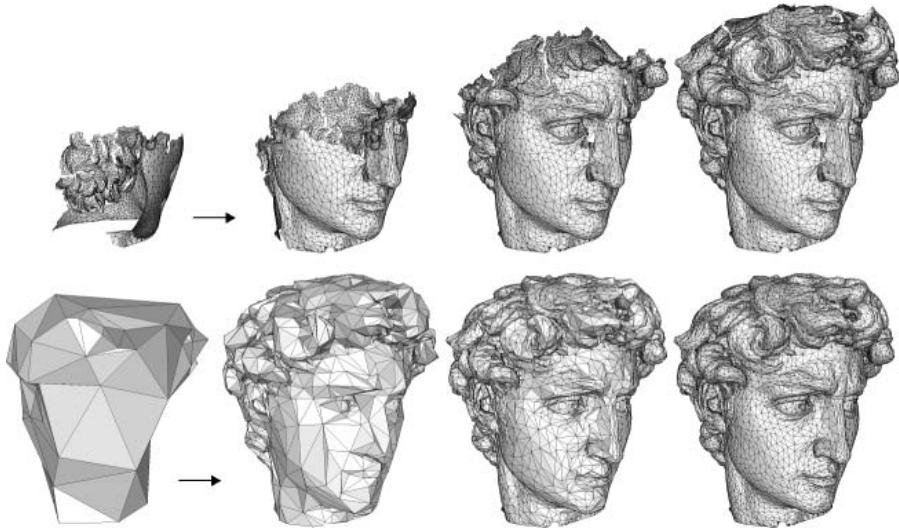


Fig. 7. [Reproduced in colour in Plate 4.] Intermediate stages during the decoding of a mesh using a single-rate (top) or a progressive technique (bottom).

4.1 General Techniques

We call lossless the methods that restore the original mesh connectivity and geometry once the transmission is complete. This is even though intermediate stages are obviously lossy. These techniques mostly proceed by decimating the mesh while recording the (minimally redundant) information required to reverse this process. The basic ingredients behind most of progressive mesh compression techniques are (i) the choice of an atomic mesh decimation operator, (ii) the choice of a geometric distance metric to determine the elements to be decimated, and (iii) an efficient coding of the information required to reverse the decimation process (i.e. to refine the mesh). At the intuitive level, one has to encode for the decoder both the location of the refinement (“where” to refine) and the parameters to perform the refinement itself (“how” to refine).

The progressive mesh technique introduced by Hoppe [22] transforms a triangle surface mesh into a stream of refinements. During encoding the input mesh undergoes a sequence of *edge collapses*, reversed during decoding as a sequence of *vertex splits*. The symbols generated provide the *explicit* location of each vertex being split and a designation of two edges incident to this vertex. This is a very flexible, but rather expensive code. In order to reduce the bit consumption due to the explicit vertex location, several researchers have proposed to specify these locations implicitly, using *independent sets* defined on the mesh. This approach improves the compression ratios, at the price of additional constraints during decimation (the decimation sequence cannot be arbitrary). Pajarola and Rossignac [49] group some edge collapses into a series of independent sets, each of them corresponding to a level of

detail. The location of each vertex to decimate is done by a 2-colouring of the mesh vertices, leading to 1 bit per vertex, for each set. Experimental results show an amortised cost of 3 bits per vertex for vertex location for all sets, plus the cost of local refinements inverting the edge collapses, leading to 7.2 bits per vertex. Cohen-Or et al. [8] define an alternation of 4- and 2-colouring over the triangles in order to locate an independent set of vertices to decimate. A local, deterministic retriangulation then fills the holes generated by vertex removal at no cost, leading to 6 bits per vertex.

Observing the local change of repartition of valences when removing a vertex, Alliez and Desbrun [2] improved the previous approaches by generating an alternation of independent sets composed of patches centred on the vertices to be removed. Each independent set thus corresponds to one decimation pass. The even decimation passes remove valence ≤ 6 vertices, while the odd ones remove only valence 3 vertices. Such a selection of valences reduces the dispersion of valences during decimation, the latter dispersion being further reduced by a deterministic patch retriangulation designed to generate valence-3 vertices, later removed by odd decimation passes. This way the decimation is coordinated with the coding, and for “progressively regular” meshes the decimation generates a regular inverse $\sqrt{3}$ -subdivision, and coding one valence per vertex is sufficient to rebuild the connectivity. For more general meshes some additional symbols are necessary. The latter approach can be seen as a progressive version of the TG coder [62].

Using the edge collapse as the atomic mesh decimation operator, Karni et al. [32] build a sequence of edge collapses arranged along a so called “vertex sequence” that traverses all the mesh vertices. The mesh traversal is optimised so that the number of jumps between two non-incident vertices is minimised. The decoding process is this way provided with an access to the mesh triangles optimised for efficient rendering using the modern vertex buffers. Compression rates are similar to the progressive valence approach [2], with the additional benefit of fast rendering.

4.2 Geometry-driven Coding

For progressive compression of a discrete point set in arbitrary dimension, Devillers and Gandois [11] decompose the space into a kD -tree and transmit only a set of point occurrences, i.e. the number of points located in each cell of the kD -tree hierarchy (see Fig. 8). They demonstrate that transmitting only occurrences of points during successive space subdivision is enough to reconstruct in a progressive manner – and lossless in the end – a discrete point set. The compression achieved by this is due to *bit sharing* intrinsic to the notion of transmission of occurrences, rather than transmission of explicit point locations. For example, transmitting the information “300 points” located in one cell at the beginning of the transmission process is equivalent to *sharing* the first high-level bits of 300 points, simultaneously. Some compression gain due to the sharing is thus obtainable for all cells containing more than one

point. More precisely, the more populated the cell, the higher the compression gain due to the bit-sharing principle. When all points are *separated* – each cell containing only one point – the bits subsequently used are equivalent to bit-plane coding for progressively refining the point locations in space.

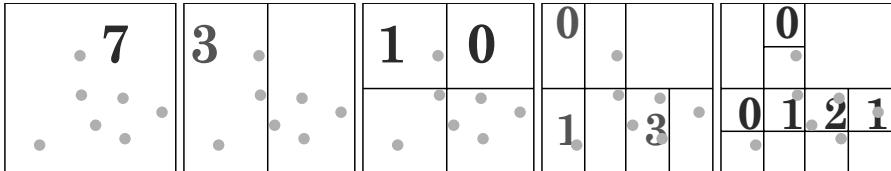


Fig. 8. The geometry coder on a two-dimensional example. The number of points located in each cell of the 2D-tree hierarchy is encoded (data courtesy P.-M. Gandonin).

During the decoding process, the only available information corresponds to the number of occurrences in each cell, i.e. a progressively refined location of the positions. At the end of the transmission, and if one does not care about any particular ordering of the original points, the original information has been restored in a lossless manner since every point is correctly located in a cell that corresponds to the initial precision over the points. Compared to a plain enumeration of the point coordinates, the information of the order over the points is lost. This is precisely what saves some bits for compression. It is proven that these savings are never less than $\log n - 2.402$ bits per point. Moreover, in this case – and contrary to other prediction-based methods that benefit from a uniform sampling – the uniform distribution corresponds to the worst-case scenario for compression since it minimises the possibility of bit-sharing.

The approach described in [11] codes only a set of discrete points. The authors have shown how a geometric triangulation (e.g. Delaunay) of the points allows progressive transmission of a meshed surface. More recently, Devillers and Gandonin [12] have adapted this technique for progressive coding of simplicial complexes (possibly non-manifold) by using the edge collapse operator for coding the connectivity. Contrary to other methods, the connectivity coding process is driven by the geometry alone.

4.3 Remeshing for Progressive Geometry Compression

When the original mesh is considered as one instance of the surface geometry that is to be compressed, *geometry compression* has to be considered rather than mesh compression. To this end, geometry compression techniques proceeding by *semi-regular remeshing* are among the best reported to date.

The main idea behind semi-regular remeshing techniques [38, 19, 36, 50] is to consider a mesh representation as having three components: geometry,

connectivity and parameterization, and assume that the last two components (connectivity and parameterization) are not important for the representation of the geometry. The common goal of these approaches is therefore to reduce these two components as much as possible. This is achieved through semi-regular remeshing of an input irregular mesh, and efficient compression of the newly generated model. The remesher proceeds by building a semi-regular mesh hierarchy designed to best approximate the original geometry. An irregular base mesh, homeomorphic (i.e. topologically equivalent) to the original mesh, is first built by mesh simplification. This base mesh constitutes the coarsest level in the semi-regular hierarchy. The hierarchy is then built by regular or adaptive subdivision (typically by edge bisection) of the base mesh. In the case of *regular* subdivision by edge bisection of a triangle base mesh, all new vertices have valence 6. The finest level of the mesh hierarchy is therefore built from patches of regular vertices, separated by (possibly irregular) vertices from the base mesh. In the case of *adapted* subdivision, some irregular vertices may be generated by adding a few conforming edges (note that this choice can be decided on the decoder side, depending if it cares about reconstructing the adaptivity pattern or not). We now describe how the connectivity and the parametric components are reduced:

- *Reducing the connectivity component.* The regularity intrinsic to the subdivision process deliberately removes almost all of the connectivity information from the mesh since much of the resulting vertices have valence 6 for triangle meshes, and valence 4 for quad meshes.
- *Reducing the parametric component.* In a semi-regular mesh hierarchy generated by subdivision for purpose of geometric approximation, we use the term *detail* to describe the differential vector component stored at each newly inserted vertex during the construction of the mesh hierarchy. The normal component of the detail coefficients stores the geometric information, whereas the tangential component carries the parametric information. Experiments show that by doing things right almost all of the parametric components can be “predicted”, i.e. removed from the representation. Intuitively, this means that sliding a vertex in the tangent plane does not modify the local geometry.

The way the parametric component is reduced is the main distinction between the two compression methods described in this section. The first method [38] uses local frames and different quantisation of normal/tangential components, whereas the second *normal mesh* compression method [36] is specifically designed to produce detail coefficients with no tangential components. Normal meshes were introduced by Guskov et al. [19] as a new way to represent geometry. A normal mesh is a multiresolution representation where almost all the details lie in the local normal direction and hence the mesh geometry is described by a single scalar per vertex instead of three as usual (see Fig. 9). Beyond the remeshing algorithm, both progressive geometry coding

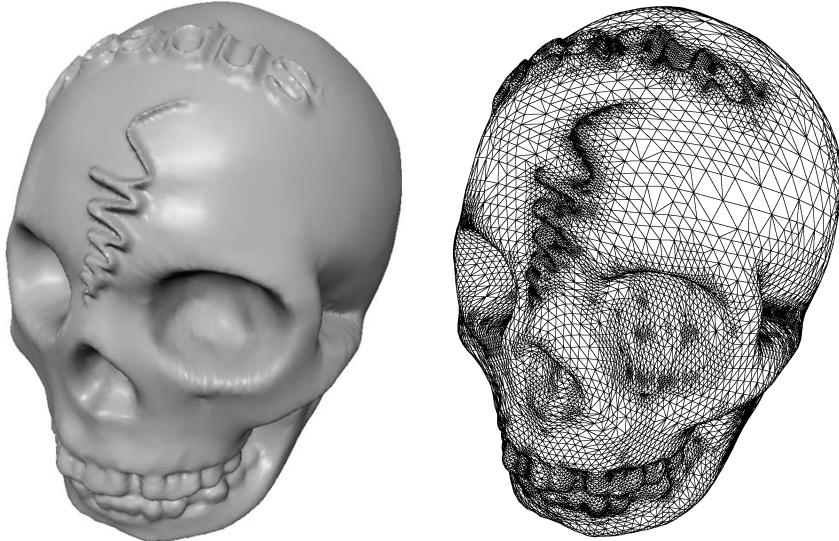


Fig. 9. [Reproduced in colour in Plate 5.] Adaptive normal mesh for the skull model (data courtesy A.Khodakovsky).

methods proposed by Khodakovsky et al. [38, 36] require a wavelet transform and a zerotree coder that we now briefly describe.

Wavelet transform. A semi-regular surface representation is a sequence of approximations at different levels of resolution. The corresponding sequence of nested refinements is transformed by the *wavelet transform* into a representation that consists of a base mesh and a sequence of wavelet coefficients that express differences between successive levels of the mesh hierarchy. The art of compression then consists of choosing appropriate wavelets to best model the statistical distribution of the wavelet coefficients. “Best” means decorrelating the geometry so as to obtain a distribution of wavelet coefficients favourable to efficient compression. For subdivision schemes designed to produce C2-differentiable surfaces almost everywhere (e.g. Loop), it produces excellent results for smooth surfaces since the geometric correlation can be exploited through the prediction of finer level geometry based on the coarser level geometry (by low-pass filtering intrinsic to the subdivision scheme). The reconstruction artifacts at low bit-rates depend mainly on the shape of subdivision basis functions. Hence a surface reconstructed from Loop wavelets has a visually more pleasing shape compared to Butterfly whose basis functions exhibit some “spikes” which look unnatural on a smooth surface.

The zerotree coder (a popular method for wavelet-based image coding [55]) extends also to geometric wavelets. It is shown in [38] that a semi-regular mesh can be represented as a hierarchy over the edges since the wavelet coefficients are attached to the edges (and not the faces). The wavelet coefficients are

therefore naturally encoded in a forest of trees, where each wavelet coefficient at the first level is the root of a tree. The branches of this tree may have variable depth since all regions need not be subdivided to the finest level. For the latter case the edges of the base mesh are subdivided in the tree until an approximation quality is met during coding, and until an adaptive flatness threshold is met during decoding (the subdivision scheme is prolonged to produce a smooth surface even with null wavelet coefficients). Note that an approximating wavelet scheme such as Loop requires uniform mesh subdivision. A zerotree coder is therefore used as a separate procedure to reflect this adaptivity, a “zerotree” symbol coding at a given branching node in the tree representing a sub-tree filled entirely with coefficients below the significance threshold. It then remains to compress the zerotree symbols (significance bits, sign bits and refinement bits), which is done using an arithmetic coder.

When using lossless compression, performance is measured by plotting the rate-distortion curve. Measuring bits per vertex here would be irrelevant since the initial mesh is not considered as optimal for approximation, and the remeshing stage changes the number of vertices. The main observation of [38] is that in many cases surface representation does not really matter for geometry (i.e. there are many degrees of freedom in the meshing), but can lead to high penalty for compression. Therefore, several degrees of sophistication in the remeshing process can lead to significant gains for compression:

- semi-regular remeshing reduces the connectivity penalty.
- uniform remeshing reduces the parameterization penalty compared to non-uniform meshes.
- uniform remeshing while considering local frames with different quantisation for different components reduces the influence of the parameterization even further.
- normal remeshing, explicitly eliminating the tangential component by building a normal mesh representation makes a significant improvement for certain classes of wavelets (e.g. normal Butterfly is better than normal Loop) [36].

Another observation of [36] is that normal Loop behaves better than [38] because the normal parameterization is smoother than MAPS [44], which leads to faster decaying wavelet coefficients and therefore more efficient compression. Moreover, recent experiments confirm the importance of the smoothness of the parameterization for semi-regular remeshing and hence for geometry compression [37]. Finally, a model-based bit-allocation technique has been proposed by Payan and Antonini [50] to efficiently allocate the bits across wavelet sub-bands according to their variance.

4.4 Comparison and Discussion

Most of the recent techniques for “lossless” progressive coding of carefully designed meshes use the *independent set* concept to drive the mesh refinement

operations, be they organised into a set of patches or along a chain of edges optimised for efficient rendering. Vertex positions are coded using various prediction schemes (barycentric, etc.) after uniform quantisation is applied in vertex coordinate space. As already observed in Sect. 3, less work has been done for geometry coding than for connectivity coding. There is even less work on progressive coding techniques, since they are obviously lossy (at least at intermediate stages), and the difficulty to objectively quantify the loss makes it difficult to analyse their performance.

Although the successful single-rate valence-based approach generalises to progressive coding of triangle meshes [2], nothing has been done for progressive coding of polygonal meshes. The key problem here is to find a mesh decimation scheme capable of transforming an arbitrary mesh into a polygon mesh during decimation so that the so-called rate-distortion trade-off is optimised. At the intuitive level, each bit transmitted during decoding should lead to the largest decrease in geometric error. Although this problem is similar to that encountered when designing a decimation scheme, the amount of information represented by a given refinement operation has yet to be quantified.

Wavelet coding schemes, coupled with an initial semi-regular remeshing stage to generate a good base mesh, have proven to be very successful for shape compression [38]. One important question there is how to generate the “best” base mesh, in the sense that it best reflects the mesh geometry and, mainly, features. Another question is the choice of a wavelet scheme suited to best decorrelate the geometric “signal” present in the mesh. Is it a subdivision-related scheme or something more elaborate? The choice of an error metric for driving both the approximation and the measurement of the rate-distortion trade-off also plays a key rôle. The latter point has already proven to be of crucial importance for applications related to visualisation (see the concept of visual metric in [33, 57]). In fact, the optimisation of the rate-distortion trade-off involves many challenging issues linked to sampling and approximation theory, differential geometry, wavelets and information theory.

Acknowledgement

This work was supported in part by the EU research project “Multiresolution in Geometric Modelling (MINGLE)” under grant HPRN-CT-1999-00117.

References

1. P. Alliez, D. Cohen-Steiner, O. Devillers, B. Levy, and M. Desbrun. Anisotropic Polygonal Remeshing. In *Proc. ACM SIGGRAPH*, 2003.
2. P. Alliez and M. Desbrun. Progressive Encoding for Lossless Transmission of 3D Meshes. In *Proc. ACM SIGGRAPH*, pages 198–205, 2001.
3. P. Alliez and M. Desbrun. Valence-Driven Connectivity Encoding of 3D Meshes. In *Eurographics Conference Proceedings*, pages 480–489, 2001.

4. 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.
5. M. Ben-Chen and C. Gotsman. On the Optimality of the Laplacian Spectral Basis for Mesh Geometry Coding. *ACM Transactions on Graphics*. to appear.
6. A. Bogomjakov and C. Gotsman. Universal Rendering Sequences for Transparent Vertex Caching of Progressive Meshes. *Computer Graphics Forum*, 21(2):137–148, 2002.
7. R.-C-N. Chuang, A. Garg, X. He, M.-Y. Kao, and H.-I. Lu. Compact Encodings of Planar Graphs via Canonical Orderings and Multiple Parentheses. In *ICALP: Annual International Colloquium on Automata, Languages and Programming*, pages 118–129, 1998.
8. D. Cohen-Or, D. Levin, and O. Remez. Progressive Compression of Arbitrary Triangular Meshes. In *IEEE Visualization Conference Proceedings*, pages 67–72, 1999.
9. R. Cohen D. Cohen-Or and T. Ironi. Multi-way Geometry Encoding, 2002. Technical report.
10. M. Deering. Geometry Compression. *Proc. ACM SIGGRAPH*, pages 13–20, 1995.
11. O. Devillers and P.-M. Gaidon. Geometric Compression for Interactive Transmission. *IEEE Visualization Conference Proceedings*, pages 319–326, 2000.
12. P.-M. Gaidon and O. Devillers. Progressive Lossless Compression of Arbitrary Simplicial Complexes. *ACM Transactions on Graphics*, 21:372–379, 2002. Proc. ACM SIGGRAPH.
13. C. Gotsman. On the Optimality of Valence-Based Connectivity Coding. *Computer Graphics Forum*, 22(1):99–102, 2003.
14. C. Gotsman, S. Gumhold, and L. Kobbelt. Simplification and Compression of 3D Meshes, 2002. In *Tutorials on Multiresolution in Geometric Modelling (Munich Summer School Lecture Notes)*, A. Iske, E. Quak, M. Floater (Eds.), Springer, 2002.
15. X. Gu, S. Gortler, and H. Hoppe. Geometry Images. In *Proc. ACM SIGGRAPH*, pages 355–361, 2002.
16. S. Gumhold. Improved Cut-Border Machine for Triangle Mesh Compression. *Erlangen Workshop'99 on Vision, Modeling and Visualization*, 1999.
17. S. Gumhold. New Bounds on the Encoding of Planar Triangulations. Technical Report WSI-2000-1, Univ. of Tübingen, 2000.
18. S. Gumhold and W. Strasser. Real Time Compression of Triangle Mesh Connectivity. In *Proc. ACM SIGGRAPH*, pages 133–140, 1998.
19. I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal Meshes. In *Proc. ACM SIGGRAPH*, pages 95–102, 2000.
20. Henry Helson. *Harmonic Analysis*. Wadsworth & Brooks/Cole, 1991.
21. J. Ho, K.-C. Lee, and D. Kriegman. Compressing Large Polygonal Models. In *IEEE Visualization Conference Proceedings*, pages 357–362, 2001.
22. H. Hoppe. Progressive meshes. In *Proc. ACM SIGGRAPH*, pages 99–108, 1996.
23. M. Isenburg. Compressing Polygon Mesh Connectivity with Degree Duality Prediction. In *Graphics Interface Conference Proc.*, pages 161–170, 2002.
24. M. Isenburg and P. Alliez. Compressing Hexahedral Volume Meshes. In *Pacific Graphics Conference Proceedings*, pages 284–293, 2002.

25. M. Isenburg and P. Alliez. Compressing Polygon Mesh Geometry with Parallelogram Prediction. In *IEEE Visualization Conference Proceedings*, pages 141–146, 2002.
26. M. Isenburg, P. Alliez, and J. Snoeyink. A Benchmark Coder for Polygon Mesh Compression, 2002. <http://www.cs.unc.edu/~isenburg/pmc/>.
27. M. Isenburg and S. Gumhold. Out-of-Core Compression for Gigantic Polygon Meshes. In *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, 22(3):935–942, 2003.
28. M. Isenburg and J. Snoeyink. Mesh Collapse Compression. In *Proc. of SIBGRAPI'99, Campinas, Brazil*, pages 27–28, 1999.
29. M. Isenburg and J. Snoeyink. Face Fixer: Compressing Polygon Meshes With Properties. In *Proc. ACM SIGGRAPH*, pages 263–270, 2000.
30. M. Isenburg and J. Snoeyink. Spirale Reversi: Reverse Decoding of the Edge-breaker Encoding. In *Proc. 12th Canadian Conference on Computational Geometry*, pages 247–256, 2000.
31. M. Isenburg and J. Snoeyink. Binary Compression Rates for ASCII Formats. In *Proc. Web3D Symposium*, pages 173–178, 2003.
32. Z. Karni, A. Bogomjakov, and C. Gotsman. Efficient Compression and Rendering of Multi-Resolution Meshes. In *IEEE Visualization Conference Proceedings*, 2002.
33. Z. Karni and C. Gotsman. Spectral Compression of Mesh Geometry. In *Proc. ACM SIGGRAPH*, pages 279–286, 2000.
34. Keeler and Westbrook. Short Encodings of Planar Graphs and Maps. *Discrete Appl. Math.*, 58:239–252, 1995.
35. A. Khodakovsky, P. Alliez, M. Desbrun, and P. Schröder. Near-Optimal Connectivity Encoding of 2-Manifold Polygon Meshes. *Graphical Models, special issue*, 2002.
36. A. Khodakovsky and I. Guskov. Compression of Normal Meshes. In *Proc. ACM SIGGRAPH*. Springer-Verlag, 2003.
37. A. Khodakovsky, N. Litke, and P. Schröder. Globally Smooth Parameterizations with Low Distortion. In *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, 22(3):350–357, 2003.
38. A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive Geometry Compression. *Proc. ACM SIGGRAPH*, pages 271–278, 2000.
39. D. King and J. Rossignac. Guaranteed 3.67V bit Encoding of Planar Triangle Graphs. In *11th Canadian Conference on Computational Geometry*, pages 146–149, 1999.
40. D. King, J. Rossignac, and A. Szmczak. Compression for Irregular Quadrilateral Meshes. Technical Report TR-99-36, GVU, Georgia Tech, 1999.
41. D. Knuth. Exhaustive Generation, 2003. volume 4 of *The Art of Computer Programming*, in preparation, available electronically, <http://www.cs.stanford.edu/~knuth>.
42. B. Kronrod and C. Gotsman. Efficient Coding of Non-Triangular Mesh Connectivity. *Graphical Models*, 63(263–275), 2001.
43. B. Kronrod and C. Gotsman. Optimized Compression of Triangle Mesh Geometry Using Prediction Trees. *Proc. 1st International Symposium on 3D Data Processing, Visualization and Transmission*, pages 602–608, 2002.
44. A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. *Computer Graphics*, 32(Annual Conference Series):95–104, August 1998.

45. E. Lee and H. Ko. Vertex Data Compression For Triangular Meshes. In *Proc. Pacific Graphics*, pages 225–234, 2000.
46. H. Lee, P. Alliez, and M. Desbrun. Angle-Analyzer: A Triangle-Quad Mesh Codec. In *Eurographics Conference Proceedings*, pages 383–392, 2002.
47. M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginsburg, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The Digital Michelangelo Project. In *Proc. ACM SIGGRAPH*, pages 131–144, 2000.
48. J. Li and C.-C. Jay Kuo. Mesh Connectivity Coding by the Dual Graph Approach, July 1998. MPEG98 Contribution Document No. M3530, Dublin, Ireland.
49. R. Pajarola and J. Rossignac. Compressed Progressive Meshes. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):79–93, 2000.
50. F. Payan and M. Antonini. 3D Mesh Wavelet Coding Using Efficient Model-based Bit Allocation. In *Proc. 1st Int. Symposium on 3D Data Processing Visualization and Transmission*, pages 391–394, 2002.
51. D. Poulalhon and G. Schaeffer. Optimal Coding and Sampling of Triangulations, 2003. 30th international colloquium on automata, languages and programming (ICALP’03).
52. J. Rossignac. Edgebreaker : Connectivity Compression for Triangle Meshes. *IEEE Transactions on Visualization and Computer Graphics*, 1999.
53. P. Sander, S. Gortler, J. Snyder, and H. Hoppe. Signal-Specialized Parametrization. In *Eurographics Workshop on Rendering 2002*, 2002.
54. P. Sander, Z. Wood, S. Gortler, J. Snyder, and H. Hoppe. Multi-Chart Geometry Images. In *Proc. Eurographics Symposium on Geometry Processing*, 2003.
55. J.M. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, 1993.
56. D. Shikhare, S. Bhakar, and S.P. Mudur. Compression of Large 3D Engineering Models using Automatic Discovery of Repeating Geometric Features. In *proceedings of 6th International Fall Workshop on Vision, Modeling and Visualization*, 2001.
57. O. Sorkine, D. Cohen-Or, and S. Toledo. High-Pass Quantization for Mesh Encoding. In *Proc. of Eurographics Symposium on Geometry Processing*, 2003.
58. V. Surazhsky and C. Gotsman. Explicit Surface Remeshing. In *Proc. Eurographics Symposium on Geometry Processing*, pages 20–30, 2003.
59. A. Szymczak, D. King, and J. Rossignac. An Edgebreaker-Based Efficient Compression Scheme for Regular Meshes. *Computational Geometry*, 20(1-2):53–68, 2001.
60. A. Szymczak, J. Rossignac, and D. King. Piecewise Regular Meshes: Construction and Compression. *Graphical Models*, 64(3–4):183–198, 2003.
61. G. Taubin, W. Horn, J. Rossignac, and F. Lazarus. Geometry Coding and VRML. In *Proc. IEEE*, volume 86(6), pages 1228–1243, June 1998.
62. C. Touma and C. Gotsman. Triangle Mesh Compression. *Graphics Interface 98 Conference Proceedings*, pages 26–34, 1998.
63. G. Turan. Succinct Representations of Graphs. *Discrete Applied Mathematics*, 8:289–294, 1984.
64. W. Tutte. A Census of Planar Maps. *Canadian Journal of Mathematics*, 15:249–271, 1963.
65. Viewpoint. *Premier Catalog (2000 Edition)* www.viewpoint.com. Viewpoint editor, 2000.

Shape Compression using Spherical Geometry Images

Hugues Hoppe¹ and Emil Praun²

¹ Microsoft Research, Redmond, Washington, USA

hhoppe@microsoft.com

² University of Utah, Salt Lake City, Utah, USA

emilp@cs.utah.edu

Summary. We recently introduced an algorithm for spherical parametrization and remeshing, which allows resampling of a genus-zero surface onto a regular 2D grid, a spherical geometry image. These geometry images offer several advantages for shape compression. First, simple extension rules extend the square image domain to cover the infinite plane, thereby providing a globally smooth surface parametrization. The 2D grid structure permits use of ordinary image wavelets, including higher-order wavelets with polynomial precision. The coarsest wavelets span the entire surface and thus encode the lowest frequencies of the shape. Finally, the compression and decompression algorithms operate on ordinary 2D arrays, and are thus ideally suited for hardware acceleration. In this paper, we detail two wavelet-based approaches for shape compression using spherical geometry images, and provide comparisons with previous compression schemes.

1 Introduction

In previous work [20], we introduce a robust algorithm for spherical parametrisation, which smoothly maps a genus-zero surface to a sphere domain. This sphere domain can in turn be unfolded onto a square, to allow remeshing of surface geometry onto a regular 2D grid – a geometry image. One important use for such a representation is shape compression, the concise encoding of surface geometry. In this paper, we explore the application of shape compression in more detail, describing two wavelet-based approaches.

As we will show, spherical geometry images are a powerful representation for the concise description of shape.

2 Related Work on Shape Compression

The compression of geometric shape has recently been a very active area of research. Since we will not be able to cover every paper here, we refer the reader

to recent comprehensive surveys [3, 10, 22]. The many compression techniques can be categorised into two broad approaches: *irregular mesh* compression and *remeshing* compression, depending on whether or not they preserve the original mesh connectivity.

Irregular Mesh Compression

Preserving the connectivity of the original mesh is important for accurately modelling sharp features such as creases and corners, particularly in manufactured parts. Also, meshes designed within graphical modelling systems may have face connectivities that encode material boundaries, shading discontinuities, or desired behaviour under deformation.

The compression of irregular meshes involves two parts: connectivity and geometry. The mesh connectivity is a combinatorial graph; it can be encoded using approximately 2 bits per vertex [2]. The mesh geometry is given by continuous (x,y,z) vertex positions; these are typically quantised to 10, 12, or 14 bits per coordinate prior to entropy coding.

The compression of irregular meshes was pioneered by Deering [8], who describes a scheme for streaming decompression in the graphics system. Gumhold and Strasser [12] advance a front through a mesh using a state machine, and compress the necessary state changes. Touma and Gotsman [28] use a similar technique based on vertex-valence encoding. Many other papers have refined this approach, including the Edge Breaker scheme of Rossignac [21] and several methods for non-triangular meshes.

Several schemes support progressive representations, whereby coarser approximations can be displayed as the data stream is incrementally received. These include progressive meshes [14], progressive forest split compression [27], and the valence-driven simplification approach of Alliez and Desbrun [2].

Compressing the geometry of irregular meshes is difficult because the irregular sampling does not admit traditional multiresolution wavelet hierarchies. Most compression schemes predict the position of each vertex from its partially reconstructed neighbourhood. A good example is the “parallelogram rule” of Touma and Gotsman [28]. The drawback of basing the prediction model on a local neighbourhood is that it cannot capture the low-frequency features of the model. In other words, the local prediction rules cannot give rise to the broad basis functions that one would obtain in the coarsest levels of a traditional multiresolution hierarchy. One exception is the scheme of Karni and Gotsman [15], which constructs smooth basis functions using spectral analysis of the mesh adjacency matrix. However, this spectral analysis is costly and unstable, and therefore becomes practical only when performed piecewise on a partitioned model.

Remeshing Compression

For many applications, preserving the connectivity of the given mesh is unnecessary. In particular, many models are obtained through 3D scanning tech-

nologies (e.g. laser range scanners, computed tomography, magnetic resonance imaging), and the precise connectivities in these dense meshes is somewhat arbitrary. Since shape compression is generally lossy, resampling the geometry onto a new mesh (with different connectivity) is quite reasonable.

In the remeshing approach of Attene et al. [5], an irregular-mesh compression algorithm resamples geometry as it traverses the mesh, by incrementally wrapping the mesh with isosceles triangles.

A number of methods use a *semi-regular* remeshing structure. Such a remesh is obtained by repeated quaternary subdivision of a coarse triangle mesh (i.e. each triangle face is regularly subdivided into 4 sub-faces). Lounsbery et al. [19] develop a wavelet-like framework over these semi-regular structures. Eck et al. [9] present a scheme for semi-regular remeshing of arbitrary triangle meshes, and achieve shape compression by removing small wavelet coefficients. Khodakovsky et al. [16] obtain better compression results using zero-trees; also, they express wavelet coefficients with respect to local surface coordinate frames, and assign fewer bits to the tangential components of the wavelet coefficients. The globally smooth parametrization of Khodakovsky et al. [18] reduces the entropy of these tangential components by constructing a remesh that is parametrically smooth across patch boundaries. The “normal mesh” representation of Khodakovsky et al. [17] attempts to remove tangential information altogether; each subdivision of the remesh is obtained by displacing most of the newly introduced vertices along the surface normal; only a small fraction of vertices require full 3D vector displacements.

Another approach, and the one pursued in this paper, is to form a *completely regular* remesh. As shown by Gu et al. [11], an arbitrary mesh can be resampled onto a regular 2D grid, a *geometry image*. The given mesh is cut along a network of cut paths to form a topological disk, and this disk is then parametrized over a square. The geometry image is obtained by creating a regular grid over the square and sampling the surface using the parametrization. Due to their simple regular structure, geometry images can be compressed using ordinary 2D image wavelets. However, one difficulty is that lossy decompression leads to “gaps” along the surface cut paths. Gu et al. [11] overcome these gaps by re-fusing the boundary using a topological sideband, and diffusing the resulting step function into the image interior.

In this work, we construct geometry images for genus-zero surfaces using a spherical remeshing approach, as described in the next section. By defining spherical extension rules beyond the geometry image boundaries, we avoid boundary reconstruction problems altogether.

3 Review of Spherical Parametrization and Remeshing

In previous work [20] we have presented a method for parametrizing a genus-zero model onto the sphere and remeshing it onto a geometry image, as illustrated in Figs. 1 and 2. Since the geometric signal is too smooth to be



Map of original mesh onto sphere, octahedron domain, and image.

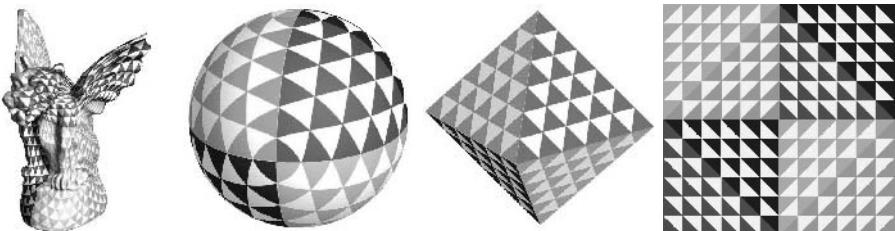


Illustration of the same map using image grid samples.

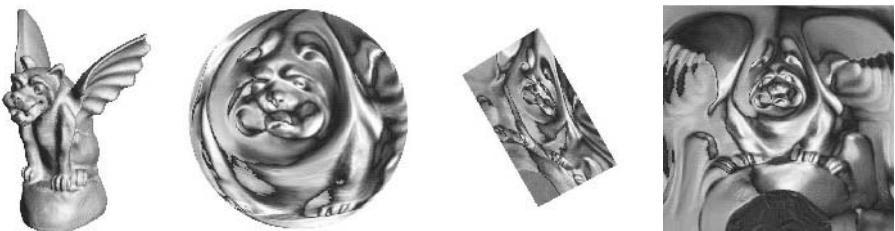
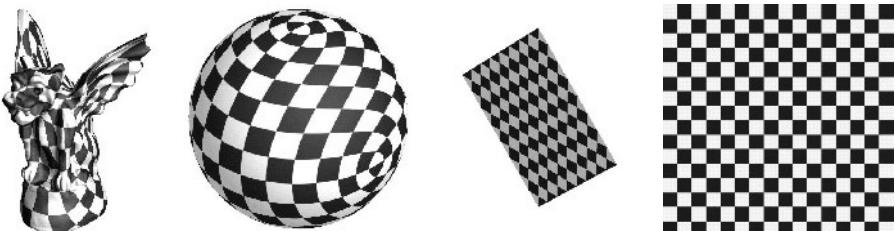
Map of original mesh onto sphere, *flat* octahedron domain, and image.

Illustration of the same map using image grid samples.

Fig. 1. [Reproduced in colour in Plate 6.] Illustration of remeshing onto octahedron and flat octahedron domains.

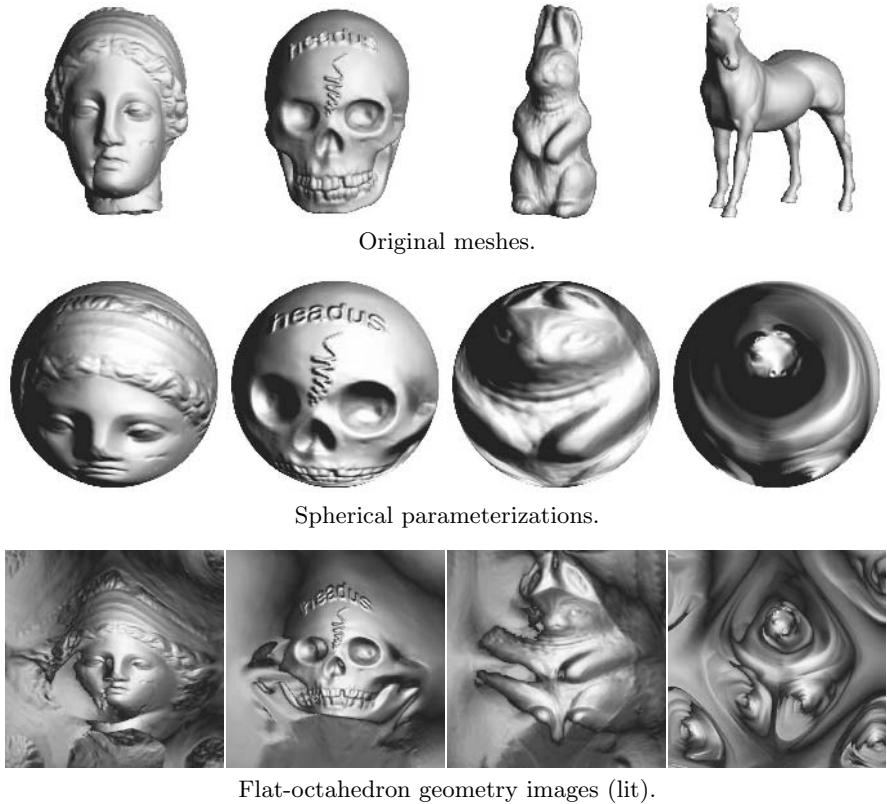


Fig. 2. Spherical parametrization and remeshing applied to the four test models. The geometry images are shown shaded to better illustrate the parametrization.

visualised by directly mapping (x,y,z) to the (R,G,B) channels of an image, we chose to visualise these geometry images in a different way. We compute for each pixel an approximated normal by taking neighbour differences, and shade the geometry image based on these normals (using two antipodal lights).

Spherical Parametrization

The first step maps the original surface onto a sphere domain. For genus-zero models, the sphere is the most natural domain, since it does not require breaking the surface using any *a priori* cuts, which would otherwise artificially constrain the parametrization.

To be suitable for subsequent remeshing, the spherical map must satisfy two important properties: (1) it must be one-to-one, and (2) it must allocate enough domain area to all features of the mesh. Our spherical parametrization

achieves these goals by employing a robust coarse-to-fine construction, and by minimising a stretch distortion measure to prevent later undersampling.

Coarse-to-fine construction. The mesh is converted to a progressive mesh format [14] by repeatedly applying half-edge-collapse operations. For triangulated genus-zero models, one can always reach a tetrahedron as the base domain [25]. This base tetrahedron is mapped to a regular tetrahedron inscribed in the unit sphere. We then visit the progressive mesh sequence in coarse-to-fine order, adding vertices back into the mesh and mapping them onto the sphere. To guarantee that the map is one-to-one, a vertex must lie inside the kernel of the spherical polygon formed by its neighbours. Fortunately, if the map is one-to-one prior to inserting a new vertex, it can be shown that the new vertex's neighbourhood is always non-empty, and thus new vertices can always be inserted into the parametrization while maintaining a bijection.

Stretch metric. To adequately sample all the features of a model, we employ a parametrization distortion metric based on stretch minimisation. We minimise this nonlinear metric each time a new vertex is introduced, by locally optimising its location and those of its immediate neighbours. Each time the number of vertices grows by a factor of 1.5, we also perform a global pass, optimising all vertices introduced so far. When optimising any vertex, we constrain it to the kernel of its neighbourhood, to prevent flips. Degenerate triangles are prevented naturally by the stretch metric, which becomes infinite in that case.

Spherical Remeshing

Once we have a spherical parametrization for the mesh, we seek to resample the sphere onto a geometry image. For simplicity, this image should be square and should have simple boundary conditions. We have explored two schemes for unfolding the sphere onto the square, one based on a *regular octahedron* domain and the other based on a *flattened octahedron* domain (see Fig. 1). In either case, we regularly subdivide the octahedron, and map it to the sphere using the spherical parametrization procedure described earlier. (The one difference is that we measure stretch in the opposite direction, from the domain to the sphere.)

The samples of the octahedron are then associated with the grid locations of a square geometry image by cutting four edges of the octahedron meeting at a vertex, and unfolding the four faces incident to the vertex-like flaps. In Fig. 1, rows 2 and 4 illustrate the sampling pattern imposed by the regular grid. For the octahedron, we use the linear 3-tap triangular reconstruction filter, and the filter footprint varies across the four quadrants of the geometry image, according to the faces of the base octahedron (shown in different colours for easy identification). For the flat octahedron, we use the traditional 4-tap bilinear reconstruction filter, and this filter is uniform across the whole geometry image.

The geometry of the regular octahedron corresponds nicely with the use of spherical wavelets (Sect. 4.1), since it offers derivative continuity across edges under the equilateral triangle sampling pattern. Similarly, the geometry of the flattened octahedron corresponds with the use of image wavelets (Sect. 4.2), since the flattened octahedron unfolds isometrically (i.e. without distortion) onto the square image.

4 Compression Using Spherical Geometry Images

To compress a model, we first apply a wavelet transform to the geometric signal, using either spherical wavelets or image wavelets. The bands resulting from the wavelet transform are then run through a general-purpose quantiser and entropy coder [7]. The process is summarised in Fig. 3, and in the pseudo-code of Fig. 4.

Previous research [13] has shown that the geometric information associated with displacement of samples from their predicted location along the surface *normal* is more important than the “parametric” information associated with the *tangential* components of the displacement. Accordingly, we express the fine-scale detail (the high-pass bands or wavelet coefficients from each step of the wavelet transform) in local coordinate frames predicted using the low-pass band. During the quantisation and entropy coding, we assign greater perceptual importance to transformed components normal to the surface than to the tangential components (we found a factor of 3 to give the best results for our models).

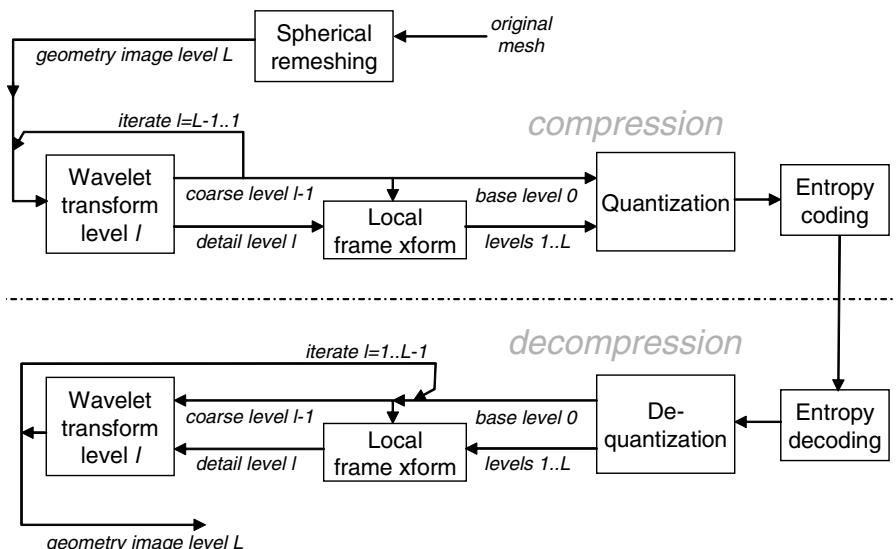


Fig. 3. Overview of compression process.

Compression:

```

Read finest level L
For all levels from fine to coarse: l = L..1:
  - identify "even" grid locations (i.e. those in coarser level l-1)
  - apply low-pass analysis filter centred on "even" samples
  - apply high-pass analysis filter to other ("odd") samples
    (if low-pass or high-pass filter kernels reach outside
      the geometry image, use boundary extension rules)
  - gather all "even" samples into coarse level l-1
  - gather all "odd" samples into detail plane(s) for level l
  - using level l-1, compute local tangential frames for samples on
    detail plane(s) of level l
  - express detail in local tangential frames
Run coarsest level 0 and all detail planes for levels 1..L
  through quantiser and entropy encoder to achieve target bit budget

```

Decompression:

```

Run entropy decoder and dequantiser to produce coarsest level 0
  and detail planes for levels 1..L
For all levels from coarse to fine l = 0..L-1:
  - using level l, compute local tangential frames for samples in
    detail plane for level l+1
  - transform detail from local frames to absolute coordinates
  - apply synthesis filter to level l to predict level l+1
  - apply synthesis filter to detail plane(s) for level l+1
    and combine with prediction to produce level l+1
    (using boundary extension rules as necessary)
Output finest level L

```

Fig. 4. Pseudo-code for the compression and decompression algorithms.

Note that the compression and decompression are lossy, and that the quantisation errors combine non-linearly. For example, if a coarse level is reconstructed inexactly, the errors are not simply added to the final result; they additionally cause small rotations of the local coordinate frames transforming the detail for the finer levels.

For the wavelet compression and decompression stage we present two alternative schemes. The first is based on spherical wavelets introduced by Schröder and Sweldens [24]; the base (coarsest) sampling level is an octahedron, and progressively finer levels are obtained by applying standard subdivision rules such as Loop or Butterfly. The second is based on image wavelets. Both schemes are interesting to consider since they offer different advantages. The mesh-based spherical wavelet scheme seems more natural for coding geometry, and provides good reconstruction of sharp detail at very low bit budgets. The image-based method is easier to implement (just by modifying one of the many existing image coders), and benefits from a large body of research into

image wavelets that resulted in well optimised wavelet bases with large support [4]. After presenting the implementation of both methods in this section, we contrast their results in Sect. 5.

4.1 Spherical Wavelets

Rather than having a complicated pointer-based mesh data structure, we can apply all the mesh-processing operations directly on the geometry image of the unfolded octahedron by manipulating grid location indices. The vertices of the base octahedron correspond to the samples at the corners, boundary midpoints, and centre of the geometry image. The vertices of the octahedron subdivided k times will be at locations $(i * 2^{L-k}, j * 2^{L-k})$, for $0 \leq i, j \leq 2^{k+1}$, where L is the finest level. Two samples on a given level are neighbours in the subdivided octahedron if they are 4-adjacent in the grid restricted to the samples on that level, or linked by a diagonal. The orientation of this diagonal is determined by the image quadrant, i.e. “forward slash” for the upper left and lower right one, and “backward slash” for the other two. This distinction in the use of diagonals is needed because the sample connectivity in the grid is not arbitrary but is inherited from the subdivided octahedron. We check the quadrant at the midpoint between the two samples, to avoid ambiguities created when one of the samples is a vertex of the base octahedron. Using this simple set of rules, we efficiently gather the neighbours of a vertex on a given level, to form the stencils required for wavelet analysis and synthesis (e.g. the green stencil in Fig. 5).

Boundary extension rules. To complete the wavelet transform stencils corresponding to samples near the boundaries, we sometimes need to “hal-

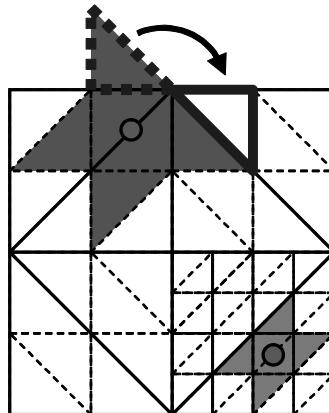


Fig. 5. Spherical wavelets on the unfolded octahedron geometry image. The dark and light grey regions highlight two Butterfly stencils at different subdivision levels. Since the dark grey stencil reaches outside the image, it uses boundary extension rules.

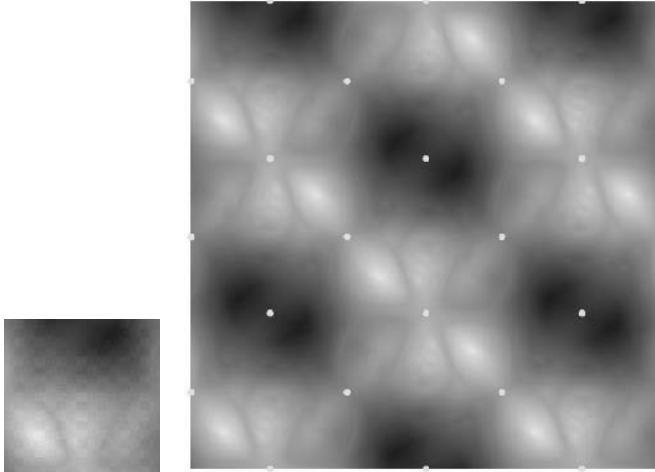


Fig. 6. A spherical geometry image and its infinite tiling in the plane. The parametrization is globally smooth except at the image boundary midpoints.

lucinate” values outside the square grid of values. Standard tricks used in signal processing to extend an image beyond its borders include replicating the boundary samples or reflecting the image across its boundaries. These methods provide a continuous signal, but introduce derivative discontinuities in the infinite lattice produced, and discontinuities are more expensive to code than smooth signals. Instead of using the standard tricks, we designed different boundary extension rules for our unfolded octahedron geometry image, which do produce an infinite lattice with derivative continuity. The general idea for these rules is that whenever we must “march” outside the image across a boundary to produce a sample, we flip the image (such that the boundary is mapped onto itself by a 180-degree rotation) and return the value located there (e.g. the red stencil in Fig. 5). Considering the image samples to be labelled in row-major order, starting from 0 (so samples in the left column are $(0, j)$), the rules are $I(-i, j) = I(i, n - 1 - j)$, $I(i, -j) = I(n - 1 - i, j)$, $I(n - 1 + i, j) = I(n - 1 - i, n - 1 - j)$ and $I(i, n - 1 + j) = I(n - 1 - i, n - 1 - j)$, where $n = 2^k + 1$ is the number of rows and columns. This is equivalent to filling the infinite plane of all sample locations by rotating the original image around the boundary midpoints (see Fig. 6). The infinite lattice produced is continuous everywhere and derivative continuous everywhere except at the repeated instances of the four boundary midpoints (dots in the figure).

Note that for $i, j = 0$ and $i, j = n - 1$, the rules provide constraints on the boundaries rather than a way to extend the image outside its borders. Therefore, to avoid sample duplication we also consider the right half of the top and bottom boundaries of the geometry image and the lower half of the left and right boundaries to be “outside” the image (so we map those locations

to the surviving half of the same boundary, using the extension rules). These “outside” boundary grid locations are not processed by the quantiser and entropy coder.

Local tangential frame. We use the lifted Butterfly scheme, as described in [24]. We compute a normal for each “odd” sample by averaging the normals of the faces from the Butterfly stencil (with weights 1,4,1,1,4,1). Note that the vertices of these faces are all “even” vertices. The Y coordinate of the frame is obtained as the cross product between this normal and the row direction in the grid of samples (obtained from differences of neighbouring samples, similarly to the image wavelet case). We take a cross product again to obtain the X axis of the frame.

4.2 Image Wavelets

The other scheme we consider is based on image wavelets, using the flat octahedron parametrization. We use a general-purpose wavelet-based image compression package [7], modified to make use of boundary extension rules and local tangential frame coding.

Boundary extension rules. These rules come into play during both compression and decompression phases when we are applying a signal processing kernel to a sample that is closer to the image boundary than the kernel width (see Fig. 7). In these cases, we have to “hallucinate” samples for the grid locations outside the original image. Similarly to the spherical wavelets case, we fill these grid locations using boundary extension rules.

For image wavelets there are two types of rules, depending on whether the image boundary is located “on” the samples or “between” samples. To simplify the discussion, let’s assume we need a sample located outside the image across the left boundary. The first case corresponds to the original unfolded octahedron, and all the coarser levels. These levels have $2^k + 1$ columns, and the left column is constrained: sample $I(0, j)$ must be equal to sample $I(0, n_r - 1 - j)$, where n_r is the number of rows. In this case, the reflection rule is the same as in the spherical wavelets case $I(-i, j) := I(i, n_r - 1 - j)$.

The second type of extension rules apply to the detail planes obtained from the image wavelet transform. Some of these images have a number of columns equal to 2^k rather than $2^k + 1$ and lack the left boundary with constrained samples. Specifically, the image wavelet transform uses separable filters, so to apply a 2D transform, it first applies a 1D transform to all the rows, producing images L and H (see Fig. 8) and then a 1D transform to all the columns, producing images LL, LH, HL, and HH. Planes with an even number of columns (such as H, HL, and HH) lack the leftmost boundary with constrained samples, and use a reflection boundary located “between” grid locations: $I(-i, j) := I(i - 1, n_r - 1 - j)$. Intuitively, in the geometry image case, when we go outside the image across a “fat” edge in Fig. 8, we come back inside the image across the same boundary, skipping the sample on the

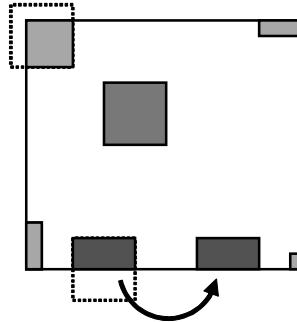


Fig. 7. Effect of domain extension rules on wavelet basis extents using the flat octahedron image wavelets.

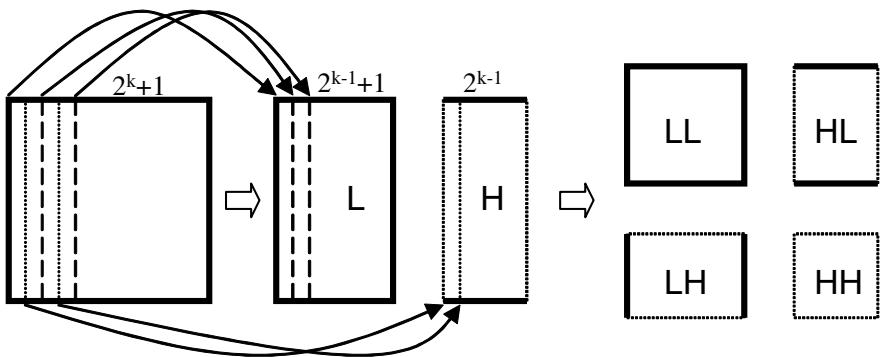


Fig. 8. Wavelet transform of an image level. Applying a 1D transform to each row results in a low-pass plane L and a high-pass (detail) one, H . After a 1D transform on each of the columns, we get the coarser level LL , and three detail planes. The thick boundary edges have flip-symmetry constraints.

boundary itself, while in the case of a dotted boundary edge of a detail plane in Fig. 8 we don't skip the sample on the boundary.

Similarly to the spherical wavelets case, the first type of boundary rules applied to the samples on the boundary provide constraints, rather than ways to extend the image. During the compression phase, those constraints are satisfied since the original geometry image was constructed using them. However, since the whole compression/decompression process is lossy, the constraints may be unsatisfied during decompression. To ensure that the resulting model is crack-free, we enforce the constraints at all the resolution levels during decompression. After we recover a level, before using it to produce the local frame and the coarse approximation for the next level, we first average together the samples on the boundary that should be equal (for example, samples $(i, 0)$ and $(n - 1 - i, 0)$ on the top row of the image are averaged together and set to the resulting value). Pairs of corresponding locations on the image boundaries are

averaged together, and the four corners are also averaged and kept consistent. It is advantageous to enforce consistency at all the resolution levels, rather than just once at the end, in order to avoid “step functions” from appearing in the result.

The boundary rules benefit the compression/decompression algorithm in two ways. First, they maintain consistency between samples on the boundaries, preventing cracks in the 3D model corresponding to the geometry image. Second, they help the prediction for the samples near the boundaries, since the image signal appears smooth not only in the interior of the image but also near its borders (see Fig. 6).

An important thing to note is the fact that applying a symmetric filter kernel to a lattice satisfying the boundary extension rules will result in a new lattice with the same extension rules. We make use of this fact since we rely on the extension rules at all the resolution levels, not just the finest resolution geometry image. For arbitrary kernels, one would need to store two instances of the processed image, one with the kernel itself and one with the reflected kernel, in order to be able to represent the whole new infinite lattice. This would be a significant drawback for compression, since the bit budget would double. We therefore use wavelets based on symmetric kernels.

Local tangential frame. To compute a local frame for each “odd” sample, we first obtain vectors corresponding to the row and column directions of the grid. If the sample is on an even row (and necessarily an odd column), we get the row direction from the difference of the two neighbouring (even) samples in the same row. If the sample is from an odd row, we average the directions computed using the two adjacent even rows. We compute the column direction in a similar fashion. Taking the cross product of the two vectors we obtain the normal direction, which we cross with the column direction to get the X axis. The Y axis is obtained by cross product between the normal and X. Finally we normalise the three vectors composing the frame.

Implementation details. We used the Antonini [4] image wavelet bases, which are symmetric separable kernels with 7 entries for the 1D high-pass and 9 entries for the low-pass. Since the wavelet kernels have large support, we do not apply the wavelet transform all the way down to the 3x3 image, but instead use a fixed number of stages (specifically, 5), starting from a fine 513x513 geometry image.

5 Results and Discussion

We have run compression experiments using the 4 test models shown in Fig. 2. The spherical parametrization process took 1–3 minutes on the original meshes with 28–134K faces. (This significant improvement in processing times over those reported in [20] are simply due to code optimisation.) The given models were remeshed into geometry images of size 513x513 and then compressed.



1,445 bytes (61.6 dB) 2,949 bytes (67.0 dB) 11,958 bytes (75.7 dB)
Compression using spherical wavelets.



1,357 bytes (60.8 dB) 2,879 bytes (66.5 dB) 11,908 bytes (77.6 dB)
Compression using image wavelets.

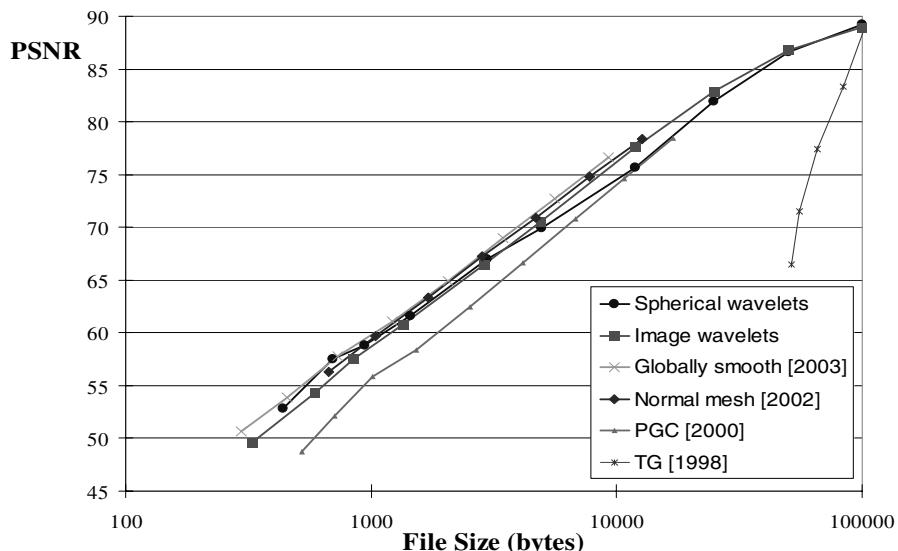
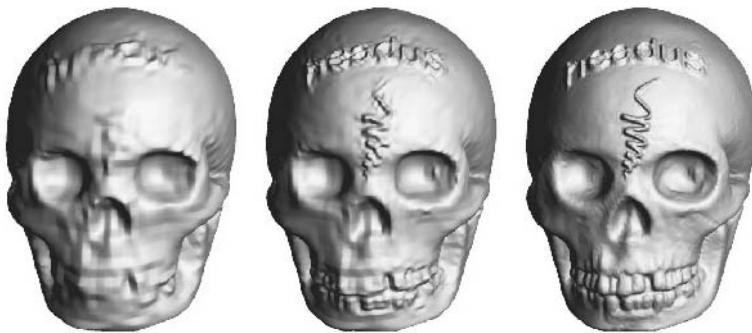


Fig. 9. [Reproduced in colour in Plate 7.] Compression results on Venus model.



1,444 bytes (59.6 dB) 2,951 bytes (65.5 dB) 11,959 bytes (76.1 dB)
Compression using spherical wavelets.



1,367 bytes (60.5 dB) 2,889 bytes (66.2 dB) 11,915 bytes (77.5 dB)
Compression using image wavelets.

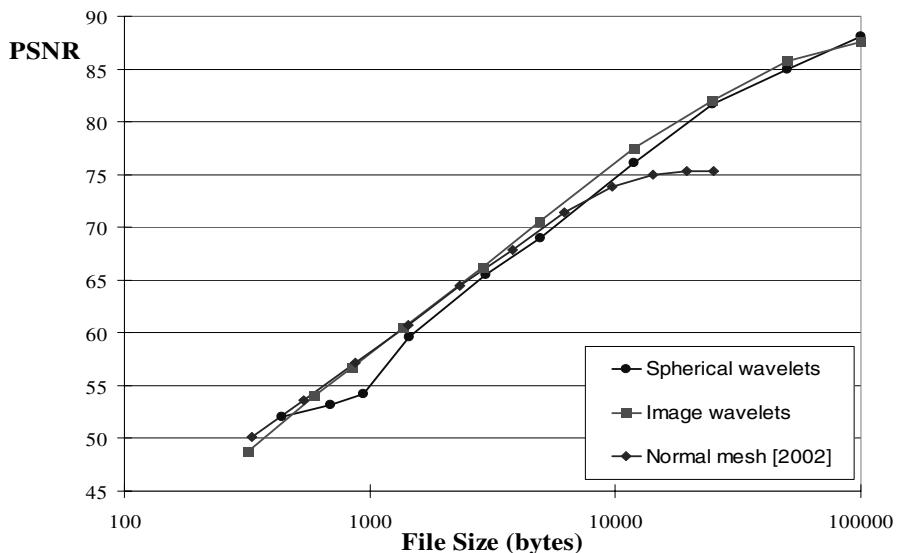


Fig. 10. [Reproduced in colour in Plate 8.] Compression results on skull model.



1,447 bytes (64.0 dB) 2,950 bytes (69.5 dB) 11,958 bytes (79.8 dB)
Compression using spherical wavelets.



1,364 bytes (63.2 dB) 2,881 bytes (70.2 dB) 11,906 bytes (81.9 dB)
Compression using image wavelets.

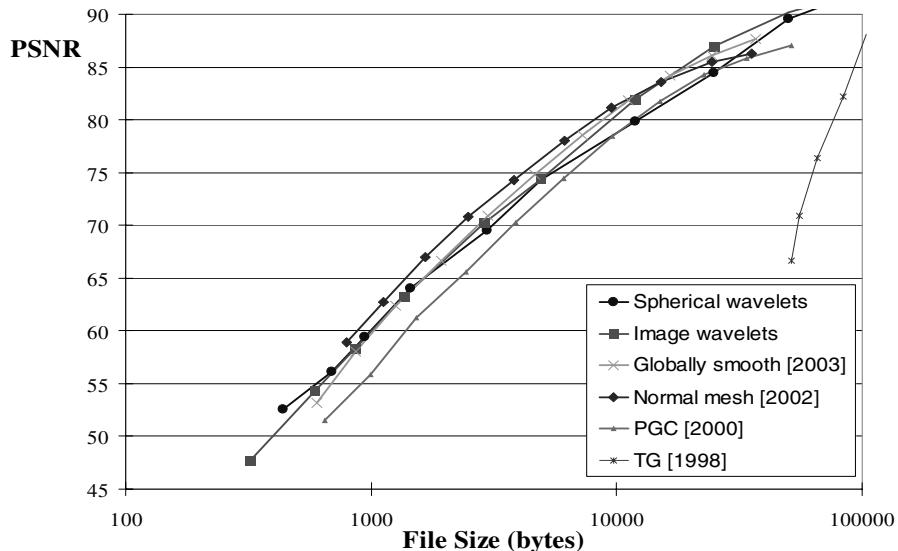
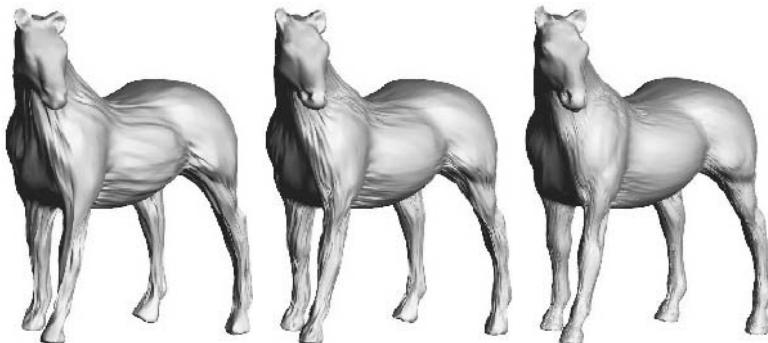
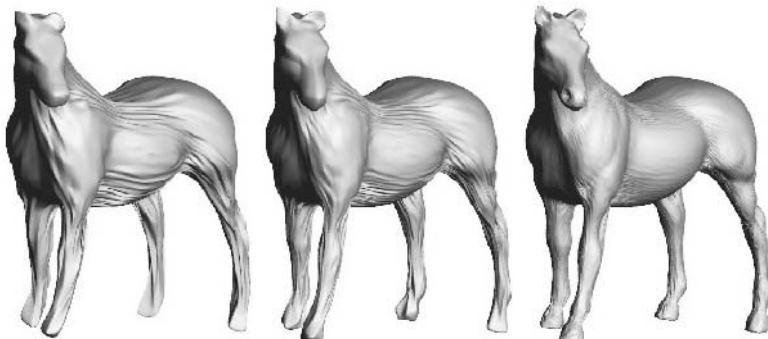


Fig. 11. [Reproduced in colour in Plate 9.] Compression results on rabbit model.



1,456 bytes (55.7 dB) 2,961 bytes (57.6 dB) 11,968 bytes (69.7 dB)
Compression using spherical wavelets.



1,376 bytes (54.2 dB) 2,900 bytes (60.6 dB) 11,932 bytes (73.1 dB)
Compression using image wavelets.

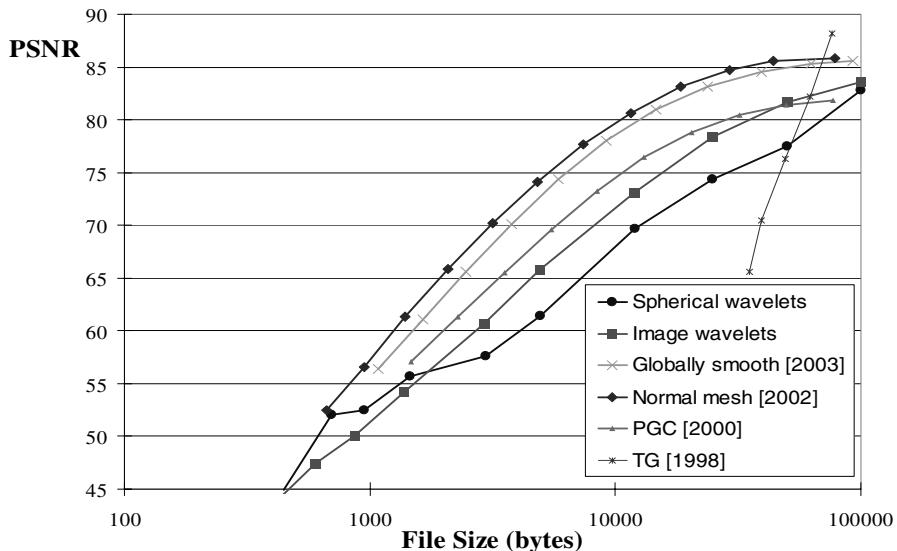


Fig. 12. [Reproduced in colour in Plate 10.] Compression results on horse model.

The results for both spherical wavelets and image wavelets are shown in Figs. 9–12. The rightmost images show each shape compressed to approximately 12,000 bytes. At this compression rate, the geometric fidelity is excellent, and these images should be considered as references for the more aggressive compressions to their left. At approximately 3,000 bytes (middle images), compression effects become evident in the blurring of sharp features. At 1,500 bytes (left images), effects are even more pronounced. It is interesting however that 1,500 bytes are generally sufficient to make the object recognisable.

The graphs in Figs. 9–12 show Peak Signal-to-Noise Ratio (PSNR) graphs for each model. We compare our rate-distortion curve with those of progressive geometry compression (PGC) [16], globally smooth parametrization (GSP) [18], and normal mesh compression (NMC) [17]. Also included for comparison is the irregular mesh compression scheme of Touma and Gotsman [28] (which preserves mesh connectivity). As can be seen from the graphs, our rate-distortion curves are generally better than PGC, but just below GSP and NMC.

The spherical wavelets generally offer better visual reconstruction, as is most evident on the skull model (Fig. 10). The reason is that the spherical wavelet kernels have more localised support than the particular image wavelets that we used, and therefore adapt more quickly to the fine detail. However, the PSNR graphs indicate that the error as measured using L^2 Hausdorff distance is generally lower when using the image wavelets. Thus, it can be argued that L^2 error is not an accurate visual norm [28], and that one should attempt to recover high-frequency detail first [26]. A more comprehensive comparison using other image wavelets (with more local support) would be useful.

The horse model (Fig. 12) shows a limitation of our spherical parametrization approach. For shapes containing many extremities, the parametrization onto the sphere suffers from distortion, and these distortions give rise to rippling effects under lossy reconstruction. In such cases, our compression is much less effective than semi-regular remeshing.

6 Summary and Future Work

We have described a spherical parametrization approach to remeshing genus-zero surfaces for shape compression. The surface is remeshed into a regular 2D grid of samples, which is then compressed using wavelets. The compression and decompression algorithm have great potential for hardware acceleration, since they do not involve any pointer-based data structures. We have presented a wavelet scheme based on ordinary 2D image wavelets, and applied it to the spherical domain using boundary extension rules, effectively creating spherical topology over a square domain. Like prior semi-regular remeshing schemes, our geometry image remeshing approach naturally supports progressive compression.

Experiments show that spherical geometry images are an effective representation for compressing shapes that parametrize well onto the sphere. Although the scheme is robust on arbitrary models, shapes with long extremities suffer from rippling artefacts during lossy decompression. One area of future work is to attempt to reduce these rippling effects by modifying the parametrization process.

Also, our approach should be extended to support surfaces with boundaries. One possibility would be to encode a separate bit-plane indicating which subset of samples lie in the “holes” of the remeshed model.

The accuracy of remesh representations can be improved by refitting to the original model as an optimisation (e.g. [23]). Such optimisation would likely help rate-distortion behaviour, particularly using an appropriate visual error norm. However, local geometry optimisation does increase the entropy of the “tangential” signal within the surface remesh, so it would be important to introduce a smoothing term to minimise such entropy away from significant geometric features.

In this work, we have used the sphere as an intermediate domain for parametrizing a surface onto an octahedron or flattened octahedron. One could also consider parametrizing the surface directly onto the octahedron, thus bypassing the sphere. This task would be more challenging, since the octahedron is not everywhere smooth like the sphere. However, it may allow the construction of improved (more stretch-efficient) parameterizations.

Briceño et al. [6] explore the compression of animated meshes using the geometry images of [11]. It would be interesting to apply a similar framework using spherical geometry images.

Acknowledgements

We gratefully thank Andrei Khodakovsky for providing results for our comparisons. We thank Cyberware for the Venus, rabbit and horse models, and Headus for the skull model.

References

1. Alliez, P., and Desbrun, M.: Progressive encoding for lossless transmission of 3D meshes. Proc. ACM SIGGRAPH 2001.
2. Alliez, P., and Desbrun, M.: Valence-driven connectivity encoding for 3D meshes. Proc. Eurographics 2001.
3. Alliez, P., and Gotsman, C.: Recent advances in compression of 3D meshes. *Advances in Multiresolution for Geometric Modelling*, N. A. Dodgson, M. S. Floater, and M. A. Sabin (eds.), Springer, 2004, pp. 3–26 (this book).
4. Antonini, M., Barlaud, M., Mathieu, P., and Daubechies, I.: Image coding using wavelet transform. IEEE Transactions on Image Processing, 205–220, 1992.

5. Attene, M., Falcidieno, B., Spagnuolo, M., and Rossignac, J.: SwingWrapper: Retiling triangle meshes for better EdgeBreaker compression. ACM Transactions on Graphics, to appear.
6. Briceño, H., Sander, P., McMillan, L., Gortler, S., and Hoppe, H.: Geometry videos. Symposium on Computer Animation 2003.
7. Davis, G.: Wavelet image compression construction kit. <http://www.geoffdavis.net/dartmouth/wavelet/wavelet.html> (1996).
8. Deering, M.: Geometry compression. Proc. ACM SIGGRAPH 1995, 13–20.
9. Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W.: Multiresolution analysis of arbitrary meshes. Proc. ACM SIGGRAPH 1995, 173–182.
10. Gotsman, C., Gumhold, S., and Kobbelt, L.: Simplification and compression of 3D meshes. In *Tutorials on Multiresolution in Geometric Modelling*, A. Iske, E. Quak, M. S. Floater (eds.), Springer, 2002, pp. 319–361.
11. Gu, X., Gortler, S., and Hoppe, H.: Geometry images. Proc. ACM SIGGRAPH 2002, 355–361.
12. Gumhold, S., and Strasser, W.: Real time compression of triangle mesh connectivity. Proc. ACM SIGGRAPH 1998, 133–140.
13. Guskov, I., Vidimce, K., Sweldens, W., and Schröder, P.: Normal meshes. Proc. ACM SIGGRAPH 2000, 95–102.
14. Hoppe, H.: Progressive meshes. Proc. ACM SIGGRAPH 1996, 99–108.
15. Karni, Z., and Gotsman, C.: Spectral compression of mesh geometry. Proc. ACM SIGGRAPH 2000, 279–286.
16. Khodakovsky, A., Schröder, P., and Sweldens, W.: Progressive geometry compression. Proc. ACM SIGGRAPH 2000.
17. Khodakovsky, A., and Guskov, I.: Normal mesh compression. Geometric Modeling for Scientific Visualization, Springer-Verlag, Heidelberg, Germany (2002).
18. Khodakovsky, A., Litke, N., and Schröder, P.: Globally smooth parameterizations with low distortion. Proc. ACM SIGGRAPH 2003.
19. Lounsbery, M., DeRose, T., and Warren, J.: Multiresolution analysis for surfaces of arbitrary topological type. ACM Transactions on Graphics, 16(1), 34–73 (1997).
20. Praun, E., and Hoppe, H.: Spherical parametrization and remeshing. Proc. ACM SIGGRAPH 2003, 340–349.
21. Rossignac, J.: EdgeBreaker: Connectivity compression for triangle meshes. IEEE Trans. on Visualization and Computer Graphics, 5(1), 47–61 (1999).
22. Rossignac, J.: 3D mesh compression. Chapter in *The Visualization Handbook*, C. Johnson and C. Hanson, (eds.), Academic Press, to appear (2003).
23. Sander, P., Wood, Z., Gortler, S., Snyder J., and Hoppe, H.: Multi-chart geometry images. Symposium on Geometry Processing 2003, 157–166.
24. Schröder, P., and Sweldens, W.: Spherical wavelets: Efficiently representing functions on the sphere. Proc. ACM SIGGRAPH 1995, 161–172.
25. Shapiro, A., and Tal, A.: Polygon realization for shape transformation. The Visual Computer, 14 (8-9), 429–444 (1998).
26. Sorkine, O., Cohen-Or, D., and Toledo, S.: High-pass quantization for mesh encoding. Symposium on Geometry Processing, 2003.
27. Taubin, G., Gueziec, A., Horn, W., and Lazarus, F.: Progressive forest split compression. Proc. ACM SIGGRAPH 1998.
28. Touma, C., and Gotsman, C.: Triangle mesh compression. Graphics Interface 1998.

Part II

— Data Structures

A Survey on Data Structures for Level-of-Detail Models

Leila De Floriani¹, Leif Kobbelt² and Enrico Puppo¹

¹ Department of Computer Science (DISI), University of Genova, Italy

{deflo|puppo}@disi.unige.it

² Computer Graphics Group, RWTH Aachen, Germany

kobbelt@cs.rwth-aachen.de

Summary. In this paper we survey some of the major data structures for encoding Level Of Detail (LOD) models. We classify LOD data structures according to the dimensionality of the basic structural element they represent into point-, triangle-, and tetrahedron-based data structures. Within each class we will review single-level data structures, general data structures for LOD models based on irregular meshes as well as more specialised data structures that assume a certain (semi-) regularity of the data.

1 Introduction

Due to the rapidly increasing complexity of three-dimensional data sets such as geometric free-form shapes, terrain models or volumetric scalar fields, the investigation of hierarchical methods to control and adjust the Level Of Detail (LOD) of a given data set has been an active research area – and it still is. A LOD model essentially permits different representations of an object at different levels of detail, where the level can also vary over the object. The operation of extracting one such representation from a LOD model is called *selective refinement* and most applications require that it must be supported either in real time, or at least on-line for objects that often exhibit fairly complicated geometry. Performance requirements impose several challenges in the design of systems based on LOD models where geometric data structures play a central rôle. There is a necessary trade-off between time efficiency and storage costs. And also there is a trade-off between generality and flexibility of models on one hand, and optimisation of performance (both in time and storage) on the other hand.

In this paper, we provide a survey of geometric data structures that have been proposed in the literature to implement LOD models. We consider free-form surfaces described through point primitives or triangle meshes and two-dimensional or three-dimensional scalar fields whose domain is partitioned

into meshes of triangles or tetrahedra. We consider mesh-based LOD models described by hierarchies of meshes, which define the underlying discretisation of the surfaces or of the domain of the field. In Sect. 2, we introduce some background notions on meshes and we give a generic definition of LOD models. In Sect. 3, we review point-based data structures that process surface data without the need to construct a mesh explicitly. In Sect. 4, we focus on 3D objects represented with meshes of triangles and we describe triangle-based data structures in the three cases of plain geometric models, progressive models, and LOD models supporting selective refinement. In Sect. 5, we review, with a similar approach, tetrahedron-based data structures used to represent volume data. Sect. 6 makes some concluding remarks.

2 Background Notions

In this Section, we introduce some background notions which we use in the paper.

2.1 Meshes

A *k-dimensional cell*, or a *k-cell*, for brevity, is a subset of the d -dimensional Euclidean space E^d homeomorphic to a closed k -dimensional ball, where $k \leq d$. Let M be a connected finite set of cells of heterogeneous dimension embedded in the Euclidean space E^d , where d is the maximum of the dimensions of the cells of M , such that the boundary of each cell in M is a collection of cells of lower dimensions, called *facets*, belonging to M . Then, M is a *d-dimensional mesh* if and only if the interiors of any pair of d -dimensional cells of M are disjoint, and any k -cell of M , with $k < d$, bounds at least one d -cell of M . The union as a set of points of the cells of a mesh M is called the *carrier*, or the *domain* of M .

A special and interesting case of meshes is that of *simplicial meshes*. A *k-dimensional simplex*, or *k-simplex*, for brevity, in E^d is the locus of the points in E^d that can be expressed as the convex combination of $k+1$ affinely independent points. A *simplicial mesh* Σ is a mesh in which all cells are simplexes. In a d -dimensional simplicial mesh, every k -simplex with $k < d$ is generated by a subset of vertices of some d -simplex. We call the set of simplexes bounded by simplex σ the *star* of σ .

A mesh is said to be *conforming* if and only if, for each pair of d -cells σ_1 and σ_2 , the intersection of the boundaries of σ_1 and σ_2 is either empty, or it consists of a k -facet belonging to the boundary of both σ_1 and σ_2 , for some $k < d$. The use of conforming meshes as decompositions of the domain of a scalar field, which is sampled at a finite set of points on a manifold, provides a way of ensuring at least C^0 continuity for the resulting approximation, without requiring the modification of the values of the field at the facets where discontinuities may arise.

The *valence* of a d -cell C in a mesh M is the number of $(d + 1)$ -cells also in M that overlap C . The most important examples are the vertex valence in a triangle mesh being the number of edges that meet at a vertex or the edge valence in a tetrahedral mesh being the number of triangles meeting at an edge.

We call *regular grids* conforming meshes in which all cells are hypercubes (squares or cubes in two and three dimensions, respectively) and all d -cells have the same valence respectively. Moreover, we call *regular meshes* those meshes which are defined by the uniform subdivision of a d -cell into scaled copies of it. Note that the vertices of a regular mesh are a subset of the vertices of a regular grid and that all sub-cells with a given dimension d have the same valence v_d .

A *semi-regular* mesh is generated by starting with an irregular mesh and applying a uniform refinement operator to each cell. The uniform refinement does not generate new vertices or edges with irregular valences nor does it change the valences of existing elements. As a consequence, a semi-regular mesh is piecewise regular and only has some isolated irregular (a.k.a. extraordinary) vertices or edges which correspond to the elements of the original (unrefined) mesh.

2.2 Elements of a LOD Model

The basic ingredients of a LOD model of a spatial object are a *base mesh*, that defines the coarsest approximation to the object, a set of *updates*, that, when applied to the base mesh, provide variable resolution mesh-based representations of the spatial object, and a *dependency relation* among updates, which supports combining them to extract consistent intermediate representations.

Intuitively, an *update* on a d -dimensional mesh M consists of replacing a set of cells in M with another set in such a way that the result M' is still a mesh (see [14] for a formal definition). An update is conforming if, when applied to a conforming mesh, it produces a conforming mesh as result. An update is either described explicitly, as the set of cells involved in it, or implicitly, by encoding the operation which produces it. The *dependency relation* can be a containment hierarchy, or a representation of the possible orders in which updates can be performed to extract meshes at different resolutions. We call the mesh at full resolution *reference mesh* which can be obtained by applying all available updates to the base mesh.

The basic operation in a LOD data structure, called *selective refinement*, consists of extracting a conforming mesh satisfying some application-dependent requirements based on level of detail, such as approximating a surface or scalar field with a certain accuracy which can be uniform or variable in space. We assume a Boolean function τ , that we call a *LOD criterion*, defined on the set of nodes in a LOD model, which returns a value *true* if an entity satisfies the requirements, a value *false* otherwise. The general *selective refinement query* on a LOD model can thus be formulated as follows: given a

LOD criterion τ , extract from the model the mesh M of minimum size that satisfies τ .

3 Point-Based Data Structures

The most primitive geometric object is a *point*. Yet we can (approximately) represent an arbitrarily complex geometric object if we use a sufficiently dense set of point samples. Modern 3D scanning devices such as laser range finders often provide dense point clouds as their raw data output.

Points scattered in a k -dimensional region that carry optional attribute values, can be considered as samples of a scalar- or vector-valued function defined over this region. Obviously, we can distinguish the irregular case, where samples are randomly scattered within the k -dimensional region, from the regular case, where the samples are aligned to the nodes of a regular grid.

The easiest (mesh-less) way to reconstruct a continuous approximation of the sampled function is to convolve the point set with a (weighted) reconstruction kernel, e.g., a radial basis function. This reconstruction, however, does not provide any explicit structure and is mostly used for algorithms whose only access to the geometry data is by function evaluation.

Another (more explicit) reconstruction technique is to generate a tessellation of the k -dimensional region, e.g., into k -simplices with the points as their vertices. In such a tessellation the function values at the sample points are usually interpolated linearly across cells in order to obtain a piecewise linear approximation of the sampled function.

A different situation occurs when the samples are taken from a k -dimensional manifold embedded in $\mathbb{R}^{(k+1)}$. In this case we can still construct a k -dimensional tessellation but additional topological information is necessary to determine the proper (geodesic) neighbourhood relation between the samples [51]. If the underlying manifold is topologically equivalent to a k -dimensional disk, we can find a map of the sample points to some region in \mathbb{R}^k , construct a tessellation in \mathbb{R}^k , and then use the same topological graph structure for the samples embedded in \mathbb{R}^k to obtain a piecewise linear approximation of the manifold in $\mathbb{R}^{(k+1)}$.

The general goal in the design of point-based data structures is to efficiently process large sets of points scattered in \mathbb{R}^k without explicitly constructing a simplex tessellation. Since in this case no topological consistency criteria (like conformity) have to be checked and enforced, many operations such as re-sampling become much more efficient.

When designing data structures for point sets one is usually interested in space partitions that are optimised for spatial queries since finding the closest n neighbours to a given point in space or finding all neighbours within some ε -ball are the most frequent operations. As a consequence octrees and binary space partitions are most common [63].

Space partitions are mostly built top-down. In order to optimally balance the resulting tree hierarchies, one usually prefers *median cut* techniques where each node is split such that both children contain the same number of sample points instead of splitting the node into sub-cells with equal volumes. The optimal orientation of the splitting (hyper-) plane is usually found by computing the eigenstructure of the inertia tensor (assuming that the samples represent unit masses distributed in space) [59].

By carefully analysing the approximation properties of discrete point samples (= piecewise constant functions) it turns out that the approximation error to a smooth function or manifold is proportional to the distance between samples. Hence, in contrast to triangle meshes, the sampling density for point-based representations is determined by surface area and not by surface curvature [6]. The minimum redundancy in a point-based representation is obtained when the coordinate quantisation for the sample points is proportional to their density.

A convenient way to link the quantisation precision to the sampling resolution is to discretise the space surrounding the given object into a uniform voxel grid with grid size h and then label all voxels which the surface passes through as *full* while all other voxels are *empty*. If we place sample points in the centres of the full voxels then each sample is as most $\sqrt{3} h/2$ away from the exact surface while the maximum distance between neighbouring samples is h .

If $h = 2^{-r}$ for some r then the set of full voxels (and hence the set of samples) can be stored as the leaves of an r -level octree. In [6], a very efficient storage and processing technique for such hierarchical point clouds is presented which only needs 2.67 bits per point (uncompressed) and usually less than one bit per point if the data set is compressed by entropy encoding.

In order to guarantee a visually continuous appearance when displaying a point-based geometry representation, the points are usually generalised to *splats*, i.e., infinitesimal points are replaced by ellipses or rectangles – either in object space or image space [72, 6]. Although surface splats require the derivation of attributes such as orientation and radius for each splat, the conceptual simplicity of point-based representations is largely preserved. However, in a strict sense, we are dealing with a piecewise linear surface representation just like triangle meshes with the important difference that the linear pieces join in a C^{-1} fashion rather than C^0 .

4 Triangle-Based Data Structures

In this section, we review data structures for describing models based on triangle meshes. In Sect. 4.1, we review data structures for encoding a triangle mesh. In Sect. 4.2, we describe data structures for progressive models. A survey on progressive models can be found in [31]. Sect. 4.3 describes data structures

for LOD models based on regular meshes, while Sect. 4.4 presents data structures for LOD models based on irregular meshes. A survey on triangle-based LOD models can be found in [14]. An extensive treatment of LOD models for view-dependent visualisation and of their applications can be found in [47].

4.1 Data Structures for Triangle Meshes

There are essentially two classes of data structures for triangle meshes, those in which edges are encoded as primary entities, called *edge-based representations*, and those in which triangles are considered as primary entities, called *triangle-based representations*. In describing and analysing these data structures, we will only consider structural information, while disregarding geometric information (which generally consists just of vertex coordinates) and attribute information (which is application dependent).

Edge-based representations have been developed for general meshes and can be used for triangle meshes without any change. The *winged-edge data structure* originally proposed to represent polyhedra with polygonal faces [3], explicitly encodes all entities forming the mesh. For each edge e , the data structure maintains the indexes of the two extreme vertices of e , of the two faces bounded by e , and of the four edges that are both adjacent to e and are on the boundary of the two faces bounded by e . Vertices and faces are also encoded: for each vertex v , a pointer to any edge incident in v , and for each face f , a pointer to an edge bounding f is maintained. Links emanating from edges support efficient navigation of the mesh by traversing either the boundary of a face or the star of a vertex, in either counterclockwise or clockwise order. By assuming the size of an index as a unit, the total storage cost (disregarding geometric and attribute information) is about $27n$, where n is the number of vertices in the mesh.

A simplified version of the winged-edge data structure is the *Doubly-Connected Edge List (DCEL)* [50] where, for each edge, indexes of just two adjacent edges are maintained. This data structure has a lower cost (about $21n$), but it allows traversing the star of vertices only counterclockwise, and the boundary of faces only clockwise.

A widely-used data structure is the *half-edge data structure* [48] in which two copies of the same edge are encoded, oriented in opposite directions. For each edge of the mesh, the data structure stores two half-edges, and information about the edge is split between such two halves. Each half-edge stores the index of its origin, the index of the face on its left, two indexes of its previous and next edges along the boundary of that face, and the index of its twin half-edge, which is oriented in the opposite direction. Information attached to vertices and faces are the same as in the winged-edge and in the DCEL structures. In the implementation discussed in [64], the cost of the half-edge data structure is equal to that of the winged-edge.

In [8] an efficient implementation of the half-edge data structure is proposed that combines the advantages of the pointer-based and the index-based

approaches by grouping the three half-edges belonging to the same triangle in the subsequent entries $E[3i]$, $E[3i+1]$, $E[3i+2]$ of a global array. By this the next and previous pointers can be replaced by simple index computations *modulo* 3 while the index of the corresponding triangle is obtained through integer division by 3. As a consequence, each half edge requires only two pointers: one to its starting vertex and one to its opposite half edge. In addition each vertex has a pointer to one incident half-edge. The triangle to half-edge pointer can be replaced with an index multiplication by 3. This sums up to 13 pointers per vertex just as for indexed triangle structures. Yet, the data structure enables the simple mesh navigation provided by half-edge data structures.

In [5] a comprehensive public source implementation of a half-edge data structure for manifold polygon meshes is described. Since the data structure allows for arbitrary n -sided faces, it cannot group the half-edges per face as in [8]. As an alternative it groups corresponding half-edges $E[2i]$, $E[2i+1]$ such that the opposite half-edge pointer can be replaced by index computations *modulo* 2. What remains are three pointers per half-edge: starting vertex, left face, next half-edge. A (full) edge is implicitly represented by a group to two successive half-edges and hence does not require any storage. Vertices and faces each store one pointer to an associated half-edge. In total this sums up to about 21 pointers per vertex if most of the faces in the mesh are triangles. The complete source code including some sample applications can be downloaded at [1].

The most common triangle-based data structure is the so-called *indexed data structure*: for each triangle t , three indexes to its vertices are maintained. The cost of this data structure (disregarding geometry and attributes) is just about $6n$, where n denotes the number of vertices. The indexed data structure has been extended in order to support mesh traversal through edges by storing, for each triangle t , also the indexes to the three triangles adjacent to t [39]. The resulting data structure is called the *indexed data structure with adjacencies*. To support an efficient traversal of the mesh passing through vertices, it is convenient to attach to each vertex of the mesh a pointer to one of its incident triangles. The storage cost of the extended incidence data structure with adjacencies (disregarding geometry and attributes) is $13n$.

4.2 Progressive Meshes

Progressive Meshes (PM) [33] are built by considering a sequence of edge collapses that simplify a mesh step by step. An *edge collapse* consists of contracting an edge e to a vertex v , which can either be a new vertex (*full-edge collapse*) or one of the extreme vertices of e (*half-edge collapse*). Given a mesh M at high resolution, edge collapses are applied until a drastically simplified mesh M_0 is obtained. The whole sequence of edge collapses is recorded and, for each of them, the corresponding reverse update (called a *vertex split*) is stored (see Fig. 1). A PM representation of a mesh M hence consists of a

base mesh M_0 plus the reverse sequence of vertex splits. M can be obtained from M_0 through progressive refinement by applying all vertex split updates in the sequence. If just a prefix of the sequence is applied then a model at an intermediate LOD is obtained.

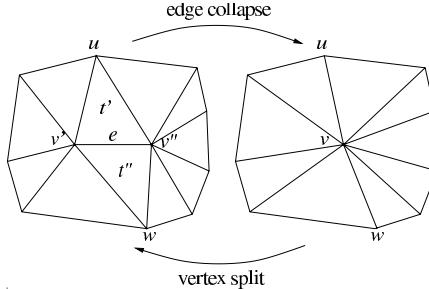


Fig. 1. Edge collapse and vertex split updates.

In a PM a vertex split is encoded as follows (refer to Fig. 1 for notation):

- a pointer to the vertex v to be split;
- the positions of vertices v' and v'' : if either one of them coincides with v , or v is the midpoint of e , then one offset vector is sufficient to specify them both;
- a pointer to u : if the vertices incident at v are arranged in a conventional order, it is sufficient to store an offset to find u in such a vertex list.
- an offset to find w by rotating around v starting at u .

It has been shown that encoding a PM requires less space than encoding the mesh M at full resolution in an edge-based data structure [35]. In [56], an alternative representation for PMs is described, in which vertex splits are clustered into batches with the advantage of doubling the compression ratio.

4.3 LOD Data Structures for Regular and Semi-regular Meshes

LOD data structures for regular triangle meshes have been proposed in the literature, for the case of terrain data and, more generally, for bivariate scalar fields, where data are sampled at the nodes of a regular grid [20, 23, 25, 43, 54].

To allow for some flexibility with respect to the topology of the base domain, regular meshes are usually handled as a special case of semi-regular meshes. Since conceptually these meshes can be considered as being generated by the recursive application of some basic refinement operator, the natural data structures for such meshes are *trees*. In some cases when the association of mesh elements from the $(n+1)$ -st refinement level with elements from the n th level is not unique, more general data structures such as *directed acyclic graphs* are necessary.

The most basic refinement operator for triangle-based data structures is the quadriseciton where on each edge a new vertex is inserted and each original triangle is split into four sub-triangles (cf. Fig. 2). The resulting (nested) hierarchy of triangles can be stored straightforwardly in a quadtree. The neighbourhood search in this data structure after r refinement steps is $O(r)$ in the worst case but only $O(1)$ in the average case. Neighbour finding algorithms that work in worst-case constant time by using arithmetic operations and bit manipulation are presented in [42].

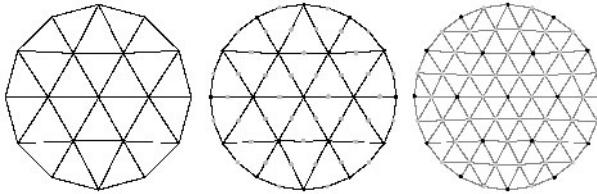


Fig. 2. Mesh refinement by quadriseciton.

In order to avoid the exponential growth in the number of triangles, one usually tries to apply the subdivision operator only to those triangles that do not meet a certain quality criterion. This selective refinement, however, leads to non-conforming meshes if quadriseciton is used locally [40]. The non-conforming edges can be fixed by introducing temporary triangle bisections that are undone once further refinement is necessary (red-green triangulation) [2].

An alternative refinement operator introduces new vertices per triangle (and not per edge). In this case each triangle is split into three sub-triangles leading to a situation where all new vertices have valence three while the valences of the original vertices are doubled. In order to make this refinement operator a uniform one, we have to flip all original edges. By this we guarantee that all newly introduced vertices have valence six and all other vertices keep their original valence (cf. Fig. 3). This operator is called $\sqrt{3}$ subdivision, since the double application produces a mesh that corresponds to a uniform 3-nary refinement of the original.

Because the $\sqrt{3}$ -operator uses only 1-3-splits of triangles and edge flips it never generates non-conforming meshes – even if it is applied selectively. This is why adaptive refinement strategies are much easier to implement based on this refinement operator [38].

Several other data structures for regular LOD models are based on recursive *triangle bisection*. A square universe S is initially subdivided in two right triangles. The bisection rule subdivides a triangle into two similar triangles by splitting it at the midpoint of its longest edge. The recursive application

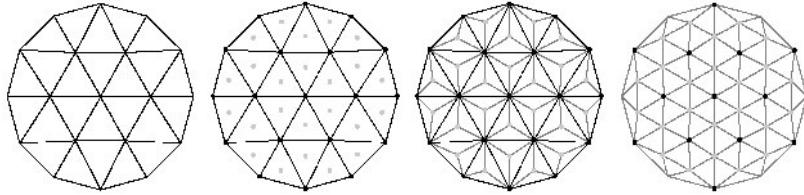


Fig. 3. Mesh refinement by the $\sqrt{3}$ -operator.

of this splitting rule to S defines a binary tree of right triangles, in which the children of a triangle t are the two triangles obtained by splitting t .

In order to extract conforming meshes, every inserted vertex must be introduced into two adjacent triangles at the same time (except for vertices at the boundary). Any pair of triangles which need to be split at the same time forms a *diamond*. Each diamond D is split into four triangles (as shown in Fig. 4) by the vertex v , called the *split vertex* of D , which bisects the edge shared by the two triangles forming D . The four triangles splitting a diamond form a *split set*.

If we consider the split vertex of each diamond, a dual hierarchy of vertex dependencies can be derived, which is a Directed Acyclic Graph (DAG) where each vertex v has exactly two parents (namely, those two vertices that generate the parents of triangles forming the split set centred at v ; vertices at the boundary have only two children) and exactly four children (namely, those four vertices that split triangles forming the split set centred at v). An example of such a hierarchy is depicted in Fig. 5. In [14, 61] an interpretation of this model in the framework of the Multi-Triangulation, which is described in the next section, is presented.



Fig. 4. Split sets corresponding to the two types of diamonds generated by recursive triangle bisection.

The data structures for representing nested meshes produced through triangle bisections either encode the DAG of the vertex dependencies, or they encode a forest of triangles which describes the nested structure of the meshes. The DAG of vertex dependencies can be traversed easily to perform selective refinement while guaranteeing that the result is a conforming mesh [43, 54].

The binary forest of triangles can be traversed to perform selective refinement, but neighbour finding is necessary to guarantee that when a triangle t is split, also the triangle adjacent to t along its longest edge is split at the same

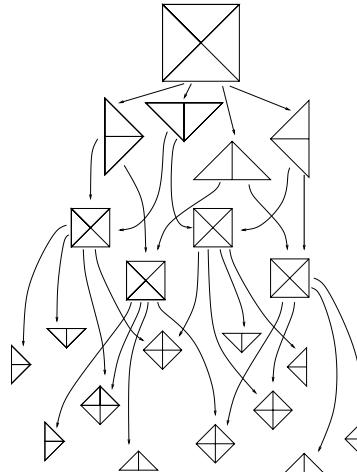


Fig. 5. The hierarchy of vertices, where each node represents both a vertex and its corresponding diamond (diamonds on the border contain just two triangles).

time. In [53] a mechanism is proposed, which is based on error saturation, that can extract conforming meshes by a simple traversal of the binary forest, without neighbour finding.

If vertices are available at all nodes of the supporting regular grid, then the forest of triangles is complete, and it can be represented implicitly [20, 23]. Each triangle can be assigned a location code, with a mechanism similar to that adopted for linear encoding of triangle quadtrees [42]. Location codes are not stored, but they are used for neighbour finding as well as to retrieve the vertices of a triangle, the value of the field associated with a vertex, etc. An efficient implementation involving arithmetic manipulation and a few bit operations allows performing such computations in constant time [23]. On the contrary, if the grid of vertices, and thus, the binary forest of triangles, are not complete, then the forest must be represented explicitly, thus resulting in a more verbose data structure [25].

In [25, 44], out-of-core implementations are proposed, the former oriented to representing non-complete hierarchies in terrain modelling, the latter targeted at visualisation and view-dependent refinement of large terrain data sets.

4.4 LOD Data Structures for Irregular Meshes

The data structures for encoding LOD models based on irregular triangle meshes can be classified into *explicit data structures*, which represent the triangles in the model, and into *implicit data structures*, which do not encode the triangles, but a procedural description of the operations through which

the LOD model is generated, i.e., edge collapse/vertex split, or vertex insertion/vertex removal.

Explicit data structures

The *Multi-Triangulation (MT)*, as proposed in [61], has been conceived as a general framework for triangle-based LOD models, which is independent of the specific type of operation used for generating it. For instance, all models described in the previous sections are special cases of the MT [14].

The general idea is to have a mesh M which is simplified through a sequence of generic updates into a base mesh M_0 , or, conversely, mesh M_0 is refined into M through an inverse sequence of refinement updates. A generic (refinement) update u is encoded as a pair (u^-, u^+) of sets of triangles. Applying an update u to a mesh M' consists of deleting the triangles of u^- from M' and replacing them with the (larger) set of triangles of u^+ . The dependency relation is defined among updates as follows: an update u is dependent on another update w if and only if u deletes some triangle introduced by w , i.e., $w^+ \cap u^- \neq \emptyset$. This rule imposes a strict partial order on the set of updates, which can be represented as a DAG. The partial order encodes all and only those dependencies that are necessary to perform either refinement or coarsening updates in the same situation in which they were performed during construction, thus guaranteeing that the result is a conforming mesh. Selective refinement operations are based on DAG traversal [15].

An implementation of the MT consists of representing the base mesh M_0 through an indexed data structure, and the DAG through a set of nodes, in which each node contains a description of the triangles corresponding to sets u^- and u^+ . Each triangle is described by the three indices of its vertices. Selective refinement can be performed very efficiently on such data structure, without the need for explicitly modifying the currently extracted mesh through operations involving adjacencies. On the other hand, efficiency in selective refinement leads to higher storage costs. A relatively compact implementation of the MT through arrays is described in [15], which requires about four times the space required by an indexed representation of the mesh M at full resolution.

An explicit LOD triangle-based data structure has been proposed by Luebke and Erikson [46] for view-dependent rendering. Each triangle t in the LOD data structure is stored as three references to the vertices of the mesh at full resolution (reference mesh) from which the vertices of t derive. During selective refinement, each vertex v of the reference mesh is associated with the ancestor of v that is the currently extracted mesh. Triangles for which two or more vertices are associated with the same vertex are degenerate, and thus they are not rendered.

A special case of an MT for semi-regular meshes, called a *hierarchical 4-k mesh*, has been proposed in [68, 69] in the context of subdivision surfaces. A hierarchical 4-k mesh is based on vertex insertion on an edge and on edge

swap, and it is described by a simplified DAG in which each node has exactly two parents, and either two or four children, as in the case of nested regular meshes generated through triangle bisection.

Data structures based on edge collapse/vertex split

LOD data structures based on edge collapse/vertex split extend progressive meshes by allowing extracting variable resolution representations in which the LOD varies in different parts of the object.

A naïve approach to extracting variable resolution meshes from a PM would consist in scanning the sequence of vertex splits and performing only those updates that are necessary to achieve the desired LOD according to the refinement criterion τ , while skipping all others [33]. However, a vertex split cannot be performed unless the corresponding vertex actually belongs to the current mesh. If a split is skipped, this prevents splitting the vertices it generates (which could be considered for split at a later stage) and all their descendants. This may lead to an under-refinement of the mesh.

To overcome the above problem, a dependency among vertices, which is represented through a binary forest of vertices [34, 70], is defined: roots of the forest are vertices of M_0 and the two children of a vertex v are the vertices, v' and v'' , generated by splitting v . Each node in the forest contains information necessary to perform a vertex split, together with information that subsumes the content of its subtree (such as the bounding sphere of the region spanned by all vertices in the subtree, and the maximum approximation error for such vertices) that are used to decide whether one split must be performed or not.

In [36] a technique is presented which even allows for re-ordering the vertex splits arbitrarily while still guaranteeing a proper mesh connectivity which corresponds to the original PM connectivity whenever a complete prefix of the vertex splits is executed.

However, this simple data structure does not guarantee that a split will be performed in the same situation, i.e., the vertex to be split might not have the same star of triangles as when it was created in the original simplification sequence. This may cause undesirable fold-overs in the mesh (see Fig. 6). The solution is to encode further dependencies for a vertex v so that v can be

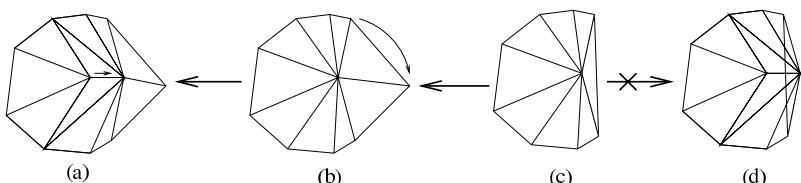


Fig. 6. Foldover: original sequence of collapses is (a)–(b)–(c) and the reverse sequence of splits is correct; if the central vertex is split as in sequence (c)–(d) a foldover occurs.

safely split: vertex v will depend on a set of vertices which completely define the triangles incident at v produced by the edge collapse which generated v in the simplification sequence [34, 70]. In [14], it has been shown that in some cases such additional dependencies can be either insufficient to prevent foldovers, or redundant, thus making the data structure quite verbose [70].

In [21], an effective mechanism based on vertex numbering is introduced, which guarantees that all splits and collapses identified during selective refinement will be performed in the same situation as in the original simplification sequence, thus avoiding foldovers. Vertices are assigned integer labels during construction. In order to check whether or not a split [collapse] can be performed, it is sufficient to compare the label of the vertex to be split [extreme vertices of the edge to collapse] with the labels of its [their] adjacent vertices. This is also sufficient to propagate necessary splits/collapses through the hierarchy, in order to support dynamic adjustment of the selectively refined LOD. The resulting data structure, called a *view-dependent tree*, is very compact. Each internal node of the forest corresponds to a vertex created through edge collapse and contains information necessary to perform the corresponding vertex split, encoded in the same way as in progressive meshes. Three pointers are then maintained to encode the forest. The mechanism proposed in [21] is valid for hierarchies built through full-edge collapse (i.e., when vertex v is different from both v' and v''). A new version of the view-dependent tree to encode models based on half-edge collapse (i.e., where vertex v coincides with either v' or v'') is presented in [13]. An external memory data structure based on a block partition of the view-dependent tree is described in [22].

A different edge-based LOD data structure, called a *FastMesh*, has been proposed in [55], which is specific for LOD meshes generated through half-edge collapse. In FastMesh, a variant of the half-edge data structure is used to encode the triangle mesh, and a forest of half-edges is stored, where each node represents a half-edge collapse operation. The forest of half-edges requires half the number of nodes compared to a binary forest of vertices. In [18], an external memory data structure based on a FastMesh has been described.

Data structures based on vertex insertion/removal

Vertex insertion in a mesh M consists of deleting a connected set of triangles from M , which defines the region of influence of the vertex v to be inserted, and replacing it with a set of new triangles all incident at v (see Fig. 7a). The region of influence is defined by the specific mesh refinement algorithm (a technique often based on incremental Delaunay triangulation). *Vertex removal* consists of removing vertex v together with all the triangles in the star of v and re-triangulating the resulting star-shaped polygon, called the *influence polygon* of v (see Fig. 7b).

Encoding a vertex insertion requires storing the vertex v to be inserted as well as the subdivision of the influence region into triangles. This provides also an encoding of vertex removal, since we need to know how to re-triangulate

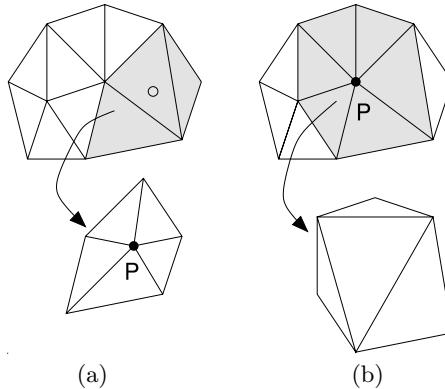


Fig. 7. (a) Vertex insertion; (b) Vertex removal.

the influence polygon after deleting a vertex v and all the triangles in the star of v .

An implicit encoding for vertex insertion/removal and a compact DAG encoding are described in [37]. The DAG encoding is based on connecting the updates, on which a given update u depends on, in a loop containing u plus all its direct ancestors. Thus, an update u with j direct descendants belongs to $j + 1$ loops: the one associated with u plus those associated with its direct descendants. The total number of links to describe the DAG is equal to $a + h$, where a and h denote the number of arcs and of nodes in the DAG, respectively.

In [37] a method for encoding the influence region is proposed which consists of storing a table with all the possible triangulations (up to rotations) of a polygon with s edges (for instance, for $s = 10$, there are 7147 equivalence classes), and identifying the triangulation of the influence polygon by specifying an index in the table. The length of the index is equal to 13 bits, by assuming that there are at most 10 triangles incident at a vertex in any update. The table with the equivalence classes must be stored as well.

There are several ways for compactly encoding a simple polygon, mainly developed in the context of mesh compression algorithms. The analysis performed in [11] showed that good performances can be obtained by encoding the triangulation of the influence polygon through a technique proposed by Taubin et al. [66]. Such an encoding consists of a bit stream, which is generated through a recursive procedure that visits the region of influence R and encodes one triangle $t \in R$ at a time by starting with an edge of its bounding polygon. The average length of the bit stream is equal to 8 bits, since the number of bits in the bit stream is twice the number of triangles in the region of influence. In [11], a compact encoding for the initial edge is described, which requires just 10 bits. The storage cost of the above data structure combined with the compact DAG encoding by Klein and Gumhold requires about

70% of the space required by an indexed representation of the reference mesh, which becomes 30% when this latter is encoded as an indexed data structure with adjacencies.

5 Tetrahedron-based Data Structures

In this section, we present data structures developed for models of volume data sets described by tetrahedral meshes. A *volume data set* consists of a set of points in the three-dimensional Euclidean space, and of a collection of field values associated with the points of V . In Sect. 5.1, we briefly review data structures for tetrahedral meshes, while in Sect. 5.2, we describe data structures for progressive models. Sect. 5.3 describes data structures for LOD models based on regular meshes, while Sect. 5.4 presents data structures for LOD models based on irregular meshes.

5.1 Data Structures for Tetrahedral Meshes

Data structures for tetrahedral meshes are a direct extension of triangle-based data structures (see Sect. 4.1) to the three-dimensional case. The indexed data structure and the indexed data structure with adjacencies [52, 57] are defined in a completely similar way as in the case of triangle meshes. If n denotes the number of vertices and n_t the number of tetrahedra of a tetrahedral mesh M , encoding connectivity information in an indexed data structure requires $4n_t$ indexes. Since n_t is approximately equal to $6n$ and we assume the size of an index to be a unit, the total storage cost of the indexed data structure is equal to $24n$ (disregarding geometric and attribute information). Encoding the connectivity in an indexed data structure with adjacencies requires $8n_t$ indexes. This leads to a total cost of $48n$, that becomes $49n$, if, for each vertex v , we also encode an index to one of its incident tetrahedra.

A tetrahedral mesh can also be represented as a simplified incidence graph [7]. In such representation, the following information are encoded: for each vertex v , a pointer to one edge incident at v ; for each edge e , the indexes of the two extreme vertices of e , and of one of the triangles sharing e ; for each triangle t , the indexes of the three edges bounding t , and the indexes of the two tetrahedra sharing it; for each tetrahedron σ , the indexes of the four triangles bounding σ . The storage cost is about twice as the cost of the indexed data structure with adjacencies. The advantage of a simplified incidence graph is representing all cells in a tetrahedral mesh explicitly, which permits the attachment of attribute information to edges and triangles as well.

Data structures designed for encoding three-dimensional cell complexes like the *Facet-Edge* [19] and the *Handle-Face* data structures [45] could also be used to encode tetrahedral meshes. Both the Facet-Edge and the Handle-Face data structures describe the three-dimensional cells in the complex implicitly, by encoding the two-manifold complexes that form the boundary of

such cells. The space requirements of the Facet-Edge and the Handle-Face data structures, when applied to tetrahedral meshes, are equal to $n_t + 18n_f$ and $78n_t$ indexes, respectively, where n_t is the number of tetrahedral and n_f the number of faces. Thus, they are definitely higher than the requirements of the indexed data structures r of the simplified incidence graph.

5.2 Progressive Tetrahedral Meshes

Progressive Tetrahedral Meshes (PTMs) encode a coarse mesh plus a linear sequence of updates that can be applied to the mesh in order to progressively refine it [24, 29, 56, 60, 67]. These models support the extraction of a mesh only at those intermediate resolutions that can be obtained by truncating the sequence of refinements at some point.

A PTM is built from the reference mesh through a sequence of full-edge collapses, which produces the base mesh, i.e., the coarsest tetrahedral approximation. A full-edge collapse consists of contracting an edge $e = (v', v'')$ to a vertex v which is an internal point of edge e , for instance, its midpoint (see Fig. 8). Like a PM, a PTM is formed by the base mesh M_0 plus the reverse sequence of full-vertex splits. A full-vertex split expands vertex v into edge $e = (v', v'')$, and partition the tetrahedra in the star of v into tetrahedra incident on v' or on v'' (see Fig. 8).

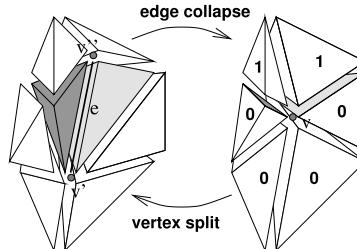


Fig. 8. Modification of a tetrahedral mesh through a full-edge collapse and a full-vertex split. On the left, tetrahedra that degenerate into triangles after full-edge collapse are shaded. On the right, tetrahedra marked with 0 and with 1 result from the deformation of tetrahedra incident at v' and at v'' , respectively.

Each full-vertex split is encoded in a PTM [29] by specifying split vertex v plus the fan of triangles incident at v which will be expanded into tetrahedra as effect of the split. Such triangles are called *cut faces*. The split vertex can be identified using $\log_2(n)$ bits in a tetrahedral mesh with n vertices, the cut faces can be identified locally with respect to the split vertex. Since a vertex in a tetrahedral mesh has about 36 incident faces, the cut faces can be encoded with roughly $6\log_2(36)$ bits.

In [56], a more compact encoding for PTMs is proposed for efficient mesh compression and transmission. Such compressed format consists of a base

mesh plus a series of batched vertex splits, called *implant sprays*, that are performed simultaneously to achieve the next level of detail. The cut faces are encoded with about 15 bits for each edge collapse. The total cost of the connectivity encoding of such PTMs has been evaluated to be less than 6 bits for each tetrahedron in the reference mesh, that is, less than 36 bits per vertex.

5.3 LOD Data Structures for Regular Meshes

In the finite element and computer graphics literature, there has been a burst of research on nested tetrahedral meshes generated through tetrahedron refinement.

The class of red/green triangulation methods subdivide a tetrahedron σ into eight tetrahedra, four of which are obtained by cutting off the corners of σ at the edge midpoints and are congruent with the parent. The remaining four tetrahedra are obtained by splitting the octahedron resulting from cutting σ [4, 30, 65]. There are different ways of splitting such octahedron which may result in a different number of congruent tetrahedral shapes. A tree of tetrahedra is used to describe the nested structure of such meshes. To reduce the number of congruent shapes generated by the subdivision process, a domain partition technique into tetrahedra and octahedra has been introduced in [28]. A tetrahedron σ is partitioned into four congruent tetrahedra as before and into an octahedron, which is in turn subdivided into six octahedra and eight tetrahedra. Greiner and Gross [28] show that this refinement rule generates only two congruent shapes. As in the case of 2D meshes, if selective refinement is performed, irregular refinement rules for tetrahedra and octahedra must be introduced. In [28], nine types of edge refinement patterns are shown to be necessary for irregular tetrahedron refinement.

Another common way of generating nested meshes consists of recursively bisecting tetrahedra along their longest edge [49, 62, 58]. *Tetrahedron bisection* consists of replacing a tetrahedron σ with the two tetrahedra obtained by splitting σ at the middle point of its longest edge and by the plane passing through such point and the opposite edge in σ [32, 49, 62]. This rule is applied recursively to an initial decomposition of the cubic domain obtained by splitting it into six tetrahedra, all sharing one diagonal. This gives rise to three congruent tetrahedral shapes, that we call *1/2 pyramids*, *1/4 pyramids* and *1/8 pyramids*, respectively (see Fig. 9).

When we apply a tetrahedron bisection, all tetrahedra that share a common edge with the tetrahedron being split must be split at the same time to guarantee that a conforming mesh is generated. The tetrahedra which share their longest edge and that thus must be split at the same time form a *diamond* (sometimes called a *cluster*) [27, 41, 58]. There are three types of diamonds generated by the three congruent tetrahedral shapes, which are formed by 1/2, 1/4 and 1/8 pyramids, respectively (see Fig. 10).

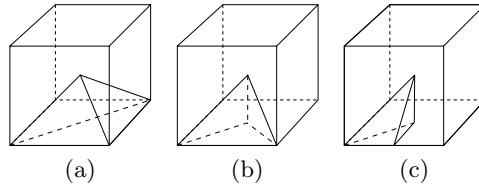


Fig. 9. Subdivision of the initial cubic domain into six tetrahedra. Examples of (a) a 1/2 pyramid, (b) a 1/4 pyramid, and (c) a 1/8 pyramid.

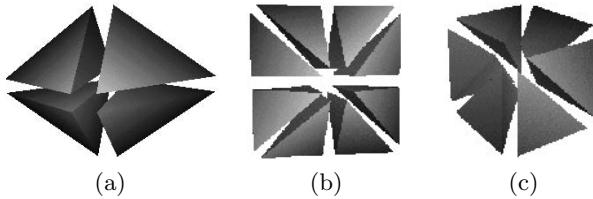


Fig. 10. (a) Diamond formed by four 1/2 pyramids. (b) Diamond formed by eight 1/4 pyramids. (c) Diamond formed by six 1/8 pyramids.

The data structures for encoding nested regular meshes produced through tetrahedron bisections encode (implicitly or explicitly) either a binary forest of tetrahedra, or a DAG of diamonds with their direct dependencies.

In a forest of tetrahedra, the roots correspond to the six tetrahedra in which the initial cube is subdivided. Any other node describes a tetrahedron σ and the two children of σ are the two tetrahedra obtained by splitting σ along its longest edge. Each of the six trees is a full binary tree which can thus be encoded implicitly. A forest of tetrahedra directly extends the forest of right triangles used for LOD models based on triangle meshes (see Sect. 4.3). Note that a forest of tetrahedra does not need to be explicitly encoded unless approximation errors or other attribute information must be associated with tetrahedra.

In order to extract conforming meshes from a forest of tetrahedra, error saturation [26, 71] has been applied, thus implicitly forcing all parents to be split before their descendants (see also [44] for an effective saturation technique for terrains). A very efficient alternative consists of assigning to each tetrahedron a *location code*, with a mechanism similar to that adopted for linear encoding of triangle quadtrees [42] or hierarchies of right triangles [23]. Location codes are not stored, but they are computed when extracting a mesh during selective refinement. In [32] parents, children, and neighbours of a tetrahedron in a nested tetrahedral mesh are computed in a symbolic way, but finding neighbours still takes time proportional to the depth in the hierarchy. A worst-case constant-time implementation of the neighbour find-

ing technique has been first proposed in [41]. The experimental comparisons, described in [16], performed on the basis of mesh extractions at uniform resolution, have shown that the meshes extracted using error saturation have, on average, 5% more tetrahedra than those extracted with the neighbour finding algorithm. On the other hand, the computing times for top-down mesh extraction are the same for the saturated and non-saturated versions.

In [27], a data structure is proposed based on the direct encoding of the diamonds corresponding to the three classes of tetrahedral shapes generated by the tetrahedron bisection process. We call this data structure a *DAG of diamonds*: it extends the DAG of vertex dependencies discussed in Sect. 4.3. The root of the DAG is the initial subdivision of the cube (a non-aligned diamond), any other node is a diamond and the arcs describe the parent-child relation. Given a diamond D , the *parents* of D are those diamonds that must be split to create the tetrahedra of D . The diamonds that are created when D is split are the *children* of D . In [27], a compact data structure for encoding a DAG of diamonds is described in which the DAG structure does not have to be explicitly recorded. Diamond information as well as error information is attached to each vertex together with its field value. Only three bytes per diamond are used to store the pre-computed error information for the diamond.

5.4 LOD Data Structures for Irregular Meshes

A LOD model based on d -dimensional simplicial meshes, called a *Multi-Tessellation (MT)*, has been defined, and it is independent of the dimension of the complex and of the specific strategy through which the model is built [17]. This model extends the Multi-Triangulation (see Sect. 5.4) to three and higher dimensions. A data structure implementing the 3D Multi-Tessellation describes the DAG and represents all the tetrahedra in the 3D MT as a 4-tuple of vertex indexes [12]. The total cost of the 3D MT data structure, when built through half-edge collapse, has been shown to be equal 360 n bytes, which is about 3.5 times the cost for encoding the reference mesh as an indexed data structure, and about 1.8 times the cost for encoding the reference mesh as an extended indexed structure with adjacencies (see [12]).

In [9], a LOD data structure for irregular tetrahedral meshes, called a *Full-Edge Multi-Tessellation (MT)*, has been developed to encode efficiently a LOD model generated through full-edge collapse. The direct dependency relation is encoded through a view-dependent tree [21]. Let us consider a full-edge collapse, which contracts an edge $e = (v', v'')$ to a vertex v which is an internal point of edge e (see Fig. 8). An internal node of the view-dependent tree corresponds to a full-edge collapse and to its inverse vertex split and contains:

- an offset vector, which is used to find the positions of vertices v' and v'' from that of v , and vice-versa;

- an offset field value, which is used to obtain the field value at v' and v'' from that of v , and vice-versa;
- a bit mask, which is used to partition the set of tetrahedra incident at v .

The bit mask contains one bit for each tetrahedron incident at v . The following rule is applied: tetrahedra marked with 0 must replace v with v' ; tetrahedra marked with 1 must replace v with v'' ; each triangular face shared by two differently marked tetrahedra must be expanded into a tetrahedron incident at both v' and v'' (see Fig. 8).

The total cost of the Full-Edge MT implementation is thus equal to $28n$ bytes, which is about $1/3$ the cost for encoding the reference mesh as an indexed data structure, and about 15% the cost for encoding the reference mesh as an indexed structure with adjacencies [9].

In [10], a compact data structure for encoding a Half-Edge MT, i.e., a LOD model generated through half-edge collapse, is described. The DAG is encoded by using the technique proposed by Klein and Gumhold [37]. A half-edge collapse of an edge (v, w) into a vertex w and its corresponding half-vertex split are encoded by storing the coordinates of vertex v and the field value at vertex v ; an implicit encoding of vertex w ; a compact encoding of the region of influence of the vertex split (which is a portion of the star of w affected by inserting vertex v) represented as a bit stream generated by traversing the region of influence in a breadth-first fashion (see Fig. 11).

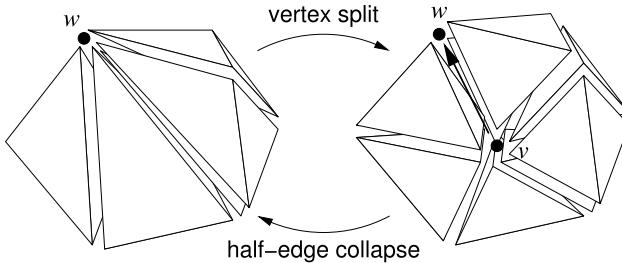


Fig. 11. Modification of a tetrahedral mesh through a half-edge collapse and a half-vertex split.

The total cost of a Half-Edge MT data structure (including the cost of encoding the direct dependencies) is thus equal to $52n$ bytes [10], which is about $1/2$ the cost for encoding the reference mesh as an indexed data structure, and about $1/4$ the cost for encoding the reference mesh as an indexed structure with adjacencies. In [13], we have designed a more space-efficient data structure for a Half-Edge MT, in which the direct dependency relation is encoded implicitly through an extension of the view-dependent tree.

6 Concluding Remarks

We have presented a survey of data structures for LOD models of free-form geometry. As a general observation it turns out that LOD data structures based on regular meshes are naturally more space efficient than the ones based on irregular meshes. This is obvious since the regular topology of the mesh as well as globally uniform refinement operations can be encoded implicitly. Among the semi-regular LOD models, those generated by triangle or tetrahedron bisection or by $\sqrt{3}$ subdivision are very effective for the extraction of adaptively refined representations because they guarantee conformity.

Most LOD models for irregular meshes are generated through edge collapses and thus the corresponding internal representations encode procedural descriptions of the coarsening and refinement operations. There are inherent difficulties when generating a LOD model of a free form surface in a top-down fashion and performing vertex removal requires special attention in re-triangulating the resulting “hole” in the mesh. Moreover, there are theoretical difficulties in refining a representation of a 3D scalar field with a non-convex domain. Vertex insertion and vertex removal cannot always be performed in a tetrahedral mesh since the resulting polyhedral “bubble” cannot always be triangulated without adding auxiliary points.

A problem which has not been solved completely for LOD models is to deal with very large meshes containing millions to billions of triangles or tetrahedra. While some out-of-core implementations for regular and irregular meshes have been proposed, a general strategy for dealing with LOD models in secondary memory is missing.

Another interesting research direction is to extend the LOD concept beyond the use of mere geometric refinement criteria. Similar to the high level abstraction of non-photo-realistic rendering, one would like to represent an object’s shape at different levels of abstraction such as sub-component-based descriptions of a 3D shape or a morphological description of a scalar field.

Acknowledgements

This work has been partially supported by the EC Research Training Network *Multiresolution in Geometric Modelling (MINGLE)*, under contract HPRN-CT-1999-00117, by project *Algorithmic and Computational Methods for Geometric Object Representation (MACROGeo)* funded by the Italian Ministry of Education, University, and Research (MIUR), and by project *Augmented Reality for Teleoperation of Free-Flying Robots (AUREA)* funded by the Italian Space Agency (ASI).

References

1. <http://www.openmesh.org>. 2004.
2. R. E. Bank, A. H. Sherman, and A. Weiser. Refinement algorithms and data structures for regular local mesh refinement. In R. Stepleman, M. Carver, R. Peiskin, W. F. Ames, and R. Vichnevetsky, editors, *Scientific Computing, IMACS Transactions on Scientific Computation*, volume 1, pages 3–17. North-Holland, Amsterdam, The Netherlands, 1983.
3. B. G. Baumgart. A polyhedron representation for computer vision. In *Proc. AFIPS National Computer Conference*, volume 44, pages 589–596, 1975.
4. J. Bey. Tetrahedral mesh refinement. *Computing*, 55:355–378, 1995.
5. M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt. Openmesh – a generic and efficient polygon mesh data structure. In *Proc. OpenSG Symposium*, 2002.
6. M. Botsch, A. Wiratanaya, and L. Kobbelt. Efficient high quality rendering of point sampled geometry. In *Proc. Eurographics Workshop on Rendering*, 2002.
7. E. Bruzzone and L. De Floriani. Two data structures for building tetrahedralizations. *The Visual Computer*, 6(5):266–283, 1990.
8. S. Campagna, L. Kobbelt, and H.-P. Seidel. Directed edges - a scalable representation for triangle meshes. *ACM Journal of Graphics Tools*, 3(4):1–12, 1998.
9. P. Cignoni, L. De Floriani, P. Magillo, E. Puppo, and R. Scopigno. Selective refinement queries for volume visualization of unstructured tetrahedral meshes. *IEEE Transactions on Visualization and Computer Graphics*, 10(1):29–45, 2004.
10. E. Danovaro and L. De Floriani. Half-edge multi-tessellation: A compact representation for multiresolution tetrahedral meshes. In *Proc. 1st International Symposium on 3D Data Processing Visualization and Transmission*, pages 494–499. IEEE Computer Society, 2002.
11. E. Danovaro, L. De Floriani, P. Magillo, and E. Puppo. Compressing multiresolution triangle meshes. In C.S. Jensen, M. Schneider, V.J. B. Seeger, and Tsotras, editors, *Advances in Spatial and Temporal Databases, Lecture Notes in Computer Science*, volume 2121, pages 345–364. Springer Verlag, Berlin, July 2001.
12. E. Danovaro, L. De Floriani, P. Magillo, and E. Puppo. Data structures for 3d multi-tessellations: an overview. In H. Post, G.P.Bonneau, and G.M.Nielson, editors, *Proc. Dagstuhl Scientific Visualization Seminar*. Kluwer Academic Publishers, 2002.
13. E. Danovaro., L. De Floriani, P. Magillo, and N. Sokolovsky. Data structures for encoding lod models generated through half-edge collapse. Technical Report DISI-TR-01-06, Department of Computer and Information Science, University of Genova, Genova (Italy), 2003.
14. L. De Floriani and P. Magillo. Multiresolution mesh representation: Models and data structures. In A. Iske, E. Quak, and M. S. Floater, editors, *Tutorials on Multiresolution in Geometric Modelling*, Springer Verlag, Heidelberg (D), 2002.
15. L. De Floriani, P. Magillo, and E. Puppo. Efficient implementation of multi-triangulations. In *Proc. IEEE Visualization'98*, pages 43–50, Research Triangle Park, NC (USA), October 1998. IEEE Computer Society.
16. L. De Floriani and M.Lee. Selective refinement in nested tetrahedral meshes. In G.Brunnett and an H.Mueller B.Hamann, editors, *Geometric Modeling for Scientific Visualization*. Springer Verlag, New York, 2003. (to appear).

17. L. De Floriani, E. Puppo, and P. Magillo. A formal approach to multiresolution modeling. In R. Klein, W. Straßer, and R. Rau, editors, *Geometric Modeling: Theory and Practice*, pages 302–323. Springer Verlag, 1997.
18. C. DeCoro and R. Pajarola. Xfastmesh: Fast view-dependent meshing from external memory. In *Proc. IEEE Visualization 2002*, pages 263–270, Boston, MA, October 2002. IEEE Computer Society.
19. D. Dobkin and M. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 5(4):3–32, 1989.
20. M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein. ROAMing terrain: Real-time optimally adapting meshes. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization'97*, pages 81–88, Phoenix, AZ, October 1997. IEEE Computer Society.
21. J. El-Sana and A. Varshney. Generalized view-dependent simplification. *Computer Graphics Forum*, 18(3):C83–C94, 1999.
22. J. El-Sana and Y. Chiang. External memory view-dependent simplification. *Computer Graphics Forum*, 19(3):C139–C150, 2000.
23. W. Evans, D. Kirkpatrick, and G. Townsend. Right-triangulated irregular networks. *Algorithmica*, 30(2):264–286, 2001.
24. P. Gandois and O. Devillers. Progressive lossless compression of arbitrary simplicial complexes. *ACM Transactions on Graphics*, 21(3):372–379, July 2002.
25. T. Gerstner. Multiresolution visualization and compression of global topographic data. *GeoInformatica*, 7(1):7–32, 2003.
26. T. Gerstner and M. Rumpf. Multiresolutional parallel isosurface extraction based on tetrahedral bisection. In *Proc. 1999 Symposium on Volume Visualization*. ACM Press, 1999.
27. B. Gregorski, M. Duchaineau, P. Lindstrom, V. Pascucci, and K. Joy. Interactive view-dependent rendering of large isosurfaces. In *Proc. IEEE Visualization 2002*, Boston, MA, October 2002. IEEE Computer Society.
28. G. Greiner and R. Gross. Hierarchical tetrahedral-octahedral subdivision for volume visualization. *The Visual Computer*, 16:357–369, 2000.
29. M.H. Gross and O.G. Staadt. Progressive tetrahedralizations. In *Proc. IEEE Visualization'98*, pages 397–402, Research Triangle Park, NC, 1998. IEEE Computer Society.
30. R. Gross, C. Luerig, and T. Ertl. The multilevel finite element method for adaptive mesh optimization and visualization of volume data. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 387–394, Phoenix, AZ, October 1997.
31. C. Gotsman S. Gumhold and L. Kobbelt. *Simplification and compression of 3D meshes*. 2002.
32. D. J. Hebert. Symbolic local refinement of tetrahedral grids. *Journal of Symbolic Computation*, 17(5):457–472, May 1994.
33. H. Hoppe. Progressive meshes. In *Proc. ACM SIGGRAPH*, pages 99–108, 1996.
34. H. Hoppe. View-dependent refinement of progressive meshes. In *Proc. ACM SIGGRAPH*, pages 189–198, Los Angeles, August 1997.
35. H. Hoppe. Efficient implementation of progressive meshes. *Computers & Graphics*, 22(1):27–36, 1998.
36. J. Kim and S. Lee. Truly selective refinement of progressive meshes. In *Proc. Graphics Interface 2001*, pages 101–110, 2001.
37. R. Klein and S. Gumhold. Data compression of multiresolution surfaces. In *Visualization in Scientific Computing '98*, pages 13–24. Springer Verlag, 1998.

38. L. Kobbelt. $\sqrt{3}$ subdivision. In *Proc. ACM SIGGRAPH*, pages 103–112. ACM, 2000.
39. C.L. Lawson. Software for C1 Surface Interpolation. In J.R. Rice, editor, *Mathematical Software III*, pages 161–164. Academic Press, 1977.
40. A. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In *Proc. ACM SIGGRAPH*, 1998.
41. M. Lee, L. De Floriani, and H. Samet. Constant-time neighbor finding in hierarchical tetrahedral meshes. In *Proc. International Conference on Shape Modeling & Applications*, pages 286–295, Genova, Italy, May 2001.
42. M. Lee and H. Samet. Navigating through triangle meshes implemented as linear quadtrees. *ACM Transactions on Graphics*, 19(2):79–121, April 2000.
43. P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, and G. A. Turner. Real-time continuous level of detail rendering of height fields. In *Proc. ACM SIGGRAPH*, pages 109–118, New Orleans, August 1996.
44. P. Lindstrom and V. Pascucci. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):239–254, 2002.
45. H. Lopes and G. Tavares. Structural operators for modeling 3-manifolds. In *SMA ’97: Proc. the Fourth Symposium on Solid Modeling and Applications*, pages 10–18. ACM, May 1997.
46. D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. In *Proc. ACM SIGGRAPH*, pages 199–207, 1997.
47. D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, , and R. Huebner. *Level of Detail for 3D Graphics*. Morgan-Kaufmann, Inc., 2003.
48. M. Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, 1987.
49. J. M. Maubach. Local bisection refinement for n -simplicial grids generated by reflection. *SIAM Journal on Scientific Computing*, 16(1):210–227, January 1995.
50. D.E Mueller and F.P. Preparata. Finding the intersection of two convex polyhedra. *SIAM Theoretical Computer Science*, 7:217–236, 1978.
51. S. Choi N. Amenta and R. Kolluri. The power crust. In *Proc. 6th ACM Symposium on Solid Modeling*, pages 249–260, Ann Arbor, Michigan, June 2001.
52. G. M. Nielson. Tools for triangulations and tetrahedralizations and constructing functions defined over them. In G. M. Nielson, H. Hagen, and H. Müller, editors, *Scientific Visualization: Overviews, Methodologies and Techniques*, chapter 20, pages 429–525. IEEE Computer Society, Silver Spring, MD, 1997.
53. M. Ohlberger and M. Rumpf. Hierarchical and adaptive visualization on nested grids. *Computing*, 56(4):365–385, 1997.
54. R. Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proc. IEEE Visualization’98*, pages 19–26, Research Triangle Park, NC, October 1998. IEEE Computer Society.
55. R. Pajarola. Fastmesh: Efficient view-dependent meshing. In *Proc. Pacific Graphics 2001*, pages 22–30. IEEE Computer Society, 2001.
56. R. Pajarola and J. Rossignac. Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):79–93, 2000.
57. A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics*, 12(1):56–102, January 1993.

58. V. Pascucci. Slow growing subdivision (SGS) in any dimension: towards removing the curse of dimensionality. *Computer Graphics Forum*, 21(3), 2002.
59. M. Pauly, M. Gross, and L. Kobbelt. Efficient simplification of point-sampled surfaces. In *Proc. IEEE Visualization*. IEEE Computer Society, October 2002.
60. J. Popovic and H. Hoppe. Progressive simplicial complexes. In *Proc. ACM SIGGRAPH*, pages 217–224, 1997.
61. E. Puppo. Variable resolution triangulations. *Computational Geometry Theory and Applications*, 11(3-4):219–238, December 1998.
62. M. Rivara and C. Levin. A 3D refinement algorithm for adaptive and multigrid techniques. *Communications in Applied Numerical Methods*, 8:281–290, 1992.
63. H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison Wesley, Reading, MA, 1990.
64. H. Samet. *Foundations of Multi-Dimensional Data Structures*. 2003, to appear.
65. S.Zhang. Successive subdivision of tetrahedra and multigrid methods on tetrahedral meshes. *Houston J. Mathematics*, 21:541–556, 1995.
66. G. Taubin, A. Guéziec, W. Horn, and F. Lazarus. Progressive forest split compression. In *Proc. ACM SIGGRAPH*, pages 123–132. ACM Press, 1998.
67. I.J. Trott, B. Hamann, and K.I. Joy. Simplification of tetrahedral meshes with error bounds. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):224–237, 1999.
68. L. Velho, L. Henriquez de Figueiredo, and J. Gomes. A unified approach for hierarchical adaptive tessellation of surfaces. *ACM Transactions on Graphics*, 4(18):329–360, 1999.
69. L. Velho and J. Gomes. Variable resolution 4-k meshes: Concepts and applications. *Computer Graphics Forum*, 19(4):195–214, 2000.
70. J.C. Xia, J. El-Sana, and A. Varshney. Adaptive real-time level-of-detail-based rendering for polygonal models. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):171–183, 1997.
71. Y. Zhou, B. Chen, and A. Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 135–142, Phoenix, AZ, October 1997.
72. M. Zwicker, H. Pfister, J. Baar, and M. Gross. Surface splatting. In *Proc. ACM SIGGRAPH*, pages 371–378, 2001.

An Algorithm for Decomposing Multi-dimensional Non-manifold Objects into Nearly Manifold Components

M. Mostefa Mesmoudi, Leila De Floriani, Franco Morando, and Enrico Puppo

Department of Computer Science (DISI), University of Genova, Italy
`{mesmoudi|defl0|morando|puppo}@disi.unige.it`

Summary. In this paper we address the problem of building valid representations for non-manifold d -dimensional objects. To this aim, we have developed a combinatorial approach based on decomposing a non-manifold d -dimensional object into an assembly of more regular components, that we call *initial quasi-manifolds*. We present a decomposition algorithm, whose complexity is slightly super-linear in the total number of simplexes. Our approach provides a rigorous basis for designing efficient dimension-independent data structures for describing non-manifold objects.

1 Introduction

A *manifold* object [with boundary] is a subset of the Euclidean space for which the neighbourhood of each point is locally equivalent [either] to an open ball [or to a closed half-space]. Objects that do not fulfil this property at one or more points are called *non-manifold* objects. Non-manifold objects are usually described through cell complexes with a non-manifold domain, and with possibly mixed-dimensional elements (see for instance the complexes in Figs. 1(a) and 6(a)). Since most objects encountered in applications contain a relatively small number of singularities, it is important to develop representations that scale well with the degree of “non-manifoldness” of the object. To this aim, an approach based on decomposing the object into simpler, possibly manifold, components seems particularly suitable as the basis for an efficient representation of non-manifolds (see Fig. 1).

In three or higher dimensions, a decomposition into manifold components may need to introduce artificial cuts through the object. Fig. 6(a) shows an example of a three-dimensional non-manifold complex of tetrahedra forming a fan around a singular point p . In order to eliminate such a singularity, we need to cut the complex along a manifold face (for instance, triangle pqr). We obtain a complex that is (combinatorially) equivalent to a ball. In six or higher dimension, a decomposition into manifold parts is not feasible in

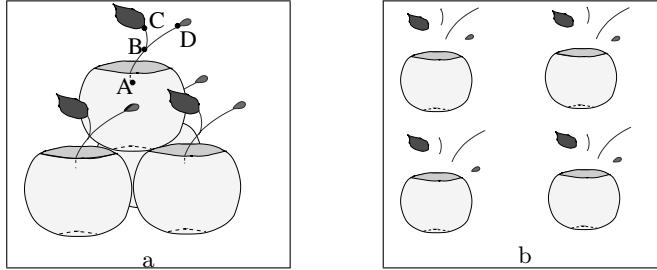


Fig. 1. In (a), an example of a non-manifold object made of a basket of four apples. All contact points between apples are singularities. For each apple, points like A, B, C and D are singular. In (b), the decomposition of the object in (a) into manifold parts.

general, since it has been proven that there do not exist algorithms that recognise the $(d - 1)$ -sphere when $d \geq 6$. Thus, for $d \geq 6$, d -manifolds are not recognisable algorithmically [15].

In [3], we have defined a natural decomposition of d -dimensional non-manifold objects described through simplicial complexes that removes (almost all) singularities by splitting the object only at non-manifold entities. The components of the resulting decomposition belong to a class, that we called *initial quasi-manifolds*, which admits, for any $d \geq 1$, a local characterisation in terms of combinatorial properties around each vertex. In dimension two, initial quasi-manifolds coincide with manifolds, and, in higher dimensions, the class of initial quasi-manifolds forms a superset of that of manifolds. The decomposition process produces a unique decomposition of the complex that we called the *standard decomposition*. The remaining singularities may be removed by introducing some artificial cuts, as illustrated in Figure 6.

In this paper, we describe an algorithm for generating the standard decomposition of a simplicial complex through a decomposition process that removes singularities by splitting the object at non-manifold entities. More details and proofs can be found in [3, 14]. The remainder of this paper is organised as follows. In Sect. 2, we review some related work. In Sect. 3, we present some basic notions in combinatorial topology that we need to develop the material. In Sect. 4, we describe the construction of our decomposition, while, in Sect. 5, we introduce an algorithm that computes a standard decomposition and we discuss its time complexity. Finally, we provide some concluding remarks in Sect. 6.

2 Related Work

Motivations for developing effective representations for non-manifold objects have been pointed out by several authors in the solid modelling literature

[9, 18, 19]. Most contributions are concerned with the representation of non-manifold surfaces. The first proposal for a topological data structure for boundary representation of non-manifold objects is the Radial-Edge data structure [20], which has been extended in [9, 11, 21]. All such data structures have been developed under the assumption that the object contains several non-manifold singularities. In [4], a compact data structure for non-manifold, non-regular two-dimensional simplicial complexes has been presented, which scales well to the manifold case.

Few approaches consider the problem of handling cell complexes in any arbitrary dimension. Approaches for dimension-independent modelling are usually quite general and thus can describe an arbitrary non-manifold complex [6, 16, 18]. Both n-G-maps [12] and Selective Geometric Complexes (SGC) [18] describe objects by cell complexes whose cells can be open and even not simply connected. The Winged Representation [16] has been developed to describe d -dimensional simplicial complexes which can be built by stitching d -simplexes at $(d - 1)$ -faces.

Some proposals [5, 7, 17] exist that pursue a decomposition of a non-manifold object into simpler and more manageable parts. However, such proposals are all limited to the two-dimensional boundaries of r-sets. In particular, the algorithm presented in [17] tries to minimise the number of duplications introduced by the decomposition process. In [8], the idea of cutting a two-dimensional non-manifold complex into manifold pieces has been exploited in order to develop a geometric compression algorithm.

3 Background

In this section, we review some basic notions about simplicial complexes in arbitrary dimensions, and about some relevant class of such Complexes. We use *abstract simplicial complexes* as basic tools to capture the combinatorial structure of geometric simplicial complexes.

3.1 Abstract Simplicial Complexes

Let V be a finite set of points that we call *vertices*. An *abstract simplicial complex* on V is a subset Ω of the set of (non-empty) parts of V , such that $\{v\} \in \Omega$ for every vertex $v \in V$, and if $\gamma \subseteq V$ is an element of Ω , then every non-empty subset of γ is also an element of Ω . Each element of Ω is called an *abstract simplex*. Whenever no ambiguity arises, we will use the terms *cell* or *simplex* to denote an abstract simplex. We will also use the term *complex* to denote an abstract simplicial complex.

The *dimension* of a simplex $\gamma \in \Omega$, denoted $\dim(\gamma)$, is defined by $\dim(\gamma) = |\gamma| - 1$, where $|\gamma|$ is the number of vertices in γ . A cell of dimension s is called an s -cell. A complex Ω is called *d -dimensional*, or a *d -complex*, if $\max_{\gamma \in \Omega}(\dim(\gamma)) = d$. Each d -cell of a d -complex Ω is called a *maximal cell* of

Ω . A subset of an abstract simplicial complex Ω that is an abstract simplicial complex is called a *sub-complex* of Ω . The *boundary* $\partial\gamma$ of a cell γ is defined to be the set of all proper parts of γ . Cells ξ in $\partial\gamma$ are called *faces* of γ . Similarly, the *co-boundary*, or *star*, of a cell γ is defined as $\star\gamma = \{\xi \in \Omega \mid \gamma \subseteq \xi\}$. Cells ξ in $\star\gamma$ are called *co-faces* of γ . The *link* of a cell γ , denoted by $lk(\gamma)$, is the set of all faces of co-faces of γ , that are not incident at γ . Any cell γ such that $\star\gamma = \{\gamma\}$ is called a *top cell* of Ω .

Two distinct cells are said to be *incident* if they share a common face. Otherwise, we say that they are *disjoint*. Two simplexes are called *s-adjacent* if they share an *s*-face; in particular, two p -simplexes, with $p > 0$, are said to be *adjacent* if they are $(p - 1)$ -adjacent. Two vertices (i.e., 0-cells) are called adjacent if they are both incident at a common 1-cell.

An *h-path* is a sequence of simplexes $(\gamma_i)_{i=0}^k$ such that two successive simplexes γ_{i-1} and γ_i are *h*-adjacent. Two simplexes γ and γ' are *h-connected*, if and only if there exist an *h-path* $(\gamma_i)_{i=0}^k$ such that γ is a face of γ_0 and γ' is a face of γ_k . A subset Ω' of a complex Ω is called *h-connected* if and only if every pair of its vertices are *h-connected*. Any maximal *h-connected* sub-complex of a complex Ω is called an *h-connected component* of Ω . The term *connected* is used as a shortcut for 0-connected.

3.2 Relevant Classes of Complexes

A d -complex Ω , where all top simplexes are maximal (i.e., of dimension d), is called *regular*, or *uniformly d-dimensional*. All complexes in Fig. 2, with the exception of the complex in Fig. 2(a), are regular 2-complexes. The $(d - 1)$ -sub-complex $\partial\Omega$ of a regular complex Ω , such that its top simplexes are all the $(d - 1)$ -simplexes of Ω that have only one incident d -simplex, is called the *boundary* of Ω . All simplexes of $\Omega - \partial\Omega$ are said to be *internal*.

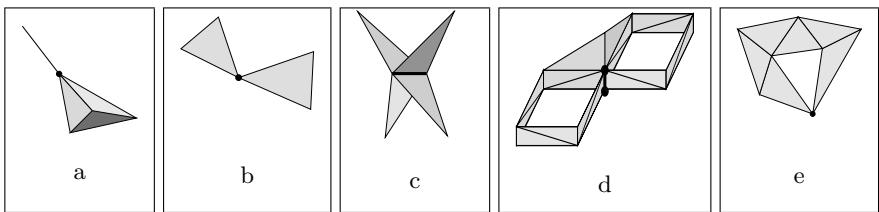


Fig. 2. Some complexes illustrating the various classes described in the text. Non-manifold simplexes are depicted in bold.

A $(d - 1)$ -simplex γ in a regular d -complex Ω is said to be a *manifold* $(d - 1)$ -simplex (in Ω) if and only if there are at most two d -simplexes incident at γ . Otherwise, γ is called a *non-manifold* $(d - 1)$ -simplex. Two d -simplexes γ and γ' are said to be *manifold-connected* if and only if there exists a $(d - 1)$ -path such that any two consecutive simplexes in the path are adjacent

through a manifold $(d - 1)$ -simplex. Such a path is called a *manifold path*. A regular complex in which every pair of d -simplexes are manifold connected is a *manifold-connected complex*. As in the case of standard connectivity, we can also define *manifold-connected* components of a general regular complex. The complexes in Fig. 2(a), (b) and (c) are not manifold-connected, while those in Fig. 2(d) and (e) are manifold-connected.

A regular $(d - 1)$ -connected d -complex in which all $(d - 1)$ -simplexes are manifold is called a *combinatorial pseudo-manifold*. A pseudo-manifold is also manifold-connected, while the reverse is not true. For instance, complex (e) in Fig. 2 is pseudo-manifold, while the remaining complexes are not.

Manifold objects are usually defined in topology through a local characterisation, which uses the concept of homeomorphism (topological equivalence). A combinatorial definition of manifolds is based on the similar concept of *combinatorial equivalence*. Its formal definition is not reported here for brevity (see [10]). Informally we can say that two complexes are combinatorially equivalent if there is a way of subdividing and welding their cells which provides two isomorphic complexes. Combinatorial equivalence of abstract complexes guarantees the homeomorphism of their geometrical realisation.

Thus we can define combinatorial balls and spheres. Let Δ^d denote a generic abstract d -simplex. A *combinatorial d -ball* is any abstract simplicial complex that is combinatorially equivalent to Δ^d . A *combinatorial d -sphere* is any simplicial complex that is combinatorially equivalent to $\partial\Delta^{d+1}$. From a combinatorial point of view, any vertex can be seen as a 0-ball. A sequence of adjacent 1-cells bounded by vertices $(v_i)_{i=0}^k$ is combinatorially equivalent to

- 1-ball if $v_i \neq v_j$ for any integers $i \neq j$ between 0 and k ,
- and to a 1-sphere if $v_0 = v_k$, and $v_i \neq v_j$ for any integers $i \neq j$ between 1 and $k - 1$.

In dimension two, any cone from an external vertex to a 1-sphere or to a 1-ball is combinatorially equivalent to a 2-ball (i.e., a disk). In dimension three, any cone from an external vertex to a 2-ball is combinatorially equivalent to a 3-ball. Therefore, the boundary complex formed by all 2-simplexes adjacent to only one 3-cell is combinatorially equivalent to a 2-sphere. Combinatorial balls and spheres of higher dimension can be generated by this method.

A vertex v in a regular d -complex Ω is a *manifold vertex* if and only if its link is combinatorially equivalent either to a $(d - 1)$ -sphere, if v is an internal vertex; or to a $(d - 1)$ -ball, if v is a boundary vertex. Vertex v is called a *non-manifold vertex* otherwise. A regular d -complex in which all vertices are manifold is a *combinatorial d -manifold*. A combinatorial d -manifold is also a combinatorial d -pseudo-manifold, while the reverse is not true. None of the examples in Fig. 2 are manifold complexes.

4 The Standard Decomposition

In this section, we briefly describe a sound decomposition of non-manifold complexes in arbitrary dimension into components of a well-understood class, that we have developed in [3, 14].

Intuitively, a complex Ω' is a decomposition of another complex Ω whenever Ω' can be obtained from Ω by *cutting* Ω through some of its faces. If Ω' is a decomposition of Ω , then any other decomposition of Ω' will be also a decomposition for Ω . Since each face is a finite collection of vertices, then any decomposition can be performed in finite steps via successive atomic splits where each atomic operation splits one vertex into two copies. This fact induces a partial order, with respect to the degree of decomposition, in the set of all possible decompositions of a complex. The minimum, with respect to the partial order, is given by Ω , while the maximum is given by the complex obtained by decomposing Ω into the collection of its top simplexes. We will call such a decomposition the *totally exploded* decomposition of Ω , and denote it with Ω^\top . Fig. 3 shows a complex and its totally exploded decomposition. We will label vertices in Ω with letters. To build the totally exploded version of a complex, we have to introduce a distinct copy of a vertex v for each top simplex v belongs to. Therefore we label vertices in Ω^\top with strings of the form vn , where v is the original name of the vertex from Ω and n is the label for the top simplex from Ω . Thus, vn belongs to n^\top in Ω^\top .

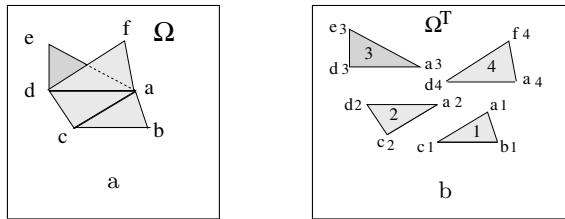


Fig. 3. A complex (a) and its totally exploded decomposition (b).

Conversely, in this framework any decomposition of Ω can be seen as obtained by *pasting* together simplexes in Ω^\top . Pasting occurs also through atomic operations that identify two vertices of the form vn and vm at a time. In [3], we have modelled pasting through the notion of *quotient* of complex Ω^\top with respect to an equivalence relation between vertices. The set of all possible decompositions of a complex Ω forms a lattice, see [14]. Two complexes adjacent in the lattice can be transformed into each other through an atomic split/join involving just a pair of vertices. Fig. 4(a) shows the Hasse diagram of the first level of the lattice of decompositions for the complex Ω . The complete decomposition lattice is too large to be shown. For simplicity,

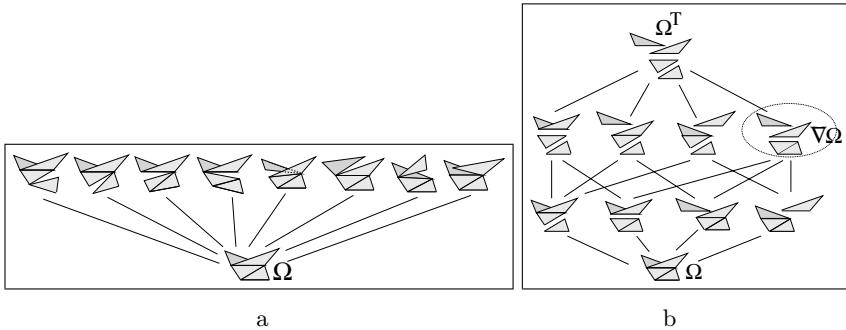


Fig. 4. In (a), the Hasse diagram for the decomposition lattice of complex Ω showing only the first decomposition level involving atomic splits of vertexes. In (b), a sub-lattice involving only two adjacent vertices splits. The complex labelled $\nabla\Omega$ in the dotted frame is highlighted for later reference.

we depict in Fig. 4(b) a sub-lattice involving only two adjacent vertex splits (which define edge splits).

The standard decomposition of a complex is a specific element of the lattice, which is obtained by discarding a whole set of “non-interesting” decompositions, and taking the “most general” of the remaining decompositions.

Usually one perceives non-manifold simplexes as “joints” between manifold parts, and it might seem reasonable to build a decomposition by splitting the complex just at them. On the other hand, it does not seem desirable to introduce cuts along non-singular (i.e., manifold) faces. Therefore we consider decompositions that in some sense are *essential* since they cut only at singularities. We say that a decomposition Ω' is an *essential* decomposition of Ω if and only if all simplexes of Ω' that are pasted together in Ω are glued at some non-manifold faces of Ω . In other words, Ω' is obtained by splitting Ω only at non-manifold faces.

In Fig. 5, we sketch a non-manifold complex (a) and four decompositions of it. The thick vertices and the thick edge in Fig. 5(a) and (b) are non-manifold simplexes. The decomposition in Fig. 5(b) is essential but it is still a non-manifold complex. The decompositions in Fig. 5(c) and (e) are essential

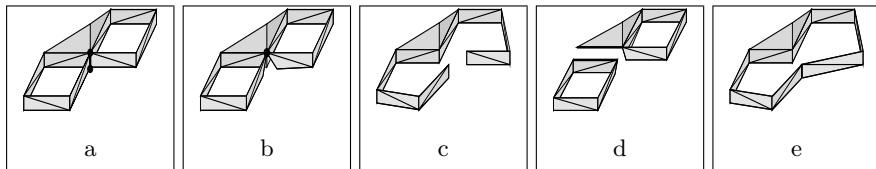


Fig. 5. Four decompositions of the complex in (a): decompositions in (b), (c) and (e) are essential, while decomposition in (d) is not essential.

decompositions and form a manifold complex. The decomposition in Fig. 5(d) is a manifold complex but is not essential because we split along the thick edge which is a manifold edge.

We define the standard decomposition $\nabla\Omega$ as the most decomposed of essential decompositions, thus it has been decomposed at *all* singularities that can be eliminated by cutting only through non-manifold faces. Therefore, the standard decomposition $\nabla\Omega$ is uniquely determined.

Thus, the standard decomposition $\nabla\Omega$ is the least upper bound of the essential decompositions in the lattice of decompositions.

It is easy to see that $\nabla\Omega$ must be a complex with regular connected components. Moreover, all connected components belong to a class, that we call *initial quasi-manifold*, which admits the following characterisation:

A regular h -complex Ω is an initial quasi-manifold if and only if the star of every vertex in Ω is $(h-1)$ -manifold-connected.

The above property means that we can always traverse the top cells in the star of each vertex through manifold adjacencies. This fact is relevant in designing efficient traversal algorithms on the data structure describing an initial quasi-manifold.

In general, initial quasi-manifolds form a super-class of manifolds. More specifically, in dimension two, the class of initial quasi-manifold complexes coincides with that of two-manifolds, while in higher dimensions there are initial quasi-manifolds that are not manifold. An example is provided by the “pinched pie” depicted in Fig. 6(a).

The relation between initial quasi-manifolds and pseudo-manifolds is more involved. Already in dimension 2, there exist pseudo-manifolds that are not initial quasi-manifolds. An example is provided by the squeezed band depicted in Fig. 2 (e). In dimension three or higher, there also exist initial quasi-manifolds that are not pseudo-manifolds. Examples of such complexes are not easy to build and definitely hard to realise visually (see [3] for an example). However, from functional analysis, we have that two disjoint convex

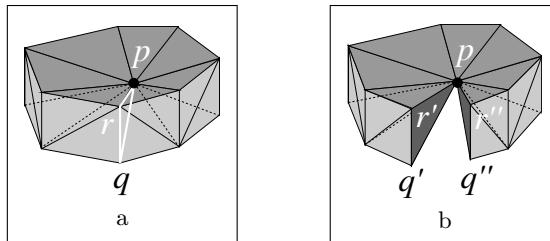


Fig. 6. In (a), the *pinched pie* formed of a fan of tetrahedra, all incident at a singular vertex p . It is an initial quasi-manifold but non-manifold complex. This complex can be decomposed into a 3-manifold (b) only by cutting along a face, such as triangle pqr .

open sets in the d -dimensional Euclidean space can always be separated by a hyper-plane (Hahn-Banach theorem). Therefore, if an initial quasi-manifold of dimension d is embedded (realisable) in the d -dimensional Euclidean space, it must be also a pseudo-manifold (while the reverse is not true). This is important since in the applications we are most often interested in 3- or 4-dimensional embedded simplicial complexes.

5 An Algorithm for Computing the Standard Decomposition

In this section, we present a decomposition algorithm that generates the standard decomposition $\nabla\Omega$ by splitting Ω at vertices that violate the condition for initial quasi-manifold. The algorithm works iteratively on the vertices of the input complex and recursively on its dimension. The decomposition of either a 0-complex or of an empty complex trivially coincides with the complex itself.

A pseudo-code description of the process generating $\nabla\Omega$ is given in Algorithm 1. This algorithm defines a recursive procedure $\text{DECOMPOSE}(\Omega, d)$ that returns the connected components of the standard decomposition for a d -complex Ω . This procedure starts by initialising a variable Ω_c with a copy of Ω . Complex Ω_c holds the current decomposition of Ω and the algorithm splits Ω_c until it contains $\nabla\Omega$. The algorithm considers each vertex v of Ω and computes recursively the decomposition of the link of v in Ω (not the link in Ω_c). Based on such decomposition, the algorithm decides whether and how Ω_c should be split at v .

The general idea on which this algorithm is based is to test vertices of Ω for the local property that characterises initial quasi-manifolds (i.e., the star of each vertex v of an initial quasi-manifold must be manifold-connected) and to split the complex where a vertex violates this property. We know that the star of a vertex is manifold-connected if and only if its link (which has a lower dimension) is manifold-connected. This is true because by adding the vertex v to all simplices in the link we obtain exactly the simplices in the closed vertex star. Therefore we want to decompose the link of v into manifold-connected components. This process induces only all those splits that are necessary to obtain the standard decomposition. Note that the recursive algorithm actually decomposes the link of a vertex into initial quasi-manifold components rather than manifold-connected components. This result is equivalent for our purposes, because the partition of top simplexes among connected components is the same in the two cases. The proof of correctness of Algorithm 1 is quite involved and omitted here for brevity [14].

Now let us consider the 2-complex Ω in Fig. 7(a), continuing the running examples of Figs. 3 and 4.

Vertices e and f have the same link in complex Ω , we have $lk(e, \Omega) = lk(f, \Omega) = ad$. The link of vertex b is segment ac and the link of vertex c

Algorithm 1 Computes the connected components in $\nabla\Omega$ for the d -complex Ω

```

Function DECOMPOSE( $\Omega, d$ )
   $\Omega_c \leftarrow \Omega$ 
  if  $d > 0$  and  $\Omega \neq \emptyset$  then
    for all vertices  $v$  of  $\Omega$  do
       $LK \leftarrow lk(v, \Omega)$  { $LK$  is the link of  $v$  in  $\Omega$ }
       $h \leftarrow \dim(LK)$  { $h$  is the dimension of  $LK$ }
       $L \leftarrow \text{DECOMPOSE}(LK, h)$  {compute the components of  $\nabla LK$ }
      if ( $h > 0$  and  $|L| > 1$ ) or ( $h = 0$  and  $|L| > 2$ ) then { $v$  must be split}
        for all  $\Psi \in L$  do {split  $v$  in  $\Omega_c$ }
          Create  $v_\Psi$  {a new copy  $v_\Psi$  for  $v$ }
          Replace  $v$  with  $v_\Psi$  in all simplices of  $\text{star}(v, \Omega_c)$  incident to  $\Psi$ 
        end for {the decomposition of vertex  $v$  has been completed}
      end if
    end for
  end if
  {returns the connected components of  $\Omega_c$ }
  return CONNECTED_COMPONENTS( $\Omega_c$ )

```

is a chain of two consecutive segments ba and ad . For all these vertices, the link is a 1-manifold. Vertices b, c, e and f do not have to be split. The link of vertex a is a non-manifold 1-complex formed by four edges, dc, de, df and cb , (see Fig. 7(b)). The link of vertices e and f in complex $lk(a, \Omega)$ is the 0-manifold $\{d\}$, the link of c has two 0-manifold components d and b , so we are in the situation $h = 0$ and $|L| \leq 2$. However, the link of vertex d in $lk(a, \Omega)$ has three 0-manifold components $\{c\}, \{e\}$ and $\{f\}$ ($h = 0$ and $|L| > 2$, see

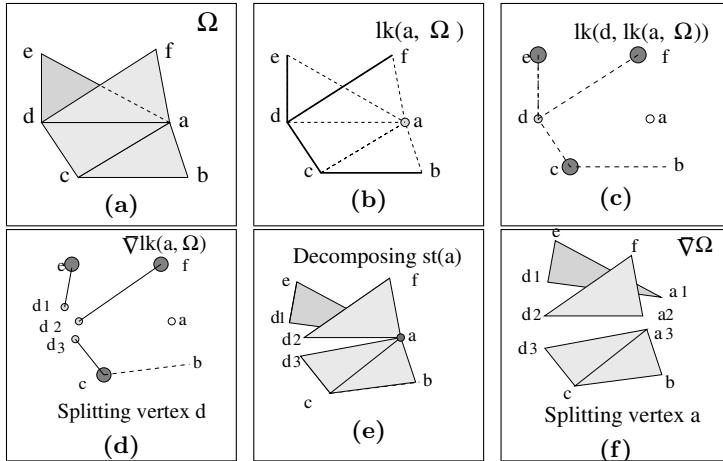


Fig. 7. Example of the decomposition process for the 2-complex in (a)

Fig. 7(c)). Hence, vertex d has to be split into three copies d_1, d_2 and d_3 . The new link $\nabla lk(a, \Omega)$ is a 1-manifold with three connected components ($h > 0$ and $|L| > 1$, see Fig. 7(d)). Therefore vertex a has to be split into three vertex copies a_1, a_2 and a_3 . The resulting complex is a 2-manifold with three components. The algorithm terminates the decomposition at this step and considers the last output as the standard decomposition $\nabla\Omega$, as shown in Fig. 7(f). We note that the same result can be obtained if we consider first vertex d rather than vertex a . Indeed, vertex a is in the link $lk(d, \Omega)$ and has three 0-manifold components in its own link $lk(a, lk(d, \Omega))$. Then, vertex a has to be split into three copies a_1, a_2 and a_3 . Therefore, the new link $\nabla lk(d, \Omega)$ is a 1-manifold with three connected components ($h > 0$ and $|L| > 1$). Then vertex d is split into three copies d_1, d_2 and d_3 and the same standard decomposition $\nabla\Omega$, as shown in Fig. 7(f) is produced.

The computation of $\text{DECOMPOSE}(\Omega, d)$ can be done in $O(d!(t \log t))$, where t is the number of top simplices in the d -complex Ω . We assume that vertices and top simplices are encoded as integers and that each top simplex in complex Ω is described as a tuple of indexes to its vertices. It has been shown in [14] that all operations, with the exception of the computation of the connected components and of the decomposition of the link LK , can be performed in $O(dt \log t)$. The subdivision of a complex into connected components is performed through a connected component labelling technique in graph with $(d+1)t$ arcs and $t+n$ nodes, where n is the number of vertices in $\nabla\Omega$ (note that $n \leq (d+1)t$). This is known (see [13]) to take $\Theta(dt + n)$ and thus less than $O(dt)$. If we denote by $T^d(n, t)$ the order of time complexity for the computation of $\text{DECOMPOSE}(\Omega, d)$ we have that

$$T^d(n, t) = O(dt \log t + n) + \sum_{v \in V} T^{(d-1)}(n_{lk(v)}, |lk(v)|)$$

where V is the set of vertices in Ω , and $n_{lk(v)}$ is the number of vertices in $lk(v)$ and $|lk(v)|$ is the number of top simplices in $lk(v)$. From the above recurrence relation, we have that $T^d(n, t) = O(n + d!(t \log t))$.

6 Concluding Remarks

We have presented an algorithm that decomposes an arbitrary d -simplicial complex into simpler components, which belong to a well-understood class that we called *initial quasi-manifolds*. This class is a decidable superset of the d -manifolds for $d \geq 3$ and coincides with that of d -manifolds for $d \leq 2$. The decomposition, that we called the *standard decomposition*, is unique, since it does not make any arbitrary choice in deciding where the object should be decomposed. This decomposition removes all singularities that can be removed without introducing artificial cuts (see Fig. 6). The standard decomposition is a useful basis for defining a data structure for non-manifold objects in arbitrary dimensions as described in [1].

Further developments of the work presented in this paper include the definition of Level-Of-Detail (LOD) models for d -dimensional objects described by simplicial complexes, which extend the results in [4] to higher dimensions. Moreover, since our decomposition operates locally, we can use this approach to define a measure of “shape complexity” at each non-manifold singularity, and thus to guide a complexity-preserving shape simplification process in extracting an *iconic* description for the simplified shape [2].

Acknowledgements

This work has been partially supported by the EC Research Training Network *Multi-resolution in Geometric Modelling (MINGLE)*, under contract HPRN-CT-1999-00117, and by project *Algorithmic and Computational Methods for Geometric Object Representation (MACROGeo)* funded by the Italian Ministry of Education, University, and Research (MIUR).

References

1. De Floriani, L., Morando, F., Puppo, E.: A Representation for Abstract Simplicial Complexes: An Analysis and a Comparison. In: Proc. 11th Int. Conf. on Discrete Geometry for Computer Imagery (2003).
2. De Floriani, L., Magillo, P., Morando, F., Puppo, E.: Non-manifold Multi-Tessellation: from meshes to iconic representation of 3D objects. In: Proceed. of 4th Intern. Workshop on Visual Form (IWVF4), C. Arcelli, L.P. Cordella, and G. Sannitidi Baja, editors, LNCS **2059** page 654, Berlin (2001), Springer-Verlag.
3. De Floriani, L., Mesmoudi, M.M., Morando, F., Puppo, E.: Decomposing Non-manifold Objects in arbitrary Dimensions. Graphical Models, **65**, 2–22 (2003)
4. De Floriani, L., Magillo, P., Puppo, P., Sobrero, D.: A multi-resolution topological representation for non-manifold meshes, *Computer-Aided Design*, **36**(2):141-159.
5. Desaulnier, H., Stewart, N.: An extension of manifold boundary representation to r-sets. ACM Trans. on Graphics, **11**(1), 40–60, (1992)
6. Elter, H., Lienhardt, P.: Different combinatorial models based on the map concept for the representation of sunsets of cellular complexes. In: Proc. IFIP TC 5/WG 5.10 Working Conference on Geometric Modeling in Computer Graphics, 193–212 (1993)
7. Falcidieno, B., Ratto, O.: Two-manifold cell-decomposition of r-sets. In: A. Kilgour and L. Kjelldahl, Eds., *Proceedings EUROGRAPHICS '92*, **11**, 391–404, September (1992)
8. Gueziec, A., Bossen, F., Lazarus, F., Horn, W.: Converting sets of polygons to manifold surfaces by cutting and stitching In: Conference abstracts and applications: SIGGRAPH '98, July 14–21, (1998)
9. Gursoz, E. L., Choi, Y., Prinz, F. B.: Vertex-based representation of non-manifold boundaries, In: M. J. Wozny, J. U. Turner, and K. Preiss, Eds., *Geometric Modeling for Product Engineering*, North Holland, 107–130, (1990)
10. Hudson, J.F.P. : Piecewise Linear Topology. W.A. Benjamin, Inc., New York (1969)

11. Lee S.H., Lee K., Partial Entity structure: a fast and compact non-manifold boundary representation based on partial topological entities, in *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*, Ann Arbor, Michigan, 2001, pp.159-170
12. Lienhardt, P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *Int. Journal of Comp. Geom. and Appl.*, **4**(3), 275–324, (1994)
13. Melhorn, K.: *Data Structures and Algorithms*. Springer Publishing Company (1984)
14. Morando, F.: Decomposition and Modeling in the Non-Manifold domain, PhD Thesis, Department of Computer and Information Science, University of Genova, Genova (Italy), February 2003
15. Nabutovsky, A.: Geometry of the space of triangulations of a compact manifold. *Comm. Math. Phys.*, **181**, 303–330 (1996)
16. Paoluzzi, A., Bernardini, F., Cattani, C., Ferrucci, V.: Dimension-independent modeling with simplicial complexes, *ACM Transactions on Graphics*, **12**(1), 56–102, (1993)
17. Rossignac, J., Cardoze, D.: Matchmaker: Manifold BRepS for non-manifold r-ssets. In: Willem F. Bronsvoort and David C. Anderson, editors, *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, 31–41, ACM, June (1999)
18. Rossignac, J.R., O'Connor, M.A.: SGC: A dimension-independent model for point sets with internal structures and incomplete boundaries. In: J.U. Turner, M. J. Wozny and K. Preiss, Eds., *Geometric Modeling for Product Engineering*, North-Holland, 145–180 (1990)
19. Weiler, K.: The Radial Edge structure: A topological representation for non-manifold geometric boundary modeling. In: M.J. Wozny, H.W. McLaughlin, J.L. Encarnaçao (eds), *Geometric Modeling for CAD Applications*, North-Holland, 1988, 3–36.
20. Weiler, K.: Topological Structures for Geometric Modeling. PhD Thesis, Troy, NY, August (1986)
21. Yamaguchi, Y., Kimura, F.: Non-manifold topology based on coupling entities. *IEEE Computer Graphics and Applications*, **15**(1):42–50, (1995)

Encoding Level-of-Detail Tetrahedral Meshes

Neta Sokolovsky¹, Emanuele Danovaro², Leila De Floriani², and Paola Magillo²

¹ Department of Computer Science, Ben Gurion University of the Negev, Beer Sheva, Israel

`netaso@cs.bgu.ac.il`

² Department of Computer and Information Science (DISI), University of Genova, Italy

`{danovaro|deflo|magillo}@disi.unige.it`

Summary. Level-Of-Detail (LOD) techniques can be a valid support to the analysis and visualisation of volume data sets of large size. In our previous work, we have defined a general LOD model for d -dimensional simplicial meshes, called a Multi-Tessellation (MT), which consists of a partially ordered set of mesh updates. Here, we consider an instance of the MT for tetrahedral meshes, called a *Half-Edge MT*, which is built through a common simplification operation, half-edge collapse. We discuss two compact encodings for a Half-Edge MT, based on alternative ways to represent the partial order.

1 Introduction

Several applications, including scientific visualisation, medical imaging, and finite element analysis, deal with increasingly large sets of three-dimensional data describing scalar fields, called volume data sets. In order to analyse volume data sets of large size and to accelerate their rendering, Level-Of-Detail (LOD) mesh-based models can be used. LOD models have been applied to the description of surfaces and two-dimensional height fields (see [5] for a survey). They encode the steps performed by a simplification process in a compact data structure, in such a way that a virtually continuous collection of simplified meshes at different LODs can be extracted on-line. An extracted mesh may have a variable resolution (i.e., density of the cells) which is focused in certain parts of the field domain (e.g., inside a box, or along a cutting plane), or in the proximity of interesting field values. This will enable a user to interactively explore large volume data using simplified approximations, and to inspect specific areas of interest. Fig. 1 shows some isosurfaces computed from a variable-resolution mesh extracted from a LOD model.

In the computer graphics and finite element literature, several research efforts have been devoted to nested tetrahedral meshes generated by recur-

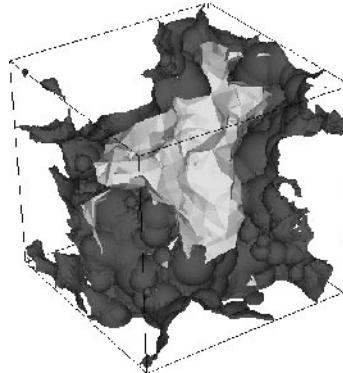


Fig. 1. [Reproduced in colour in Plate 11.] Variable LOD based on field values: the isosurface with a field value equal to 1.27 (shown in dark grey) is extracted in high LOD. The second isosurface, with a field value equal to 1.45 (shown in light grey), illustrates the lower resolution of the mesh.

sive decomposition, which are suitable for regularly distributed data points (see [9, 8, 13, 14, 15, 19]). LOD models based on unstructured tetrahedral meshes are desirable when dealing with irregularly-spaced data, since they are highly adaptive and can capture the shape of the field domain accurately. Such models are a rather new research issue. There have been proposals for simplification algorithms for unstructured tetrahedral meshes, based on edge collapse [1, 10, 18], or on vertex insertion [11, 17], and for LOD models, based either on a progressive [10, 16] or on a multi-level approach [3].

In [6], we have defined a general LOD model based on d -dimensional simplicial complexes, called a *Multi-Tessellation* (*MT*), which is both dimension- and application-independent. Here, we consider an *MT* based on unstructured tetrahedral meshes, and built through an edge-collapse simplification strategy, that we call a *Half-Edge Multi-Tessellation* (*Half-Edge MT*). We describe two compact data structures for a *Half-Edge MT*: a DAG-based structure in which the dependency relation is encoded as a Directed Acyclic Graph (DAG), similar to approach of Klein and Gumhold [12], and a tree-based *Half-Edge MT* in which a partial order is encoded as a tree by extending an approach proposed by El-Sana and Varshney [7].

2 Background

A *volume data set* consists of a set S of points spanning a domain D in the three-dimensional Euclidean space, with a field value f associated with each of them. A *tetrahedral mesh* Σ is a connected set of tetrahedra such that the union of all tetrahedra in Σ covers D . Any two distinct tetrahedra of Σ have disjoint interiors and the intersection of the boundaries of any two tetrahedra (if the intersection is non-empty) consists of lower dimensional simplexes

which belong to the boundaries of both tetrahedra. Although, theoretically, the number m of tetrahedra in a mesh Σ can be quadratic in the number n of vertices of Σ , in practice, we often find $m \approx 6n$.

The two most common data structures to encode tetrahedral meshes are the so called *indexed data structure*, and the *indexed data structure with adjacencies*. In both data structures, the vertices are stored in an array in which each entry contains three coordinates, and the field value. Both store, for each tetrahedron σ , the references to its four vertices of σ . In addition, the indexed data structure with adjacencies stores, for each tetrahedron σ , the references to the four tetrahedra sharing a face with σ .

The storage requirement for encoding a mesh with n vertices and m tetrahedra in an indexed structure is $8n$ bytes for vertices (assuming that coordinates and field value are stored in 2 bytes each), and $4m$ vertex references, i.e., $16m$ bytes. The indexed structure with adjacencies requires, in addition, another $16m$ bytes for referring to the adjacent tetrahedra. Since $m \approx 6n$, we have $104n$ bytes for the indexed data structure, and $200n$ bytes for the indexed data structure with adjacencies.

Given a volume data set S , an *approximated* tetrahedral mesh is a mesh Σ' having m' ($m' < m$) tetrahedra and vertices at a subset V' of the original data set V , with n' ($n' < n$) points. A scalar field f' is defined on Σ' , similarly to f , with the convention that values of f and f' are the same on each vertex that belongs to both V and V' . The approximation error associated with Σ' is the error that we perform in using Σ' instead of Σ for describing S . The *error* associated with each tetrahedron is a combination of the *field error* and of the *domain error*. In the simplification algorithm that we use [1], the field error at a tetrahedron $\sigma \in \Sigma'$ is computed as the maximum of the absolute value of the difference between the actual field value at the points of $V \setminus V'$ inside σ and the field value at the same points linearly interpolated within σ .

The domain error measures the variation in the domain shape (warping). Let us say that a point p of the domain is associated with a tetrahedron $\sigma \in \Sigma'$ if σ is the nearest tetrahedron to p in Σ' . The domain error at a tetrahedron $\sigma \in \Sigma'$ is computed as the maximum value of the distance from σ among the points associated with σ . The one-sided Hausdorff distance is used for such computation.

3 The Half-Edge Multi-Tessellation

The basic ingredients of a LOD model M are a coarse mesh Σ_b subdividing the domain, that we call the *base mesh*, a set of updates $U = \{u_1 \dots u_k\}$, and a relation \prec of direct dependency among updates.

An *update* applied to a mesh Σ consists of a pair of meshes $u = (u^-, u^+)$, where u^- is a sub-mesh of Σ , and the boundaries of u^- and u^+ are coincident. Intuitively, u replaces u^- with u^+ in Σ . The relation \prec of direct dependency is defined as follows: an update u' depends on an update u'' (denoted $u'' \prec u'$)

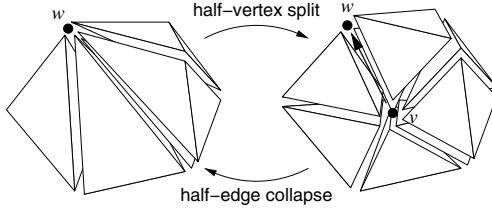


Fig. 2. An example of a half-edge collapse and of a half-vertex split.

if and only if u' removes some tetrahedra introduced by u'' . The transitive closure of relation \prec is a partial order. The updates in M will be also called the *nodes* of the MT. The mesh at the full resolution, that we term the *reference* mesh, can be obtained by applying all updates in U to the base mesh.

We say that a subset U' of U is *consistent* if, for every update $u'' \in U'$, each node u' such that $u' \prec u''$ is also in U' . The updates which form a consistent subset U' can be applied to the base mesh in any total order that extends the partial order, thus producing a mesh at an intermediate LOD.

A *Half-Edge Multi-Tessellation* (or *Half-Edge MT*) is a LOD model based on a specific update, called a *half-edge collapse*, applied to an unstructured tetrahedral mesh. A half-edge collapse consists of contracting an edge $e = (v, w)$ into one of its extreme vertices, say w . The reverse modification of a half-edge collapse is a *half-vertex split*, which expands a vertex w into an edge e by inserting the other extreme vertex v of e (see Fig. 2).

In [2], we have defined a LOD model based on a *full-edge collapse* applied to an unstructured tetrahedral mesh, called a *Full-Edge Multi-Tessellation* (or *Full-Edge MT*). A *full-edge collapse* consists of contracting an edge e , with extreme vertices v' and v'' , to a new vertex v (often the mid-point of e). The data structure proposed in [2] is specific for full-edge collapses. A full-edge collapse has the disadvantage of producing larger updates in comparison with those generated by a half-edge collapse, which imply less flexibility in extracting variable-LOD meshes [4].

In order to encode a Half-Edge MT, we need to encode the direct dependency relation, and the updates. The base mesh is stored separately in an indexed data structure with or without adjacencies. In the following section, we present two data structures: a DAG-based structure and a tree-based one which differ in the way they store the direct dependency relation.

4 A DAG-based Data Structure for a Half-Edge MT

The *direct dependency relation* is described as a Directed Acyclic Graph (DAG) that we store by using a technique proposed by Klein and Gumbhold [12]. For each node in the DAG, corresponding to an update u , a cyclic linked list, called a *loop*, is defined, which contains u followed by all its parents in

the DAG. Thus, an update u appears in its own loop and in all the loops defined by its children. For each node u , we store the number of loops to which u belongs (1 byte), and, for each loop, a forward pointer implementing the linked list plus the loop identifier (4 and 1 bytes, respectively).

The number of nodes (updates) is about the number n of vertices in the reference mesh. The total number of links to describe partial order as a DAG is equal to $n + a$, where a is the number of arcs in the DAG. Experimentally, we have found that a is equal to $6n$ on average. Thus, the cost of storing the DAG is equal to $36n$ bytes.

The *updates* are described by storing information sufficient to perform the half-vertex split and half-edge collapse associated with an update u (i.e., replacing u^- with u^+ and vice-versa) on a current mesh.

To perform a half-edge collapse, we need the vertex v and the vertex w to which the edge $e = (v, w)$ is contracted. To perform a half-vertex split, we need the coordinates of the introduced vertex v , the value of the field at v , and a compact encoding of the topological structure of u^- .

We also store an error value, which is used to decide whether to perform the half-edge collapse / half-vertex split represented by u . The error value $\varepsilon(u)$ provides an estimate of the approximation error associated with u and is computed as the maximum of the errors associated with the tetrahedra forming u^- . We store the error associated with an update and not with each tetrahedron to obtain a more economical representation.

The cost of encoding the coordinates, and the field value is equal to 8 bytes, while the error value is encoded on 2 bytes.

Since an update u corresponds to the insertion of a vertex v , updates and vertices are re-numbered in such a way that a node u and its corresponding vertex v have the same label. Thus, vertex v is encoded at a null cost.

We describe the topology of u^- by encoding a face f of the star-shaped polyhedron Π bounding u^- plus a bit stream which describes a traversal of the tetrahedral subdivision u^- starting at f . The bit stream contains three bits for each tetrahedron of u^- and is constructed as follows. We start from the tetrahedron that contains the face f and traverse the graph in which the nodes are tetrahedra of u^- and the arcs are faces of such tetrahedra. We label the faces of the tetrahedra encountered in the traversal in breadth-first order. A face of a tetrahedron, which is common to another tetrahedron of u^- is labelled 1 and 0 otherwise. If u^- contains k tetrahedra, then the bit stream contains $3k$ bits. Our experiments have shown that we can safely assume $k = 15$, thus, 45 bits for the stream, i.e., 6 bytes.

Vertex w and face f are identified by means of a tetrahedron index plus an index which identifies vertex w [face f] among the vertices [faces] of such a tetrahedron. In turn, a tetrahedron index consists of the index of the update u' such that u'^+ contains it, plus $\log P$ bits to discriminate it among the tetrahedra of u'^+ (where P denotes the maximum of tetrahedra created in a half-vertex split). For the tetrahedron containing f , update u' is the first parent of u in the DAG (we sort the loop of u according to this convention),

and, for the tetrahedron containing w , it is u itself, thus we do not need to store it. For details, see [4]. Thus, we have $2 \log P$ bits per update to identify these two tetrahedra, i.e., 10 bits, since the construction algorithm [1] enforces P to be equal to 32. The indexes of face f and vertex w within such tetrahedra require 2 bits each.

Summing up all the contributions, the storage cost for the information associated with a single update turns out to be equal to 18 bytes. The total cost of this DAG-based data structure is $18n + 36n = 54n$ bytes.

4.1 Performing Updates

Now, we explain how the stored information for an update u is used to perform corresponding half-vertex split and half-edge collapse on a current mesh Σ .

In order to split vertex w into an edge $e = (v, w)$, we start from the encoded boundary face f , and use the bit stream to retrieve all tetrahedra of u^- by visiting them in the same order as they have been visited when creating the bit stream. Tetrahedra of u^- , found in this way, are updated by replacing vertex w with vertex v , and new tetrahedra, incident in e , are inserted.

In order to collapse edge $e = (v, w)$ into vertex w , we identify vertex w among the vertices of u^+ in the current mesh, with tetrahedra of u^+ being those incident on v . Such tetrahedra are modified by replacing v with w , and degenerate tetrahedra resulting from such operation are removed.

5 A Tree-based Data Structure for a Half-Edge MT

The *dependency relation* is encoded as a tree by extending an approach proposed by El-Sana and Varshney [7] for triangle meshes simplified through full-edge collapse.

We store a forest of binary trees of vertices. The leaves of the forest correspond to the vertices of the reference mesh. Each internal node represents the vertex generated by collapsing an edge, and its two children are the endpoints of this edge. Since a half-edge collapse does not create a new vertex, we rename the surviving endpoint of the edge and consider it as another vertex. For example, if an edge (v, w) collapses to vertex w , then \hat{w} is a renamed copy of vertex w and appears in the binary tree as a parent of vertices v and w . By convention, vertex w is a left child of \hat{w} and is called a *false* child, while vertex v is a right child of \hat{w} and called a *true* child. Accordingly, \hat{w} is a *false* parent of w and a *true* parent of v (see Fig. 3).

In addition, we use a vertex enumeration mechanism. The n vertices of the reference mesh are labelled arbitrarily from 1 to n , the remaining vertices are labelled with consecutive numbers as they are created. In this way the label of parent \hat{w} is greater than labels of its children w and v .

We also define a true parent for the vertices that are false children in the tree. A *true parent* of v is a true parent of the nearest ancestor \bar{v} of v , such

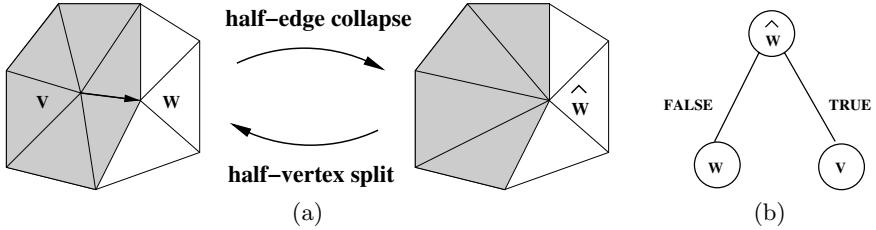


Fig. 3. (a) Half-edge collapse and half-vertex split. Note that \hat{w} is the same vertex as w but with a different label. The polygon affected by updates is shadowed. (b) Binary tree describing the update.

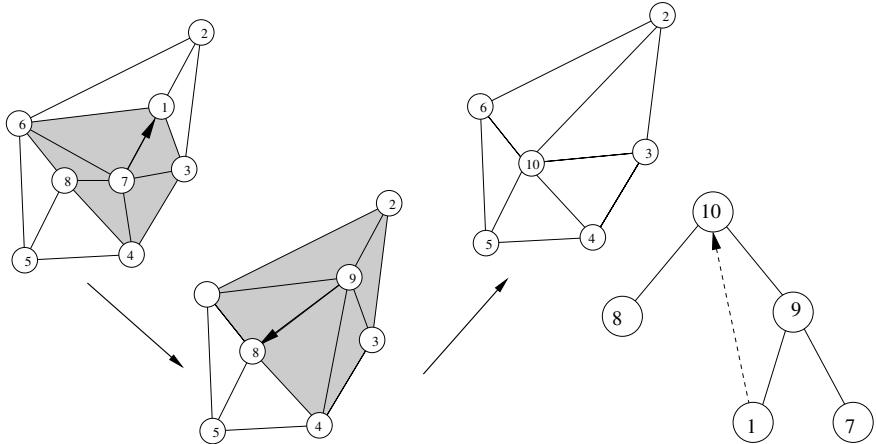


Fig. 4. An example of the sequence of half-edge collapses and the corresponding binary forest (irrelevant trees are omitted).

that \bar{v} is a true child. The true parent of v may not exist, in this case vertex v is the result of iterated renaming of a vertex of the base mesh (see Fig. 4).

The forest is implemented as an array in which every node u is stored at the position corresponding to its label. The entry for u stores an index pointing to the left child of u (stored only if u is not a leaf), and an index which points either to the right sibling of u , if u is the left child, or to the parent of u , if u is the right child (by convention, in this case a negative index is stored). In addition, each left child has a link to its true parent. The storage cost for the forest is equal to $4 n + 8 n_{in} + 4 n_l \simeq 16n$ bytes, where n is the number of vertices of the reference mesh (i.e., of leaves in the forest), n_{in} is the number of internal nodes ($n_{in} \simeq n$), and n_l is the number of left children ($n_l = n_{in}$).

Updates are encoded in the same way as in the DAG-based data structure, except for the encoding of face f and vertex w . Here, the index of the update u' , such that the tetrahedron containing f in u'^+ , must be stored explicitly in

additional 4 bytes, while vertex w is explicitly given in the tree, thus saving 7 bits. Therefore, the storage cost is 21 bytes per update.

The total cost of the tree-based data structure is thus equal to $21n + 16n = 37n$ bytes.

5.1 Correctness of the Dependency Encoding

The tree-based data structure does not store direct dependency links. Nevertheless, it contains sufficient information to retrieve the updates u' such that $u' \prec u$, when we are going to apply an update u on a current mesh Σ .

The *neighbours* of a vertex v are the vertices adjacent to v . We call *relevant* neighbours of vertex v the vertices (different from v) of the tetrahedra of u^- .

Let the half-vertex split associated with u split \hat{w} into $e = (v, w)$, and let the current mesh Σ contain \hat{w} . Such a split can be performed if all updates u' , such that $u' \prec u$, have already been performed in Σ . We claim that this condition is equivalent to the condition that each relevant neighbour of \hat{w} has a label lower than \hat{w} (see Fig. 5 (a)).

1. We prove that, if some update u' , such that $u' \prec u$, has not been performed, then some relevant neighbour of \hat{w} has a label greater than \hat{w} . This is true because, if u' is not performed and $u' \prec u$, then \hat{w}' (the vertex split by u') is greater than \hat{w} . On the other hand, \hat{w}' must be a relevant neighbour of \hat{w} because u and u' are related in the partial order.
2. We prove that, if some relevant neighbour v' of vertex \hat{w} has a label greater than \hat{w} , then there exists update u' , such that $u' \prec u$, and u' has not been performed. This is true for the update u' that splits v' . In fact, since v' is a relevant neighbour of \hat{w} , then u and u' must be related in the partial order, therefore $u' \prec u$.

Let the half-edge collapse associated with u collapse $e = (v, w)$ into \hat{w} , and let the current mesh Σ contain e . Such collapse can be performed if no update u' , such that $u \prec u'$, has been performed in the current mesh Σ . We claim that this condition is equivalent to the condition that all neighbours of

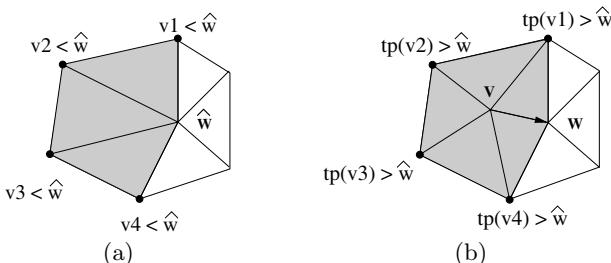


Fig. 5. Conditions for half-vertex split (a) and half-edge collapse (b). Note that $tp(v)$ denotes the true parent of vertex v .

vertex v (except w) have either true parent with a label greater than \hat{w} , or no true parent (see Fig. 5 (b)).

1. We prove that, if some update u' , such that $u \prec u'$, has been performed, then there exist at least one neighbour v' of v , such that the true parent of v' is lower than \hat{w} . This is true for at least one of the children of the vertex \hat{w}' split in update u' . Since $u \prec u'$, then the label of \hat{w}' is lower than \hat{w} . At least one of the children of \hat{w}' is a neighbour, because u and u' are related in the partial order.
2. We prove that, if some neighbour v' of v has a true parent w' with label lower than \hat{w} , then some update u' has been performed, with $u \prec u'$. We can see that this is true based on the following remarks. Since v' and v are neighbours, then updates u and u' (the update that has split the true parent of v') must be related in the partial order. Therefore, $u \prec u'$.

6 Comparisons on Storage Costs

Table 1 reports space requirements for the two data structures that we have presented in term of n (the number of vertices). The last two columns show the compression ratio with respect to the space required for encoding the reference mesh as an indexed data structure, and as an indexed data structure with adjacencies, respectively.

We have built Half-Edge MTs for a number of data sets:

- *Fighter* (courtesy of NASA): it represents the air flow over a jet Fighter from a wind tunnel model;
- *Small Buckyball* (courtesy of AVS Inc.): it represents electron density around a Carbon 60 molecule;
- *Blunt Fin* (courtesy of NASA): it represents the air flow over a flat plate with a blunt fin rising from the plate;
- *F117* (courtesy of MIT) represents the flow over a F117 aircraft;
- *Plasma* (courtesy of Italian National Research Council- Pisa); it represents three-dimensional Perlin noise.

Table 2 reports the characteristics of these data sets, while Table 3 reports the sizes of the Half-Edge MTs encoded with both data structures. Columns *real* show the space required to store updates and dependencies, while columns *bound* give the theoretical estimate.

7 Concluding Remarks

We have proposed data structures for a class of LOD tetrahedral meshes, called a Half-Edge Multi-Tessellation. These data structures act as a compression mechanism also with respect to storing the original mesh at full

Table 1. Space required to encode Half-Edge MT with the DAG-based and tree-based data structures.

| | Updates | Dependency | Total | Reference | | Compression | |
|------------|---------|------------|--------|-----------|---------|-------------|---------|
| | (byte) | (byte) | (byte) | ind | ind-adj | ind | ind-adj |
| DAG-based | 18n | 36n | 54n | 104n | 200n | 1.9 | 3.7 |
| Tree-based | 21n | 16n | 37n | 104n | 200n | 2.8 | 5.4 |

Table 2. Space required for storing the reference mesh as indexed structure with and without adjacencies.

| Data Set | Vertices (k) | Tetra (k) | Index (kB) | Index-Adjacent (kB) |
|-----------|--------------|-----------|------------|---------------------|
| Fighter | 13 | 68 | 1203 | 2299 |
| Bucky | 35 | 160 | 2840 | 5400 |
| Bluntnfin | 40 | 217 | 3795 | 7270 |
| F117 | 47 | 234 | 4130 | 7882 |
| Plasma | 268 | 1280 | 22625 | 43105 |

Table 3. Space required for DAG-based and tree-based data structures.

| Data Set | DAG-based | | Compression | | Tree-based | | Compression | |
|-----------|--------------|---------------|-------------|---------|--------------|---------------|-------------|---------|
| | real (kB) | bound (kB) | ind | ind-adj | real (kB) | bound (kB) | ind | ind-adj |
| Fighter | 586 | 644 | 2.1 | 3.9 | 432 | 466 | 2.8 | 5.3 |
| Bucky | 1700 | 1894 | 1.7 | 3.2 | 1244 | 1298 | 2.3 | 4.3 |
| Bluntnfin | 1818 | 2127 | 2.1 | 4.0 | 1404 | 1467 | 2.7 | 5.2 |
| F117 | 2093 | 2512 | 2.0 | 3.8 | 1653 | 1735 | 2.5 | 4.8 |
| Plasma | 13072 | 14481 | 1.7 | 3.3 | 9512 | 9922 | 2.4 | 4.5 |

resolution. A Half-Edge MT is more selective than a data structure developed for encoding multi-resolution tetrahedral meshes built through full-edge collapses, as shown in [4].

The DAG-based data structure has been implemented, while the tree-based data structure is currently under implementation. We plan to compare it experimentally with the DAG-based Half-Edge MT. Further development of the work presented in this paper will be the use of the tree-based data structure in a client-server application. Using our compact data structure will enable both a progressive and a selective download of the extracted mesh by a client and allow for a dynamic selective refinement at each new request from a client.

Acknowledgements

This work has been partially supported by the Research Training Network EC Project on *Multiresolution in Geometric Modelling* (MINGLE), under contract HPRN-CT-1999-00117, and by two projects funded by the Italian Ministry of Education, University, and Research (MIUR) on *Algorithmic and Computa-*

tional Methods for Geometric Object Representation (MACROGeo), Protocol N. RBAU01MZJ5, and on *Representation and Management of Spatial and Geographical Data in the Web*, Protocol N. 2003018941, respectively.

References

1. P. Cignoni, D. Costanza, C. Montani, C. Rocchini, and R. Scopigno. Simplification of tetrahedral volume with accurate error evaluation. In *Proceedings IEEE Visualization 2000*, pages 85–92. IEEE Computer Society, 2000.
2. P. Cignoni, L. De Floriani, P. Magillo, E. Puppo, and R. Scopigno. Selective refinement queries for volume visualization of unstructured tetrahedral meshes. *IEEE Transactions on Visualization and Computer Graphics*, 2003.
3. P. Cignoni, L. De Floriani, C. Montani, E. Puppo, and R. Scopigno. Multi-resolution modeling and visualization of volume data based on simplicial complexes. In *Proceedings 1994 Symposium on Volume Visualization*, pages 19–26, Washington, DC, October 1994.
4. E. Danovaro and L. De Floriani. Half-edge Multi-Tessellation: a compact representations for multiresolution tetrahedral meshes. In *Proceedings 1st International Symposium on 3D Data Processing Visualization Transmission*, pages 494–499, 2002.
5. L. De Floriani and P. Magillo. Multiresolution mesh representation: Models and data structures. In M. Floater, A. Iske, and E. Quak, editors, *Tutorials on Multiresolution in Geometric Modelling*, pages 363–418. Springer-Verlag, 2002.
6. L. De Floriani, E. Puppo, and P. Magillo. A formal approach to multiresolution modeling. In R. Klein, W. Straßer, and R. Rau, editors, *Geometric Modeling: Theory and Practice*, pages 302–323. Springer-Verlag, 1997.
7. J. El-Sana and A. Varshney. Generalized view-dependent simplification. *Computer Graphics Forum*, 18(3):C83–C94, 1999.
8. B. Gregorski, M. Duchaineau, P. Lindstrom, V. Pascucci, and K. Joy. Interactive view-dependent rendering of large isosurfaces. In *Proceedings IEEE Visualization 2002*, 2002.
9. G. Greiner and R. Gross. Hierarchical tetrahedral-octahedral subdivision for volume visualization. *The Visual Computer*, 16:357–365, 2000.
10. M. H. Gross and O. G. Staadt. Progressive tetrahedralizations. In *Proceedings IEEE Visualization'98*, pages 397–402, Research Triangle Park, NC, 1998. IEEE Comp. Soc. Press.
11. B. Hamann and J. L. Chen. Data point selection for piecewise trilinear approximation. *Computer Aided Geometric Design*, 11:477–489, 1994.
12. R. Klein and S. Gumhold. Data compression of multiresolution surfaces. In *Visualization in Scientific Computing '98*, pages 13–24. Springer-Verlag, 1998.
13. M. Lee, L. De Floriani, M., and H. Samet. Constant-time neighbor finding in hierarchical meshes. In *Proceedings International Conference on Shape Modeling*, pages 286–295, Genova (Italy), May 7-11 2001.
14. M. Ohlberger and M. Rumpf. Adaptive projection operators in multiresolution scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):74–93, 1999.

15. V. Pascucci and C. L. Bajaj. Time-critical isosurface refinement and smoothing. In *Proceedings 2000 Symposium on Volume Visualization*, pages 33–42, October 2000.
16. J. Popovic and H. Hoppe. Progressive simplicial complexes. In *Proc. ACM SIGGRAPH '97*, pages 217–224, 1997.
17. K. J. Renze and J. H. Oliver. Generalized unstructured decimation. *IEEE Computational Geometry & Applications*, 16(6):24–32, 1996.
18. I. J. Trotts, B. Hamann, and K. I. Joy. Simplification of tetrahedral meshes with error bounds. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):224–237, 1999.
19. Y. Zhou, B. Chen, and A. Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. In *Proceedings IEEE Visualization'97*, pages 135–142. IEEE Computer Society, 1997.

Multi-Scale Geographic Maps

Raquel Viaña¹, Paola Magillo², and Enrico Puppo²

¹ Department of Mathematics, University of Alcalá, Spain
raquel.viana@uah.es

² Department of Computer Science(DISI), University of Genova, Italy
{magillo|puppo}@disi.unige.it

Summary. We consider geographic maps represented as plane graphs, which undergo a process of generalisation performed through sequences of local updates. Generalisation transforms a highly detailed map into one with fewer details, spanning many different scales of representation through the sequence of updates. We study intrinsic dependency relations among updates in the sequence and, on this basis, we derive a multi-scale model that supports efficient retrieval of maps at different scales, possibly variable through the domain.

1 Introduction

We consider geographic maps represented in vector format, i.e., where spatial entities are represented explicitly as the elements (points, lines, regions) of a plane graph. Modern Geographic Information Systems (GISs) deal with maps at different scales, which can ideally span from a global (worldwide) scale to a very local (single house) scale. In this context, the concept of scale is related to the amount of information and the level of detail of entities represented in a map for a given area, rather than to the classical ratio between the size of objects in a paper map and the size of real entities they represent. A GIS should be able to relate representations of the same entity at different scales, and possibly to dynamically generate variable-scale maps, according to user needs. For instance, a variable-scale map may contain more detail for a certain class of entities (e.g., rivers, roads, property boundaries, etc.), as shown in Fig. 1(b), or in a given focus area, as shown in Fig. 1(c).

As far as we know, commercial databases have no consistent combinatorial treatment of multi-scale maps. Usually, collections of maps at different scales are maintained, and each map is treated as a whole. Links between a map and its corresponding maps at higher/lower scales are maintained through geo-referencing, but no links are present among different representations of the same spatial entity at several levels of detail. There are several applications available on the Internet (www.multimap.com, www.mapquest.com, etc.) which are based on this principle.

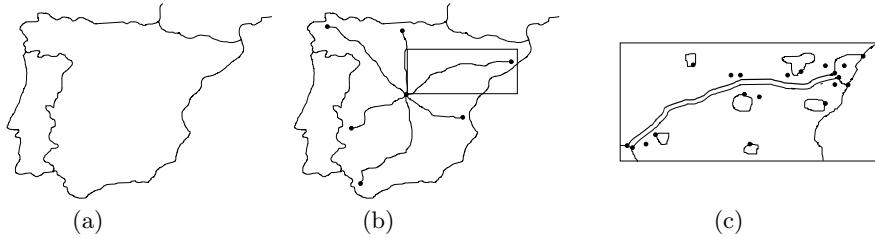


Fig. 1. (a) The Iberian peninsula. (b) The level of detail has been increased in one of the countries, showing its main roads. (c) Detail has been increased in the area outlined in (b).

Multi-scale representation provides a support for progressive and selective transmission of vector maps over a network (e.g., to download a map from the Web). Progressive transmission consists of sending a coarse map first, followed by details which incrementally improve its level of detail. It can also be seen as a form of map compression. Selective transmission allows the user to concentrate the transmission of details in some parts of the map. In [4] a method is proposed for the progressive transmission of a set of lines on the basis of the Douglas-Pücker simplification algorithm [6]. This algorithm does not guarantee, however, the topological consistency of the simplified model. This problem has been addressed in [2, 8]. Based on such ideas, in [3] a solution for progressively transmitting geographic maps in vector format is proposed. It is a hierarchical model maintaining vertical links between representations of the same entity on two consecutive levels. Its main drawback is that it does not support selective refinement.

In previous work [8], it has been shown that maps at a small scale (i.e., low level of detail) can be generalised to a larger scale (i.e., higher level of detail) through a restricted set of operators that modify the map. Based on such representation and set of operators, we develop here a combinatorial framework to represent multi-scale maps. Such a framework is inspired by the Multi-Triangulation, proposed in [9], which encompasses a variety of models developed in the literature to represent surfaces and scalar fields at multiple levels of detail through meshes of triangles [5]. Multi-scale representation of maps introduces, however, new issues due to the fact that each element in a map has its own identity and semantics, whereas the elementary patches (simplexes) used to describe a surface are just parts of a whole.

The rest of this paper is organised as follows. Sect. 2 introduces basic definitions, Sect. 3 defines the atomic updates we use to change the scale of a map, Sect. 4 defines a dependency relation among updates, Sect. 5 defines a multi-scale model for maps, and, finally, Sect. 6 contains some concluding remarks and directions for future developments of the work.

2 Map Representation Through Plane Graphs

A vector map M can be represented effectively as a plane graph, i.e., a set of *points* and a set of *lines* that are the geometric realisation in the plane of a graph where no two lines can intersect, except at their endpoints. The connected components of the plane obtained by removing the points and lines of M are the *regions* of M . The boundary of a region is formed by the points and lines delimiting it, and is classified into *proper boundary* and *features*. The proper boundary of a region is formed by those points and lines which bound both it, and at least another different region. It is formed by one *outer boundary*, and zero or more *inner boundaries*. The features of a region are those points and lines bounding the region, and not being on the boundary of any other region. Irregular situations, such as regions with holes, isolated points, and feature lines dangling inside regions occur in practice and admit a representation in this model.

In the remainder, we will use the term *entity* to denote a point, a line, or a region of a map. An entity (line or region) and another one (point or line) bounding it are said to be mutually *incident*. We refer to [8] for more details on this representation.

3 Update Operations on a Map

An *abstraction update* u on a map M is a function between the entities of M and the entities of another map M' . Given an entity e in M' , we call e an *abstracted entity* through u if its inverse image contains more than one entity of M . Update u is said to be *atomic* if there exists exactly one abstracted entity e through u in M' , and every entity in M which does not belong to the inverse image of e is transformed by u onto the same entity in M' .

A function which maps the set of entities of a map M onto the set of entities of another map M' is said to be *consistent* if it is *surjective*, *monotonic*, and *preserves connected sets of entities by inverse image* [1]. For a consistent function to be an abstraction update, the function must also be strictly non-injective.

In [8], a set of seven functions called *atomic abstraction updates* were defined, which have been proven in [1] to be necessary and sufficient to generate *all* consistent combinatorial transformations between maps. The formal proof is based on a categorical framework; specifically on the analysis of the category formed by all possible combinatorial maps, and all valid morphisms that generalise such maps. These operators provide a basis for building multi-scale models which guarantee to preserve the topological consistency of a map.

We are interested in a model in which details are either added or discarded depending on the user requirements, while maintaining the overall structure consistent. In order to obtain this, we define the inverse transformation of each atomic abstraction update, that we call an *atomic refinement update*.

Each atomic abstraction update is a function mapping two or three entities onto one entity, or, in other words, it deletes two or three entities from a map containing them, and replaces them with a new entity. Its inverse atomic refinement update restores the previous situation, by removing one entity and replacing it with two or three other entities.

We denote an update as $u : \{a_1, a_2, \dots\}[b_1, b_2, \dots] \rightarrow \{c_1, c_2, \dots\}$, where $\{a_1, a_2, \dots\}$ is the set of entities deleted by u , $\{c_1, c_2, \dots\}$ is the set of entities added to replace them, and $[b_1, b_2, \dots]$, if existing, are entities which do not disappear from the map but are needed in order to perform the combinatorial changes required to apply u . The seven pairs of mutually inverse atomic updates are described in the following:

(a) *line-to-point* $ltp : \{p, p', l\} \rightarrow \{p_0\}$

point-to-line $ptl : \{p_0\}[l_{p_1}, \dots, l_{p_j}; l'_{p'_1}, \dots, l'_{p'_k}; r_0] \rightarrow \{p, p', l\}$

Let l be a line and let p and p' ($p \neq p'$) be its endpoints. The application of update *line-to-point* to a map containing p , p' and l consists of removing these three entities, adding p_0 , and making every line and region incident on p and/or p' to become incident on p_0 (see Fig. 2 (a)).

In order to restore the original map, which is done by update *point-to-line*, it is necessary to know which of the lines incident on p_0 become incident on p (l_{p_1}, \dots, l_{p_j}) and which ones become incident on p' ($l'_{p'_1}, \dots, l'_{p'_k}$). In case l is a feature-line, it is also necessary to know the region incident on p_0 that will contain l , which is called r_0 .

(b) *region-to-point* $rtp : \{p, l, r\} \rightarrow \{p_0\}$

point-to-region $ptr : \{p_0\}[r_0] \rightarrow \{p, l, r\}$

Let r be a region whose boundary is formed just by line l (which must be a loop) and point p . The application of *region-to-point* consists of removing p , l , and r , adding p_0 , and making every line and region that was incident on p to be incident on p_0 (see Fig. 2 (b)).

In update *point-to-region*, r_0 is the region, among the ones incident on p_0 , that will become incident on line l .

As the only combinatorial difference between p and p_0 is that p is incident on l and r , and p_0 is not, from now on points p_0 and p are considered to be the same point.

(c) *region-to-line* $rtl : \{l, l', r\} \rightarrow \{l_0\}$

line-to-region $ltr : \{l_0\} \rightarrow \{l, l', r\}$

Let r be a region whose boundary is formed by two points which are not equal, and two lines, l and l' (with $l \neq l'$), each of which is incident on both points. The application of *region-to-line* to a map containing l , l' and r consists of removing these three entities, adding l_0 , and making every entity incident on l or l' to be incident on l_0 (see Fig. 2 (c)).

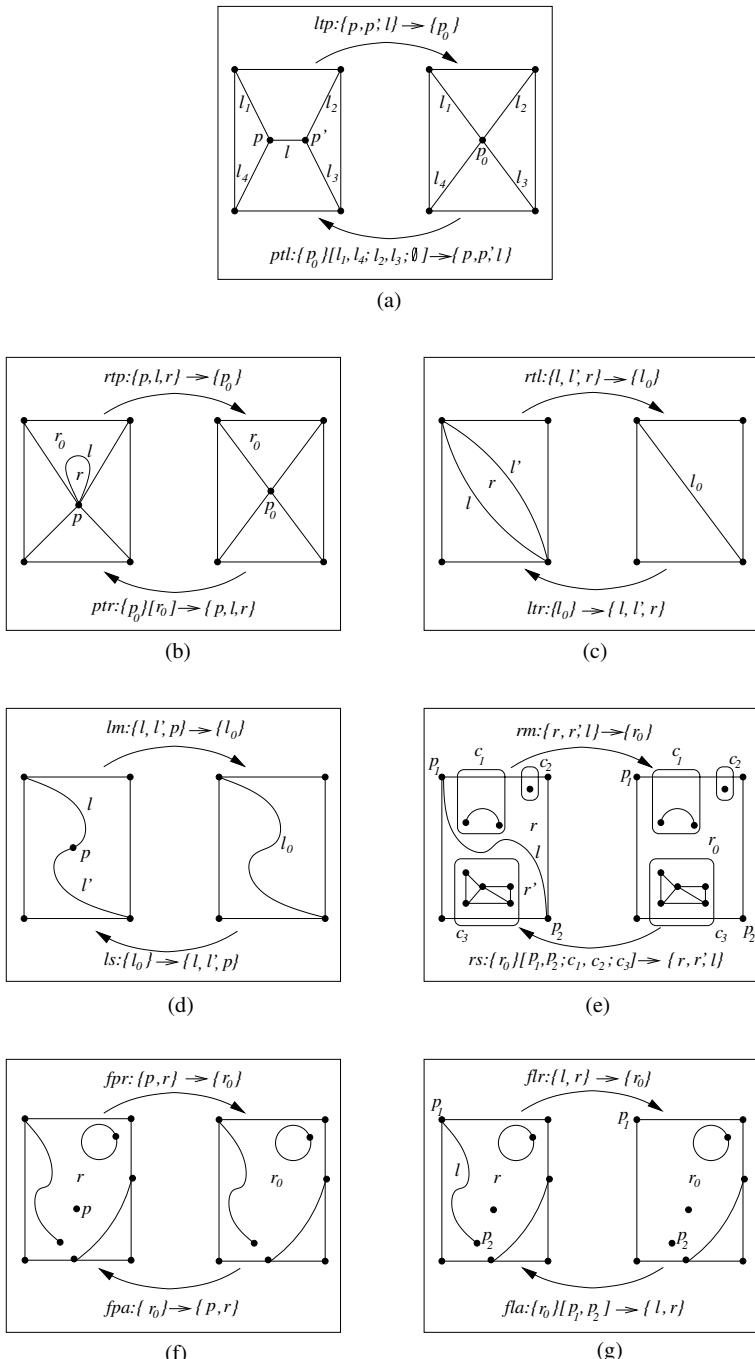


Fig. 2. Set of pairs of atomic/refinement updates.

(d) *line-merge* $lm : \{l, l', p\} \rightarrow \{l_0\}$ *line-split* $ls : \{l_0\} \rightarrow \{l, l', p\}$

Let l, l' , with $l \neq l'$, be two lines incident on a common point p which does not bound any other line. Update *line-merge* consists of removing l , l' and p , adding l_0 , and making every entity incident on l, l' or p to be incident on l_0 (see Fig. 2 (d)).

(e) *region-merge* $rm : \{r, r', l\} \rightarrow \{r_0\}$ *region-split* $rs : \{r_0\}[p_1, p_2; c_{r_1}, \dots, c_{r_l}; c_{r'_1}, \dots, c_{r'_m}] \rightarrow \{r, r', l\}$

Let r, r' , with $r \neq r'$, be two regions having a common bounding line l , with endpoints p_1 and p_2 . The application of *region-merge* on a map consists of removing r, r' and l from it, adding r_0 to it, and making every point and line incident on r or r' to be incident on r_0 (see Fig. 2 (e)).

To perform the inverse update, *region-split*, we need to know the two points p_1 and p_2 that will bound line l , and which of the entities in each inner boundary or feature of r_0 will bound either r (c_{r_1}, \dots, c_{r_l}) or r' ($c_{r'_1}, \dots, c_{r'_m}$), where $c_{r_i}, 1 \leq i \leq l$, denotes an inner boundary or feature of r , and $c_{r'_j}, 1 \leq j \leq m$, denotes an inner boundary or feature of r' .

(f) *feature-point-removal* $fpr : \{p, r\} \rightarrow \{r_0\}$ *feature-point-addition* $fpa : \{r_0\} \rightarrow \{p, r\}$

Let p be an isolated point inside a region r . The application of *feature-point-removal* consists of removing both p and r , adding r_0 , and making every point and line incident on r to become incident on r_0 (see Fig. 2 (f)).

As the only combinatorial difference between r_0 and r is that r is incident on point p and r_0 is not, we can consider r to be the same region as r_0 .

(g) *feature-line-removal* $flr : \{l, r\} \rightarrow \{r_0\}$ *feature-line-addition* $fla : \{r_0\}[p_1, p_2] \rightarrow \{l, r\}$

Let l be a feature-line in region r . The application of *feature-line-removal* to a map consists of removing l and r , adding r_0 , and making every point and line incident on r to become incident on r_0 (see Fig. 2 (g)).

To perform update *feature-line-addition*, the endpoints of line l , points p_1 and p_2 , must be known.

As the only combinatorial difference between r_0 and r is that r is incident on l and r is not, we consider r_0 to be the same region as r .

4 Dependencies in a Sequence of Atomic Refinement Updates

The sequential application of atomic abstraction (resp. refinement) updates allows us to build sequences of maps in which the level of detail continually

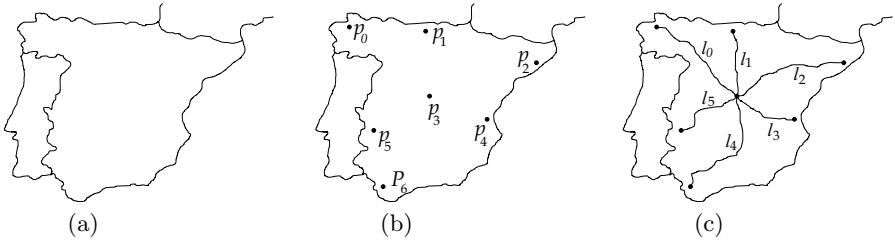


Fig. 3. The sequential application of seven updates, all of which are of type *feature-point-addition*, to the map in (a), produces (b). Then, updates of type *feature-line-addition* can be applied on the resulting map, producing (c).

decreases (resp. increases). Let us consider the map of Fig. 3 (a). The sequential application of updates of type *feature-point-addition* produces the map of Fig. 3 (b). And the application of updates of type *feature-line-addition* to this last map produces the one of Fig. 3 (c).

Let us assume an user provided with map of Fig. 3 (a) needs to increase the level of detail, so that lines l_0, l_1, l_2 of Fig. 3 (c) are shown. The process would be more efficient if all the points of Fig. 3 (b) were not added, but simply points p_0, p_1, p_2 and p_3 were created. In general, given a set of updates which have been performed in a given order, we are interested in performing just some of the updates, not necessarily consecutive in the sequence. This is not straightforward, as some updates require before being applied that other updates have been previously performed. For example, line l_0 cannot be added directly in the map of Fig. 3 (a), unless points p_0 and p_3 have been previously added to the map. This fact will be formalised by means of the dependency concept.

4.1 Sequences of Updates

Let M be a map and (u_1, u_2, \dots, u_n) be a collection of updates such that u_1 is an update on M , and every u_i , $2 \leq i \leq n$, is an update on the map obtained by applying to M all updates preceding u_i in the sequence. Then, the pair $S = (M, (u_1, u_2, \dots, u_n))$ is called a *sequence of updates* for M .

We consider *monotonic sequences*, i.e., sequences where updates are either all atomic abstraction updates, or all atomic refinement updates. Monotonic sequences of the two types are mutually inverse. We use the following notation: a sequence of atomic abstraction updates is denoted by $\underline{S} = (\overline{M}, (\underline{u}_1, \underline{u}_2, \dots, \underline{u}_n))$, and its inverse sequence of atomic refinement updates is denoted by $\overline{S} = (M, (\overline{u}_1, \overline{u}_2, \dots, \overline{u}_n))$, where, for $1 \leq i \leq n$, \overline{u}_{n-i+1} and \underline{u}_i are mutually inverse updates, and \overline{M} [M] is the map obtained from M [\overline{M}] by applying all modifications of \overline{S} [S] to it. Fig. 4 shows a sequence of atomic abstraction updates (from right to left), and its inverse sequence of atomic refinement updates (from left to right).

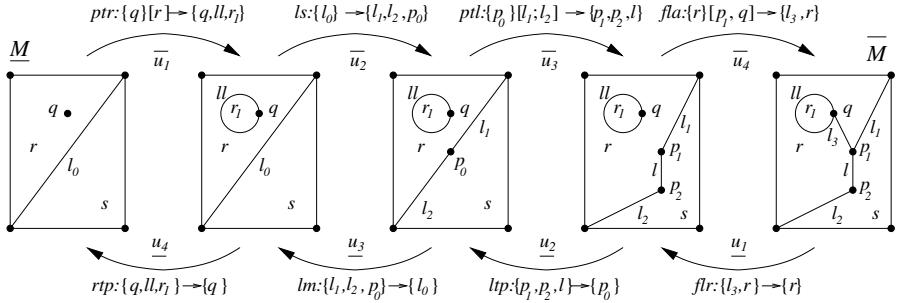


Fig. 4. $S = (\bar{M}, (\bar{u}_1, \bar{u}_2, \bar{u}_3, \bar{u}_4))$ is a sequence of atomic abstraction updates and $\bar{S} = (\underline{M}, (\bar{u}_1, \bar{u}_2, \bar{u}_3, \bar{u}_4))$ is the inverse sequence of atomic refinement updates.

4.2 Naïve Dependency

In the remainder, we focus on refinement sequences. We are interested in increasing the level of detail of \underline{M} selectively, by applying to \underline{M} only a subset of the sequence of refinement updates. Not all subsets of \bar{S} , however, make sense as sequences on \underline{M} .

Given an update \bar{u}_i , with $1 \leq i \leq n$, in general \bar{u}_i cannot be applied unless some other updates in \bar{S} have been applied previously. For example, in Fig. 4, update \bar{u}_3 needs update \bar{u}_2 , because point p_0 is necessary to perform \bar{u}_3 , and this point is created in \bar{u}_2 .

Given update \bar{u}_i , a sufficient condition to apply \bar{u}_i to a map is that the map contains every entity affected by \bar{u}_i in the original sequence. For example, in the upper part of Fig. 5 we have depicted how update \bar{u}_2 can be directly applied on \underline{M} . Update \bar{u}_3 can be applied after \bar{u}_2 , and update \bar{u}_4 can be applied after \bar{u}_3 . Update \bar{u}_4 is independent of update \bar{u}_1 , whereas \bar{u}_4 is dependent on \bar{u}_3 . Fig. 6 (a) shows the dependency links existing among the updates of Fig. 4, based on this convention.

4.3 Refined Dependency

The naïve dependency is too strict, and it tends to create unnecessarily long sequences of dependent updates. In fact, an update \bar{u}_i may be performed on a map although not every entity affected by it in the original sequence is present in the map. It is sufficient that the map contains, for each such entity, an entity representing it at a possibly different scale. For example, consider update \bar{u}_4 in Fig. 4. We want to apply \bar{u}_4 even if p_1 does not exist, but some other point representing it, for instance point p_0 , exists. In this view, \bar{u}_4 depends just on \bar{u}_2 , while \bar{u}_3 and \bar{u}_4 are independent, and can be applied in any order, as we can observe in Fig. 5. The dependency relation in this case is depicted in Fig. 6 (b).

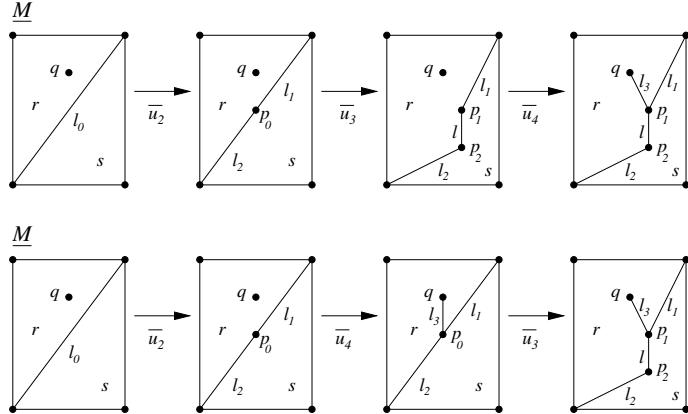


Fig. 5. Examples of feasible subsequences obtained from the sequence in Fig. 4.

Forests

Let a and a' be two entities of the same type (points, lines or regions), belonging to sequence \overline{S} . We say that a and a' are mutually *representative* if either $a \equiv a'$, or $a' [a]$ can be obtained from $a [a']$ through the application of updates in the sequence. For instance, in Fig. 4, points p_0 and p_1 are mutually representative, and so are points p_0 and p_2 . In order to keep track of representatives, we define forests of points, lines and regions, called *point-forest*, *line-forest*, and *region-forest*, respectively (see Fig. 6 (c)).

In the point-forest, roots are points created in updates of types *line-split* or *feature-point-addition*, and the children of a node, if any, are the two points created from it in an update of type *point-to-line*.

The roots of the line-forest are those lines that have been created in any update of type *point-to-line*, *point-to-region*, *region-split*, or *feature-line-addition*, and the two children of a line, if existing, are those lines obtained from it through updates of type *line-split* or *line-to-region*. Each branch of such trees is labelled with the corresponding type of atomic update.

Finally, the roots in the region-forest are those regions obtained in updates of type *point-to-region* or *line-to-region*, and the children of a region, if any, are the two regions obtained from it by an update of type *region-split*.

Two entities are mutually representative if they lie on the same path in the appropriate forest. The *most abstract representative* of an entity a is the eldest ancestor of a in the forest. In a more restrictive view, we define a and b to be mutually representative *through a given set of update types* if they lie on a path formed only of updates belonging to types of the given set.

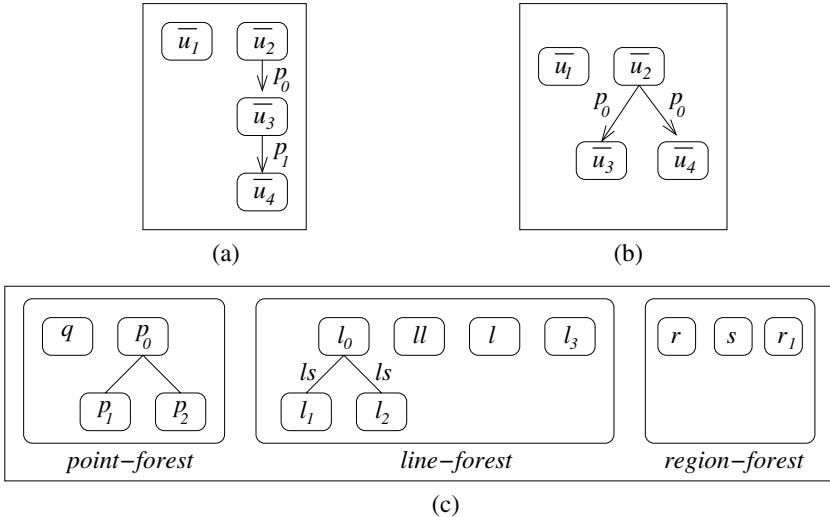


Fig. 6. Dependencies among updates according to the naïve ((a)) and refined ((b)) formulation. An arc from \bar{u}_i to \bar{u}_j means that \bar{u}_j directly depends on \bar{u}_i , and such arc is labelled with the entity created by \bar{u}_i and either removed or needed by \bar{u}_j (c) Entity forests corresponding to the sequence in Fig. 4.

Feasible Subsequences

Let $\bar{u}_{i_1} : \{a\}[b_1, \dots, b_m] \rightarrow \{c_1, c_2, c_3\}$, with possibly $c_3 = \emptyset$, be an update in sequence $\bar{S} = (\underline{M}, (\bar{u}_1, \dots, \bar{u}_n))$. Let us assume that entity a has some representative entity a' in \underline{M} , and that \bar{u}_{i_1} is directly applied on \underline{M} by considering a' instead of a . Those entities incident on a' become incident on some of the entities already existing in \underline{M} , or created by \bar{u}_{i_1} , so that the result of the application of \bar{u}_{i_1} on \underline{M} is a map, denoted by $\underline{M} \oplus \bar{u}_{i_1}$.

Let us now assume that the process is repeated until a map $\underline{M} \oplus \bar{u}_{i_1} \oplus \dots \oplus \bar{u}_{i_p}$ is obtained, with $\bar{u}_{i_1}, \dots, \bar{u}_{i_p} \in \bar{S}$. We say that subsequence $(\bar{u}_{i_1}, \dots, \bar{u}_{i_p})$ is *feasible* on \underline{M} with respect to \bar{S} if the application on \underline{M} of updates $(\bar{u}_1, \dots, \bar{u}_{i_p})$ produces the same map as the application on \underline{M} of the updates in $(\bar{u}_{i_1}, \dots, \bar{u}_{i_p})$ followed by those updates in $(\bar{u}_1, \dots, \bar{u}_{i_p})$ which are not in $(\bar{u}_{i_1}, \dots, \bar{u}_{i_p})$, i.e., if:

$$\underline{M} \oplus \bar{u}_{i_1} \oplus \dots \oplus \bar{u}_{i_p} \oplus \sum_{m=1}^{i_p-1} \bar{u}_m = \underline{M} \oplus \sum_{\substack{m=1 \\ m \neq i_1, \dots, i_p}}^{i_p} \bar{u}_m$$

where the summations are performed by \oplus operation. Examples of feasible subsequences of the sequence in Fig. 4 are $\bar{S}' = (\underline{M}, (\bar{u}_2, \bar{u}_4, \bar{u}_3))$, and $\bar{S}'' = (\underline{M}, (\bar{u}_2, \bar{u}_3, \bar{u}_4))$ (see Fig. 5). On the other hand, for instance, $\bar{S}''' = (\underline{M}, (\bar{u}_1, \bar{u}_3, \bar{u}_4))$ is not a feasible subsequence.

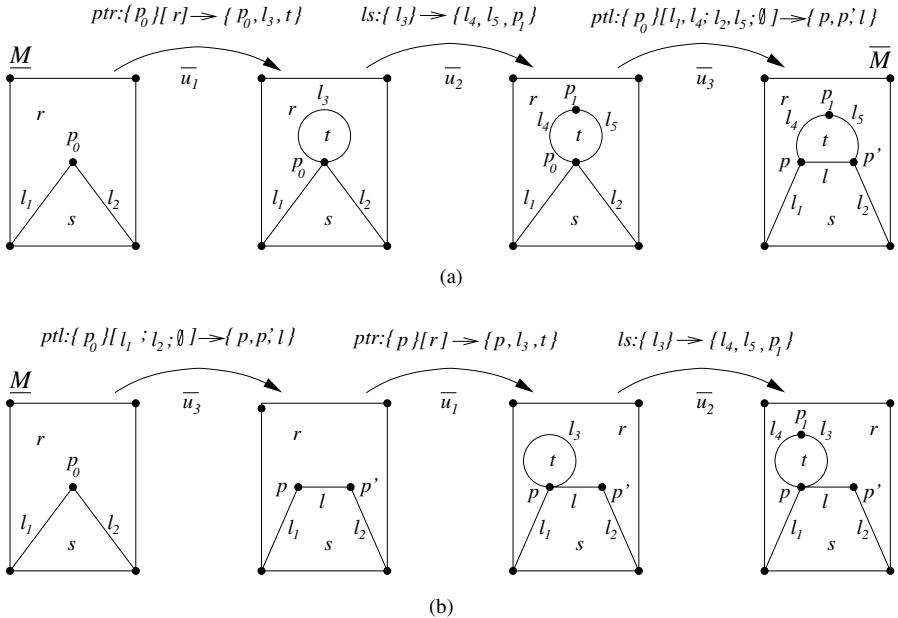


Fig. 7. (a) Sequence of atomic refinement updates (b) Line l_3 belongs to the minimal domain of \bar{u}_3 , and as this line has not been created before the application of \bar{u}_3 , subsequence $(\underline{M}, (\bar{u}_3, \bar{u}_1, \bar{u}_2))$ is not feasible.

Minimal Domain

We are interested in characterising the set of feasible subsequences which can be obtained from any sequence $\bar{S} = (\underline{M}, (\bar{u}_1, \dots, \bar{u}_n))$. It can be proved that, given any subset $\{\bar{u}_{i_1}, \dots, \bar{u}_{i_j}\}$ of the updates in \bar{S} , a sufficient condition for $\bar{S}' = (\underline{M}, (\bar{u}_{i_1}, \dots, \bar{u}_{i_j}))$ being a feasible subsequence of \bar{S} is that, for each update \bar{u}_{i_k} in \bar{S}' , some entities have been created previously to the application of \bar{u}_{i_k} . Such entities form what we call the *minimal domain* of \bar{u}_{i_k} . The fact that an entity a has been created means that \underline{M}' must contain either a , or some representative of a which is a descendant of a in the appropriate entity tree.

In general, the minimal domain for an atomic refinement update $\bar{u} : \{a\}[b_1, \dots, b_m] \rightarrow \{c_1, c_2, c_3\}$ contains the most abstract representatives, through the same type of update as \bar{u} , of entities a, b_1, \dots, b_m . There are, however, two exceptions, which are listed below:

- (a) If \bar{u} is of type $ptr : \{p_0\}[r] \Rightarrow \{p_0, l_3, t\}$, the minimal domain contains:
 - the most abstract representative of point p_0
 - if l is a feature-line, the most abstract representative of region r_0

- the line(s), if existing, created in an update of type *point-to-region* or *line-to-region*, such that either such line(s), or some of its (their) descendants in the line-forest, have one endpoint at p and the other one at p' in \bar{S} . In Fig. 7 (a), a sequence of atomic refinement updates is shown, and (b) illustrates that it is not possible to retrieve map \bar{M} if line l_3 has not been created before point p_0 is expanded to $\{p, p', l\}$.
- (b) If \bar{u} is of type *rs* : $\{r_0\}[p_1, p_2; c_{r_1}, \dots, c_{r_l}; c'_{r'_1}, \dots, c'_{r'_m}] \rightarrow \{r, r', l\}$, the minimal domain consists of:
- the most abstract representatives of r_0, p_1, p_2
 - every line which has been created in an update of type *feature-line-addition*, and such that, in sequence \bar{S} , either such line or some of its representatives belong to the outer boundary of r or r' . As we can observe in Fig. 8 (b), line l_5 must be created before region r_0 is split into two regions, otherwise the insertion of line l_6 would not split r_0 .

Direct Dependency Relation

Given a sequence $\bar{S} = (\underline{M}, (\bar{u}_1, \bar{u}_2, \dots, \bar{u}_n))$, we have explained that, for $\bar{S}' = (\underline{M}, (\bar{u}_{i_1}, \bar{u}_{i_2}, \dots, \bar{u}_{i_j}))$ being a feasible subsequence, it is necessary that for each \bar{u}_{i_k} in \bar{S}' , the entities forming its minimal domain have been created in updates $\bar{u}_{i_{(k_1)}}, \dots, \bar{u}_{i_{(k_m)}}$ previous to \bar{u}_{i_k} in the subsequence. We say that \bar{u}_j *directly depends* on \bar{u}_i , $1 \leq i < j \leq n$, if some entity belonging to the minimal domain of \bar{u}_j has been created in \bar{u}_i .

A sequence $\bar{S} = (\underline{M}, (\bar{u}_1, \bar{u}_2, \dots, \bar{u}_n))$ is *non-redundant* if there are no two different updates in the sequence, $\bar{u}_i : \{a_i\}[b_{i1}, \dots, b_{im}] \rightarrow \{c_{i1}, c_{i2}, c_{i3}\}$ and $\bar{u}_j : \{a_j\}[b_{j1}, \dots, b_{jp}] \rightarrow \{c_{j1}, c_{j2}, c_{j3}\}$, with $1 \leq i < j \leq n$, such that $a_i \notin \{c_{i1}, c_{i2}, c_{i3}\}$ and $a_i \in \{c_{j1}, c_{j2}, c_{j3}\}$. Intuitively, it means that an entity that has been removed due to the application of one update in the sequence, cannot reappear again as one of the entities created by any other posterior update in the sequence.

Let $\bar{S} = (\underline{M}, (\bar{u}_1, \bar{u}_2, \dots, \bar{u}_n))$ be a non-redundant sequence of atomic refinement updates. It is easy to see that the transitive closure of the direct dependency relation is a *strict partial order*, \prec .

A subsequence of updates from \bar{S} is called *closed* with respect to the relation of direct dependency if, for each update \bar{u} in it, \bar{S} contains also the updates \bar{u} depends on. It can be shown that any closed subsequence is *feasible* [7].

5 The Multiresolution Model

In this Section, we define our multi-scale model for maps, which is inspired by the Multi-Triangulation developed for triangle meshes [5, 9].

We define a *Multi-Map* as the pair (\bar{S}, \prec) , where:

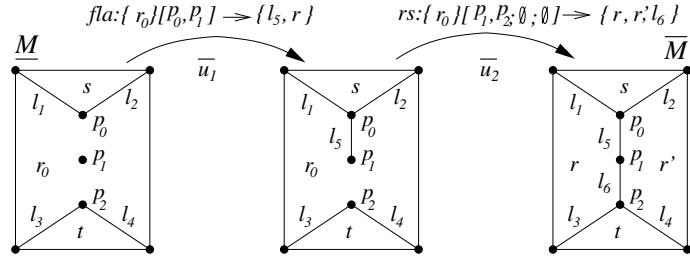


Fig. 8. Line l_5 belongs to the minimal domain of update \bar{u}_2 , because \bar{u}_2 cannot be directly applied on \underline{M} unless line l_5 has been previously created.

- $\bar{S} = (\underline{M}, \bar{u}_1, \dots, \bar{u}_n)$ is a non-redundant sequence of atomic refinement updates. In practice, \bar{S} is the reverse of a sequence $S = (M, u_1, \dots, u_n)$ of abstraction updates that have been performed during a simplification process.
- \underline{M} is called the *base map* and corresponds to the coarsest map available in the model.
- \bar{M} , the map obtained by applying all refinement updates to \underline{M} , is called the *reference map*, and corresponds to the most detailed map available in the model.
- \prec is the partial order on $\{\bar{u}_1, \dots, \bar{u}_n\}$ defined by direct dependency, as in the previous section.

A *subMulti-Map* is the restriction of a Multi-Map to a set of updates closed with respect to the relation of direct dependency. It follows that the application of all the updates in a subMulti-Map to the base map \underline{M} produces a map. The map obtained after the application of all the updates in a subMulti-Map is called an *extracted map*.

We are now interested in obtaining a map whose scale is variable in space, according to arbitrary user requirements. We assume that user requirements are given by means of an external Boolean function $\tau()$, defined over the updates of a Multi-Map, which decides whether an update is *necessary* or not in order to achieve (locally) the level of detail needed by the user/application. For instance, if all updates acting in a certain focus area are necessary, then refinement will produce the maximum level of detail inside that area, while leaving the rest of the map at a lower detail.

The selective refinement will produce the smallest map, extracted from the Multi-Map, in which all updates necessary according to $\tau()$ have been performed. Such map is generated from the minimal set of updates which contains all updates verifying $\tau()$ and is closed with respect to the relation of direct dependency.

The general principles underlying the selective refinement algorithm for maps are similar to those for Multi-Triangulations that we have developed in our previous work [5, 9], although the specific concepts and details for deciding

where to refine, the rules for defining refinement, and the data structures to encode the model and the map are very different from those used in Multi-Triangulations.

6 Conclusions and Future Work

We have presented a combinatorial description of a multi-scale model for maps represented by plane graphs. The model is theoretically sound, it allows for selective refinement and is promising in terms of flexibility and applicability to real data. In our current work, we are designing a data structure for encoding it, as well as algorithms operating on it. In future work, also geometry and semantics must be taken into account, and generalisation algorithms and criteria for building and querying the model on the basis of metrics, topology and semantics will be developed.

Acknowledgements

The kind help of Leila De Floriani for helpful discussions and a careful reading of this paper is gratefully acknowledged. This work has been partially supported by the Research Training Network EC Project on *Multiresolution in Geometric Modelling* (MINGLE), under contract HPRN-CT-1999-00117, and by the project funded by the Italian Ministry of Education, University, and Research (MIUR) on *Representation and Management of Spatial and Geographical Data in the Web*, Protocol N. 2003018941.

References

1. M. Bertolotto. Geometric modeling of spatial entities at multiple levels of resolution. Ph.D.Thesis, Department of Computer Science, University of Genova, DISI-TH-1998-01, 1998.
2. M. Bertolotto, L. De Floriani, and E. Puppo. Multiresolution topological maps, *Advanced Geographic Data Modelling – Spatial Data Modelling and Query Languages for 2D and 3D Applications*, M. Molenaar, S. De Hoop (eds.), Publications on Geodesy – New Series, N. 40, Netherland Geodetic Commission, pp. 179-190, 1994.
3. M. Bertolotto and M. J. Egenhofer. Progressive transmission of vector map data over the world wide web. *GeoInformatica* 5(4), pp. 345-373, 2001.
4. B. Buttenfield. Progressive transmission of vector data on the internet: a cartographic solution. *Proceedings 19th International Cartographic Conference*, Ottawa, Canada, pp.581-590, 1999.
5. L. De Floriani and P. Magillo. Multiresolution mesh representation: models and data structures, *Tutorials on Multiresolution in Geometric Modelling*, A. Iske, E. Quak, M. S. Floater (eds.), Springer-Verlag, pp. 363–418, 2002.

6. D. H. Douglas and T. K. Pücker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10, 2, pp. 112-122, 1973.
7. P. Magillo, E. Puppo, and R. Viaña. A Multi-Scale Model for Geographic Maps. Technical Report, DISI-TR03-10, Department of Information and Computer Sciences, University of Genova, 2003.
8. E. Puppo and G. Dettori. Towards a formal model for multiresolution spatial maps. *Advances in Spatial Databases*, M. J. Egenhofer, J. R. Herring (eds.), LNCS Vol.951, Springer-Verlag, pp.152-169, 1995.
9. E. Puppo. Variable resolution triangulations. *Computational Geometry Theory and Applications* 11, pp.219-238, 1998.

Part III

— Modelling

Constrained Multiresolution Geometric Modelling

Stefanie Hahmann¹ and Gershon Elber²

¹ Laboratoire LMC-IMAG, Institut National Polytechnique de Grenoble, France

Stefanie.Hahmann@imag.fr

² Technion, Haifa, Israel

gershon@cs.technion.ac.il

Summary. This paper surveys the state-of-the-art of geometric modelling techniques that integrate constraints, including direct shape manipulation, physics-based modelling, solid modelling and freeform deformations as well as implicit surface modelling. In particular, it focuses on recent advances of multiresolution modelling of shapes under constraints.

1 Introduction

Freeform curves, surface and solids are generally represented in the B-spline basis. Various geometric quantities, such as control points, knots and weights have to be specified. Controlling the shape of an object under complex deformations by manipulating the control points directly is often difficult. The movement of control points gives an indication of the resulting deformation, but being extraneous to the object, the control points do not allow for precise control of the shape. In addition, large deformations of complex objects with many details to be preserved become nearly impossible without any “higher level” control mechanisms. User-friendly shape-control tools, therefore, generally make use of modelling techniques that integrate constraints. The present paper surveys the state-of-the-art of geometric modelling techniques that integrate constraints, including direct shape manipulation, physics-based modelling, solid modelling and freeform deformations as well as implicit surface modelling. In particular, we focus on recent advances of multiresolution modelling of shapes under constraints. Going beyond the limits of traditional modelling techniques, they allow for editing of complex objects while automatically preserving the details.

2 Interactive Freeform Techniques

Controlling the shape of an object under complex deformations is often difficult. The traditional approach to interacting with deformable objects is to manipulate control points since they allow precise control over models. CAGD textbooks by Farin [35], Hoschek and Lasser [53], and Cohen et al. [22] cover the complete theory of parametric freeform curve and surface representations such as NURBS curves and tensor product surfaces, triangular Bézier patches, n -sided patches, but also Coons and Gregory surfaces. Limited by the expertise and patience of the user, the direct use of control points as the manipulation handles necessitates an explicit specification of the deformation. Therefore, large deformations can be extremely difficult to achieve because they mandate moving a huge number of individual control points by hand, and the precise modification of the freeform object can be laborious. Deformation tools based on geometric constraints offer more direct control over the shape. In this section high-level interactive freeform curve and surface manipulation techniques are presented. These use either geometric constraints as direct deformation handles (Sect. 2.1) or as definitions of functional behaviour via geometric properties (Sect. 2.2). Finally, geometry-driven (freeform) solid modelling techniques are described (Sect. 2.3).

2.1 Direct Curve and Surface Manipulation

Rather than manipulating control points, Bartels and Beatty show in [2] how to pick 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 minimising the control points' offsets. In [40] Fowler and Bartels control 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 [39]. 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 [12] deform B-spline surfaces by determining the displacement and radius of influence for each constrained surface point. Hsu et al. [54] propose point picking for freeform deformations. Curve constraints, i.e. forcing the surface to contain a given curve or to model a character line, have been considered in [18, 46, 73]. Direct shape manipulation techniques are closely related to Variational Design, where the objective of obtaining fair and graceful shapes is achieved by minimising some energy, see Sect. 4.1. 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 prescribed by minimising some energy functional, such as bending. For example, Welch et al. [97] maintains the imposed constraints while calculating a surface that is as smooth as possible. Celniker and Welch [18] derive interactive sculpting techniques for

B-spline surfaces based on energy minimisation, keeping some linear geometric surface-constrained features unchanged. Celniker and Gossard [17] enforce linear geometric constraints for shape design of finite elements governed by some surface energy. While energy minimisation affects the surface globally, finite element methods allow for local control. Forsey and Bartels [38] later used the technique of hierarchical B-splines in an attempt to overcome this drawback for B-spline surfaces.

2.2 Feature Modelling

Constrained geometric modelling also occurs in feature modelling – a quite different context. Geometric modelling tools are commonly used in various phases of product development, for example, to generate product images or NC-code. Many applications, however, require functional information that is not contained in geometric models. A feature in a product model combines geometric information with functional information, such as information about its function for the user in a design application, or its manufacturing process in a manufacturing application. Features are higher level entities compared to the underlying geometry and as such are easier to maintain and manipulate at the user level.

The concept of features has been investigated mainly in mechanical environments [26, 81]. This is due to the fact that classical mechanical parts are defined by canonical geometry shapes, which can easily be classified. Constraints occur at different stages in feature modelling. In [3], a semantic feature modelling approach is presented. All properties of features including their geometric parameters, their boundaries, their interactions, and their dependencies, are declared by means of constraints. Another issue in feature modelling is feature validation, which concerns the meaning of a feature, given by its information content [29]. A feature modelling system should ensure that product modifications by a user are in accordance with the meaning of the features. Here, constraints are used to specify such feature validity conditions; constraint satisfaction techniques are applied to maintain feature validity under product modifications from multiple views.

More recently, freeform feature modelling approaches have been developed in [16, 92, 91]. In contrast to the feature-based approach adopted by CAD systems for classical mechanical design, freeform features are strongly related to aesthetic or styling aspects when modelling with freeform surfaces. The Brite-Euram project FIORES (Formalization and Integration of an Optimized Reverse Engineering Styling Workflow) focused on the development of modelling tools for direct shape modifications closer to the stylist's way of thinking [25]. Here again, properties of aesthetic features are expressed in terms of constraints, including convexity, shape-preserving deformations, eliminations and cuts, and continuity conditions [37].

Surface pasting is a freeform feature modelling technique that composes surfaces to construct surfaces with varying levels of detail. A feature is placed

on top of an existing surface to provide a region of increased detail. In [1] B-spline surfaces, called features, are applied to any number of B-spline surfaces, called base surface, in order to add details on the base surface. Improvements and further developments of pasting techniques for B-spline surfaces can be found in [23, 66].

2.3 Solid Modelling

The history of solid modelling goes back to the 1980s when the term “solid modelling” was introduced; see survey papers [79, 80]. This was also the period when early advances were motivated primarily by the mechanical engineering industry. Traditional solid modelling approaches include implicit functions (CSG and blobby models), boundary representations and cell decompositions. The use of constraints has mainly been developed from interaction with freeform solids.

Sederberg and Parry [84] developed a technique for globally deforming solid models in a free-form manner, called free-form deformation (FFD). The three-dimensional object to be deformed is embedded in a three-dimensional parametric space, called the control lattice. The vertices of the object are assigned parametric values that depend on their positions inside the parametric solid (usually a Bézier or B-spline solid). A deformation applied to the solid via its control points deforms the embedded object in response. FFD lattices can be patched together to provide local control over the deformation. Coquillart [24] extended FFD to non-paralleliped lattices, represented by rational splines. Hsu et al. [54] improved the traditional FFD with a technique that allows the user to manipulate the embedded object directly. It computes how the Bézier (or B-spline) control points must move in order to produce the desired deformation. Shi-Min et al. [85] proposed a similar scheme in which an FFD function is computed based on the manipulation and translation of a single point. Complex deformations are then achieved via the composition of several such single-point FFDs. MacCracken and Joy [64] generalised FFD by incorporating arbitrary-topology subdivision-based lattices.

Rappoport et al. [76] derived a method for modelling tri-variate Bézier solids while *preserving the volumes*. Different solids can be patched together at their boundaries to create a more complex object. Their algorithm uses an energy minimisation function whose purpose is to preserve the volume during sculpting. In addition to the volume-preserving constraint, their system can satisfy inter-patch continuity constraints, positional constraints, attachment constraints, and inter-point constraints.

Hirota et al. [51] developed an algorithm for *preserving the global volume* of a B-rep solid undergoing a free-form deformation. Following a user-specified deformation the algorithm computes the new node positions of the deformation lattice, while minimising an energy functional subject to the volume preservation constraint. During initialisation, each triangle in the surface is projected onto the x-y plane, and the volume under the triangle is stored.

During the deformation process, this volume is constantly re-computed and compared to the original. By taking the difference between the volumes of the original and deformed volume elements, the total change in volume is computed.

Self-intersection could clearly occur in the FFD function. Local self-intersection can be identified via the vanishing Jacobian of the FFD, an approach proposed in [41].

3 Implicit Surfaces

Implicit surfaces have sparked great interest in the computer graphics and animation community [100, 7, 27, 71, 101], with applications for geometric modelling and scientific visualisation [62]. Deformations of implicit surfaces can be obtained intuitively by articulating the skeleton or by changing the parameters of implicit primitives that hierarchically define the surface [14, 13]. Another, more intricate way to deform implicit models is to change the iso-surface progressively by modifying the sample field function defining it [98, 28].

Two kinds of constraints are particularly easy to integrate. First, collision detection can be accelerated, since in-out functions are provided. Second, implicit surfaces provide a good tool for physics-based animation; see Sect. 4.

The volume of an implicit object is another constraint that it is important to preserve during deformation [27, 13, 28]. For example, volume constant deformations in a morphing process can make virtual objects look like real ones. A more complete overview on implicit surface modelling can be found in [8].

4 Physics-based Modelling

Physics-based modelling attaches physical properties to geometric structures in order to achieve better or more fair shapes for design purposes, or in order to increment realism in computer animations. The constraints are formulated in terms of energy functionals or kinetic and mass laws that are, in many cases, non-linear.

4.1 Variational Shape Design

Although it is difficult to exactly define, in mathematic terms, what *fairness* of a curve or surface is, it is commonly accepted that smooth and graceful shapes are obtained by minimising the amount of energy stored in the surface. The energy functionals originating from elasticity theory are in general non-linear, such as the bending energy for curves $\int \kappa^2(t)dt$ or the thin-plate energy for surfaces $\int \kappa_1^2 + \kappa_2^2 dA$. These and other higher order non-linear energy functionals have been used in [70, 45].

In order to accelerate computations, linearised versions of these energy functionals are generally used; see, for example, [17, 18, 97, 44]

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

where α and β are weights on stretching and bending. This produces a surface which tends to minimise 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 linearised in the optimisation process.

Historically, use of such energy functionals goes back to early spline and CAGD literature [69, 78] and has led to a research area, called Variational Design of smooth curves and surfaces, today [34, 47, 48, 9, 49].

4.2 Dynamic Modelling

Deformations of objects are obtained by externally applying forces. The dynamic approach based on well-established laws of physics aims to produce smooth and natural motions in order to create realistic-looking computer animation. Traditional animation techniques [59] have to be considered as well. To synthesise convincing motions, the animator must specify the variables at each instant in time, while also satisfying kinematic constraints.

Terzopoulos et al. [88] introduced freeform deformable models to computer graphics, pioneering the development of dynamic parametric curves, surfaces and solids. Animation of implicit surfaces goes back to [100]. Gravitational, spring, viscous and collision forces applied to the geometric model act as constraints when deforming objects. Non-linear dynamic behaviour [89] results from simulating inelastic deformation. Different dynamic behaviour of deformable objects has been developed by many varying the imposed constraints, the numerical solution method or by applying these to different geometric models, including modal dynamics [72], animation of non-rigid articulated objects [99], FEM-based methods [17], D-NURBS [90], implicit surfaces [27], deformable voxel methods [19] and dynamic subdivision surfaces [75]. In [74], dynamic parameters are directly evaluated over B-spline curves, while parameterization of the curve is ignored.

5 Multiresolution Editing

Multiresolution analysis has received considerable attention in recent years in many fields of computer graphics, geometric modelling and visualisation [86, 94]. It provides a powerful tool for efficiently representing functions at multiple levels-of-detail with many inherent advantages, including compression, LOD display, progressive transmission and LOD editing.

In the literature the term multiresolution (MR) is employed in different contexts, including wavelets, subdivision and hierarchies or multigrids. Multiresolution representations based on wavelets have been developed for parametric curves [20, 61, 36], and can be generalised to tensor-product surfaces, to surfaces of arbitrary topological type [63], to spherical data [83], and to volume data [21]. Wavelets provide a rigorous unified framework. Here, a complex function is decomposed into a “coarser” low resolution part, together with a collection of detail coefficients, necessary to recover the original function. Other multiresolution representations exist for data defined on irregular meshes [10, 11], for arbitrary meshes [102, 58, 30, 52], for tensor product surfaces, known as hierarchical B-splines [38], and for volumetric data sets represented using tri-variate functions [77].

In the context of geometric modelling, LOD editing is an attractive MR application because it allows the 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 the manipulation of a lot of control points (see Sect. 2), MR methods can achieve the same effect by manipulating only a few control points of some low resolution representation; see [36, 86]. However, there are application areas, including CAGD and computer animation, where deformations under constraints are needed. As stated in the introduction, it is obvious that constraints offer an additional and finer control of the deformation applied to curves and surfaces.

Continuing the previous sections, the present section reports on constrained modelling methods using MR representations. Sect. 5.1 presents an LOD editing method for B-spline curves and surfaces that allows the integration of linear and non-linear geometric constraints, including fixed position, symmetry and constant area. Sect. 5.2 presents wavelet-based MR curve editing methods preserving area and length of curves. Sect. 5.3 is about variational MR methods, where minimum energy is the constraint to be satisfied. Finally, Sect. 5.4 describes MR subdivision methods.

5.1 Constrained Multiresolution Control

Multiresolution Editing of Freeform Curves

In [36, 44], a wavelet decomposition for uniform cubic B-splines is presented toward interactive and intuitive manipulation of freeform shape. In [55], results from [61] are similarly employed toward the support of non uniform knot sequences. While local support is considered the major advantage of the B-spline representation, it is also its Achilles heel. Global changes are fundamentally difficult to apply to a highly refined shape and a painstaking laborious manual effort is required to move one control point at a time. The ability to decompose a given freeform B-spline curve or a surface as offered by [36, 44, 55] is a large step in the direction that alleviates these difficulties. The



Fig. 1. Multiresolution manipulation of a non uniform quadratic B-spline curve with 138 control points. In all six images (a)–(f), a single select-and-drag operation was applied to the top of the 's' letter in the upward direction.

user can now modify the shape locally or globally as he/she see fits. In Fig. 1, a single select-and-drag operation is applied to a non uniform quadratic B-spline curve at six different resolutions. The outcome clearly shows the power of multiresolution editing, allowing for both local and global control.

Let $C(t) = \sum_{i=0}^{n-1} P_i B_{i,\tau,k}(t)$ be a planar non uniform B-spline curve of order k and n control points. Let the knot sequence of $C(t)$ be

$$\tau = \{t_0, t_1, \dots, t_{k-1}, \dots, t_n, \dots, t_{n+k-1}\}.$$

$C(t)$ is defined for the domain $[t_{k-1}, t_n]$. The knots from t_k to t_{n-1} are denoted the *interior knots* and their removal does not affect the domain of $C(t)$.

The knot sequence of τ , together with the order k , define a subspace Φ of piecewise polynomial functions. This subspace contains all polynomial functions but also piecewise polynomials with potential discontinuities at each of the interior knots, depending on the multiplicity of the knot. Let $\tau_0 = \tau$ and further let $\tau_{i+1} \subset \tau_i$ by removing only interior knots from τ_i . Then:

- The domain spanned by all the τ_i is the same and equal to $[t_{k-1}, t_n], \forall i$.
- The subspace Φ_{i+1} induced by τ_{i+1} and k is a strict subspace of Φ_i . That is $\Phi_{i+1} \subset \Phi_i$.

Clearly $C(t) \in \Phi_0$. Denote $C(t)$ as $C_0(t)$. Let Φ_1 be a new subspace formed out of Φ_0 , by removing a single knot $t_j = \tau_0/\tau_1$. We seek to find the orthogonal projection, under the L_2 norm, of $C_0(t)$ onto Φ_1 . Denote this



Fig. 2. Projections (in thick grey) of the original “multiresolution” curve from Fig. 1 (in thin line) over different spline subspaces are presented. The top left is the smallest space (single quadratic polynomials) all the way to the bottom right which is the original space.

projection by $C_1(t) \in \Phi_1$ and let the difference be $D_1(t) = C_0(t) - C_1(t)$. We call $C_1(t) = \sum_{i=0}^{n-2} Q_i B_{i,\tau_{1,k}}(t)$ a low resolution version of $C_0(t)$ and $D_1(t)$ the details. $D_1(t)$ is in a new subspace $\Psi_0 \subset \Phi_0$ which means we can express $D_1(t)$ in terms of the basis functions of Ψ_0 as

$$D_1(t) = \sum_{i=0}^{n-1} d_i B_{i,\tau_{0,k}}(t).$$

$D_1(t) \in \Psi_1$ is orthogonal to the space of Φ_1 . Hence, the following must hold,

$$0 = \langle D_1(t), B_{m,\tau_{1,k}} \rangle = \sum_{i=0}^{n-1} d_i \langle B_{i,\tau_{0,k}}, B_{m,\tau_{1,k}} \rangle. \quad (1)$$

It turns out that equation (1) completely prescribes the coefficients of $D_1(t)$ up to uniform scaling of the function. This $D_1(t)$ is also known as the *B-wavelet* function of knot t_j in subspace Ψ_0 . Fig. 2 presents the orthogonal projection of our “multiresolution” curve onto several subspaces, all the way to a single Bézier curve.

Unfortunately, the computation of the coefficients of $D_1(t)$, following Equation (1) is expensive, as it necessitates the resolution of products and integrals of B-spline basis functions,

$$\langle B_{i,\tau,k}(t), B_{j,\tau,k}(t) \rangle = \int B_{i,\tau,k}(t) B_{j,\tau,k}(t) dt.$$

One option is to limit these computations to uniform knot sequences only, removing half the knots each time, effectively doubling the knot spacing. This approach was taken by [36, 44] and it allows one to precompute the B-wavelets once for each order.

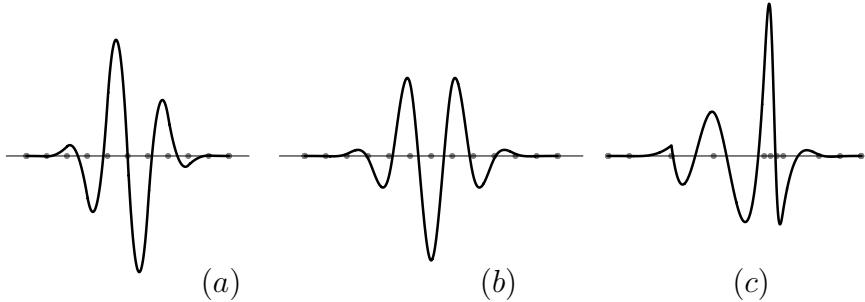


Fig. 3. B-Wavelets of a uniform quadratic **(a)**, a uniform cubic **(b)**, and a non uniform knot sequence of a cubic curve **(c)**. The third knot from the left in **(c)** is a triple knot, resulting in a C^1 discontinuity in the B-Wavelet.

For non uniform knot sequences the B-wavelets must be computed every time and while one can reach interactive rates for curves with dozens of control points, this computation as already stated is expensive. Fig. 3 presents few examples of B-wavelets. See [61, 31, 55] for more on the computation of products and integrals of B-spline basis functions as well as more on this B-wavelet decomposition. A similar computation is also necessary toward the computation of $C_{i+1}(t)$ from $C_i(t)$, given the subspace Φ_i .

Due to the computational costs, alternatives were sought. One alternative is to approximate the low resolution projection using a simple least squares fit [32]. Given $C_i(t) \in \Phi_i$, find a least squares fit $C_{i+1}(t) \in \Phi_{i+1}$ to $C_i(t)$ by sampling $C_i(t)$ at m locations, $m \gg n_{i+1}$, n_{i+1} the number of coefficients in $C_{i+1}(t)$. Nevertheless, for the task in hand of interactive multiresolution manipulation with constraints, this B-wavelet decomposition is not really necessary! Consider curve $C_i(t) \in \Phi_i$, $\Phi_i \subset \Phi_0$. Now consider a change of a single control point in $C_0(t)$ against a change of a single control point in $C_i(t)$. The latter will clearly affect a larger domain of the original curve $C(t) = C_0(t) \in \Phi_0$ compared to a change in $C_0(t)$. A single control point P_j is supported along the non zero domain of its basis function $B_j(t)$. The less interior knots there are, the larger the domain of $B_j(t)$ is.

Then, a modification to the shape using a change in curve $C_i(t) \in \Phi_0$ could be added to the original curve $C_0(t)$ using knot insertion [42], refining $C_i(t)$ at all the knots of τ_0/τ_i . In practice, direct manipulation is preferred over control point manipulation, hiding the representation (i.e. control points) from the novice user. If point $C(t_1)$ is directly selected and dragged along the vector \mathbf{V} to $C(t_1) + \mathbf{V}$, a new $\Delta_i(t) \in \Phi_i$ curve could be constructed as

$$\Delta_i(t) = \frac{1}{\sigma} \sum_{i=0}^{n_i} B_{i,\tau_i,k}(t_1) B_{i,\tau_i,k}(t),$$

using the support of the different basis functions at t_1 as the weights and



Fig. 4. Multiresolution editing without (a), and with two positional (b), and three tangential (c) linear constraints.

$$\sigma = \sum_{i=0}^{n_i} (B_{i,\tau_i,k}(t_1))^2 = \sum_{i=J-k+1}^J (B_{i,\tau_i,k}(t_1))^2, \quad t_J \leq t_1 < t_{J+1},$$

yielding $\Delta_i(t_1) = 1$.

Linear Constraints

Multiresolution editing has a drawback we already discussed. It can be imprecise. We now aim to add support for constraints to our multiresolution editing capabilities. To begin with, we consider the two simple linear constraints of position and tangency.

Recall curve $C(t) = \sum_{i=0}^n P_i B_{i,\tau_i,k}(t)$. A positional constraint could be prescribed as $C(t_P) = P$. Then, if the original curve satisfies the constraint or $C(t_p) = C_0(t_p) = P$, we are now required to have $\Delta_i(t_p) = 0$, an additional linear constraint that is easy to satisfy. In practice, two possible simple approaches could be employed to solve this underconstrained linear system, having $\Delta_i(t_p) = 0$ and $\Delta_i(t_1) = 1$ as constraints and achieving an L_2 minimising solution elsewhere along the domain. Either the singular value decomposition (SVD) or the QR factorisation [43] of the linear systems of equations would do. Interestingly enough, the QR factorisation is also employed by [97] for similar reasons.

A tangential constraint can be supported in an almost identical way. Here, $C'(t_T) = T$ and $C'(t)$, that is expressed in terms of basis functions one degree lower, is elevated back to the same function space using degree elevation, resulting again in a linear alternative constraint to satisfy $\Delta'_i(t_T) = 0$. Second order or even higher derivatives constraints can easily be incorporated as well, in a similar fashion. Fig. 4 shows one example of multiresolution editing with positional and tangential constraints.

Other linear constraints can also be supported with some more effort. A planar curve, having the domain of $t \in [0, 1]$, is considered x -symmetric if $x(t) = x(1-t)$ and $y(t) = -y(1-t)$. Analogously, one can define y -symmetry and even rotational symmetry as $x(t) = -x(1-t)$ and $y(t) = -y(1-t)$. Assuming a symmetric knot sequence, that is $\tau_{i+1} - \tau_i = \tau_{k+n-i-1} - \tau_{k+n-i-2}$, $0 \leq i \leq n/2$,

$$c(1-t) = \sum_{i=0}^{n-1} P_i B_i(1-t) = \sum_{i=0}^{n-1} P_i B_{n-1-i}(t) = \sum_{i=0}^{n-1} P_{n-1-i} B_i(t)$$

due to the symmetry of the basis functions. But now the constraint of $x(t) = x(1-t)$ reduces to

$$\sum_{i=0}^{n-1} x_i B_i(t) = \sum_{i=0}^{n-1} x_{n-1-i} B_i(t), \quad \text{or} \quad \sum_{i=0}^{n-1} (x_i - x_{n-1-i}) B_i(t) = 0.$$

Hence and because of the independence of the basis functions, the symmetry constraint is now reduced to $O(n/2)$ linear constraints of the form

$$x_i = x_{n-1-i}, \quad i = 0, \dots, \left\lfloor \frac{n}{2} \right\rfloor - 1,$$

and

$$y_i = -y_{n-1-i}, \quad i = 0, \dots, \left\lceil \frac{n}{2} \right\rceil - 1.$$

Finally, we consider area constraints. The area of a closed curve equals

$$\begin{aligned} \mathcal{A} &= \frac{1}{2} \oint -x'(t)y(t) + x(t)y'(t) dt, \\ &= \oint -\sum_i x_i B'_{i,k}(t) \sum_j y_j B_{j,k}(t) + \sum_i x_i B_{i,k}(t) \sum_j y_j B'_{j,k}(t) dt \\ &= \sum_i x_i \sum_j y_j \oint -B'_{i,k}(t) B_{j,k}(t) + B_{i,k}(t) B'_{j,k}(t) dt. \\ &= [x_0, x_1, \dots, x_{n-1}] \begin{bmatrix} \xi_{0,0} & \xi_{0,1} & \cdots & \xi_{0,n-1} \\ \xi_{1,0} & \xi_{1,1} & \cdots & \xi_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \xi_{n-1,0} & \xi_{n-1,1} & \cdots & \xi_{n-1,n-1} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix}, \end{aligned}$$

where

$$\begin{aligned} \xi_{i,j} &= \oint -B'_{i,k}(t) B_{j,k}(t) + B_{i,k}(t) B'_{j,k}(t) dt \\ &= \oint -(k-1) \left(\frac{B_{i,k-1}(t)}{t_{i+k-1} - t_i} - \frac{B_{i+1,k-1}(t)}{t_{i+k} - t_{i+1}} \right) B_{j,k}(t) \\ &\quad + (k-1) \left(\frac{B_{j,k-1}(t)}{t_{j+k-1} - t_j} - \frac{B_{j+1,k-1}(t)}{t_{j+k} - t_{j+1}} \right) B_{i,k}(t) dt. \end{aligned}$$

The area constraint is not linear. Nonetheless, it is a bilinear constraint so one can fix the x_i coefficients, resulting in a linear constraint in y_i and then reverse the rôle of x_i and y_i , in the next iteration. During an interactive session, when the user selects and drags the curve's location, we need to

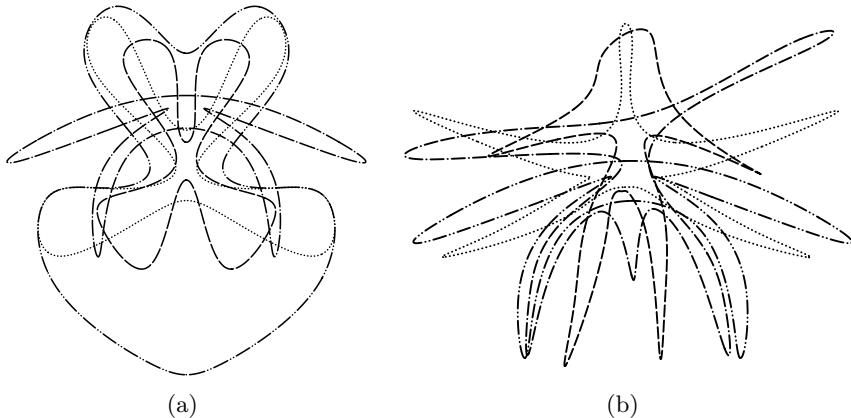


Fig. 5. Y -symmetry constraint **(a)** and area constraint **(b)** are employed in multiresolution context. These two examples were created in few seconds using direct curve manipulation under constraints.

solve these constraints at every mouse event or at almost every pixel. This interleaving process becomes fully transparent to the end user at such low granularity.

Fig. 5 shows two examples of direct manipulation of freeform curves under symmetry and area constraints. The curves were directly manipulated in real time while the symmetry and/or area constraints are fully preserved. More on the symmetry and area constraints in multiresolution editing as well as the special case of linear curves and the extension to freeform surfaces could be found in [33].

5.2 Area and Length Preserving MR Curve Editing

In a wavelet based multiresolution setting complex objects can be edited at a chosen scale with mainly two effects: First, modifying some low-resolution control points and add back the details modifies the overall shape of the object. Second, modifying a set of fine detail coefficients modifies the character of the object without affecting its overall shape. In this section a wavelet based multiresolution editing method is presented, that integrates the constant area constraint completely into the multiresolution formulation of the deformation.

Wavelet Based MR Curves

Let us briefly sketch the notation of the wavelet based multiresolution analysis that will be used in this section. For more details see [65, 36, 86]. Suppose

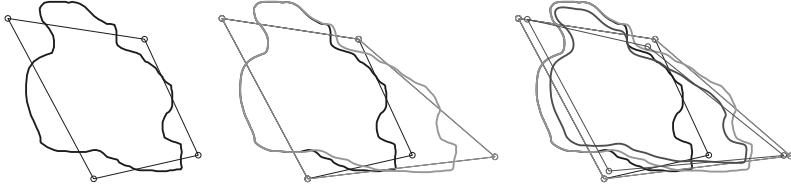


Fig. 6. Area preserving multiresolution deformation. Left: original curve and coarse control polygon, $n = 7, L = 2$. Centre: deformed curve without area constraint (in grey). Right: Deformed area preserving curve (between the other two curves).

we have a certain functional space E and some nested linear *approximation spaces* $V^j \subset E$ with $V^0 \subset V^1 \subset \dots \subset V^n$. Since we are dealing with closed curves, these spaces have finite dimension. Let V^j be spanned by a set of basis functions $\varphi^j = [\varphi_1^j, \dots, \varphi_m^j]^T$, called *scaling functions*. A space W^j being the complement of V^j in V^{j+1} is called the *detail space*. Its basis functions $\psi^j = [\psi_1^j, \dots, \psi_{N-m}^j]^T$ are such that together with φ^j they form a basis of V^{j+1} . The functions ψ_i^j are called *wavelets*. The space V^n can therefore be decomposed as follows:

$$V^n = V^{n-1} \oplus W^{n-1} = V^{n-2} \bigoplus_{j=n-2}^{n-1} W^j = \dots = V^0 \bigoplus_{j=0}^{n-1} W^j. \quad (2)$$

A *multiresolution curve* is then defined as a planar parametric curve $c(t) = (x^n)^T(\varphi^n)$, element of V^n , where x^n is a column of control points $x_0^n, \dots, x_{D_{2^n}}^n \in \mathbb{R}^2$. Due to property (2) the same curve can be expressed in terms of the basis functions of the different decompositions of V^n , each of it corresponding to a certain resolution of the curve. The multiresolution curve at any *level of resolution* $L \in [0, n]$, i.e. element of $V^L \bigoplus_{j=L}^{n-1} W^j$ is then given by some coarse control points x^L that form approximations of the initial control polygon and by the detail coefficients d^L, \dots, d^{n-1} as follows:

$$\mathbf{c}(t) = (\mathbf{x}^L)^T(\varphi^L) + (\mathbf{d}^L)^T(\psi^L) + \dots + (\mathbf{d}^{n-1})^T(\psi^{n-1}), \quad L = 0, \dots, n.$$

The *filter bank* algorithm [65, 36] is used to compute the coefficients of all levels of resolutions from the initial coefficients x^n and vice versa.

Area Preserving Deformation of a MR Curve

An advantage of a MR representation of the curve is that LOD editing consists of simply applying deformations on the coarse control points at some level L , the overall shape of the curve is therefore modified and the fine details are preserved, see Fig. 6 (a,b).

However the enclosed area of a (closed) modified curve is generally not preserved. In [50] it has been shown that the constant area constraint can be

integrated completely into the MR editing process. To this end a MR formula of the area constraint has been developed that allows one to compute the area of a curve in terms of the coefficients at any resolution level L .

The area (see Sect. 5.1) of a multiresolution curve can now be evaluated at any level of resolution L in terms of the bilinear equation

$$2\mathcal{A} = (\mathbf{X}^L) \begin{bmatrix} M^L \end{bmatrix} (\mathbf{Y}^L)^T, \quad \forall L \in \{0, \dots, n\},$$

where X^L and Y^L are the line vectors of the x- and y-coordinates resp. of all coarse and wavelet coefficients of the MR representation of the curve, i.e.

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

and

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

Note that φ^L and ψ^k are vector notations. Therefore the elements of the previous area-matrix are in fact block matrices whose elements are of type $I(\varphi_i, \psi_j) = \oint \varphi_i(t)\psi_j(t) - \varphi'_i(t)\psi_j(t)dt$ and whose sizes vary in function of the resolution level L . It has been shown in [50] that the area matrices M^L can be computed efficiently by recursively applying the refinement equations.

The *area preserving editing process* now works as follows: Let \mathcal{A}_{ref} be the reference area to be preserved. After choosing the *decomposition* level L , the user modifies one or more coarse control points (Fig. 6(b)), defining the desired *deformation*. Let $(X_0, Y_0)^T$ denote the coefficient vectors of the deformed MR curve at level L . The algorithm then computes new positions, denoted by $(X, Y)^T$, of the coarse control points (and possibly the detail coefficients) such that they are as close as possible to the user defined deformation while *preserving the area* \mathcal{A}_{ref} , see Fig. 6(c). The last step remains to solve the following min-max problem:

$$\max_{\lambda} \min_{X, Y} \quad (|X - X_0|^2 + |Y - Y_0|^2) + \lambda(XMY^T - 2\mathcal{A}_{ref}).$$

If only local area preserving deformations are desired, the degrees of freedom in $(X, Y)^T$ can be reduced to a user-defined subset of control points. Fig. 7 shows an example, where the upper left coarse control point has been kept fixed during deformation and area preservation.

Length preserving deformation of a MR curve

Deformation of curves with constant length is needed typically if one wants to create wiggles or folding of a curve. Sauvage et al. [82] developed a multiresolution approach of length preserving curve deformation for the particular case



Fig. 7. Local area preserving multiresolution deformation. $n = 7$, $L = 2$. original curve (black, big curve), deformed curve at level 2 (grey, small curve), area preserving deformed curve at level 2 (mid size curve).

of piecewise linear curves using the Lazy wavelets [87]. Let $c(t)$ be a polyline of control points c_i^n . Coarse coefficients and wavelet coefficients are then computed by

$$\begin{cases} x_i^j = x_{2i}^{j+1}, \\ d_i^j = x_{2i+1}^{j+1} - \frac{1}{2}(x_{2i}^{j+1} + x_{2i+2}^{j+1}) . \end{cases}$$

In the case of polylines the length is given by $L = \sum_{i=0}^{N-2} \|c_{i+1}^n - c_i^n\|_2$. One can either keep the total length constant or preserve the length of each segment. We choose the second way because of two main reasons:

- It ensures a balance between the lengths of the segments: the control points do not gather in a small part of the curve.
- It allows the length constraints to be expressed in such a way that computationally inefficient square root evaluations can be avoided.

The length constraint being a non-linear functional has no multiresolution representation as does the area constraint. However in [82] it is shown that length preserving MR curve editing offers a direct control of wrinkle generation. The level of resolution L where the length adaptation is performed has two advantages. First, wrinkles can be generated locally on a user defined extent, and magnitude and frequency of the wrinkles can be controlled.

The algorithm works in two steps. Once the user has defined the deformation by modifying some coarse control points at an arbitrary scale, he fixes the level of resolution L where he wants the length preserving being done. In other words, with L he chooses the extent and frequency of wrinkle creation. Following some geometric rules, the detail coefficients of the deformed curve belonging to level $L + 1$ are then modified in order to make the control polygon at level $L + 1$ have the same length as the level $L + 1$ control polygon of the initial curve. The second step of the algorithm consists then of length preserving by smoothing via an optimisation method and precisely satisfies the length constraint.

5.3 Variational MR Curves

The variational modelling paradigm is used in order to find the “best” curve or surface amongst all solutions that meet the constraints. The constraints may

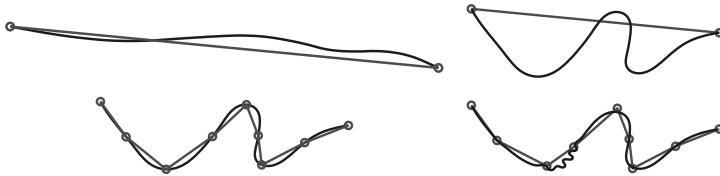


Fig. 8. Length preserving MR editing: Two successive deformations at different levels of decomposition are shown. The initial curve (top left) is edited at the coarsest level. (top right) Its length is adapted at the scale $L = 1$ resulting in large wrinkles. (bottom left) and (bottom right) Two neighbouring control points are moved closer at the scale 3 and length preserving at scale $L = 6$ creates small wrinkles.

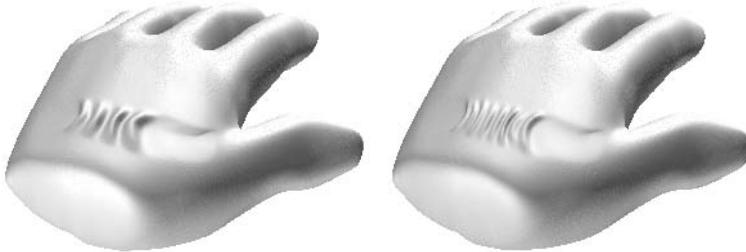


Fig. 9. Wrinkles on the back of the hand by length preserving MR deformation. The scheme has been applied on several lines of a triangular mesh modelling a hand. It creates wrinkles at the back of the hand automatically by pinching the skin. The skin is also stretched around the wrinkles. The model is purely geometric, no time consuming physical simulation is used.

result from the particular modelling technique used, for example sample point approximation, or direct curve manipulation (see Sect. 2.1). In the context of smooth curve and surface design the notion of “best” is formulated by minimising some energy functional, see Sect. 4.1.

Gortler and Cohen [44] show how the variational constraint, which generalises least squares, can be solved through a MR formulation of a planar curve. A wavelet based MR curve satisfying some linear constraints and minimising a linearised bending energy functional may be found by solving the following linear system [97]

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

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. Gortler and Cohen show then how wavelets accelerate the iterative conjugent-gradient-solving of the variational problem.

5.4 Multiresolution Subdivision Methods with Constraints

Subdivision has become a popular tool in computer graphics. Much literature derives and analyses new subdivision algorithms for curves, surfaces and solids. For an overview see the SIGGRAPH 2000 course notes [103] and the textbook [94]. Subdivision curves and surfaces are intrinsically hierarchical. Different levels of subdivision of a coarse mesh provide different levels of resolution. Constrained modelling techniques can then interact with different subdivision levels in order to obtain particular local design effects.

MacCracken and Joy [64] developed an extension of Catmull-Clark subdivision surfaces to the volumetric setting, mainly for the purpose of freeform deformation in 3D space. Qin et al. [75] introduced dynamic Catmull-Clark subdivision surfaces. McDonnell and Qin [67] simulate volumetric subdivision objects using a mass-spring model. A generalisation of McDonnell et al. [68] includes haptic interaction. Capell et al. [15] use the subdivision hierarchies to construct a hierarchical basis to represent displacements of a solid model for dynamic deformations. Additionally, some linear constraints, such as point displacements can be added at any level of subdivision.

Variational subdivision is another modelling technique, where constraints are combined with classical subdivision. Instead of applying explicit rules for the new vertices, Kobbelt's [56] variational subdivision scheme computes the new vertices such that a fairness functional is minimised. At each step a linear system has to be solved. The resulting curves have minimal total curvature. Furthermore, in [57] it is shown how wavelets can be constructed by using the Lifting Scheme [87] which are appropriate for variational subdivision curves. Weimer and Warren [93, 95, 96] developed variational subdivision schemes that satisfy partial differential equations, for instance, fluid or thin-plate equations.

Multiresolution subdivision surfaces extend subdivision surfaces by introducing details at each level. Each finer mesh is obtained from a coarse mesh by using a set of fixed refinement rules, e.g. Loop subdivision or Catmull-Clark subdivision. Each time a finer mesh is computed, it is obtained by adding detail offsets to the subdivided coarse mesh. Details are represented in local coordinate frames, which are computed from the coarser level. Various forms of multiresolution subdivision surfaces can be found in [63, 102, 60]. Several constrained modelling techniques have been developed by Zorin, Biermann and co-workers for multiresolution surface. A cut-and-paste editing technique for multiresolution surfaces has been proposed in [6]. In [5] Biermann et al. describe a method for creating sharp features and trim regions on multiresolution subdivision surfaces along a set of user-defined curves. A method for approximating results of boolean operations (union, intersection, difference) applied to free-form solids bounded by multiresolution subdivision surfaces can be found in [4].

6 Conclusion

In this paper geometric modelling techniques have been surveyed that all make use of constraints of different nature in order to provide high-level user friendly manipulation tools of geometric objects. Basic research, developing new curve and surface representation, is going on and new deformation and editing tools have to be invented. For example, it is still a challenge to develop modelling tools for subdivision surfaces equivalent to those existing for NURBS surfaces.

Acknowledgements

This work was partially supported by the European Community Fifth framework program, with the Research Training Network MINGLE (Multiresolution IN Geometric modELLing, HPRN-CT-1999-00117) and in part by the Fund for Promotion of Research at the Technion, IIT, Haifa, Israel.

References

1. Barghiel C., Bartels R., Forsey D.: Pasting Spline Surfaces. In Daehlen M., Lyche T., Schumaker L.L. (eds.), *Mathematical Methods for Curves and Surfaces*, Ulvik, Vanderbilt University Press, 31-40 (1999).
2. Bartels, R., Beatty, J.: A technique for the direct manipulation of spline curves. In *Graphics Interface Conference Proceedings '89*, 33-39 (1989).
3. Bidarra R., Bronsvoort W.F.: Semantic feature modelling. *Computer Aided Design* **32**, 201–225 (2000).
4. Biermann H., Kristjansson D., Zorin D.: Approximate Boolean Operations on Free-form Solids. In Proc. ACM SIGGRAPH. Los Angeles, 185-194 (2001).
5. Biermann H., Martin I.M., Zorin D., Bernardini F.: Sharp Features on Subdivision Surfaces. In Pacific Graphics Conference Proceedings. Tokyo, Japan (2001).
6. Biermann H., Martin I.M., Bernardini F., Zorin D.: Cut-and-Paste Editing of Multiresolution surfaces. In Proc. ACM SIGGRAPH, 187–196 (2002).
7. Bloomenthal J., Shoemake K.: Convolution surfaces. In Proc. ACM SIGGRAPH, Las Vegas, Nevada, (1991).
8. Bloomenthal J. (editor): *Introduction to Implicit Surfaces*. Morgan Kaufmann (1997).
9. Bonneau G.P., Hagen H.: Variational design of rational Bézier curves and surfaces. In Laurent P.J., Le Méhauté A., Schumaker L.L. (eds.) *Curves and Surfaces II*, 51–58 (1994).
10. Bonneau G.-P., Hahmann S., Nielson G.: Blac-wavelets: a multiresolution analysis with non-nested spaces. In *Visualization Conference Proceedings*, 43–48 (1996).
11. Bonneau G.-P.: Multiresolution analysis on Irregular Surface Meshes. *IEEE Transactions on Visualization and Computer Graphics*, **4**, 365-378 (1998).

12. Borel P., Rappoport A.: Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics*, **13**(2), 137–155 (1994).
13. Cani-Gascuel M.-P., Desbrun M.: Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics* **3**(1), 39–50 (1997).
14. Cani-Gascuel M.-P.: Layered deformable models with implicit surfaces. In *Graphics Interface Conference Proceedings*, Vancouver, Canada (1998).
15. Capell S., Green S., Curless B., Duchamp T., Popovic Z.: A Multiresolution Framework for Dynamic Deformations. In *ACM SIGGRAPH Symposium on Computer Animation*, 41–48 (2002).
16. Cavendish J.C., Marin S.P.: A procedural feature-based approach for designing functional surfaces. In Hagen H. (ed.) *Topics in Surface Modeling*. SIAM, Philadelphia (1992).
17. Celniker G., Gossard, D.: Deformable curve and surface finite-elements for free-form shape design. In Proc. ACM SIGGRAPH, 257–266 (1991).
18. Celniker G., Welch, W.: Linear constraints for deformable b-spline surfaces. In *Symposium on Interactive 3D Graphics*, 165–170 (1992).
19. Chen Y., Zhu Q., Kaufman A., Muraki S.: Physically-based animation of volumetric objects. In *Proceedings of IEEE Computer Animation*, 154–160 (1998).
20. Chui C., Quak E.: Wavelets on a bounded interval. In Braess D., Schumaker L.L. (eds.) *Numerical Methods of Approximation Theory*. Birkhäuser Verlag, Basel, 1–24 (1992).
21. Cignoni P., Montani C., Puppo E., Scopigno R.: Multiresolution Representation and Visualization of Volume Data. *IEEE Trans. on Visualization and Comp. Graph.*, **3** (4), 352–369 (1997).
22. Cohen E., Riesenfeld R.F., Elber G.: *Geometric Modeling with Splines: An Introduction*. AK Peters (2001).
23. Conrad B., Mann S.: Better Pasting via Quasi-Interpolation. In Laurent P.J., Sablonniere P., Schumaker L.L. (eds.), *Curve and Surface Design*, Saint-Malo, Vanderbilt University Press, 27–36 (1999).
24. Coquillart S.: Extended free-form deformation: A sculpturing tool for 3D geometric modeling. In Proc. ACM SIGGRAPH, 187–196 (1990).
25. Dankwort C.W., Podehl G.: FIORES - A European Project for a New Workflow in Aesthetic Design. *VDIBerichte Nr. 1398*, 177–192 (1998).
26. De Martino T., Falcidieno B., Giannini F., Hassinger S., Ovtcharova J.: Integration of Design-by-features and Feature Recognition approaches. *Computer Aided Design*, **26**, 646–653 (1994).
27. Desbrun M., Gascuel M.-P.: Animating soft substances with implicit surfaces. In Proc. ACM SIGGRAPH, 287–290 (1995).
28. Desbrun M., Cani-Gascuel M.-P.: Active implicit surface for computer animation. In *Graphics Interface Conference Proceedings*, Vancouver, Canada (1998).
29. Dohmen M., de Kraker K.J., Bronsvoort W.F.: Feature validation in a multiple-view modeling system. In *Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, Irvine (1996).
30. Eck M., DeRose T., Duchamp T., Hoppe H., Lounsbery T., Stuetzle W.: Multiresolution analysis of arbitrary meshes. In Proc. ACM SIGGRAPH, 173–182 (1995).

31. Elber G.: Free Form Surface Analysis Using A Hybrid of Symbolic and Numerical Computation. PhD thesis, Department of Computer Science, The University of Utah (1992).
32. Elber G., Gotsman G.: Multiresolution Control for Nonuniform Bspline Curve Editing. In The third Pacific Graphics Conference on Computer Graphics and Applications, Seoul, Korea, 267-278 (1995).
33. Elber G.: Multiresolution Curve Editing with Linear Constraints. The Journal of Computing & Information Science in Engineering, **1**(4), 347-355 (2001).
34. Farin G., Rein G., Sapidis N., Worsey A.J.: Fairing cubic B-spline curves. Computer Aided Geometric Design **4**, 91-103 (1987).
35. Farin G.: Curves and Surfaces for Computer Aided Geometric Design. Academic Press, New York, 4th edition (1996).
36. Finkelstein A., Salesin D.H.: Multiresolution curves, In Proc. ACM SIGGRAPH, 261-268 (1994).
37. Fontana M., Giannini F., Meirana M.: A Free Form Feature Taxonomy. In Proceedings of EUROGRAPHICS, Computer Graphics Forum, **18**(3), 646-653 (1994).
38. Forsey D., Bartels R.: Hierarchical B-spline refinement. In Proc. ACM SIGGRAPH, 205-212 (1988).
39. Fowler B.: Geometric manipulation of tensor product surfaces. In Symposium on Interactive 3D Graphics, 101-108 (1992).
40. Fowler B., Bartels R.: Constraint-based curve manipulation. IEEE Computer Graphics and Applications, **13**(5), 43-49 (1993).
41. Gain J.E., Dodgson N.A.: Preventing self-intersection under free-form deformation. IEEE Transactions on Visualization and Computer Graphics, **7**(4), 289-298 (2001).
42. Goldman R.N., Lyche T. (Eds.): Knot Insertion and Deletion Algorithms for B-Spline Curves and Surfaces. SIAM, Philadelphia, ISBN 0-89871-306-4 (1993).
43. Golub G.H., Van Loan C.F.: Matrix Computation. The John Hopkins University Press, Baltimore and London, Third Edition (1996).
44. Gortler S., Cohen M.: Hierarchical and variational geometric modeling with wavelets. In 1995 Symposium on 3D Interactive Graphics, 35-41 (1995).
45. Greiner G.: Variational design and fairing of spline surfaces. In Eurographics Conference Proceedings, 143-154 (1994).
46. Greiner G., Loos J.: Data dependent thin plate energy and its use in interactive surface modeling. In Eurographics Conference Proceedings, 176-185 (1996).
47. Hagen H., Schulze G.: Automatic smoothing with geometric surface patches. Computer Aided Geometric Design, **4**, 231-236 (1994).
48. Hagen H., Santarelli P.: Variational design of smooth B-spline surfaces. In Hagen H. (ed.) Topics in geometric modeling. SIAM Philadelphia, 85-94 (1992).
49. Hahmann S.: Shape improvement of surfaces. Computing Suppl., **13**, 135-152 (1998).
50. Hahmann S., Bonneau G.-P., Sauvage B.: Area preserving deformation of multiresolution curves. Reserach Report, IMAG RR-1062-I (2002).
51. Hirota G., Maheshwari R., Lin M.C.: Fast volume-preserving free-form deformation using multi-level optimization. In Proceedings of Solid Modeling, 234-245 (1999).
52. Hoppe H.: Progressive meshes. In Proc. ACM SIGGRAPH, 99-108 (1996).
53. Hoschek J., Lasser D.: Fundamentals of Computer Aided Geometric Design. A.K. Peters (1993).

54. Hsu W.M., Hughes J.F., Kaufman H.: Direct manipulation of free-form deformations. In Proc. ACM SIGGRAPH, 177–184 (1992).
55. Kazinnik R., Elber G.: Orthogonal Decomposition of Non-Uniform Bspline Spaces using Wavelets. Computer Graphics Forum, **16**(3), 27-38 (1997).
56. Kobbelt L.: A variational approach to subdivision. Computer Aided Geometric Design, **13**, 743–761 (1996).
57. Kobbelt L., Schröder P.: A Multiresolution Framework for Variational Subdivision. ACM Trans. on Graph., **17**(4), 209-237 (1998).
58. Kobbelt L., Campagna S., Vorsatz J., Seidel HP.: Interactive multiresolution modeling on arbitrary meshes. In Proc. ACM SIGGRAPH, 105–114 (1998).
59. Lassiter J.: Principles of traditional animation applied to 3D computer animation. In Proc. ACM SIGGRAPH, 45-44 (1987).
60. Lee A., Moreton H., Hoppe H.: Displaced subdivision surfaces. In Proc. ACM SIGGRAPH, 85-94 (2000).
61. Lyche T., Mørken K.: Spline wavelets of minimal support. In Braess D., Schumaker L.L. (eds.) Numerical Methods of Approximation Theory. Birkhäuser Verlag, Basel, 177–194 (1992).
62. Lorensen W., Cline H.: Marching cubes: a high resolution 3D surface construction algorithm. In Proc. ACM SIGGRAPH, Anaheim California, 163–169 (1987).
63. Lounsbery M., De Rose T., Warren J.: Multiresolution analysis for surfaces of arbitrary topological type. ACM Transaction on Graphics, **16**(1), 34–73 (1997).
64. MacCracken R., Joy K.I.: Free-form deformations with lattices of arbitrary topology. In Proc. ACM SIGGRAPH, 181–188 (1996).
65. Mallat S.: A theory for multiresolution signal decomposition: the wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, **11**, 674–693 (1989).
66. Mann S., Yeung T.: Cylindrical Surface Pasting. In Brunnett G., Bieri H.P., Farin G. (eds.) Geometric Modeling, Springer-Wien, 233-248 (2001).
67. McDonnell K.T., Qin H.: Dynamic sculpting and animation of free-form subdivision solids. In Proceedings of IEEE Computer Animation 2000, 126-133 (2000).
68. McDonnell K.T., Qin H., Włodarczyk R.A.: Virtual clay: A real-time sculpting system with haptic toolkits. In Proceedings of the 2001 ACM Symposium on Interactive 3D Graphics, 179-190 (2001).
69. Mehlum E.: Non-linear spline. In Barnhill R.E., Riesenfeld R.F. (eds.) Computer Aided Geometric Design. Academic Press, 173–208 (1974).
70. Moreton H.P., Séquin C.H.: Functional optimisation for fair surface design. Computer Graphics, **26**(2), 167–176 (1992).
71. Pasko A., Adzhiev V., Sourin A., Savchenko V.: Function representation in geometric modeling: Concepts, implementation and applications. The Visual Computer, **11**(8), 429–446 (1995).
72. Pentland A., Williams J.: Good vibrations: Modal dynamics for graphics and animation. In Proc. ACM SIGGRAPH, 215-222 (1989).
73. Pernot J.P., Guillet S., Leon J.C., Giannini F., Falcidieno B., Catalano E.: A Shape Deformation Tool to Model Character Lines in the Early Design Phases. In Conference Proceedings of Shape Modeling International, Banff, Canada (2002).
74. Pottmann H., Leopoldseder S., Hofer M.: Approximation with active B-Spline curves and surfaces. In Pacific Graphics Conference Proceedings, 8-25 (2002).

75. Qin H., Mandal c., Vemuri B.C.: Dynamic Catmull-Clark subdivision surfaces. *IEEE Transactions on Visualization and Computer Graphics*, **4**(3), 215-229 (1998).
76. Rappoport A., Sheffer A., Bercovier M.: Volume-preserving free-form solids. In *Solid Modeling Conference Proceedings*, 361-372 (1995).
77. Raviv A., Elber G.: Three Dimensional Freeform Sculpting Via Zero Sets of Scalar Trivariate Functions. *CAD*, **32** (8/9), 513-526 (2000).
78. Reinsch C.H.: Smoothing by spline functions II. *Num. Math.*, **16**, 451-454 (1967).
79. Requicha A.A., Voelcker U.B.: Solid modeling: a historical summary and contemporary assessment. *IEEE Computer Graphics and Applications*, **2**, 9-23 (1982).
80. Requicha A.A., Rossignac J.R.: Solid modeling and beyond. *IEEE Computer Graphics and Applications*, **12**, 31-44 (1992).
81. Rossignac J.R.: Issues on feature-based editing and interrogation of solid models. *Comp. Graph.* **14**(2), 149-172 (1990).
82. Sauvage B., Hahmann S., Bonneau G.-P.: Length preserving multiresolution editing of curves. preprint (2003).
83. Schröder P., Sweldens W.: Spherical Wavelets: efficiently representing functions on the sphere. In *Proc. ACM SIGGRAPH*, 161-172 (1995).
84. Sederberg T.W., Parry S.R.: Free-form deformation of solid geometric models. In *Proc. ACM SIGGRAPH*, 151-160 (1986).
85. Shi-Min H., Hui Z., Chiew-LanT., Jia-Guang S.: Direct manipulation of FFD: Efficient explicit solutions and decomposable multiple point constraints. *The Visual Computer*, **17**(6), 370-379 (2001).
86. Stollnitz E., DeRose T., Salesin D.: *Wavelets for Computer Graphics: Theory and Applications*. Morgan-Kaufmann (1996).
87. Sweldens W.: The lifting scheme: a construction of second generation wavelets. *SIAM J. Math. Anal.*, **29**(2), 511-546 (1997).
88. Terzopoulos D., Platt J., Barr A., Fleischer K.: Elastically deformable models. In *Proc. ACM SIGGRAPH* (1987).
89. Terzopoulos D., Fleischer K.: Modeling inelastic deformation: viscoelasticity, plasticity, fracture. In *Proc. ACM SIGGRAPH*, 269-278 (1988).
90. Terzopoulos D., Qin H.: Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, **13**(2), 103-136 (1994).
91. van den Berg E., van der Meiden R., Bronsvoort W.F.: Specification of freeform features. *Proceedings Eighth Symposium on Solid Modeling and Applications* (2003).
92. Vosniakos G.C.: Investigation of feature-based product modelling for mechanical parts with free-form surfaces. *International Journal of Advanced Manufacturing Technology*, **15**, 188-199 (1999).
93. Warren J., Weimer H.: Variational subdivision for natural cubic splines. *Approximation Theory IX*, 345-352 (1998).
94. Warren J., Weimer H.: *Subdivision Methods for Geometric Design: a constructive approach*. Morgan Kaufmann Publisher (2001).
95. Weimer H., Warren J.: Subdivision schemes for thin plate splines. In *Eurographics Conference Proceedings*, 303-313 (1998).
96. Weimer H., Warren J.: Subdivision schemes for fluid flow. In *Proc. ACM SIGGRAPH*, 111-120 (1999).

97. Welch W., Witkin A.: Variational surface modeling. In Proc. ACM SIGGRAPH, 157-166 (1992).
98. Whitaker R., Breen D.: Level-set models for the deformation of solid objects. In Implicit Surfaces '98, Eurographics and ACM SIGGRAPH Workshop, Seattle, 19-36 (1998).
99. Witkin A., Welch W.: Fast animation and control for non-rigid structures. In Proc. ACM SIGGRAPH, 243-252 (1990).
100. Wyvill G., McPheeers C., Wyvill B.: Data structure for soft objects. The Visual Computer, **2**(4), 227-234 (1986).
101. Wyvill A., Galin E., Guy A.: Extending the CSG tree warping, blending and boolean operations in an implicit surface modeling system. In Implicit Surfaces, Eurographics and ACM SIGGRAPH Workshop, Seattle, 113-122 (1998).
102. Zorin D., Schröder P., Sweldens W.: Interactive Multiresolution Mesh Editing. In Proc. ACM SIGGRAPH, 259-268 (1997).
103. Zorin D., Schröder P.: Subdivision for Modeling and Animation. In ACM SIGGRAPH Course Notes (2000).

Multi-scale and Adaptive CS-RBFs for Shape Reconstruction from Clouds of Points

Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel

Max-Planck-Institut für Informatik, Saarbrücken, Germany

{ohtake|belyaev|hpseidel}@mpi-sb.mpg.de

Summary. We describe a multi-scale approach for interpolation and approximation of a point set surface by compactly supported radial basis functions. Given a set of points scattered over a surface, we first use down-sampling to construct a point set hierarchy. Then starting from the coarsest level, for each level of the hierarchy, we use compactly supported RBFs to approximate the set of points at the level as an offset of the RBF approximation computed at the previous level. A simple RBF centre reduction scheme combined with the multi-scale approach accelerates the latter and allows us to achieve high quality approximations using relatively small number of RBF centres.

We also develop an adaptive RBF fitting procedure for which the RBF centres are randomly chosen from the set of points of the level. The randomness is controlled by the density of points and geometric characteristic of the set. The support size of the RBF we use to approximate the point set at a vicinity of a point depends on the local density of the set at that point. Thus parts with complex geometry are approximated by dense RBFs with small supports.

Numerical experiments demonstrate high speed and good performance of the proposed methods in processing irregularly sampled and/or incomplete data.

1 Introduction

Among various techniques available for 3D scattered data interpolation and approximation Radial Basis Function (RBF) methods remain to be very attractive for a large variety of applications because of their ability to produce high quality shapes, process irregularly sampled and noisy data, repair incomplete data, and handle shapes of arbitrary topological complexity [19, 20, 8]. In computer graphics, implicit surface modelling with RBFs was first introduced in [16, 19]. The main limitation of early RBF-based techniques consisted of computational difficulties to handle large point data sets (see, for example, [11] and references therein). Recently it was shown that enhancing global RBFs by fast multipole techniques [5] and using RBFs with compact support

(CS-RBFs) in a hierarchical manner [6, 9] allows a user to process scattered data consisting of millions of points [14].

In this paper, we present a further development of our approach to 3D scattered data fitting with CS-RBFs [14]. In Sect. 2, we describe our multi-scale approach to 3D scattered data interpolation with CS-RBFs [14] and enhance it by a centre reduction procedure. In Sect. 3, we propose an adaptive CS-RBF approximation scheme.

This paper does not contribute to mathematical theory of RBF approximations. The interested reader is referred to [7][Chapter 7] and [4] for a comprehensive introduction to RBFs. Our approach is engineering oriented. The experimental results justify that the developed methods deliver fast and high quality reconstruction of complex shapes from real-world scattered data.

2 Multi-scale CS-RBF Interpolation and Approximation

Let us consider a set of points $\mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_N\}$ sampled from a surface and equipped with unit normals $\mathcal{N} = \{\mathbf{n}_1, \dots, \mathbf{n}_N\}$ that indicate the surface orientation. In practice, these normals are usually estimated from range data during the shape acquisition phase or by local least-square fitting. Our aim is to construct a function $y = f(\mathbf{x})$ such that its zero level-set $\{\mathbf{x} \in \mathbb{R}^3 : f(\mathbf{x}) = 0\}$ interpolates/approximates the set of points \mathcal{P} .

The implicit surface $\{\mathbf{x} \in \mathbb{R}^3 : f(\mathbf{x}) = 0\}$ separates the space into two parts: $f(\mathbf{x}) > 0$ and $f(\mathbf{x}) < 0$. Let us assume that the normals \mathcal{N} are pointing into the part of space where $f(\mathbf{x}) > 0$. Thus one can say that $f(\mathbf{x})$ has negative values outside the surface and positive values inside the surface.

Single-level Interpolation

We construct function $y = f(\mathbf{x})$ interpolating \mathcal{P} in the following form suggested in [14]

$$f(\mathbf{x}) = \sum_{\mathbf{p}_i \in \mathcal{P}} [g_i(\mathbf{x}) + \lambda_i] \phi_\sigma(\|\mathbf{x} - \mathbf{p}_i\|),$$

where $\phi_\sigma(r) = \phi(r/\sigma)$, $\phi(r) = (1 - r)_+^4 (4r + 1)$, is Wendland's compactly supported RBF [21], σ is its support size, and $g_i(\mathbf{x})$ and λ_i are unknown functions and coefficients to be determined.

For each $\mathbf{p}_i \in \mathcal{P}$ we construct $g_i(\mathbf{x})$ as a local approximation of \mathcal{P} in a small vicinity of \mathbf{p}_i . Thus the zero level-set of

$$\sum_{\mathbf{p}_i \in \mathcal{P}} g_i(\mathbf{x}) \phi_\sigma(\|\mathbf{x} - \mathbf{p}_i\|) \tag{1}$$

approximates \mathcal{P} . Now an interpolation of \mathcal{P} is achieved by the standard RBF interpolation procedure:

$$\sum_{\mathbf{p}_i \in \mathcal{P}} \lambda_i \phi_\sigma(\|\mathbf{x} - \mathbf{p}_i\|) = - \sum_{\mathbf{p}_i \in \mathcal{P}} g_i(\mathbf{x}) \phi_\sigma(\|\mathbf{x} - \mathbf{p}_i\|), \quad \mathbf{x} = \mathbf{p}_j \in \mathcal{P}, \quad (2)$$

which leads to a sparse system of linear equations with respect to λ_i . Since Wendland's compactly supported RBFs are strictly positive definite [21], $N \times N$ interpolation matrix $\{\phi_\sigma(\|\mathbf{p}_j - \mathbf{p}_i\|)\}$ is positive definite if \mathcal{P} consists of pairwise distinct points.

One can notice that (1) has the same zero level-set as a partition of unity approximation (PU)

$$\sum_{\mathbf{p}_i \in \mathcal{P}} g_i(\mathbf{x}) \frac{\phi_\sigma(\|\mathbf{x} - \mathbf{p}_i\|)}{\sum \phi_\sigma(\|\mathbf{x} - \mathbf{p}_i\|)}. \quad (3)$$

PU approximations are now very popular in computational mechanics [2, 3] and can deliver high quality approximations of scattered data [10, 22, 1, 13, 18].

Thus $\{\lambda_i\}$ are small and iterative solving of system of linear equations (2) can be done quickly if we start from initial guess $\lambda_i = 0$, $i = 1, \dots, N$. In our implementation we use the preconditioned biconjugate gradient method [15].

We find local approximations $\{g_i(\mathbf{x})\}$ as follows. For each point $\mathbf{p}_i \in \mathcal{P}$ let us determine a local orthogonal coordinate system (u, v, w) with the origin of coordinates at \mathbf{p}_i such that the plane (u, v) is orthogonal to \mathbf{n}_i and the positive direction of w coincides with the direction of \mathbf{n}_i . We approximate \mathcal{P} in a vicinity of \mathbf{p}_i by a quadric

$$w = h(u, v) \equiv Au^2 + 2Buv + Cv^2, \quad (4)$$

where the coefficients A , B , and C are determined via the following least-squares minimisation

$$\sum_{(u_j, v_j, w_j) = \mathbf{p}_j \in \mathcal{P}} \phi_\sigma(\|\mathbf{p}_j - \mathbf{p}_i\|) (w_j - h(u_j, v_j))^2 \rightarrow \min. \quad (5)$$

Finally we set

$$g_i(\mathbf{x}) = w - h(u, v). \quad (6)$$

Thus the zero level-set of $g_i(\mathbf{x})$ coincides with the graph of $w = h(u, v)$.

Parameter σ , the support size of $\phi_\sigma(\cdot)$, is estimated from the density of \mathcal{P} . We start an octree-based subdivision of a bounding box of \mathcal{P} and stop the subdivision if each leaf cell contains no more than eight points of \mathcal{P} . Then we compute the average diagonal of the leaf cells. Finally we set σ equal to three fourth of that average diagonal.

Multi-level Interpolation

The single-level CS-RBF interpolation procedure described above is quite fast since it requires solving a sparse system of linear equations. However using

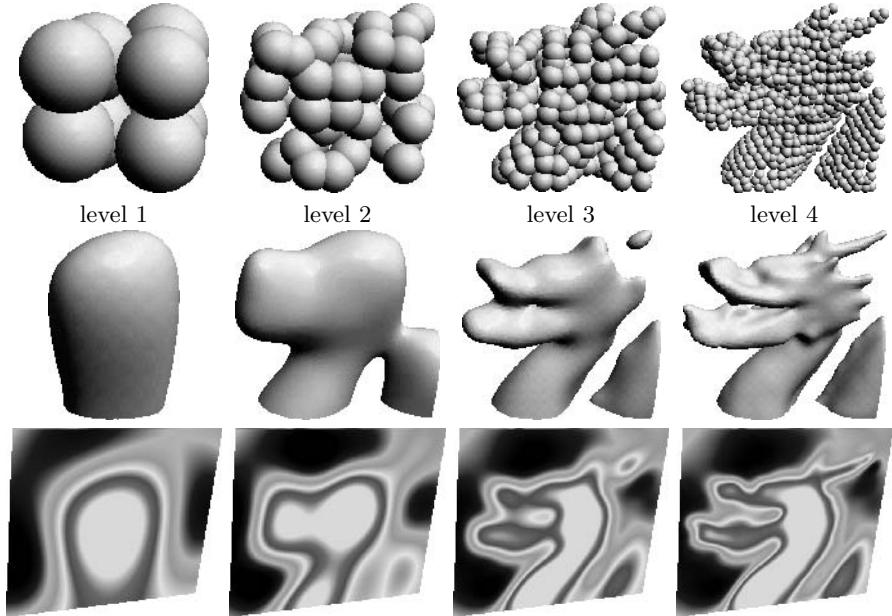


Fig. 1. [Reproduced in colour in Plate 12.] Multi-scale interpolation of the Stanford dragon model. Top row: four first levels of the multi-scale hierarchy of points; the radii of the spheres at each level of the hierarchy are proportional to the support size of the RBFs used for the interpolation at that level. Middle row: zero level-sets of the interpolating functions. Bottom row: cross-sections of the interpolating functions.

compactly supported basis functions implies several essential limitations. It can produce poor quality shapes when interpolating point data with varying density, has no ability of repairing incomplete data, and defines the interpolating implicit surface in a small vicinity of the interpolated points [14]. In order to eliminate these limitations we build a multi-scale hierarchy of point sets $\{\mathcal{P}^1, \mathcal{P}^2, \dots, \mathcal{P}^M = \mathcal{P}\}$ and interpolate a point set \mathcal{P}^{m+1} of the hierarchy by offsetting the interpolation function used in the previous level to interpolate \mathcal{P}^m . Fig. 1 demonstrates the main steps of our multi-level interpolation procedure.

A seminal idea to use RBF fitting in a hierarchical way was proposed in [6] for bivariate scattered data interpolation. Very recently it was combined with an adaptive domain decomposition in [9]. Our multi-level RBF fitting procedure described in this section can be considered as a variant of the multilevel approximation scheme [9] adapted for interpolating 3D scattered data.

To construct a multi-scale hierarchy of point sets $\{\mathcal{P}^1, \mathcal{P}^2, \dots, \mathcal{P}^M\}$ with $\mathcal{P} = \mathcal{P}^M$ we first fit \mathcal{P} into a parallelepiped and then subdivide it and its parts recursively into eight equal octants. Point set \mathcal{P} is clustered with respect to the

cells of the built octree-based subdivision of the parallelepiped. For each cell we consider the points of \mathcal{P} contained in the cell and compute their centroid. A unit normal assigned to the centroid is obtained by averaging the normals assigned to the points of \mathcal{P} inside the cell and normalising the result. Set \mathcal{P}^1 corresponds to the subdivision of the bounding parallelepiped into eight equal octants.

Now our multi-level interpolation procedure proceeds in the coarse-to-fine way. First we define a base function

$$f^0(\mathbf{x}) = -1$$

and then recursively define the set of interpolating functions

$$f^k(\mathbf{x}) = f^{k-1}(\mathbf{x}) + o^k(\mathbf{x}) \quad (k = 1, 2, \dots, M),$$

where $f^k(\mathbf{x}) = 0$ interpolates \mathcal{P}^k . An offsetting function o^k

$$o^k(\mathbf{x}) = \sum_{\mathbf{p}_i^k \in \mathcal{P}^k} [g_i^k(\mathbf{x}) + \lambda_i^k] \phi_{\sigma^k}(\|\mathbf{x} - \mathbf{p}_i^k\|).$$

has the form used in the previous section for the single-level interpolation. In particular, local approximations $g_i^k(\mathbf{x})$ are determined similar to (4),(5), (6) via least square fitting applied to \mathcal{P}^k . The shifting coefficients λ_i^k are found by solving the following system of linear equations

$$f^{k-1}(\mathbf{p}_i^k) + o^k(\mathbf{p}_i^k) = 0. \quad (7)$$

The support size σ^k is defined by

$$\sigma^{k+1} = \sigma^k/2, \quad \sigma_1 = cL,$$

where L is the length of a diagonal of the bounding parallelogram and the parameter c is chosen such that an octant of the bounding box is always covered by a ball of radius σ_1 centred somewhere in the octant. In practice we use $c = 0.75$.

Finally, the number of subdivision levels M is determined by σ_1 and σ_0 where σ_0 is the support size for the single-level interpolation. According to our experience, $M = \lceil -\log_2(\sigma_0/(2\sigma_1)) \rceil$ produces good results.

According to numerical experiments [14], our multi-scale interpolation scheme demonstrates a good performance in processing irregularly sampled and/or incomplete data (Fig. 2 demonstrates reconstruction of missed data) and works several times faster than the Fast RBF method [5] which employs globally supported RBFs.

Multi-scale RBF Centre Reduction

If an approximation of scattered data is required instead of an interpolation there is no need to use all the points as the RBF centres. A greedy RBF centre

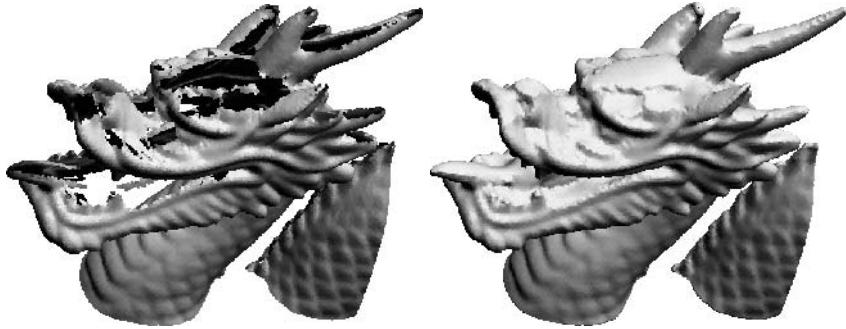


Fig. 2. Reconstruction of incomplete data obtained from range scans of the Stanford dragon model.

reduction method was proposed in [5]. Below we describe a simple RBF centre reduction strategy suitable for our multi-scale approach.

Let \mathcal{P}^k be approximated by the zero level-set of $y = f^k(\mathbf{x})$. We approximate \mathcal{P}^{k+1} by taking into account only those points of \mathcal{P}^{k+1} for which the Taubin distance [17] from $\{\mathbf{x} \in \mathbb{R}^3 : f^k(\mathbf{x}) = 0\}$ is greater than a user-specified threshold ε .

Fig. 3 shows last five levels of the multiscale CS-RBF fitting done without (the top row) and with (the bottom row) the centre reduction. Notice how the proposed centre reduction scheme reduces computational time and memory usage. Fig. 4 demonstrates presents another example: the St. Matthew model consisting of more than 3.3 million points is accurately approximated by using half a million RBF centres.

3 Adaptive CS-RBF Approximation

Adaptive RBF Support Size Selection

Given an RBF centre \mathbf{c} , let us define a local approximation error at \mathbf{c} as

$$\varepsilon(\sigma) = \sqrt{\frac{1}{\sum_j \phi_\sigma(\|\mathbf{p}_j - \mathbf{c}\|)} \sum_j \phi_\sigma(\|\mathbf{p}_j - \mathbf{c}\|) \left(\frac{g(\mathbf{p}_j)}{\|\nabla g(\mathbf{p}_j)\|} \right)^2}.$$

where $g(\mathbf{x})$ is a local quadratic approximation at centre \mathbf{c} . We obtain $g(\mathbf{x})$ from (5) and (6) with

$$w = h(u, v) \equiv Au^2 + 2Buv + Cv^2 + Du + Ev + F$$

used instead of (4) since we do not need to interpolate \mathbf{c} . We want to keep σ as large as possible while maintaining a user-defined accuracy ε_0 .

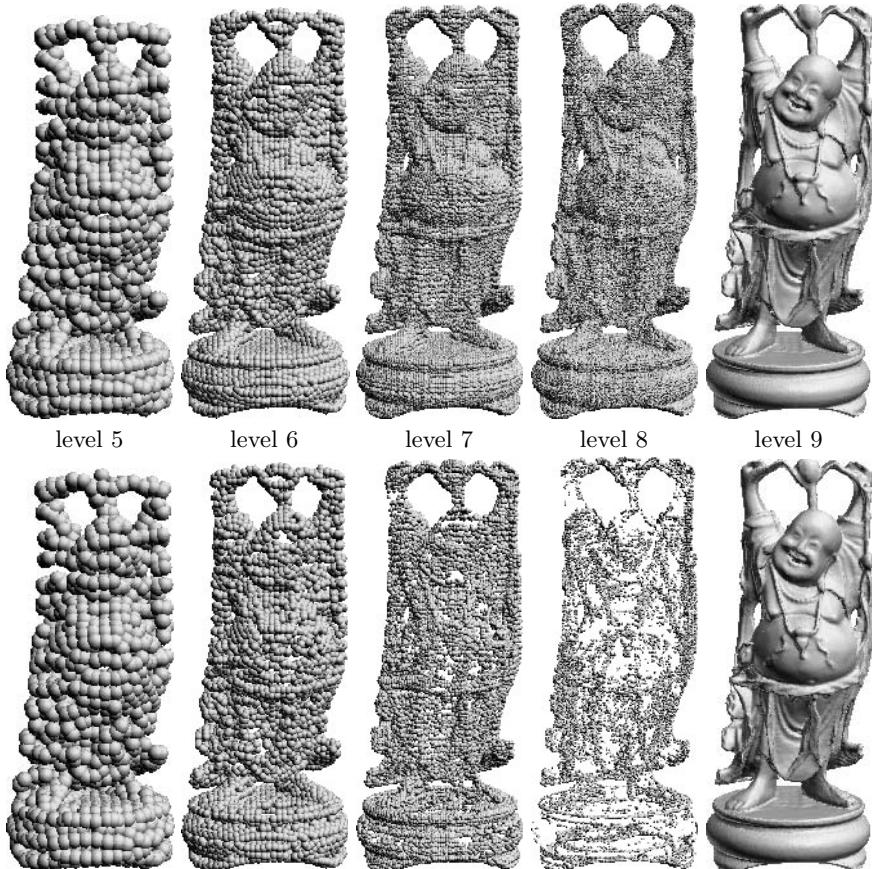


Fig. 3. Multi-level reconstruction of the Buddha model consisting of 544K points (nine levels of multi-scale hierarchy are used, computations are performed on 1.6 GHz Mobile Pentium 4). Top row: no RBF centre reduction is applied; reconstruction time is 24.5 min; peak RAM is 332 Mb; 901K RBFs are used. Bottom row: RBF centre reduction with $\varepsilon = 3 \times 10^{-4}$ accuracy is applied; reconstruction time is 5 min; peak RAM is 128 Mb; 75K RBFs are used.

It is natural to assume that $\varepsilon(\sigma)$ is monotonically decreasing as $\sigma \rightarrow 0$ and look for σ_{opt} such that

$$\varepsilon(\sigma_{\text{opt}}) = \varepsilon_0. \quad (8)$$

We use ten iterations of the bisection method to find σ_{opt} in interval $[L, R]$. Initially we set $L = 0$ and $R = 100\varepsilon_0$. If $\varepsilon(R) < \varepsilon_0$ the search interval is shifted: $L_{\text{new}} = R$ and $R_{\text{new}} = 2R$.

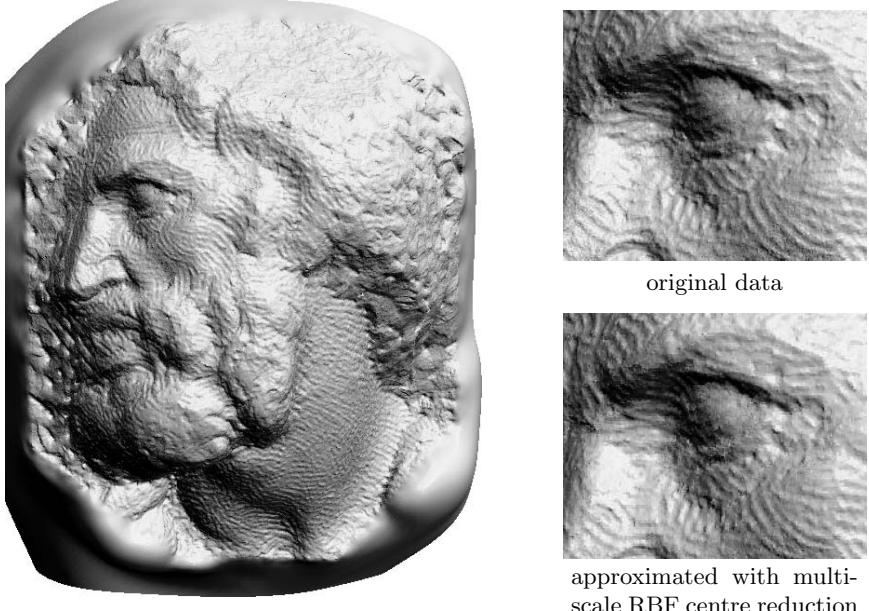


Fig. 4. Multi-level reconstruction of the Michelangelo’s St. Matthew model consisting of 3,376K points. Top: RBF interpolation. Bottom: approximation with multi-scale RBF centre reduction (507K RBFs, 24 min on 1.6 GHz Mobile Pentium 4).

RBF Centre Selection

As we already noticed before, our RBF-based approximation

$$f(\mathbf{x}) \approx \tilde{f}(\mathbf{x}) = \sum_{\mathbf{c}_i \in \mathcal{C}} [g_i(\mathbf{x}) + \lambda_i] \phi_{\sigma_i}(\|\mathbf{x} - \mathbf{c}_i\|), \quad (9)$$

where $\mathcal{C} \subset \mathcal{P}$ is a set of RBF centres, is close to partition of unity approximation (3). It is clear that a “good” cover is important for a high quality partition of unity approximation. In our case, the cover consists of domains $\text{supp } \phi_{\sigma_i}(\mathbf{c}_i)$. We choose RBF centres $\{\mathbf{c}_i\}$ such that $\{\text{supp } \phi_{\sigma_i}(\mathbf{c}_i)\}$ covers all the points of \mathcal{P} . Furthermore, we want to generate a minimal cover with an amount of overlap greater than a certain threshold.

We measure the amount of overlap at $\mathbf{p}_i \in \mathcal{P}$ by

$$v_i = \sum_j w_{\sigma_j}(\|\mathbf{p}_i - \mathbf{c}_j\|)$$

and control overlapping by a user-specified parameter T . We determine RBF centres $\{\mathbf{c}_j\}$ and their corresponding support sizes $\{\sigma_i\}$ according to the following procedure.

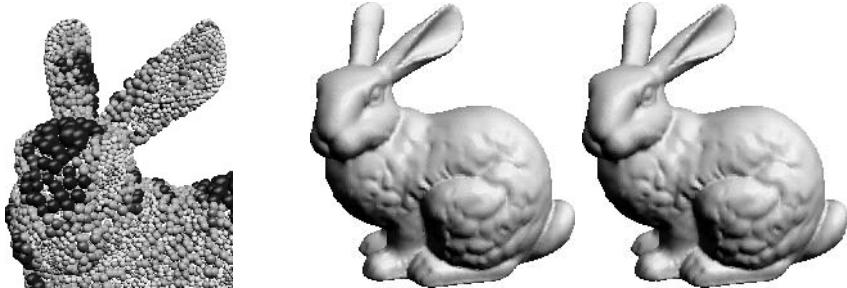


Fig. 5. [Reproduced in colour in Plate 13.] Adaptive approximation of the Stanford bunny model with $\varepsilon_0 = 2.5 \times 10^{-4}$. Right: for $T = 1.5$ each approximation centre is visualised by a sphere of radius $\sigma_k/4$; the spheres are coloured according to their sizes which increases from red to blue. Middle: 8,504 RBF centres (and local approximations) are used if $T = 1.5$; L^2 error = 1.86×10^{-4} and L^∞ error = 2.76×10^{-3} ; computational time is 7 seconds. Right: 20,813 RBF centres are used if $T = 5$; L^2 error = 1.72×10^{-4} and L^∞ error = 2.57×10^{-3} ; computational time is 19 seconds.

- Step 1. Assign $v_i = 0$ to each point $p_i \in \mathcal{P}$.
- Step 2. Choose randomly m points (in our current implementation, we use $m = 15$) with $v < T$.
- Step 3. Select a point with minimum v among the points chosen during the previous step.
- Step 4. Set the point selected at Step 3 to be an RBF centre $\mathbf{c}_k \in \mathcal{C}$. Set $v_k = T$ for \mathbf{c}_k .
- Step 5. Find optimal support size $\sigma_k = \sigma_{\text{opt}}$ and local polynomial approximation $g_k(\mathbf{x})$ at centre \mathbf{c}_k determined at the previous step.
- Step 6. Update overlapping v_i for all $\mathbf{p}_i \in \mathcal{P} \setminus \mathcal{C}$ by adding $w_{\sigma_k}(\|\mathbf{p}_i - \mathbf{c}_k\|)$.
- Step 7. If there remain points $\mathbf{p}_i \in \mathcal{P}$ with $v_i < T$, go to Step 2.

Notice that steps 2 and 3 implement a multiple choice technique, a powerful tool for randomised optimisation [12] introduced recently in geometric modelling [23].

According to our numerical experiments, choosing $T = 1.5$ produces a good cover. Bigger values of overlapping rate T lead to wasting computational power. A comparison of the approximations with $T = 1.5$ and $T = 5$ for the Stanford bunny model is given in Fig. 5.

Least Square Fitting

We use a least square fitting procedure to determine unknown coefficients $\{\lambda_i\}$ in (9). Let

$$E(\lambda_1, \dots, \lambda_M) = \sum_{i=1}^N \tilde{f}(\mathbf{p}_i)^2.$$

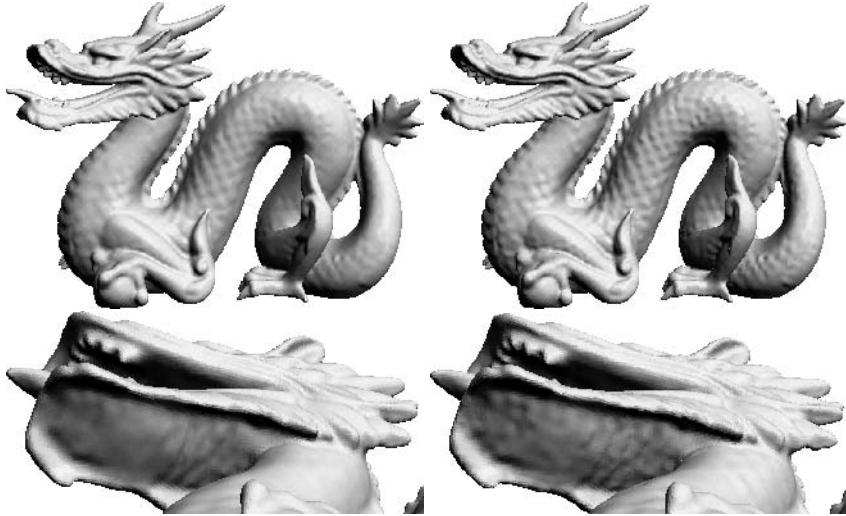


Fig. 6. The Stanford dragon model approximated with $\varepsilon_0 = 2.5 \times 10^{-4}$ (20,146 centres). Left images: only the base approximation is used; fitting time is 63 seconds; L^2 error = 1.50×10^{-4} and L^∞ error = 3.88×10^{-3} . Right images: local details are added; total fitting time is 148 seconds; L^2 error = 1.10×10^{-4} and L^∞ error = 1.66×10^{-3} .

Now coefficients $\lambda_1, \dots, \lambda_M$ are found by solving the system of linear equations

$$\partial E / \partial \lambda_i = 0, \quad i = 1, \dots, M.$$

Base Approximation and Local Details

Let us rewrite (9) in the form

$$f(\mathbf{x}) \approx \underbrace{\sum_{\mathbf{c}_i \in \mathcal{C}} g_i(\mathbf{x}) \phi_{\sigma_i}(\|\mathbf{x} - \mathbf{c}_i\|)}_{\text{base approximation}} + \underbrace{\sum_{\mathbf{c}_i \in \mathcal{C}} \lambda_i \phi_{\sigma_i}(\|\mathbf{x} - \mathbf{c}_i\|)}_{\text{local details}} \quad (10)$$

The first term of the right-hand side of (10) can be considered as a base approximation of $f(\mathbf{x})$ while the second term represents local details. Fig. 6 demonstrates this separation into a base approximation and local details for the Stanford dragon model. Surprisingly to us, the first term delivers a remarkably good approximation.

4 Conclusion

In this paper, we have reviewed our multi-scale approach to 3D scattered data interpolation with CS-RBFs [14], enhanced it by a centre reduction proce-

dure, and developed an economical and fast adaptive CS-RBF approximation scheme.

Surprisingly to us, an adaptive partition of unity approximation, which we use for initial fitting to scattered data, already delivers a high quality approximation. This should be the subject of further study.

Acknowledgements

This work was supported in part by the European Union research project “Multiresolution in Geometric Modelling (MINGLE)” under grant HPRN-CT-1999-00117.

We would like to thank the anonymous reviewers of this paper for their valuable and constructive comments.

The models are courtesy of the Digital Michelangelo Project 3D Model Repository (Michelangelo’s St. Matthew head model) and the Stanford 3D Scanning Repository (bunny, dragon, and Buddha models).

References

1. M. Alexa. Hierarchical partition of unity approximation. Technical report, TU Darmstadt, August 2002.
2. I. Babuška and J. M. Melenk. The partition of unity method. *International Journal of Numerical Methods in Engineering*, 40:727–758, 1997.
3. T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering*, 139:3–47, 1996.
4. D. Buhmann, M. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, 2003.
5. J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proc. ACM SIGGRAPH*, pages 67–76, August 2001.
6. M. S. Floater and A. A. Iske. Multistep scattered data interpolation using compactly supported radial basis functions. *Journal of Comp. Appl. Math.*, 73:65–78, 1996.
7. S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company, Inc., 1994.
8. A. Iske. Scattered data modelling using radial basis functions. In A. Iske, E. Quak, and M. S. Floater, editors, *Tutorials on Multiresolution in Geometric Modelling*, pages 205–242. Springer, 2002.
9. A. Iske and J. Levesley. Multilevel scattered data approximation by adaptive domain decomposition. Technical Report TUM-M0208, Technische Universität München, July 2002.
10. D. Lazzaro and L. B. Montefusco. Radial basis functions for multivariate interpolation of large scattered data sets. *Journal of Computational and Applied Mathematics*, 140:521–536, 2002.

11. S. K. Lodha and R. Franke. Scattered data techniques for surfaces. In H. Hagen, G. Nielson, and F. Post, editors, *Proceedings of Dagstuhl Conference on Scientific Visualization*, pages 182–222. IEEE Computer Society Press, 1999.
12. M. Mitzenmacher, A. Richa, and R. Sitaraman. The power of two random choices: A survey of techniques and results. In *Handbook of Randomized Computing, Chapter 9*. Kluwer, 2001.
13. Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics*, 22(3):463–470, July 2003. Proc. ACM SIGGRAPH 2003.
14. Y. Ohtake, A. G. Belyaev, and H.-P. Seidel. A multi-scale approach to 3D scattered data interpolation with compactly supported basis functions. In *Shape Modeling International 2003*, pages 153–161, Seoul, Korea, May 2003.
15. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1993.
16. V. V. Savchenko, A. A. Pasko, O. G. Okunev, and T. L. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 14(4):181–188, 1995.
17. G. Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, 1991.
18. I. Tobor, P. Reuter, and C. Schlick. Efficient reconstruction of large scattered geometric datasets using the partition of unity and radial basis functions. In *The 12-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'04)*, February 2004.
19. G. Turk and J. O'Brien. Shape transformation using variational implicit surfaces. In *Proc. ACM SIGGRAPH*, pages 335–342, August 1999.
20. G. Turk and J. O'Brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, 21(4):855–873, October 2002.
21. H. Wendland. Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree. *Advances in Computational Mathematics*, 4:389–396, 1995.
22. H. Wendland. Fast evaluation of radial basis functions: Methods based on partition of unity. In L. Schumaker and J. Stöckler, editors, *Approximation Theory X: Wavelets, Splines, and Applications*, pages 473–483. Vanderbilt University Press, Nashville, 2002.
23. J. Wu and L. P. Kobbelt. Fast mesh decimation by multiple-choice techniques. In *Vision, Modeling, Visualization 2002 Proceedings*, pages 241–248, Erlangen, Germany, November 2002.

Part IV

— Parameterization

Surface Parameterization: a Tutorial and Survey

Michael S. Floater¹ and Kai Hormann²

¹ Computer Science Department, Oslo University, Norway

`michaelf@ifi.uio.no`

² ISTI, CNR, Pisa, Italy

`hormann@isti.cnr.it`

Summary. This paper provides a tutorial and survey of methods for parameterizing surfaces with a view to applications in geometric modelling and computer graphics. We gather various concepts from differential geometry which are relevant to surface mapping and use them to understand the strengths and weaknesses of the many methods for parameterizing piecewise linear surfaces and their relationship to one another.

1 Introduction

A parameterization of a surface can be viewed as a one-to-one mapping from the surface to a suitable parameter domain. In general, the parameter domain itself will be a surface and so constructing a parameterization means mapping one surface into another. Typically, surfaces that are homeomorphic to a disk are mapped into the plane. Usually the surfaces are either represented by or approximated by triangular meshes and the mappings are piecewise linear.

Parameterizations have many applications in various fields of science and engineering, including scattered data fitting, reparameterization of spline surfaces, and repair of CAD models. But the main driving force in the development of the first parameterization methods was the application to texture mapping which is used in computer graphics to enhance the visual quality of polygonal models. Later, due to the quickly developing 3D scanning technology and the resulting demand for efficient compression methods of increasingly complex triangulations, other applications such as surface approximation and remeshing have influenced further developments.

Parameterizations almost always introduce distortion in either angles or areas and a good mapping in applications is one which minimises these distortions in some sense. Many different ways of achieving this have been proposed in the literature.

The purpose of this paper is to give an overview of the main developments over recent years. Our survey [20] of 2002 attempted to summarise advances

in this subject up to 2001. However, a large number of papers have appeared since then and wherever possible we will focus on these more recent advances. This paper also differs from [20] in that we build up the discussion from some classical differential geometry and mapping theory. We further discarded the classification of methods into linear and non-linear ones and rather distinguish between their differential geometric properties. We believe that this helps to clarify the strengths and weakness of the many methods and their relationship to one another.

2 Historical Background

The Greek astronomer *Claudius Ptolemy* (100–168 A.D.) was the first known to produce the data for creating a map showing the inhabited world as it was known to the Greeks and Romans of about 100–150 A.D. In his work *Geography* [89] he explains how to project a sphere onto a flat piece of paper using a system of grid lines – longitude and latitude.

As we know from peeling oranges and trying to flatten the peels on a table, the sphere cannot be projected onto the plane without distortions and therefore certain compromises must be made. Fig. 1 shows some examples. The orthographic projection (a), which was known to the Egyptians and Greeks more than 2000 years ago, modifies both angles and areas, but the directions from the centre of projection are true. Probably the most widely used projection is the stereographic projection (b) usually attributed to *Hipparchus* (190–120 B.C.). It is a conformal projection, i.e., it preserves angles (at the expense of areas). It also maps circles to circles, no matter how large (great circles are mapped into straight lines), but a *loxodrome* is plotted as a spiral. A loxodrome is a line of constant bearing and of vital importance in navigation. In 1569, the Flemish cartographer *Gerardus Mercator* (1512–1594), whose goal was to produce a map which sailors could use to determine courses [87], overcame this drawback with his conformal cylindrical *Mercator projection* (c) which draws every loxodrome as a straight line. Neither the stereographic nor the Mercator projections preserve areas however. *Johann Heinrich Lambert*

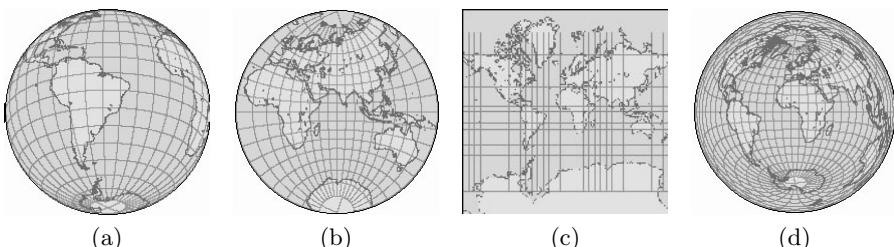


Fig. 1. (a) orthographic, (b) stereographic, (c) Mercator, and (d) Lambert projections of the Earth.

(1728–1777) found the first equiareal projection (d) in 1772 [86], at the cost of giving up the preservation of angles.

All these projections can be seen as functions that map a part of the surface of the sphere to a planar domain and the inverse of this mapping is usually called a *parameterization*. Many of the principles of parametric surfaces and differential geometry were developed by *Carl Friedrich Gauß* (1777–1855), mostly in [81].

Conformal projections of general surfaces are of special interest due to their close connection to complex analytic functions, and the Riemann Mapping Theorem. This theorem, due to *Bernhard Riemann* (1826–1866) in his dissertation [91] of 1851, states that any simply-connected region of the complex plane can be mapped conformally into any other simply-connected region, such as the unit disk. It implies, similarly, that any disk-like surface can be mapped conformally into any simply-connected region of the plane.

3 Differential Geometry Background

We take some basic theory of mappings from Kreyszig [85, Chap. VI]. Suppose a surface $S \subset \mathbb{R}^3$ has the parametric representation

$$\mathbf{x}(u^1, u^2) = (x_1(u^1, u^2), x_2(u^1, u^2), x_3(u^1, u^2))$$

for points (u^1, u^2) in some domain in \mathbb{R}^2 . We call such a representation *regular* if (i) the functions x_1, x_2, x_3 are smooth, i.e., differentiable as many times as we need for our discussion, and (ii) the vectors

$$\mathbf{x}_1 = \frac{\partial \mathbf{x}}{\partial u^1}, \quad \mathbf{x}_2 = \frac{\partial \mathbf{x}}{\partial u^2}$$

are linearly independent at every point (their cross product $\mathbf{x}_1 \times \mathbf{x}_2$ is non-zero).

Many properties of S are characterised by its *first fundamental form*, which is the square of the element of arc of a curve in S , the quadratic form

$$ds^2 = \mathbf{x}_1 \cdot \mathbf{x}_1 (du^1)^2 + 2 \mathbf{x}_1 \cdot \mathbf{x}_2 du^1 du^2 + \mathbf{x}_2 \cdot \mathbf{x}_2 (du^2)^2.$$

Writing

$$g_{\alpha\beta} = \mathbf{x}_\alpha \cdot \mathbf{x}_\beta, \quad \alpha = 1, 2, \quad \beta = 1, 2,$$

and arranging the coefficients in a symmetric matrix

$$\mathbf{I} = \begin{pmatrix} g_{11} & g_{12} \\ g_{12} & g_{22} \end{pmatrix}$$

we have

$$ds^2 = (du^1 \ du^2) \mathbf{I} \begin{pmatrix} du^1 \\ du^2 \end{pmatrix}.$$

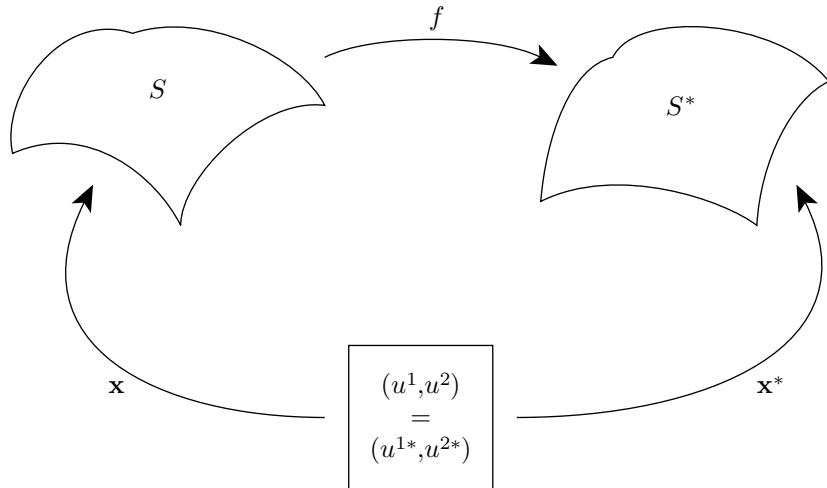


Fig. 2. The mapping f from S to S^* and the parameterization \mathbf{x} of S induce the parameterization $\mathbf{x}^* = f \circ \mathbf{x}$ of S^* .

Often, the matrix \mathbf{I} is itself referred to as the first fundamental form. Under the assumption of regularity, this matrix has a strictly positive determinant

$$g = \det \mathbf{I} = g_{11}g_{22} - g_{12}^2,$$

the discriminant of the quadratic form. In this case, the form is positive definite. The coefficients $g_{\alpha\beta}$ are the components of a covariant tensor of second order, called the metric tensor, denoted simply by $g_{\alpha\beta}$.

Suppose now that S is a surface with coordinates (u^1, u^2) and that f is a mapping from S to a second surface S^* . Then we can define the parameterization $\mathbf{x}^* = f \circ \mathbf{x}$ of S^* , so that the coordinates of any image point $f(\mathbf{p}) \in S^*$ are the same as those of the corresponding pre-image point $\mathbf{p} \in S$; see Fig. 2. We say that the mapping f is *allowable* if the parameterization \mathbf{x}^* is regular. With this set up we will now consider various kinds of mappings.

3.1 Isometric Mappings

An allowable mapping from S to S^* is *isometric* or *length-preserving* if the length of any arc on S^* is the same as that of its pre-image on S . Such a mapping is called an *isometry*.

For example, the mapping of a cylinder into the plane that transforms cylindrical coordinates into cartesian coordinates is isometric.

Theorem 1. *An allowable mapping from S to S^* is isometric if and only if the coefficients of the first fundamental forms are the same, i.e.,*

$$\mathbf{I} = \mathbf{I}^*.$$

Two surfaces are said to be *isometric* if there exists an isometry between them. Isometric surfaces have the same Gaussian curvature at corresponding pairs of points (since Gaussian curvature depends only on the first fundamental form).

3.2 Conformal Mappings

An allowable mapping from S to S^* is *conformal* or *angle-preserving* if the angle of intersection of every pair of intersecting arcs on S^* is the same as that of the corresponding pre-images on S at the corresponding point.

For example, the stereographic and Mercator projections are conformal maps from the sphere to the plane; see Fig. 1.

Theorem 2. *An allowable mapping from S to S^* is conformal or angle-preserving if and only if the coefficients of the first fundamental forms are proportional, i.e.,*

$$\mathbf{I} = \eta(u^1, u^2) \mathbf{I}^*, \quad (1)$$

for some scalar function $\eta \neq 0$.

3.3 Equiareal Mappings

An allowable mapping from S to S^* is *equiareal* if every part of S is mapped onto a part of S^* with the same area.

For example, the Lambert projection is an equiareal mapping from the sphere to the plane; see Fig. 1.

Theorem 3. *An allowable mapping from S to S^* is equiareal if and only if the discriminants of the first fundamental forms are equal, i.e.,*

$$g = g^*. \quad (2)$$

The proofs of the above three results can be found in Kreyszig [85]. It is then quite easy to see the following (see also Kreyszig).

Theorem 4. *Every isometric mapping is conformal and equiareal, and every conformal and equiareal mapping is isometric, i.e.,*

$$\text{isometric} \Leftrightarrow \text{conformal} + \text{equiareal}.$$

We can thus view an isometric mapping as ideal, in the sense that it preserves just about everything we could ask for: angles, areas, and lengths. However, as is well known, isometric mappings only exist in very special cases. When mapping into the plane, the surface S would have to be developable, such as a cylinder. Many approaches to surface parameterization therefore attempt to find a mapping which either

1. is conformal, i.e., has no distortion in angles, or
2. is equiareal, i.e., has no distortion in areas, or
3. minimises some combination of angle distortion and area distortion.

3.4 Planar Mappings

A special type of mappings that we will consider now and then in the following are planar mappings $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, $f(x, y) = (u(x, y), v(x, y))$. For these kind of mappings the first fundamental form can be written as

$$\mathbf{I} = J^T J$$

where $J = \begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix}$ is the Jacobian of f . It follows that the *singular values* σ_1 and σ_2 of J are just the square roots of the *eigenvalues* λ_1 and λ_2 of \mathbf{I} and it is then easy to verify

Proposition 1. *For a planar mapping $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ the following equivalencies hold:*

1. f is isometric $\Leftrightarrow \mathbf{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Leftrightarrow \lambda_1 = \lambda_2 = 1 \Leftrightarrow \sigma_1 = \sigma_2 = 1$,
2. f is conformal $\Leftrightarrow \mathbf{I} = \begin{pmatrix} \eta & 0 \\ 0 & \eta \end{pmatrix} \Leftrightarrow \lambda_1/\lambda_2 = 1 \Leftrightarrow \sigma_1/\sigma_2 = 1$,
3. f is equiareal $\Leftrightarrow \det \mathbf{I} = 1 \Leftrightarrow \lambda_1\lambda_2 = 1 \Leftrightarrow \sigma_1\sigma_2 = 1$.

4 Conformal and Harmonic Mappings

Conformal mappings have many nice properties, not least of which is their connection to complex function theory. Consider for the moment the case of mappings from a planar region S to the plane. Such a mapping can be viewed as a function of a complex variable, $\omega = f(z)$. Locally, a conformal map is simply any function f which is analytic in a neighbourhood of a point z and such that $f'(z) \neq 0$. A conformal mapping f thus satisfies the Cauchy-Riemann equations, which, with $z = x + iy$ and $\omega = u + iv$, are

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \quad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}. \quad (3)$$

Now notice that by differentiating one of these equations with respect to x and the other with respect to y , we obtain the two Laplace equations

$$\Delta u = 0, \quad \Delta v = 0,$$

where

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

is the Laplace operator.

Any mapping $(u(x, y), v(x, y))$ which satisfies these two Laplace equations is called a *harmonic mapping*. Thus a conformal mapping is also harmonic, and we have the implications

$$\text{isometric} \Rightarrow \text{conformal} \Rightarrow \text{harmonic}.$$

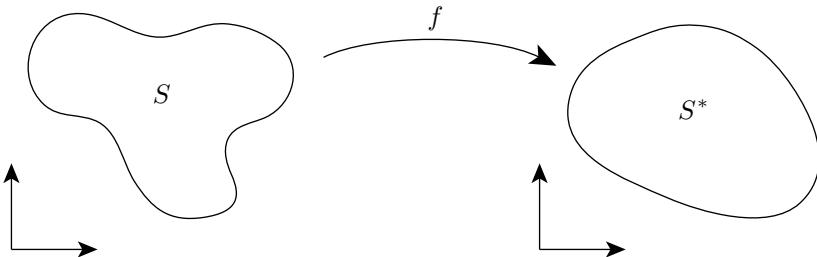


Fig. 3. One-to-one harmonic mappings.

Why do we consider harmonic maps? Well, their big advantage over conformal maps is the ease with which they can be computed, at least approximately. After choosing a suitable boundary mapping (which is equivalent to using a Dirichlet boundary condition for both u and v), each of the functions u and v is the solution to a linear elliptic partial differential equation (PDE) which can be approximated by various methods, such as finite elements or finite differences, both of which lead to a linear system of equations. Harmonic maps are also guaranteed to be one-to-one for convex regions. The following result was conjectured by Radó [90] and proved independently by Kneser [84] and Choquet [80].

Theorem 5 (RKC). *If $f : S \rightarrow \mathbb{R}^2$ is harmonic and maps the boundary ∂S homeomorphically into the boundary ∂S^* of some convex region $S^* \subset \mathbb{R}^2$, then f is one-to-one; see Fig. 3.*

On the downside, harmonic maps are not in general conformal and do not preserve angles. For example, it is easy to verify from the Cauchy-Riemann and Laplace equations that the bilinear mapping $f : [0, 1]^2 \rightarrow \mathbb{R}^2$ defined by

$$u = x(1 + y), \quad v = y,$$

is harmonic but not conformal. Indeed the figure below clearly shows that this harmonic map does not preserve angles.

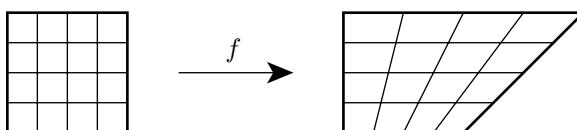


Fig. 4. A harmonic mapping which is not conformal.

Another weakness of harmonic mappings is their “one-sidedness”. The inverse of a harmonic mapping is not necessarily harmonic. Again, the bilinear example above provides an example of this. It is easy to check that the inverse

mapping $x = u/(1 + v)$, $y = v$ is not harmonic as the function $x(u, v)$ does not satisfy the Laplace equation.

Despite these drawbacks, harmonic mappings do at least minimise deformation in the sense that they minimise the Dirichlet energy

$$E_D(f) = \frac{1}{2} \int_S \|\operatorname{grad} f\|^2 = \frac{1}{2} \int_S (\|\nabla u\|^2 + \|\nabla v\|^2).$$

This property combined with their ease of computation explains their popularity.

When we consider mappings from a general surface $S \subset \mathbb{R}^3$ to the plane, we find that all the above properties of conformal and harmonic mappings are essentially the same. The equations just become more complicated. Any mapping f from a given surface S to the plane defines coordinates of S , say (u^1, u^2) . By Theorem 2, if f is conformal then there is some scalar function $\eta \neq 0$ such that

$$ds^2 = \eta(u^1, u^2)((du^1)^2 + (du^2)^2).$$

Suppose that S has given coordinates $(\tilde{u}^1, \tilde{u}^2)$. After some analysis (see Chap. VI of Kreyszig), one can show that the above equation implies the two equations

$$\frac{\partial u^1}{\partial \tilde{u}^1} = \frac{\tilde{g}_{11}}{\sqrt{\tilde{g}}} \frac{\partial u^2}{\partial \tilde{u}^2} - \frac{\tilde{g}_{12}}{\sqrt{\tilde{g}}} \frac{\partial u^1}{\partial \tilde{u}^1}, \quad \frac{\partial u^1}{\partial \tilde{u}^2} = -\frac{\tilde{g}_{22}}{\sqrt{\tilde{g}}} \frac{\partial u^2}{\partial \tilde{u}^1} + \frac{\tilde{g}_{12}}{\sqrt{\tilde{g}}} \frac{\partial u^2}{\partial \tilde{u}^2}, \quad (4)$$

which are a generalisation of the Cauchy-Riemann equations (3). Indeed, in the special case that S is planar, we can take

$$\tilde{g}_{11} = \tilde{g}_{22} = 1, \quad \tilde{g}_{12} = 0, \quad (5)$$

and we get simply

$$\frac{\partial u^1}{\partial \tilde{u}^1} = \frac{\partial u^2}{\partial \tilde{u}^2}, \quad \frac{\partial u^1}{\partial \tilde{u}^2} = -\frac{\partial u^2}{\partial \tilde{u}^1}.$$

In a similar manner to the planar case, we can differentiate one equation in (4) with respect to \tilde{u}^1 and the other with respect to \tilde{u}^2 , and obtain the two generalisations of Laplace's equation,

$$\Delta_S u^1 = 0, \quad \Delta_S u^2 = 0, \quad (6)$$

where Δ_S is the Laplace-Beltrami operator

$$\Delta_S = \frac{1}{\sqrt{\tilde{g}}} \left(\frac{\partial}{\partial \tilde{u}^1} \left(\frac{\tilde{g}_{22}}{\sqrt{\tilde{g}}} \frac{\partial}{\partial \tilde{u}^1} - \frac{\tilde{g}_{12}}{\sqrt{\tilde{g}}} \frac{\partial}{\partial \tilde{u}^2} \right) + \frac{\partial}{\partial \tilde{u}^2} \left(\frac{\tilde{g}_{11}}{\sqrt{\tilde{g}}} \frac{\partial}{\partial \tilde{u}^2} - \frac{\tilde{g}_{12}}{\sqrt{\tilde{g}}} \frac{\partial}{\partial \tilde{u}^1} \right) \right).$$

When this operator is differentiated out, one finds that it is a linear elliptic operator with respect to the coordinates $(\tilde{u}^1, \tilde{u}^2)$ (as noted and exploited by Greiner [82]). This operator generalises the Laplace operator (as can easily be checked by taking S to be planar with $\tilde{g}_{\alpha\beta}$ as in (5)), and is independent of the

particular coordinates (in this case $(\tilde{u}^1, \tilde{u}^2)$) used to define it. As explained by Klingenberg [83], it can also be written simply as

$$\Delta_S = \operatorname{div}_S \operatorname{grad}_S.$$

In a similar manner to the planar case, a harmonic map can either be viewed as the solution to equation (6), or as the minimiser of the Dirichlet energy

$$E_D(f) = \frac{1}{2} \int_S \|\operatorname{grad}_S f\|^2$$

over the surface S .

5 Equiareal Mappings

As we saw in Sect. 3, there are essentially only two quantities to consider minimising in a mapping: angle distortion and area distortion.

We know from the Riemann mapping theorem that (surjective) conformal mappings from a disk-like surface to a fixed planar simply-connected region not only exist but are also *almost unique*. For example, consider mapping the unit disk S into itself (treating S as a subset of the complex plane), and choose any point $z \in S$ and any angle θ , $-\pi < \theta \leq \pi$. According to the theorem, there is precisely *one* conformal mapping $f : S \rightarrow S$ such that $f(z) = 0$ and $\arg f'(z) = \theta$. In this sense there are only the three degrees of freedom defined by the complex number z and the real angle θ in choosing the conformal map.

What we want to do now is to demonstrate that equiareal mappings are *substantially different* to conformal ones from the point of view of uniqueness as there are many more of them. The following example is to our knowledge novel and nicely illustrates the abundance of equiareal mappings. Consider again mappings $f : S \rightarrow S$, from the unit disk S into itself. Using the polar coordinates $x = r \cos \theta$, $y = r \sin \theta$, one easily finds that the determinant of the Jacobian of any mapping $f(x, y) = (u(x, y), v(x, y))$ can be expressed as

$$\det J(f) = u_x v_y - u_y v_x = \frac{1}{r} (u_r v_\theta - u_\theta v_r).$$

Consider then the mapping $f : S \rightarrow S$ defined by

$$r(\cos \theta, \sin \theta) \mapsto r(\cos(\theta + \phi(r)), \sin(\theta + \phi(r))),$$

for $0 \leq r \leq 1$ and $-\pi < \theta \leq \pi$, where $\phi : [0, 1] \rightarrow \mathbb{R}$ is an arbitrary function. This mapping maps each circle of radius r centred at the origin into itself, rotated by the angle $\phi(r)$; see Fig. 5. If ϕ is differentiable then so is f and differentiation shows that

$$u_r v_\theta - u_\theta v_r = r,$$

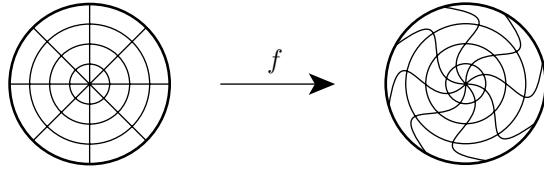


Fig. 5. An equiareal mapping.

independent of the function ϕ . We conclude that $\det J(f) = 1$ and therefore, according to Proposition 1, f is equiareal, *irrespective* of the chosen univariate function ϕ .

It is not difficult to envisage other families of equiareal mappings constructed by rotating circles about other centres in S . These families could also be combined to make further equiareal mappings.

When we consider again the formulations of conformal and equiareal mappings in terms of the first fundamental form, the lack of uniqueness of equiareal mappings becomes less surprising. For, as we saw earlier, the property of conformality (1) essentially places *two* conditions on the three coefficients of the first fundamental form $g_{11}^*, g_{12}^*, g_{22}^*$, while the property of equiarealness (2) places only *one* condition on them (the three conditions together of course completely determine the three coefficients, giving an isometric mapping).

Considering not only the non-uniqueness, but also the rather strange rotational behaviour of the above mappings, we conclude that it is hardly sensible to try to minimise area deformation alone. In order to find a well-behaved mapping we surely need to combine area-preservation with some minimisation of angular distortion.

6 Discrete Harmonic Mappings

Common to almost all surface parameterization methods is to approximate the underlying smooth surface S by a piecewise linear surface S_T , in the form of a *triangular mesh*, i.e. the union of a set $T = \{T_1, \dots, T_M\}$ of triangles T_i such that the triangles intersect only at common vertices or edges. Nowadays in fact, surfaces are frequently simply *represented* as triangular meshes, and the smooth underlying surface is often not available. We will denote by V the set of vertices. If S_T has a boundary, then the boundary will be polygonal and we denote by V_B the set of vertices lying on the boundary and by V_I the set of interior vertices.

The most important parameterization task is to map a given disk-like surface $S \subset \mathbb{R}^3$ into the plane. Working with a triangular mesh S_T , the goal is to find a suitable (polygonal) domain $S^* \subset \mathbb{R}^2$ and a suitable piecewise linear mapping $f : S_T \rightarrow S^*$ that is linear on each triangle T_i in S_T and continuous; see Fig. 6. Such a mapping is uniquely determined by the images $f(v) \in \mathbb{R}^2$ of the vertices $v \in V$.

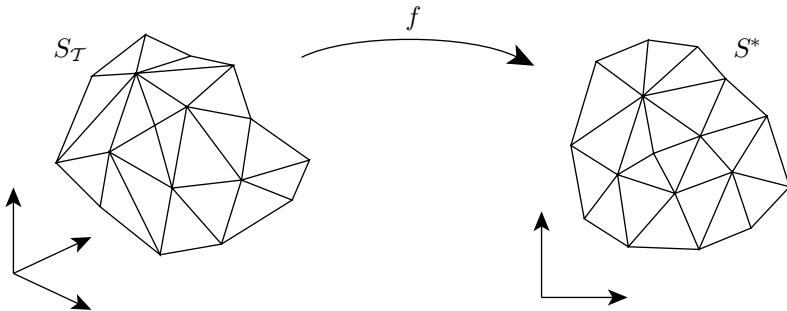


Fig. 6. Piecewise linear mapping of a triangular mesh.

6.1 Finite Element Method

One of the earliest methods for mapping disk-like surfaces into the plane was to approximate a harmonic map using the finite element method based on linear elements. This method was introduced to the computer graphics community by Eck et al. [12] and called simply a *discrete harmonic map*, although a similar technique had earlier been used by Pinkall and Polthier for computing piecewise linear minimal surfaces [56]. The basic method has two steps.

1. First fix the boundary mapping, i.e. fix $f|_{\partial S_T} = f_0$, by mapping the polygonal boundary ∂S_T homeomorphically to some polygon in the plane. This is equivalent to choosing the planar image of each vertex in the mesh boundary ∂S_T and can be done in several ways (see [14] or [33, Sec. 1.2.5] for details).
2. Find the piecewise linear mapping $f : S_T \rightarrow S^*$ which minimises the Dirichlet energy

$$E_D = \frac{1}{2} \int_{S_T} \|\text{grad}_{S_T} f\|^2,$$

subject to the Dirichlet boundary condition $f|_{\partial S_T} = f_0$.

The main advantage of this method over earlier approaches is that this is a quadratic minimisation problem and reduces to solving a linear system of equations. Consider one triangle $T = [v_1, v_2, v_3]$ in the surface S_T . Referring to Fig. 7, one can show that

$$\begin{aligned} 2 \int_T \|\text{grad}_T f\|^2 &= \cot \theta_3 \|f(v_1) - f(v_2)\|^2 \\ &\quad + \cot \theta_2 \|f(v_1) - f(v_3)\|^2 + \cot \theta_1 \|f(v_2) - f(v_3)\|^2. \end{aligned}$$

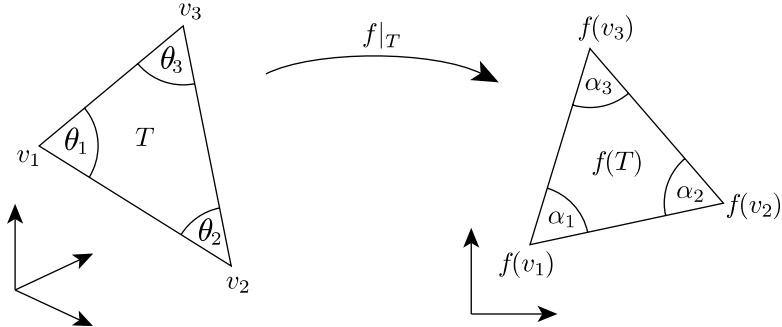


Fig. 7. Atomic map between a mesh triangle and the corresponding parameter triangle.

The normal equations for the minimisation problem can therefore be expressed as the linear system of equations

$$\sum_{j \in N_i} w_{ij} (f(v_j) - f(v_i)) = 0, \quad v_i \in V_I, \quad (7)$$

where

$$w_{ij} = \cot \alpha_{ij} + \cot \beta_{ij} \quad (8)$$

and the angles α_{ij} and β_{ij} are shown in the figure below. Here we have assumed that the vertices in V are indexed (in any random order) and that N_i denotes the set of indexes of the neighbours of the vertex v_i (those vertices which share an edge with v_i).

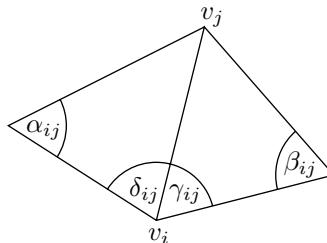


Fig. 8. Angles for the discrete harmonic map and the mean value coordinates.

The associated matrix is symmetric and positive definite, and so the linear system is uniquely solvable. The matrix is also sparse and well-conditioned enough that iterative methods are effective, e.g., conjugate gradients. Note that the system has to be solved twice, once for the x - and once for the y -coordinates of the parameter points $f(v_i)$, $v_i \in V_I$. In practice the method often gives good visual results.

6.2 Convex Combination Maps

The theory of finite elements [79] provides a well-established convergence theory for finite element approximations to second order elliptic PDEs. Extrapolating this theory, we can argue that the error incurred when discretising a harmonic map $f : S \rightarrow S^*$, $S^* \subset \mathbb{R}^2$, from a smooth surface to the plane, by a discrete harmonic map over some triangular mesh S_T of S , will tend to zero as the mesh size tends to zero (in an appropriate norm and under certain conditions on the angles of the triangles).

Due to the RKC Theorem 5, it is therefore reasonable to expect that, with S^* convex, a discrete harmonic map $f : S_T \rightarrow S^*$, like its harmonic cousin, will be one-to-one, i.e., that for every oriented triangle $T = [v_1, v_2, v_3]$ in the surface S_T , the mapped triangle $f(T) = [f(v_1), f(v_2), f(v_3)]$ would be non-degenerate and have the same orientation. It turns out that this is guaranteed to be true if all the weights w_{ij} in Equation (7) are positive. To understand this, note that if we define the normalised weights

$$\lambda_{ij} = w_{ij} / \sum_{k \in N_i} w_{ik},$$

for each interior vertex v_i , we can re-express the system (7) as

$$f(v_i) = \sum_{j \in N_i} \lambda_{ij} f(v_j), \quad v_i \in V_I. \quad (9)$$

It follows that if all the weights w_{ij} are positive then so are the weights λ_{ij} and the piecewise linear mapping f demands that each mapped interior vertex $f(v_i)$ will be a convex combination of its neighbours $f(v_j)$, and so must lie in their convex hull. It is reasonable to call *any* piecewise linear mapping of this kind a *convex combination mapping*. The special case in which the weights λ_{ij} are uniform, i.e., for each interior vertex v_i they are equal to $1/d_i$, where d_i is the valency of vertex v_i , was called a *barycentric mapping* by Tutte [92] (in a more abstract graph-theoretic setting). Each image point $f(v_i)$ is forced to be the barycentre of its neighbours. Tutte showed the following.

Theorem 6 (Tutte). *A barycentric mapping of any simple 3-connected planar graph G is a valid straight line embedding.*

It was later observed in [14] that this theorem applies to triangular meshes, and moreover, that Tutte's proof could be extended in a simple way to allow arbitrary positive weights λ_{ij} in Equation (9) satisfying $\sum_{j \in N_i} \lambda_{ij} = 1$. Recently, an independent and simpler proof of this result was given in [19].

Theorem 7. *If $f : S_T \rightarrow S^*$ is a convex combination mapping which maps ∂S_T homeomorphically into a convex polygon ∂S^* , then f is one-to-one.*

Recalling the weights of Equation (8), notice from trigonometry that

$$\cot \alpha_{ij} + \cot \beta_{ij} = \frac{\sin(\alpha_{ij} + \beta_{ij})}{\sin \alpha_{ij} \sin \beta_{ij}},$$

and so

$$w_{ij} > 0 \iff \alpha_{ij} + \beta_{ij} < \pi.$$

Therefore, it follows (see again [19]):

Proposition 2. *A discrete harmonic map $f : S_T \rightarrow S^*$ is one-to-one if it maps ∂S_T homeomorphically into a convex polygon ∂S^* and if the sum of every pair of opposite angles of quadrilaterals in S_T is less than π .*

Generally speaking, this *opposite-angle* condition is fulfilled when the triangles are “well-shaped”, and holds in particular when all angles of all triangles in S_T are less than $\pi/2$.

Conversely, counterexamples have been constructed (a numerical one in Duchamp et al. [11] and an analytical one in [15]) which show that if the opposite-angle condition does not hold then the discrete harmonic map may not be one-to-one: some triangles “flip over”, i.e. have the wrong orientation.

We envisage two possible ways of tackling this problem. The first approach is to perform some preprocessing operation on the given triangular mesh and insert new vertices to split triangles and perhaps swap some edges in order to obtain a new mesh for which the opposite angle condition holds. Of course, if the mesh is *planar*, we could simply use the well-known Delaunay swap criterion, and we would eventually end up with a Delaunay triangulation, which certainly satisfies the opposite angle condition in every quadrilateral, provided no four points are co-circular. However, we do not know of any concrete swapping procedure in the literature which provides the same guarantee for a general surface mesh. The other alternative is to *design* a convex combination map with good properties and if possible to mimic the behaviour of a harmonic map.

6.3 Mean Value Coordinates

In addition to injectivity, another natural property that we can expect from a mapping is to be an isometry whenever possible. It is well-known [83, 85] that such an isometry exists if and only if the surface S is *developable*. Piecewise linear surfaces S_T are developable if the angles around each interior vertex sum up to 2π which rarely is the case, unless S_T is planar. We therefore propose that a good piecewise linear mapping should have the following *reproduction property*: In the case that the surface mesh S_T is planar and its planar polygonal boundary is mapped affinely into the plane, then the whole mapping should be the same affine mapping.

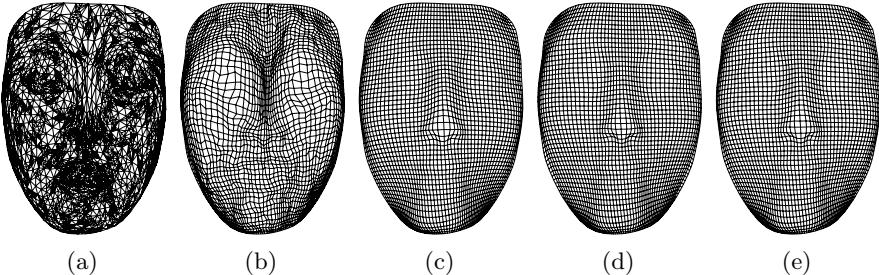


Fig. 9. Remeshing a triangle mesh with a regular quadrilateral mesh using different parameterization methods.

Discrete harmonic maps have this reproduction property but are not guaranteed to be injective. The *shape-preserving* method of [14] is a convex combination mapping (and therefore always one-to-one for convex images), designed also to have this reproduction property. In many numerical examples, the discrete harmonic map and shape-preserving maps look visually very similar, especially when the surface is not far from planar. For more complex shapes, the two methods begin to differ more, with the shape-preserving map being more robust in the presence of long and thin triangles.

A more recent paper [18] gives an alternative construction of a convex combination mapping with the reproduction property, which both simplifies the shape-preserving method of [14] and at the same time directly discretises a harmonic map. It is based on *mean value coordinates* and motivated as explained below. The numerical results are quite similar to the shape-preserving parameterization. Fig. 9 shows the result of first mapping a triangle mesh (a) to a square and then mapping a regular rectangular grid back onto the mesh. The four mappings used are barycentric (b), discrete harmonic (c), shape-preserving (d), and mean value (e).

The idea in [18] is the observation that harmonic functions (and therefore also harmonic maps) satisfy the mean value theorem. At every point in its (planar) domain, the value of a harmonic function is equal to the average of its values around any circle centred at that point. This suggests finding a piecewise linear map $f : S_T \rightarrow S^*$, for a planar triangular mesh S_T , which satisfies the mean value theorem at every interior vertex v_i of the mesh. We let Γ_i be a circle centred at v_i with radius $r_i > 0$ small enough that Γ_i only intersects triangles in T which are incident on v_i . We then demand that

$$f(v_i) = \frac{1}{2\pi r_i} \int_{\Gamma_i} f(v) ds.$$

Some algebra then shows that independently of $r_i > 0$ (for r_i small enough), the above equation is the same as Equation (7) but with the weights w_{ij} replaced by

$$w_{ij} = \frac{\tan(\delta_{ij}/2) + \tan(\gamma_{ij}/2)}{\|v_j - v_i\|},$$

with the angles shown in Fig. 8, and where now $w_{ij} \neq w_{ji}$ in general. When S_T is a surface mesh, we simply use the same weights with the angles δ_{ij} and γ_{ij} taken from the mesh triangles. For a recent in-depth comparison of computational aspects of discrete harmonic maps and mean value maps, including multilevel solvers, see Aksoylu, Khodakovskiy, and Schröder [1].

Energy Minimisation

We have seen that mean value maps discretise harmonic maps in a certain way, but in contrast to discrete harmonic maps they are not the solution of a known minimisation problem. This makes them a bit special because all other parameterization methods in the literature stem from the minimisation of some energy.

For example, discrete harmonic maps minimise the Dirichlet energy, and recently Guskov [29] showed that the shape-preserving maps minimise an energy that is based on second differences. But these are not the only energies that are minimised by convex combination maps. Greiner and Hormann [25] showed that any choice of symmetric weights $w_{ij} = w_{ji}$ in (7) minimises a *spring energy* and Desbrun, Meyer, and Alliez [10] proposed the *chi energy* that is minimised if the Wachspress coordinates [93, 94, 88] are taken as w_{ij} .

An interesting question for future research is whether there also exists a meaningful energy that is minimised by mean value mappings.

6.4 The Boundary Mapping

The first step in constructing both the discrete harmonic and the convex combination maps is to choose the boundary mapping $f|_{\partial S_T}$. There are two issues here: (i) choosing the *shape* of the boundary, and (ii) choosing the *distribution* of the points around the boundary.

Choosing the Shape

In many applications, it is sufficient (or even desirable) to map to a rectangle or a triangle, or even a polygonal approximation to a circle. In all these cases, the boundary is convex and the methods of the previous section work well.

The convexity restriction may, however, generate big distortions near the boundary when the boundary of the surface S_T does not resemble a convex shape. One practical solution to avoid such distortions is to build a “virtual” boundary, i.e., to augment the given mesh with extra triangles around the boundary so as to construct an extended mesh with a “nice” boundary. This approach has been successfully used by Lee, Kim, and Lee [43], and Kós and Várdy [40].

Choosing the Distribution

Consider first the case of a smooth surface S with a smooth boundary ∂S . Due to the Riemann Mapping Theorem we know that S can be mapped into *any* given simply-connected region $S^* \subset \mathbb{R}^2$ by a *conformal* map $f : S \rightarrow S^*$. Since any such conformal map defines a boundary mapping $f|_{\partial S} : \partial S \rightarrow \partial S^*$, this implies (assuming smooth well-behaved boundaries) that there must *exist* some boundary mapping such that the harmonic map it defines is also conformal. Such a boundary mapping seems like an ideal mapping to aim for. However, to the best of our knowledge it is not known how to find one.

Thus in the case of piecewise linear mappings, the usual procedure in the literature is to choose some simple boundary mapping such as chord length parameterization (for polygons), either around the whole boundary, or along each side of the boundary when working with triangular or rectangular boundaries.

An interesting topic for future research is to search for better ways to distribute the mapped boundary points around a fixed, chosen boundary (such as a circle). It seems likely that finding a distribution that maximises the conformality of the whole mapping will depend at least on the global shape of the surface mesh boundary and perhaps on the shape of the surface itself. As far as we know this issue has not yet been addressed in the literature.

7 Discrete Conformal Mappings

In all the parameterization methods described in the previous section, the boundary mapping $f|_{\partial S_T}$ had to be fixed in advance and preferably map to a convex polygon. There are, however, several approaches that maximise the conformality of the piecewise linear mapping *without* demanding the mesh boundary to be mapped onto a fixed shape. Instead, these methods allow the parameter values of the boundary points to be included into the optimisation problem and the shape of the parameter domain is determined by the method.

7.1 Most Isometric Parameterizations

The method of Hormann and Greiner [34] is based on measuring the conformality of a (non-degenerate) bivariate linear function $g : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ by the condition number of its Jacobian J with respect to the Frobenius-norm, which can be expressed in terms of the singular values σ_1 and σ_2 of J as follows:

$$E_M(g) = \kappa_F(J) = \|J\|_F \|J^{-1}\|_F = \sqrt{\sigma_1^2 + \sigma_2^2} \sqrt{1/\sigma_1^2 + 1/\sigma_2^2} = \frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1}.$$

According to Proposition 1 this functional clearly is minimal if and only if g is conformal. Since each atomic map $f|_T : T \rightarrow \mathbb{R}^2$ can be seen as such a

bivariate linear function, the conformality of the piecewise linear mapping f is then defined as

$$E_M(f) = \sum_{T \in \mathcal{T}} E_M(f|_T). \quad (10)$$

This energy is bounded from below by twice the number of triangles in \mathcal{T} and this minimum is obtained if and only if f is conformal. Thus, minimising (10) gives a parameterization that is as conformal as possible. Note that piecewise linear functions can only be conformal if the surface $S_{\mathcal{T}}$ is developable and conformality implies isometry in this case. Hence the term “most isometric parameterisations” (MIPS).

Interestingly, the notion of singular values is also useful to express the Dirichlet energy of a linear mapping $g(x, y) = (u(x, y), v(x, y))$. Using the identity

$$\sigma_1^2 + \sigma_2^2 = \text{trace}(J^T J) = \text{trace}(\mathbf{I}) = u_x^2 + u_y^2 + v_x^2 + v_y^2 \quad (11)$$

we find for any planar region S that

$$E_D(g) = \frac{1}{2} \int_S \|\text{grad } g\|^2 = \frac{1}{2} \int_S (\|\nabla u\|^2 + \|\nabla v\|^2) = \frac{1}{2} (\sigma_1^2 + \sigma_2^2) A(S),$$

where $A(S)$ denotes the area of S . Further considering the identity

$$\sigma_1 \sigma_2 = \det J = u_x v_y - u_y v_x = A(g(S))/A(S) \quad (12)$$

reveals a close relation between the MIPS energy of an atomic map and its Dirichlet energy,

$$E_M(f|_T) = 2 \frac{E_D(f|_T)}{A(f(T))}.$$

This underlines the conformality property of the MIPS method since it is well known that $E_D(f|_T) \geq A(f(T))$ with equality if and only if f is conformal.

It also shows that the MIPS energy in (10) is a sum of quadratic rational functions in the unknown parameter values $f(v)$ and thus the minimisation is a non-linear problem. As proposed in [36], this problem can be solved in the following way. Starting from an initial barycentric mapping, each planar vertex is repeatedly relocated in order to minimise the functional locally there. During this iteration, each vertex $p_i = f(v_i)$ in the current planar mesh lies in the kernel K_i of the star-shaped polygon formed by its neighbours. Since the MIPS energy is infinite if any mapped triangle $f(T)$ is degenerate, it is infinite on the boundary of the kernel K_i . There must therefore be a local minimum to the local functional somewhere in the interior of K_i . In fact, it has been shown in [33, Sec. 1.3.2] that the local functional is convex over the interior of K_i and that the local minimum can be found efficiently using Newton’s method. By moving p_i to this minimum, the method ensures that the updated planar mesh will not have any folded triangles.

7.2 Angle-based Flattening

While the conformality condition used in the previous section is triangle-based and can be expressed in terms of the parameter values $f(v)$, $v \in V$, the method of Sheffer and de Sturler [69] minimises a pointwise criterion that is formulated in terms of the angles of the parameter triangles.

Let us denote by θ_i the *mesh angles* in S_T and by α_i the corresponding *planar angles* in S^* . We further define $I(v)$ as the set of indices of the angles around a vertex $v \in V$ and the sum of these angles, $\theta(v) = \sum_{i \in I(v)} \theta_i$. For any interior vertex $v \in V_I$, the planar angles α_i , $i \in I(v)$ sum up to 2π , but the corresponding mesh angles usually do not. This angular deformation is inevitable for piecewise linear mappings and the best one can hope for is that the deformation is distributed evenly around the vertex. Sheffer and de Sturler therefore define for each $v \in V$ the *optimal angles* $\beta_i = \theta_i s(v)$, $i \in I(v)$ with a uniform scale factor

$$s(v) = \begin{cases} 2\pi/\theta(v), & v \in V_I, \\ 1, & v \in V_B, \end{cases}$$

and determine an optimal set of planar angles by minimising the energy

$$E(\alpha) = \sum_i (\alpha_i / \beta_i - 1)^2. \quad (13)$$

They finally construct the parameter values $f(v)$ and thus the piecewise linear mapping f itself from the angles α_i .

Though the minimisation problem is linear in the unknowns α_i , it becomes non-linear as a number of constraints (some of which are non-linear) have to be taken into account to guarantee the validity of the solution. A simplification of these constraints as well as a discussion of suitable solvers can be found in [77]. As in the previous section, the energy in (13) is bounded from below and the minimum is obtained if and only if S_T is developable with $\theta(v) = 2\pi$ at all interior vertices and f is conformal with $\alpha_i = \beta_i = \theta_i$ for all i .

7.3 Linear Methods

Lévy et al. [47] and Desbrun et al. [10] both independently developed a third method to compute discrete conformal mappings which has the advantage of being linear. For a bivariate linear function $g : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, Lévy et al. propose measuring the violation of the Cauchy-Riemann equations (3) in a least squares sense, i.e., with the *conformal energy*

$$E_C(g) = \frac{1}{2}((u_x - v_y)^2 + (u_y + v_x)^2).$$

Based on this they find the optimal piecewise linear mapping $f : S_T \rightarrow S^*$ by minimising

$$E_C(f) = \sum_{T \in \mathcal{T}} E_C(f|_T) A(T).$$

Like the MIPS energy, $E_C(g)$ can be expressed in terms of the singular values of the Jacobian of g and there is a close relation to the Dirichlet energy. Using (11) and (12) we find

$$E_C(g) = \frac{1}{2}(\sigma_1 - \sigma_2)^2$$

and

$$E_C(g) A(S) = E_D(g) - A(g(S))$$

for any planar region S . Therefore we have

$$E_C(f) = E_D(f) - A(f),$$

which also shows that $E_C(f)$ is quadratic in the unknowns $f(v)$ and that the normal equations for the minimisation problem can therefore be expressed as a linear system of equations.

Desbrun et al. take a slightly different path to arrive at the same system. They start with the finite element method (see Sect. 6.1) that yields the equations

$$D_p E_D(f) = 0$$

for all parameter points $p = f(v)$ of the interior vertices $v \in V_I$; compare (7). But instead of fixing the boundary $f|_{\partial S_T}$, they impose *natural boundary* constraints,

$$D_p E_D(f) = D_p A(f),$$

for all $p = f(v)$, $v \in V_B$. But as they also show that $D_p A(f) = 0$ at the interior vertices, this amounts to solving

$$\text{grad } E_D = \text{grad } A,$$

and is thus equivalent to minimising $E_C(f)$.

However, as $E_C(f)$ is clearly minimised by all degenerate mappings f that map S_T to a single point, additional constraints are needed to find a unique and non-trivial solution. Both papers therefore propose to fix the parameter values $f(v), f(w)$ of two vertices $v, w \in V$. The solution depends on this choice. For example, if we parameterize the pyramid in Fig. 10 (a) whose vertices lie on the corners of a cube, fixing $p_1 = f(v_1)$ and $p_2 = f(v_2)$ gives the solution in (b), while fixing $p_1 = f(v_1)$ and $p_3 = f(v_3)$ results in the parameterization shown in (c).

Note that the x - and y -coordinates of the parameter points $f(v)$ are coupled in this approach since the areas of the parameter triangles are involved. Thus the size of the system to be solved is roughly twice as large as the one for discrete harmonic maps (see Sect. 6.1).

We further remark that unlike the MIPS and the angle based flattening methods, this approach may generate folded triangles and that we do not know of any sufficient conditions that guarantee the resulting parameterization to be a one-to-one mapping.

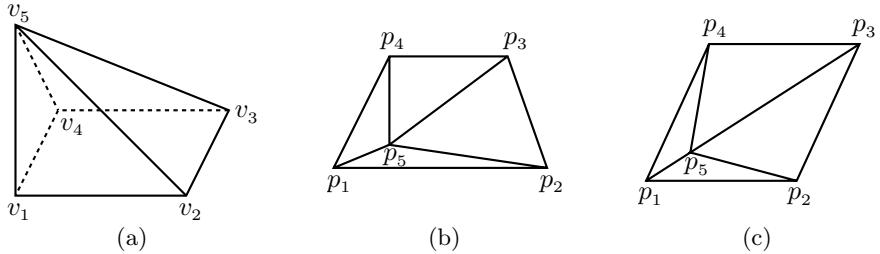


Fig. 10. Example of two different discrete conformal mappings for the same triangulation.

8 Discrete Equiareal Mappings

In view of the high degree of non-uniqueness of equiareal mappings shown in Sect. 5, it is not surprising that discrete (piecewise linear) equiareal mappings are also far from unique and also exhibit strange behaviour. For example, an obvious attempt at an area-preserving mapping $f : S_T \rightarrow S^*, S^* \subset \mathbb{R}^2$, for a triangular mesh $S_T \subset \mathbb{R}^3$ is to fix the polygonal region S^* to have the same area as that of S_T , and then to find f which minimises a functional like

$$E(f) = \sum_{T \in \mathcal{T}} (A(f(T)) - A(T))^2.$$

Unlike the discrete Dirichlet energy, this functional is no longer quadratic in the coordinates of the image points $f(v)$. Not surprisingly, there exist meshes for which E has several minima and, moreover, there may be several mappings f such that $E(f) = 0$. Fig. 11 shows an example in which the area $A(f(T))$ of each image triangle is equal to the area of the corresponding domain triangle $A(T)$ and thus $E(f) = 0$. In other words, f is a (discrete) equiareal mapping.

Other examples of minimising the functional E and its variants often produce long and thin triangles. In some cases triangles flip over. Maillot, Yahia, and Verroust [52] gave a variant of E in which each term in the sum is divided by $A(T)$, but few numerical examples are given in their paper. Surazhsky and

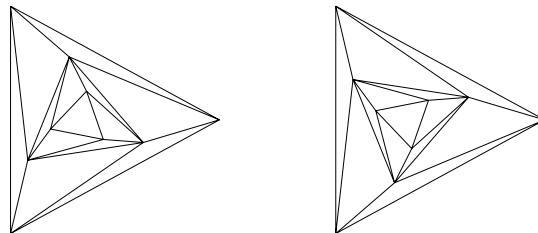


Fig. 11. Two planar meshes whose corresponding triangles have the same area.

Gotsman [76] have found area-equalisation useful for other purposes, specifically for remeshing.

Recently, Degener et al. [9] extended the MIPS method to find parameterizations that mediate between angle and area deformation. They measure the area deformation of a bivariate linear function g by

$$E_A(g) = \det J + \frac{1}{\det J} = \sigma_1 \sigma_2 + \frac{1}{\sigma_1 \sigma_2},$$

which, according to Proposition 1, clearly is minimal if and only if g is equiareal. They then minimise the overall energy

$$E(f) = \sum_{T \in \mathcal{T}} E_M(f|_T) E_A(f|_T)^q A(T),$$

where $q \geq 0$ is a parameter that controls the relative importance of the angle and the area deformation. Note that the case $q = 0$ corresponds to minimising angle deformation alone, but that no value of q gives pure minimisation of areas.

Sander et al. [62] explore methods based on minimising functionals that measure the “stretch” of a mapping. These appear to retain some degree of conformality in addition to reducing area distortion and seem to perform well in numerical examples. In the notation of Sect. 7.1 they measure the stretch of a bivariate linear mapping $g : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ by

$$E_2(g) = \|J\|_F = \sqrt{\sigma_1^2 + \sigma_2^2} \quad \text{and} \quad E_\infty(g) = \|J\|_\infty = \sigma_1$$

and minimise one of the two functionals

$$E_2(f) = \sqrt{\frac{\sum_{T \in \mathcal{T}} A(T) E_2(f|_T^{-1})^2}{\sum_{T \in \mathcal{T}} A(T)}}, \quad E_\infty(f) = \max_{T \in \mathcal{T}} E_\infty(f|_T^{-1}).$$

Note that both functionals accumulate the stretch of the *inverse* atomic maps $f|_T^{-1}$ that map from the parameter to the surface triangle. Sander et al. minimise these non-quadratic functionals with an iterative method similar to the one described in Sect. 7.1. Like the MIPS method, both stretch functionals always yield a one-to-one mapping. Numerical examples showing comparisons with discrete harmonic maps are given in [62].

9 Parameterization Methods for Closed Surfaces

9.1 Surfaces with Genus Zero

There has been a lot of interest in spherical parameterization recently and in this section we will briefly summarise recent work. Many of the methods

attempt to mimic conformal (or harmonic) maps and are very similar to those for mapping disk-like surfaces into the plane, although some of the *linear methods* now become *non-linear*.

An important point is that, according to Gu and Yau [27], harmonic maps from a closed genus zero surface to the unit sphere are conformal, i.e., harmonic and conformal maps are *the same* when we deal with (closed) sphere-like surfaces. Intuitively, this follows from the fact that the domain and image have no boundary, and it is exactly the boundary map that makes the difference between a conformal and a harmonic map in the planar case. According to Gu and Yau there are essentially only six “degrees of freedom” (the Möbius transformations) in a spherical conformal map, three of which are rotations, the others involving some kind of area distortion (angles are of course preserved by definition).

The method of Haker et al. [32] first maps the given sphere-like surface S_T into the plane and then uses stereographic projection (itself a conformal map) to subsequently map to the sphere. The planar mapping part of this construction appears to reduce to the usual discrete harmonic map described in Sect. 6.1. Unfortunately, it is not clear in [32] how the surface is split or cut to allow for a mapping into the plane and how the boundary condition is treated.

Gu and Yau [28] have later proposed an iterative method which approximates a harmonic (and therefore conformal) map and avoids splitting. Specifically, a harmonic map from a closed surface S to the unit sphere S^* is a map $f : S \rightarrow S^*$ such that at every point p of S , the vector $\Delta_S f(p) \in \mathbb{R}^3$ is perpendicular to the tangent plane of S^* at $f(p)$. In the discrete case we consider piecewise linear mappings $f : S_T \rightarrow \mathbb{R}^3$ over an approximative mesh S_T with the property that $f(v)$ lies on the unit sphere S^* for every vertex $v \in V$ of the mesh S_T . Gu and Yau propose approximating a harmonic (conformal) map in the following way. Let $\Pi_{v_i}(u)$ denote the perpendicular projection of any point u on the sphere S^* onto the tangent plane of S^* at v_i . Then they consider a map which solves the (non-linear) equations

$$\sum_{j \in N_i} w_{ij} (\Pi_{v_i}(f(v_j)) - f(v_i)) = 0, \quad v_i \in V,$$

where, as in the planar case (7), the coefficients w_{ij} are the weights of (8). Gu and Yau [28] give many nice numerical examples based on their method. However, numerical difficulties apparently arise when some of the weights w_{ij} are negative, and they propose editing the original surface mesh, so that all weights are positive, though no procedure for doing this is given.

One might expect that a piecewise linear map should be one-to-one if all the weights are positive. Gotsman, Gu, and Sheffer have dealt with this issue in [24]. They work with the alternative equations

$$\sum_{j \in N_i} w_{ij} f(v_j) = \lambda_i f(v_i), \quad v_i \in V.$$

This equation says that a certain (positive) linear combination of the neighbouring vectors $f(v_j)$ must be parallel to the unit vector $f(v_i)$, and the factor $\lambda_i > 1$ is to be determined. Such a mapping is a spherical barycentric (or convex combination) mapping. When the weights w_{ij} are constant with respect to j we get an analogue of Tutte's barycentric mapping into the plane. A theorem by Colin de Verdière, described in [24], guarantees a valid embedding into the sphere if certain conditions on the eigenvalues of the matrix formed by the left hand sides of the equation hold. Unfortunately, it is currently not known how to guarantee these conditions and examples of simple meshes can be constructed for which there are several possible barycentric mappings, some of which are not one-to-one. However, the paper by Gotsman, Gu, and Sheffer looks like a good start-point for future work in this direction.

The angle-based method of Sheffer and de Sturler [69] has been generalised in a straightforward manner to the spherical case by Sheffer, Gotsman, and Dyn [71] using a combination of angle and area distortion. The stretch metric approach of Sander et al. [62] has been generalised to the spherical case by Praun and Hoppe [58].

9.2 Surfaces with Arbitrary Genus

A well-known approach to parameterizing (mesh) surfaces of arbitrary genus over simpler surfaces of the same genus is to somehow segment the mesh into disk-like patches and then map each patch into the plane. Usually, triangular-shaped patches are constructed and each patch is mapped to a triangle of a so-called base mesh.

The challenge of this approach is to obtain mappings that are smooth across the patch boundaries and the first methods [12, 42, 59, 30, 21] suffered indeed from this problem. But recently Khodakovsky, Litke, and Schröder [38] and Gu and Yau [28] proposed two different methods to compute parameterizations that are globally smooth with singularities occurring at only a few extraordinary vertices.

10 Conclusion

We have summarised, as best we can, both early and recent advances in the topic of surface parameterization. In addition to the 35 papers we have mentioned earlier in the text, we have added to the reference list a further 43 references to papers on surface parameterization, giving a total of 78 papers.

We feel it fair to say that the topic, as we know it now, began with the 1995 paper on the discrete harmonic map by Eck et al. [12], though essentially the same method was proposed in 1993 by Pinkall and Polthier [56] for computing minimal surfaces. During the period 1995–2000, we know of 19 published papers on surface parameterization, many of which we summarised in [20]. In contrast, we know of 49 papers on this topic which have been published during

the period 2001–2003; see Fig. 12. Thus there has clearly been a significant increase in research activity in this area in the last three years. A strong focus among these recent papers has been on methods which automatically find the boundary mapping, and methods for spherical parameterizations and other topologies. These two latter topics look likely to receive further attention in the future.

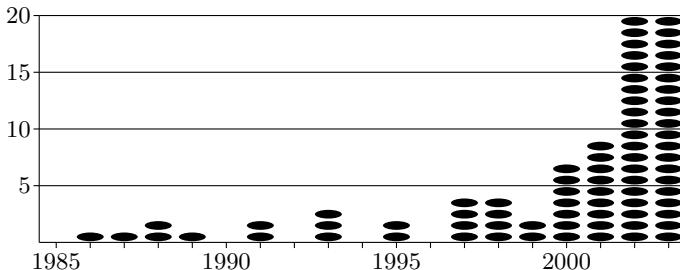


Fig. 12. Number of papers on surface parameterization per year (1985–2003).

Acknowledgements

We would like to thank several people for their helpful suggestions and feedback, especially Hugues Hoppe, Xianfeng Gu, Alla Sheffer, Craig Gotsman, Vitaly Surazhsky, and the referees. This work was supported in part by the European Union research project “Multiresolution in Geometric Modelling (MINGLE)” under grant HPRN-CT-1999-00117.

References

Papers on Surface Parameterization

1. B. Aksoylu, A. Khodakovsky, and P. Schröder. Multilevel solvers for unstructured surface meshes. Preprint, 2003.
2. N. Arad and G. Elber. Isometric texture mapping for free-form surfaces. *Computer Graphics Forum*, 16(5):247–256, 1997.
3. L. Balmelli, G. Taubin, and F. Bernardini. Space-optimized texture maps. *Computer Graphics Forum*, 21(3):411–420, 2002. Proceedings of Eurographics 2002.
4. C. Bennis, J.-M. Vézien, and G. Iglesias. Piecewise surface flattening for non-distorted texture mapping. *Proc. ACM SIGGRAPH '91*, 25(4):237–246, 1991.
5. E. Bier and K. Sloan. Two-part texture mappings. *IEEE Computer Graphics and Applications*, 6(9):40–53, 1986.
6. M. M. F. Yuen C. C. L. Wang, S. S.-F. Smith. Surface flattening based on energy model. *Computer-Aided Design*, 34(11):823–833, 2002.

7. S. Campagna and H.-P. Seidel. Parameterizing meshes with arbitrary topology. In *Proceedings of Image and Multidimensional Digital Signal Processing '98*, pages 287–290, 1998.
8. N. Carr and J. Hart. Meshed atlas for real time procedural solid texturing. *ACM Transactions on Graphics*, 21(2):106–131, 2002.
9. P. Degener, J. Meseth, and R. Klein. An adaptable surface parameterization method. In *Proceedings of the 12th International Meshing Roundtable*, pages 227–237, 2003.
10. M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21(3):209–218, 2002. Proceedings of Eurographics 2002.
11. T. Duchamp, A. Certain, T. DeRose, and W. Stuetzle. Hierarchical computation of PL harmonic embeddings. Technical report, University of Washington, July 1997.
12. M. Eck, T. D. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proc. ACM SIGGRAPH '95*, pages 173–182, 1995.
13. E. Fiume, A. Fournier, and V. Canale. Conformal texture mapping. In *Proceedings of Eurographics '87*, pages 53–64, 1987.
14. M. S. Floater. Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.
15. M. S. Floater. Parametric tilings and scattered data approximation. *International Journal of Shape Modeling*, 4(3,4):165–182, 1998.
16. M. S. Floater. Meshless parameterization and B-spline surface approximation. In R. Cipolla and R. Martin (eds.), *The Mathematics of Surfaces IX*, pages 1–18, London, 2000. Springer.
17. M. S. Floater. Convex combination maps. In J. Levesley, I. J. Anderson, and J. C. Mason (eds.), *Algorithms for Approximation IV*, pages 18–23, 2002.
18. M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.
19. M. S. Floater. One-to-one piecewise linear mappings over triangulations. *Mathematics of Computation*, 72(242):685–696, 2003.
20. M. S. Floater and K. Hormann. Parameterization of triangulations and unorganized points. In A. Iske, E. Quak, and M. S. Floater (eds.), *Tutorials on Multiresolution in Geometric Modelling*, Mathematics and Visualization, pages 287–316. Springer, Berlin, Heidelberg, 2002.
21. M. S. Floater, K. Hormann, and M. Reimers. Parameterization of manifold triangulations. In C. K. Chui, L. L. Schumaker, and J. Stöckler (eds.), *Approximation Theory X: Abstract and Classical Analysis*, Innovations in Applied Mathematics, pages 197–209. Vanderbilt University Press, Nashville, 2002.
22. M. S. Floater and M. Reimers. Meshless parameterization and surface reconstruction. *Computer Aided Geometric Design*, 18(2):77–92, 2001.
23. V. A. Garanzha. Maximum norm optimization of quasi-isometric mappings. *Numerical Linear Algebra with Applications*, 9(6,7):493–510, 2002.
24. C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3D meshes. *ACM Transactions on Graphics*, 22(3):358–363, 2003. Proc. ACM SIGGRAPH 2003.
25. G. Greiner and K. Hormann. Interpolating and approximating scattered 3D-data with hierarchical tensor product B-splines. In A. Le Méhauté, C. Rabut,

- and L. L. Schumaker (eds.), *Surface Fitting and Multiresolution Methods*, Innovations in Applied Mathematics, pages 163–172. Vanderbilt University Press, Nashville, 1997.
26. X. Gu, S. Gortler, and H. Hoppe. Geometry images. *ACM Transactions on Graphics*, 21(3):355–361, 2002. Proc. ACM SIGGRAPH 2002.
 27. X. Gu and S.-T. Yau. Computing conformal structures of surfaces. *Communications in Information and Systems*, 2(2):121–146, 2002.
 28. X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Proceedings of the 1st Symposium on Geometry Processing*, pages 127–137, 2003.
 29. I. Guskov. An anisotropic mesh parameterization scheme. In *Proceedings of the 11th International Meshing Roundtable*, pages 325–332, 2002.
 30. I. Guskov, A. Khodakovsky, P. Schröder, and W. Sweldens. Hybrid meshes: Multiresolution using regular and irregular refinement. In *Proceedings of the 18th Annual Symposium on Computational Geometry*, pages 264–272, 2002.
 31. I. Guskov, K. Vidimče, W. Sweldens, and P. Schröder. Normal meshes. In *Proc. ACM SIGGRAPH 2000*, pages 95–102, 2000.
 32. S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):181–189, 2000.
 33. K. Hormann. *Theory and Applications of Parameterizing Triangulations*. PhD thesis, Department of Computer Science, University of Erlangen, November 2001.
 34. K. Hormann and G. Greiner. MIPS: An efficient global parametrization method. In P.-J. Laurent, P. Sablonnière, and L. L. Schumaker (eds.), *Curve and Surface Design: Saint-Malo 1999*, Innovations in Applied Mathematics, pages 153–162. Vanderbilt University Press, Nashville, 2000.
 35. K. Hormann, G. Greiner, and S. Campagna. Hierarchical parametrization of triangulated surfaces. In *Proceedings of Vision, Modeling, and Visualization 1999*, pages 219–226, 1999.
 36. K. Hormann, U. Labsik, and G. Greiner. Remeshing triangulated surfaces with optimal parametrizations. *Computer-Aided Design*, 33(11):779–788, 2001.
 37. K. Hormann and M. Reimers. Triangulating point clouds with spherical topology. In T. Lyche, M.-L. Mazure, and L. L. Schumaker (eds.), *Curve and Surface Design: Saint-Malo 2002*, Modern Methods in Applied Mathematics, pages 215–224. Nashboro Press, Brentwood, 2003.
 38. A. Khodakovsky, N. Litke, and P. Schröder. Globally smooth parameterizations with low distortion. *ACM Transactions on Graphics*, 22(3):350–357, 2003. Proc. ACM SIGGRAPH 2003.
 39. S. Kolmanič and N. Guid. The flattening of arbitrary surfaces by approximation with developable stripes. In U. Cugini and M. J. Wozny (eds.), *From geometric modeling to shape modeling*, volume 80 of *International Federation for Information Processing*, pages 35–46. Kluwer Academic Publishers, Boston, 2001.
 40. G. Kós and T. Várady. Parameterizing complex triangular meshes. In T. Lyche, M.-L. Mazure, and L. L. Schumaker (eds.), *Curve and Surface Design: Saint-Malo 2002*, Modern Methods in Applied Mathematics, pages 265–274. Nashboro Press, Brentwood, TN, 2003.
 41. V. Kraevoy, A. Sheffer, and C. Gotsman. Matchmaker: constructing constrained texture maps. *ACM Transactions on Graphics*, 22(3):326–333, 2003. Proc. ACM SIGGRAPH 2003.

42. A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In *Proc. ACM SIGGRAPH '98*, pages 95–104, 1998.
43. Y. Lee, H. S. Kim, and S. Lee. Mesh parameterization with a virtual boundary. *Computers & Graphics*, 26(5):677–686, 2002.
44. B. Lévy. Constrained texture mapping for polygonal meshes. In *Proc. ACM SIGGRAPH 2001*, pages 417–424, 2001.
45. B. Lévy. Dual domain extrapolation. *ACM Transactions on Graphics*, 22(3):364–369, 2003. Proc ACM SIGGRAPH 2003.
46. B. Lévy and J.-L. Mallet. Non-distorted texture mapping for sheared triangulated meshes. In *Proc ACM SIGGRAPH '98*, pages 343–352, 1998.
47. B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics*, 21(3):362–371, 2002. Proc ACM SIGGRAPH 2002.
48. J. Liesen, E. de Sturler, A. Sheffer, Y. Aydin, and C. Siebert. Preconditioners for indefinite linear systems arising in surface parameterization. In *Proceedings of the 9th International Meshing Roundtable*, pages 71–82, 2001.
49. F. Losasso, H. Hoppe, S. Schaefer, and J. Warren. Smooth geometry images. In *Proceedings of the 1st Symposium on Geometry Processing*, pages 138–145, 2003.
50. S. D. Ma and H. Lin. Optimal texture mapping. In *Proceedings of Eurographics '88*, pages 421–428, 1988.
51. W. Ma and J. P. Kruth. Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces. *Computer-Aided Design*, 27(9):663–675, 1995.
52. J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. In *Proc ACM SIGGRAPH '93*, pages 27–34, 1993.
53. J. McCartney, B. K. Hinds, and B. L. Seow. The flattening of triangulated surfaces incorporating darts and gussets. *Computer-Aided Design*, 31(4):249–260, 1999.
54. J. McCartney, B. K. Hinds, and B. L. Seow. On using planar developments to perform texture mapping on arbitrarily curved surfaces. *Computers & Graphics*, 24(4):539–554, 2000.
55. L. Parida and S. P. Mudur. Constraint-satisfying planar development of complex surfaces. *Computer-Aided Design*, 25(4):225–232, 1993.
56. U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.
57. E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *Proc ACM SIGGRAPH 2000*, pages 465–470, 2000.
58. E. Praun and H. Hoppe. Spherical parametrization and remeshing. *ACM Transactions on Graphics*, 22(3):340–349, 2003. Proc ACM SIGGRAPH 2003.
59. E. Praun, W. Sweldens, and P. Schröder. Consistent mesh parameterizations. In *Proc ACM SIGGRAPH 2001*, pages 179–184, 2001.
60. N. Ray and B. Lévy. Hierarchical least squares conformal maps. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, pages 263–270, 2003.
61. P. Sander, Z. Wood, S. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *Proceedings of the 1st Symposium on Geometry Processing*, pages 138–145, 2003.

62. P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *Proc ACM SIGGRAPH 2001*, pages 409–416, 2001.
63. E. L. Schwartz, A. Shaw, and E. Wolfson. Applications of computer graphics and image processing to 2D and 3D modeling of the functional architecture of visual cortex. *IEEE Computer Graphics and Applications*, 8(4):13–23, 1988.
64. E. L. Schwartz, A. Shaw, and E. Wolfson. A numerical solution to the generalized mapmaker’s problem: flattening nonconvex polyhedral surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(9):1005–1008, 1989.
65. A. Sheffer. Spanning tree seams for reducing parameterization distortion of triangulated surfaces. In *Proceedings of Shape Modeling International*, pages 61–66, 2002.
66. A. Sheffer. Non-optimal parameterization and user control. In T. Lyche, M.-L. Mazure, and L. L. Schumaker (eds.), *Curve and Surface Design: Saint-Malo 2002*, Modern Methods in Applied Mathematics, pages 355–364. Nashboro Press, Brentwood, 2003.
67. A. Sheffer. Skinning 3D meshes. *Graphical Models*, 65(5):274–285, 2003.
68. A. Sheffer and E. de Sturler. Surface parameterization for meshing by triangulation flattening. In *Proceedings of the 9th International Meshing Roundtable*, pages 161–172, 2000.
69. A. Sheffer and E. de Sturler. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers*, 17(3):326–337, 2001.
70. A. Sheffer and E. de Sturler. Smoothing an overlay grid to minimize linear distortion in texture mapping. *ACM Transactions on Graphics*, 21(4):874–890, 2002.
71. A. Sheffer, C. Gotsman, and N. Dyn. Robust spherical parametrization of triangular meshes. In *Proceedings of the 4th Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics*, pages 94–99, 2003.
72. A. Sheffer and J. Hart. Seamster: inconspicuous low-distortion texture seam layout. In *Proceedings of IEEE Visualization 2002*, pages 291–298, 2002.
73. T. Shimada and Y. Tada. Approximate transformation of an arbitrary curved surface into a plane using dynamic programming. *Computer-Aided Design*, 23(2):153–159, 1991.
74. C. Soler, M.-P. Cani, and A. Angelidis. Hierarchical pattern mapping. *ACM Transactions on Graphics*, 21(3):673–680, 2002. Proc ACM SIGGRAPH 2002.
75. O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski. Bounded-distortion piecewise mesh parameterization. In *Proceedings of IEEE Visualization 2002*, pages 355–362, 2002.
76. V. Surazhsky and C. Gotsman. Explicit surface remeshing. In *Proceedings of the 1st Symposium on Geometry Processing*, pages 20–30, 2003.
77. R. Zayer, C. Rössl, and H.-P. Seidel. Variations on angle based flattening. In N. A. Dodgson, M. S. Floater, and M. A. Sabin (eds.), *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 187–199 (this book), Springer, Berlin, 2004.
78. G. Zigelman, R. Kimmel, and N. Kiryati. Texture mapping using surface flattening via multi-dimensional scaling. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):198–207, 2002.

Other References

79. S. C. Brenner and L. R. Scott. *The mathematical theory of finite element methods*, volume 15 of *Texts in Applied Mathematics*. Springer, second edition, 2002.
80. G. Choquet. Sur un type de transformation analytique généralisant la représentation conforme et définie au moyen de fonctions harmoniques. *Bulletin des Sciences Mathématiques*, 69:156–165, 1945.
81. C. F. Gauß. Disquisitiones generales circa superficies curvas. *Commentationes Societatis Regiae Scientiarum Gottingensis Recentiores*, 6:99–146, 1827.
82. G. Greiner. Variational design and fairing of spline surfaces. *Computer Graphics Forum*, 13(3):143–154, 1994. Proceedings of Eurographics ’94.
83. W. Klingenberg. *A Course in Differential Geometry*. Springer, Berlin, Heidelberg, 1978.
84. H. Kneser. Lösung der Aufgabe 41. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 35:123–124, 1926.
85. E. Kreyszig. *Differential Geometry*. Dover, New York, 1991.
86. J. H. Lambert. *Beyträge zum Gebrauche der Mathematik und deren Anwendung*, Band 3. Buchhandlung der Realschule, Berlin, 1772.
87. G. Mercator. Nova et aucta orbis terrae descriptio ad usum navigantium emendate accommodata. Duisburg, 1569.
88. M. Meyer, H. Lee, A. H. Barr, and M. Desbrun. Generalized barycentric coordinates for irregular polygons. *Journal of Graphics Tools*, 7(1):13–22, 2002.
89. C. Ptolemy. *The Geography*. Dover, 1991. Translated by E. L. Stevenson.
90. T. Radó. Aufgabe 41. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 35:49, 1926.
91. B. Riemann. *Grundlagen für eine allgemeine Theorie der Functionen einer veränderlichen complexen Größe*. PhD thesis, Universität Göttingen, 1851.
92. W. T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 13:743–768, 1963.
93. E. L. Wachspress. *A Rational Finite Element Basis*. Academic Press, New York, 1975.
94. J. Warren. Barycentric coordinates for convex polytopes. *Advances in Computational Mathematics*, 6(2):97–108, 1996.

Variations on Angle Based Flattening

Rhaleb Zayer, Christian Rössl, and Hans-Peter Seidel

Max-Planck-Institut für Informatik, Saarbrücken, Germany
`{zayer|roessl|hpseidel}@mpi-sb.mpg.de`

Summary. *Angle Based Flattening* is a robust parameterization technique allowing a free boundary. The numerical optimisation associated with the approach yields a challenging problem. We discuss several approaches to effectively reduce the computational effort involved and propose appropriate numerical solvers. We propose a simple but effective transformation of the problem which reduces the computational cost and simplifies the implementation. We also show that fast convergence can be achieved by finding approximate solutions which yield a low angular distortion.

1 Introduction

Surface parameterization is a fundamental problem in computer graphics. Intuitively, we can think of it as the *flattening* of a surface to a valid planar configuration, i.e. one without foldovers or self-intersections. More formally, consider a surface that is homeomorphic to a disk. Then the goal is to find a bijective mapping from a surface to the parameter domain, that fulfils certain quality constraints. For a triangulated surface this is a piecewise linear mapping between the original and an isomorphic planar mesh.

The importance of the problem makes surface parameterization a very active field of research, see [9] for an extensive recent survey. Numerous approaches have been proposed so far, inspired by results from different areas of research. Tutte [23] starts from graph theory and uses barycentric maps for embedding a planar graph. The shape-preserving weights [6] improve on the conformality of the mapping while still guaranteeing bijectivity. Most recently, Floater presented the mean value coordinates [8] as an alternative. Eck et al. [5] use discrete harmonic maps to minimise angular distortion. Sander et al. [19] introduce a stretch metric to reduce the distortion induced by the parameterization. All the above methods require a predefined convex boundary in the parameter domain. Hormann and Greiner construct a most-isometric parameterization [14] by minimising a non-linear deformation functional without needing to fix the boundary. Desbrun et al. [3] and Levy et al. [15] achieve

quasi-conformal mappings with an evolving boundary by solving linear systems based on the Cauchy-Riemann equation and harmonic energy minimisation respectively. Other recent approaches apply multi-dimensional scaling [24] or an iterative algorithm that locally flattens the triangulation until a prescribed distortion bound is reached [22].

While quasi-conformal parameterizations such as [3, 5, 13, 15, 18] propose several schemes to minimise angular distortion, it seems natural to formulate the problem in terms of interior angles of the flat mesh. This leads to the *Angle Based Flattening* (ABF) method introduced by Sheffer and de Sturler [20]. The ABF algorithm constructs such a parameterization by minimising a functional that punishes the angular distortion of the planar mesh with respect to the angles of the original mesh. A set of linear and non-linear equality constraints on the planar angles guarantees the validity of the parameterization. These constraints however do not prevent the boundary from self-intersection. Hence a post-processing of the flat mesh is needed to handle edge crossings at the boundary. Each post-processing step first identifies the nodes causing intersections in the flat mesh, then it adds constraints on the local configurations in order to avoid intersections. The flat mesh is recomputed as a solution of the updated nonlinear system. The post-processing algorithm is repeated until no more intersections are found.

2 Overview

In this paper, we discuss several approaches to effectively reduce the computational effort involved in *Angle Based Flattening* and discuss algorithms to efficiently solve the parameterization problem. The complexity of the constrained optimisation problem raised in the ABF method makes finding a solution in reasonable time a very challenging problem. Several numerical schemes have been proposed to speed up the convergence of the original algorithm [20] by using preconditioning [16] and smoothing [21]. We take a completely different approach by identifying the main reasons that hinder convergence within the setting of the constrained problem itself. In fact, the post-processing might be expensive as it iteratively tries to find intersections and to then solve the whole non linear system as many times as needed. We take advantage of a characterisation of convex planar drawings of tri-connected graphs to eliminate boundary intersections in the first place. This way we can steer or even avoid post-processing. Having this characterisation in hand, it can be used in association with different objective functions that reflect the criteria we would like to minimise. Such functions can be described as the angular distortion [20] or the MIPS energy introduced by Hormann and Greiner [14] as both can be expressed completely in terms of angles.

The non-linear equations in the ABF method lead to a dense sparsity pattern of the Hessian matrix of the system which increases the computational cost. We show how the convergence can be improved alternatively by a simple

yet effective transformation of the problem that relaxes the non-linear equality constraints. In fact, the Hessian becomes diagonal and its sparsity pattern becomes independent of the valences of the vertices of the input mesh. Since the system of equations is symmetric, we opt for the more appropriate symmetric numerical solvers instead of the non-symmetric ones proposed in [20, 21, 16]. We propose a practical approach that achieves fast convergence by finding approximate solutions which yield a low angular distortion.

3 Conventions

Throughout the paper, we try to restrict ourselves to the essential amount of formalism only, where the following notations are used:

- N is the total number of interior mesh angles.
- α_i^* ($i = 1, \dots, N$) denote the angles of the *original* mesh,
- α_i are the corresponding angles of the *flat* mesh. As these are the variables of the optimisation problem, the more usual notation x_i is used alternatively when appropriate.
- v denotes the central vertex in a centred drawing of a *wheel*, i.e. of its 1-neighbourhood. d is the number of direct neighbours of v or its *valence*. α_j ($j = 1, \dots, d$) refer to the angles at v , while β_j and γ_j denote the opposite left and right angles of a face with central angle α_j , respectively. All faces are oriented counter-clockwise.
- Variables and functions without subscripts may refer to multivariate vectors as explained by the context.

4 Characterisation of Drawings of Planar Graphs

Sheffer and de Sturler [20] addressed the problem of the validity of the planar embedding by requiring the following consistency condition on the set of positive angles of the planar mesh:

- *Vertex consistency*

For each internal vertex v , with central angles $\alpha_1, \dots, \alpha_d$:

$$\sum_{i=1}^d \alpha_i - 2\pi = 0 \quad (1)$$

- *Triangle consistency*

For each triangular face with angles α, β, γ the face consistency:

$$\alpha + \beta + \gamma - \pi = 0 \quad (2)$$

- *Wheel consistency*

For each internal vertex v with left angles β_1, \dots, β_d and right angles $\gamma_1, \dots, \gamma_d$:

$$\prod_{i=1}^d \frac{\sin(\beta_i)}{\sin(\gamma_i)} = 1 \quad (3)$$

These conditions guarantee the centred embedding of internal vertices without overlapping of interior edges. However they do not prevent the overlapping of boundary edges. This issue is a well-studied problem in graph theory [4, 11]. Di Battista and Vismara provide a characterisation of the convex planar straight line drawing of a tri-connected graph for a given set of positive angles [4]. Their minimal constraints for the planarity of the graph impose in addition to (1), (2), (3) the following condition:

- *Convex external face condition*

For each external vertex v , with internal angles $\alpha_1, \dots, \alpha_d$:

$$\sum_{i=1}^d \alpha_i \leq \pi \quad (4)$$

Condition (4) guarantees the *convexity of the boundary* and hence prevents boundary overlapping. Note that the inequality (4) prevents local and global self-intersection simultaneously. So it does not only prevent adjacent boundary edges from overlapping, but it also guarantees that the boundary loop as a whole does not cross itself. For the local configuration it would in fact be sufficient to require the following weakened condition to hold:

- *Adjacent boundary edges consistency*

For each external vertex v , with internal angles $\alpha_1, \dots, \alpha_d$:

$$\sum_{i=1}^d \alpha_i \leq 2\pi. \quad (5)$$

This prevents adjacent boundary triangles from crossing each other. However, condition (5) is not strong enough to globally enforce a valid mesh with no boundary intersections as shown in Fig. 1 (b).

To get a better understanding and better control of the boundary behaviour, we propose multiplying the right hand side of (4) by a positive scalar t , formally

$$\sum_{i=1}^d \alpha_i \leq t\pi. \quad (6)$$

The scalar t can be interpreted as a *boundary control coefficient* that steers the convexity of the boundary. A lower bound for this factor can be derived

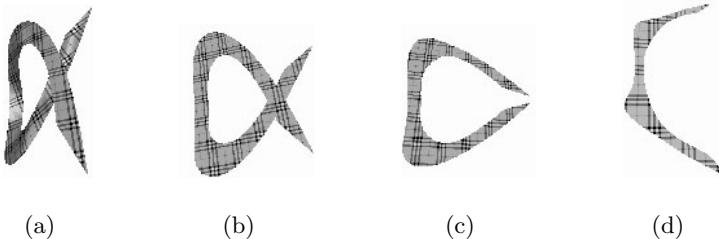


Fig. 1. [Reproduced in colour in Plate 14.] Flattening an α -shaped model: (a) Original mesh. (b) The flattened mesh with boundary control coefficient $t = 2$, (c) $t = 1.1$, (d) $t = 1.03$. (The views are scaled differently.)

using discrete curvature measure. Consider the angular defect of the flat mesh which is expressed as

$$\sum_{v=1}^n (\pi - A_v) = 2\pi, \quad (7)$$

where A_v is the sum of angles at vertex v and n is the number of boundary vertices. By a simple calculation, we establish the lower bound

$$t_0 = 1 - 2/n.$$

The trivial case is a single triangle, its angles cannot be all smaller than $\pi/3$.

We experimented with different values of t , and summarise the following interpretations that can be used as reference for choosing appropriate values of t :

- $t > 2$ results in the classic ABF method without preconditioning. No adjacent edge overlapping or boundary self crossing is taken into consideration.
- $1 < t \leq 2$ prevents adjacent edges from overlapping, but does not necessarily prevent global self-intersections of the boundary loop. We experienced such cases only for “boundary-heavy” (w.r.t. the ratio of boundary to inner vertices, see e.g. Fig. 1) surfaces with non-trivial geometry.
- $t = 1$ globally prevents the boundary loop from self-intersection for any valid input mesh, note that this suffices to induce a convex boundary.
- $t_0 < t < 1$ forces the boundary to become concave.

Figs. 1 and 2 illustrate the behaviour of the boundary for different values of t . We can take advantage of these facts in order to avoid an iterative post-processing and thus have better control over the convergence of the constrained optimisation problem. In the next sections we show how this problem with the additional inequalities included can be solved efficiently.

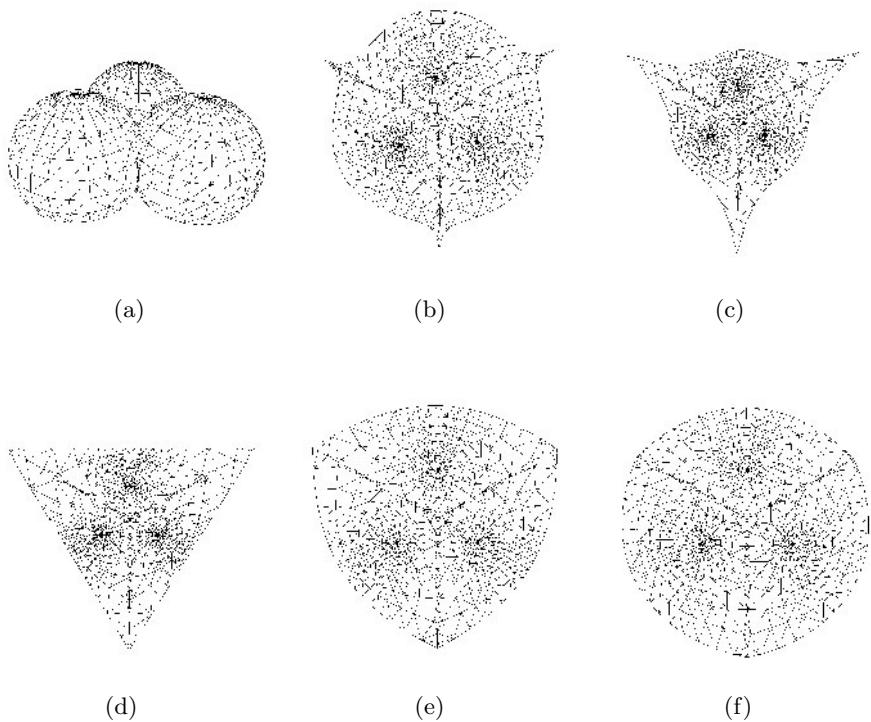


Fig. 2. Effect of the boundary control coefficient t on the *3-balls* model. Note that the ABF algorithms preserve the symmetry. (a) Original mesh. (b) Flat mesh for $t \geq 2$ (ABF). (c) $t = 1.05$, (d) $t = 1$ (convex boundary ABF), (e) $t = 0.98$, (f) $t = 0.968$.

5 Constrained Optimisation Problem

A general approach to establish a surface parameterization is to minimise an objective function $f(x)$ that quantifies distortion with respect to certain quality criteria. As the validity of the flat mesh is guaranteed by the angle constraints of Sect. 4, a typical choice of such a function would be based on angles. Examples of such objective functions are the angular distortion [20]

$$f(x) = \sum_{i=1}^N w_i (x_i - a_i)^2$$

with the weights $w_i = \frac{1}{a_i^2}$. The variables a_i represent the *optimal angles* of the flat mesh, which are

$$a_i = \begin{cases} \alpha_i^* \frac{2\pi}{\sum_{i=1}^d \alpha_i^*} & \text{around an interior vertex,} \\ \alpha_i^* & \text{around a boundary vertex.} \end{cases}$$

We can now formulate the optimisation problem as

$$\text{minimise } f(x) \text{ subject to } h(x) = 0, g(x) \leq 0, \quad (8)$$

where g and h are multivariate functions of the equality (1), (2), (3) and the inequality constraints (6) respectively. In the next section we will replace (3) by a modified equation (10).

6 Solving the Optimisation Problem

Large constrained optimisation systems of the form (8) are still open problems in the field of non-linear optimisation [2]. The adequacy of a minimisation method depends on the properties of the objective function as well as on the constraints.

In order to solve the optimisation problem we use the method of Lagrange multipliers as it guarantees the exact satisfaction of constraints. We handle the inequality constraints by means of the so called *active set* approach, a variant of Newton-like methods. It transforms inequalities to equalities which are generally easier to handle.

The active set is defined as the set of indices for which the inequality constraint (4) is active. Formally

$$A(x, \mu) = \{i | g_i \geq -\frac{\mu_i}{c}, i = 1, \dots, r\}$$

where μ_i is the Lagrange multiplier associated with g_i , and c is a fixed positive scalar.

The active set approach converts inequality constraints to equality constraints by altering the Lagrange multipliers associated with them. If a constraint does not figure in the active set, its associated multipliers are set to zero. Otherwise it is treated as an equality constraint. The numerical advantage of this method is that as the iterates get closer to the solution, the active set becomes more and more stable. A detailed description of the active set method can be found in [1].

In every Newton iteration the following system is solved

$$\begin{bmatrix} \nabla_{xx}^2 L & J_h^T & J_g^T \\ J_h & 0 & 0 \\ J_g & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \mu_h \\ \Delta \mu_g \end{bmatrix} = - \begin{bmatrix} \nabla_x L \\ h \\ g \end{bmatrix} \quad (9)$$

where the Lagrangian L is given by

$$L = f(x) + \mu_h^T h(x) + \mu_g^T g(x).$$

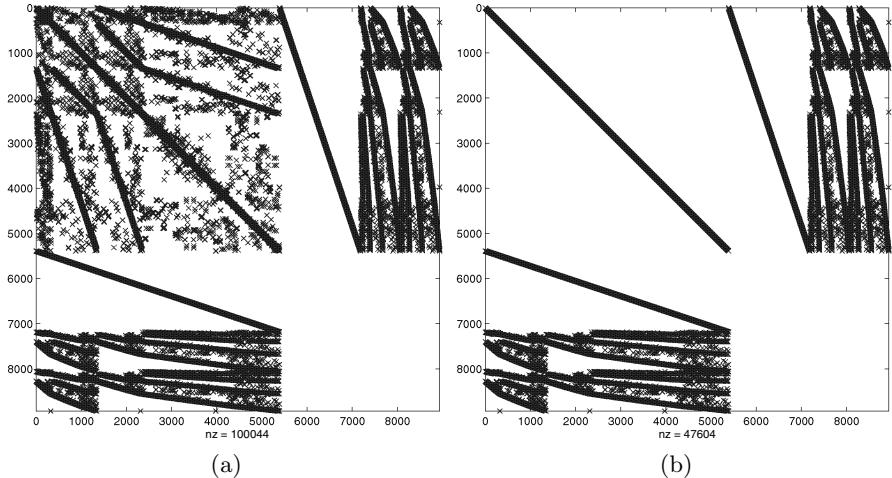


Fig. 3. System matrices of equation (9) generated from the *ear* model using (a) the original wheel condition (3), (b) the simplified wheel condition (10). The Diagonal Hessian brought the number of nonzero elements from 100044 down to 47607.

In the classic ABF algorithm, the computation of the Hessian matrix $\nabla_{xx}^2 L$ involves finding the second derivatives of the products involved in condition (3). The resulting matrix is sparse, but it still contains a considerable number of non-zero elements (cf. Fig. 3(a)). This number depends largely on the valences of the input mesh vertices.

Instead, we propose to use a *modified wheel condition* (10). Since the angles are strictly positive we can safely rewrite condition (3) as

$$\sum_{i=1}^d \log(\sin \beta_i) - \log(\sin \gamma_i) = 0 . \quad (10)$$

The virtue of this modification resides in the fact that it yields a diagonal Hessian matrix

$$\nabla_{xx}^2 L = \text{diag}(f''(x_i) + m_i \frac{-1}{\sin^2(x_i)})$$

where m_i is the linear combination of the Lagrange multipliers involved with x_i in condition (10). The amount of computation and effort by the iterative solvers is hence reduced considerably. The main reason for this is that the Hessian can be computed efficiently as this reduction also avoids the estimation of complex derivatives (which depend on the valences of vertices) with all the floating error they may induce. We note that consequently the implementation of the ABF method is simplified quite a bit. Fig. 3 illustrates the structure of a typical system matrix and the improvement induced by the *modified wheel condition*.

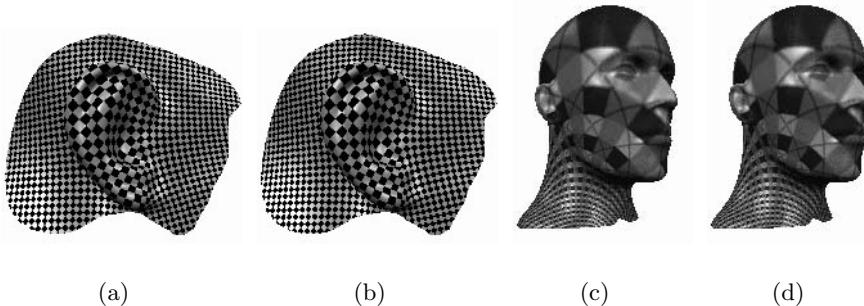


Fig. 4. [Reproduced in colour in Plate 15.] Comparison of under-determined (**a**, **c**) and minimisation (**b**, **d**) solutions. The parameterization of the *Ear* and the *Mannequin* models is visualised by mapping regular textures.

The system matrix is symmetric although not necessarily positive-definite, with the additional advantage of having a diagonal Hessian. We can exploit this structure by using adequate iterative solvers such as MINRES or SYMMLQ both developed by Paige and Saunders [17] for symmetric matrices, instead of the non-symmetric GMRES and BiCGStab that were used in [20, 16]. The latter solvers have higher cost per iteration and may suffer from breakdowns or simply stagnate while MINRES and SYMMLQ have relatively cheap cost per iteration, which is just 4 axpys (addition of a scalar multiple of a vector to a vector) higher than the iteration cost of the conjugate gradient method. Another alternative, which is relatively inexpensive, is the CGNR algorithm introduced by Hestens and Steifel [10]. In our case, there is no need to transpose the system matrix as it is symmetric. The cost per iteration is then just one matrix-vector multiplication higher than the cost of the conjugate gradient method. A comparison of the convergence of these iterative solvers for typical meshes is given in Sect. 8.

Note that the initial guess for the unknowns x is the set of the optimal angles $\{\alpha_i | i = 1, \dots, n\}$. Consequently, at every Newton iteration the solution stays within the positive domain. In order to guarantee that our algorithm does not step into the negative domain, we can apply a similar technique as in [20] that rejects negative iterates and appends increased weights to the corresponding angles. However, our experiments with different meshes show that we hardly ever run into this situation.

7 Practical Approach

In general, the method of Lagrange multipliers we use is a local optimisation method. This means that the provided solution is a local minimum which is largely dependent on the initial guess given by the user. In other words it is

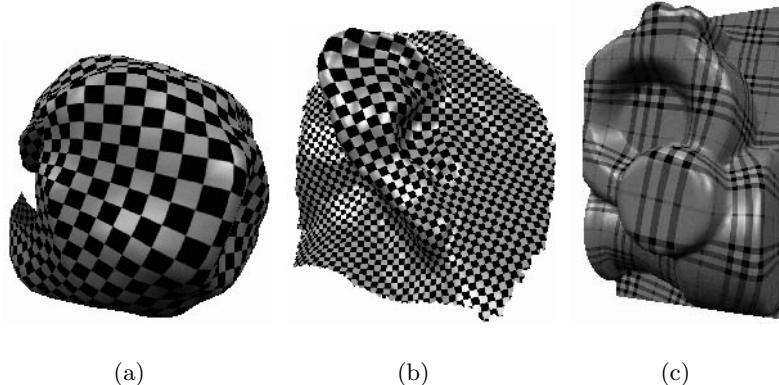


Fig. 5. [Reproduced in colour in Plate 16.] Visualisation of parameterizations from ABF by texture mapping different models. (a) *Clumpy*, (b) *Large ear*, (c) *Mechanical part*. Notice the quasi-conformality of the parameterization.

the closest minimum to the initial guess. Since the initial guess is very close to the solution, only few newton iterations are needed for convergence. Hence, we can assume that *any* feasible point that is close to the initial guess mentioned above gives a good estimate for the solution and would yield a low angular distortion.

With this consideration in mind, the problem can be restated as how to get a feasible point. The idea is to have a null objective function, i.e. $f = 0$. This means that we reduce the problem to solving the under-determined system of equality and inequality constraints. The modification leads to considerable speed-up of convergence as there is no extra load from the objective function. In the following, we call this solution the under-determined solution and the one using the angular distortion functional the minimisation solution (strictly speaking both solutions are just approximate solutions).

In practice, the difference between the minimisation and the under-determined solution is hardly noticeable. Fig. 4 shows a comparison between such solutions. Table 2 shows the numerical difference with respect to angular distortion between the two methods. The latter method seems to outperform the minimisation method as it converges much faster. Table 1 summarises the performance of both methods.

Our experiments with several different initial starting guesses indicate that if we are looking only for a valid mapping of the mesh to the plane, we can get a very fast feasible solution by setting the initial guess to zero or to $\frac{\pi}{3}$ and the objective function to zero. This solution does not reflect the geometry of the mesh and hence might not be suited for applications like texture mapping.

Another alternative method for finding a feasible point would be to use least squares methods for solving under-determined nonlinear problems. How-

ever, as these methods do not guarantee the exact satisfaction of the constraints for large problems, they generally fail to produce valid parameterizations.

8 Results and Discussion

We applied our algorithm to a set of different triangular meshes (see also Figs. 2, 4 and 5). Table 1 summarises the numerical results of our method, all timings were measured on a 1.7 GHz Intel Xeon CPU. The parameterization time depends on the number of triangles, on the geometry as well as on the connectivity of the input mesh. For consistency with the original ABF we use the same metrics as in [21] to measure angular distortion. The numerical data suggests that the under-determined method in association with the CGNR algorithm delivers high quality quasi-conformal parameterizations in very competitive time.

Table 1. Comparison of runtime (in seconds) of the minimisation and under-determined method using different iterative solvers.

| model | #Δ | CGNR | | SYMMLQ | | MINRES | |
|------------|-------|--------|--------|--------|--------|--------|--------|
| | | minim. | under. | minim. | under. | minim. | under. |
| 3 Balls | 1032 | 66 | 1 | 21 | 2 | 7 | 2 |
| Ear | 1796 | 159 | 3 | 44 | 9 | 11 | 7 |
| Mannequin | 5420 | > 900 | 26 | 292 | 69 | 126 | 56 |
| Mech. part | 7938 | > 999 | 86 | > 999 | 245 | > 999 | 192 |
| Large ear | 24914 | > 999 | 237 | > 999 | 654 | > 999 | 522 |

Table 2. Comparison of angular distortion induced by the minimisation and the under-determined method using different iterative solvers.

9 Conclusion

We presented and discussed several extensions to angle based flattening. With additional inequality constraints we can eliminate global and/or local boundary self-intersections. This leads to a nice interpretation of boundary behaviour through the introduction of the new boundary control coefficient. While its use initially targets the avoidance of iterative post-processing, we see potential use for optimising the parameterization with respect to this coefficient variable.

The arising non-linear constrained optimisation problem can be solved efficiently. With a simple and intuitive transformation we take advantage of a simply structured symmetric system matrix, enabling the application of robust iterative solvers. The use of our under-determined system solution leads to a relatively fast method for generating angle based parameterizations.

Acknowledgement

This work was supported in part by the European Union research project “Multiresolution in Geometric Modelling (MINGLE)” under grant HPRN-CT-1999-00117.

References

1. D.P. Bertsekas. *Constrained optimization and lagrange multiplier methods*. Athena Scientific, 1996.
2. R.H. Byrd and J. Nocedal. Active set and interior methods for nonlinear optimization *Doc. MATH*, Extra Volume ICM III, 1998, pp. 667–676.
3. M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of triangle meshes. *Proc. Eurographics 2002*, pp. 209–218.
4. G. Di Battista and L. Vismara. Angles of planar triangular graphs. *SIAM Journal on Discrete Mathematics*, 9 (3), 1996, pp. 349–359.
5. M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. *Proc. ACM SIGGRAPH ’95*, pp. 173–182.
6. M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Comp. Aided Geom. Design*, (14), 3, 1997, pp. 231–250.
7. M. S. Floater and K. Hormann. Parameterization of triangulations and unorganized points. *Tutorials on Multiresolution in Geometric Modelling* Springer-Verlag, Heidelberg (2002), pp. 287–315.
8. M. S. Floater. Mean value coordinates. *Comp. Aided Geom. Design*, (20), 1, 2003, pp. 19–27.
9. M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey, *Advances in Multiresolution for Geometric Modelling*, N. A. Dodgson, M. S. Floater, and M. A. Sabin (eds.), Springer, 2004, pp. 157–186 (this book).
10. M. R. Hestenes and E. Stiefel. Method of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand.* 49:409–436, 1952.

11. A. Garg. New results on drawing angle graphs. *Computational Geometry* (9), (1-2), 1998, pp. 43–82.
12. A. Greenbaum. *Iterative Methods for Solving Linear Systems* SIAM, Philadelphia, 1997.
13. S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6(2), 2000, pp. 181–189.
14. K. Hormann and G. Greiner. MIPS: an efficient global parametrization method. *Curve and Surface Design: Saint-Malo 1999*, 2000, pp. 153–162.
15. B. Levy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *Proc. ACM SIGGRAPH 2002*, pp. 362–371.
16. J. Liesen, E. de Sturler, A. Sheffer, Y. Aydin, and C. Siefert. Preconditioners for indefinite linear systems arising in surface parameterization. *Proceedings of the 10th International Meshing Round Table*, 2001, pp. 71–81.
17. C. Paige and M. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12, 1975, pp. 617–629.
18. U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(15), 1993, pp. 15–36.
19. P. Sander, J. Snyder, S. Gortler, and H. Hoppe. Texture mapping progressive meshes. *Proc. ACM SIGGRAPH 2001*, pp. 409–416.
20. A. Sheffer and E. de Sturler. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers*, 17 (3), 2001, pp. 326–337.
21. A. Sheffer and E. de Sturler. Smoothing an overlay grid to minimize linear distortion in texture mapping. *ACM Transactions on Graphics*, 21 (4), 2002.
22. O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski. Bounded-distortion piecewise mesh parameterization. *Proc. IEEE Visualization 2002*, pp. 355–362.
23. W. T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 13 (3), 1963, pp. 743–768.
24. G. Zigelmann, R. Kimmel, and N. Kiryati. Texture mapping using surface flattening via multi-dimensional scaling. *IEEE Transactions on Visualization and Computer Graphics*, 8(2), 2002.

Part V

— Subdivision

Recent Progress in Subdivision: a Survey

Malcolm Sabin

Computer Laboratory, University of Cambridge, UK
and

Numerical Geometry Ltd., Cambridge, UK
`mas33@cl.cam.ac.uk`

Summary. After briefly establishing the traditional concepts in subdivision surfaces, we survey the way in which the literature on this topic has burgeoned in the last five or six years, picking out new trends, ideas and issues which are becoming important.

Subdivision surfaces were first described by Catmull and Clark [6] in 1978, soon, in fact, after the now-ubiquitous NURBS were identified as being a sensible standard for parametric surface descriptions. For twenty years they were an interesting generalisation of (a subset of) NURBS, with a paper on one aspect or another appearing in some relevant journal every year or so.

In the last five or six years the situation has totally changed, as indicated by the numbers of papers published which relate to subdivision reasonably directly. These are plotted against date in Fig. 1. Subdivision surfaces are now one of the methods of choice in Computer Graphics, and some consider that they might succeed NURBS as the standard in engineering CAD.

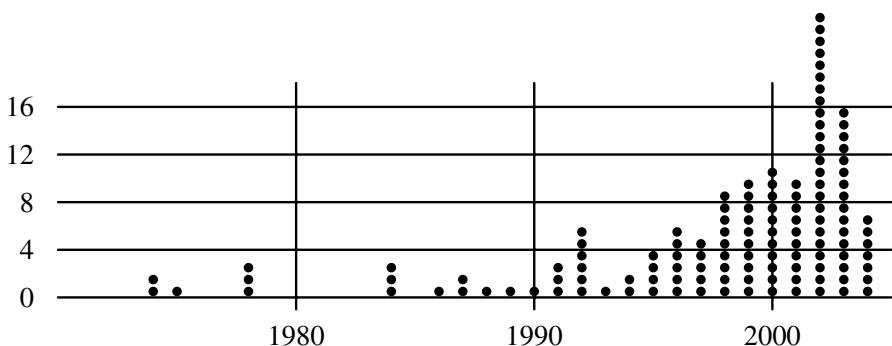


Fig. 1. Number of subdivision papers, plotted by year.

This paper looks at some of the technical changes that have happened in the last few years which are helping to drive that change in status. The paper is divided into the following aspects:

- Background – identifying the “classical” knowledge.
- New schemes and a classification.
- New domains and new ranges – a subdivision surface may be thought of as a map from a bivariate manifold into \mathbb{R}^3 . We can map from other domains, and we can map into other ranges.
- New issues – the traditional focus on smoothness has now been joined by other criteria for judging the quality of a scheme.
- New ideas – it has become clear that linear stationary schemes may not provide the solution to all our problems, and newer ideas broaden our horizons.

1 Introduction and Background

The beginnings of the subdivision story can be dated back to the papers of de Rahm [1], over fifty years ago, but the relevance to the modelling of shape started with the proposal of Chaikin [2], who devised a method of generating smooth curves for plotting. This was soon analysed by Forrest [3] and by Riesenfeld [4] and linked with the burgeoning theory of B-spline curves. It became clear that uniform B-spline curves of any degree would have such a subdivision construction.

The extension to surfaces took just a few years, until 1978, when Catmull and Clark [6] published their descriptions of both quadratic and cubic subdivision surfaces, the exciting new point being that a surface could be described which was not forced to have a regular rectangular grid in the way that the tensor product B-spline surfaces were. The definition of a specific surface in terms of a control mesh could follow the needs of the boundaries and the curvature of the surface. This was made possible by the extension of the subdivision rules to allow for ‘extraordinary points’, being either ‘extraordinary vertices’ where either other than four faces came together at a vertex, or else ‘extraordinary faces’ where a face had other than four sides.

At about the same time Doo [5] and Sabin, who had also been working on quadratic subdivision, showed a way of analysing the behaviour of these schemes at the extraordinary points, treating the refinement process in terms of matrix multiplication, and using eigenanalysis of the spectrum of this matrix [7]. This aspect was followed up by Ball and Storry [9, 12] who made this analysis process more formal and succeeded in making some improvements to the coefficients used around the extraordinary points in the Catmull-Clark scheme, so that radial curves through an extraordinary point matched curvatures there at equal distances on opposite sides. In his PhD dissertation [11], Storry identified that in the limit, the configuration around an extraordinary

point was always an affine transform (dependent on the original polyhedron) of a point distribution which was completely defined by the eigenvectors of the subdivision matrix. He called this the *natural configuration*.

The next two big ideas emerged in 1987. Loop, in his Masters' thesis [13], described a subdivision scheme defined over a grid of triangles. This not only gave a new domain over which subdivisions could be defined, but also showed that the eigenanalysis could be used explicitly in the original design of a scheme, in the choice of coefficients which should be used around extraordinary points.

The other significant publication that year was the description by Dyn, Levin and Gregory [14] of their four-point curve scheme. This was new in two ways: it was an interpolating scheme, rather than smoothing, and the limit curve did not consist of parametric polynomial pieces. The analysis of its continuity and differentiability therefore required new tools.

The first tool was provided in [14] and tools of a greater generality were provided in [19] and [20]. The method in the later paper together with the idea of the *symbol* of a subdivision scheme, presented in [19], was later expressed in terms of z -transforms [26], which turn convolution of sequences of numbers into multiplication of Laurent polynomials. Algebraic manipulation of these polynomials allows such processes as the taking of differences to be expressed very simply, and it has turned out that many of the arguments we need to deploy can be expressed very elegantly in this notation. It also provides sufficient conditions for a scheme to have a certain level of derivative continuity, whereas the eigenanalysis approach provides only necessary conditions.

The generalisation of the four-point ideas to an interpolating surface scheme came in 1990, with the description by Dyn, Levin and Gregory [17] of the butterfly scheme, an interpolating surface scheme defined over a triangular grid.

In 1995 Reif [34] showed that there was rather more to continuity than had been dreamt of. He identified that the natural configuration implies a parametrization of the rings of regular pieces which surround each extraordinary point and that it is essential, in order to obtain a scheme which generates well-behaved surfaces at the extraordinary points, to ensure that this parametrization is injective. (Later, Peters and Reif went further, and in [54] they constructed a scheme (a variant of the quadratic) for which the injectivity test fails, resulting in severe folding of the limit surface in every ring.)

The following year Reif [35] showed that the attempts to make a C2 variant of Catmull-Clark were not going to succeed, because a surface C2 at the extraordinary points would need to have regular pieces at least bi-sextic.

Thus as we passed the mid-1990s, subdivision theory stood like this:

- A surface subdivision scheme takes a manifold mesh of vertices joined by faces, usually called the polyhedron, and creates a new, finer, polyhedron by constructing new vertices as linear combinations of the old ones, in

groups defined by the connectivity of the polyhedron, and joining them up by new faces in a way related to the old connectivity.

The coefficients can be documented in diagrammatic form either in the *mask*, a diagram where the coefficients by which a given old vertex influences the surrounding new ones are laid out in the same pattern as those new vertices, or in the *stencils*, a set of diagrams where the coefficients by which nearby old vertices influence a given new one are laid out in the same pattern as those old vertices. These are totally equivalent. In the univariate case, they are the columns and rows, respectively, of the matrix by which the sequence of old vertices is multiplied to give the sequence of new ones.

- This refinement can be repeated as often as desired, and there are conditions on the scheme guaranteeing the existence of a well-defined limit surface to which the sequence of finer and finer polyhedra converges. During the refinement process the number of extraordinary points remains constant, and they become separated by regular mesh of a kind which is dependent on the topological rules of the scheme.
- The regular mesh is often well-described by box-spline theory (the butterfly scheme was almost alone in not being describable in those terms) but the z -transform analysis can always be applied to determine the smoothness of the limit surface in the regular regions. The extraordinary points are surrounded by rings of regular mesh, and close to the extraordinary point these are just affine transforms of the natural configuration, and can be parametrized by the characteristic map.
- Because every box-spline has a generating subdivision scheme [10], we had a way in principle of creating as many different subdivision schemes as we might want. Each such scheme would have to have its extraordinary point rules invented, of course, but nobody had bothered to go through the exercise. We also had a sequence of interpolating curve schemes, generated by letting an increasing number ($2n$) of points influence the new vertex in the middle of each span [16], but this had not led to a sequence of interpolating triangular surface schemes. In fact Catmull-Clark, Loop and butterfly were regarded as the significant surface schemes, and the cubic B-spline subdivision and the 4-point scheme as the significant curve schemes, any others being only of academic interest.
- The question of the behaviour of the limit surface in the immediate vicinity of the extraordinary points was still of interest. Indeed, the papers [7, 15, 18] before Reif's key result [35] on the lower bound of the polynomial order of patches surrounding an extraordinary point have been more than balanced by those after [46, 47, 58, 69, 70, 134].

This amount of classical subdivision theory is elaborated at greater length in the material prepared for the Primus Workshop earlier in the MINGLE project. Of particular relevance are the four chapters [100, 101, 102, 103].

2 New Schemes and a Classification

2.1 New Schemes

Kobbelt's 1996 scheme for interpolatory subdivision over quad grids [37] can be regarded as the last of the classical schemes, being essentially a tensor product of the four-point scheme with the awkward details sorted out.

The 'simplest' scheme, published by Peters and Reif in 1997 [41], however, had a new flavour to it. It was in fact a box-spline, and we should have been able to predict it by doing a systematic scan through all the box-splines with small numbers of shifts. This may be how they found it, but it still felt new, because the mesh changed orientation at each refinement stage.

This surprising aspect of that scheme is echoed in Kobbelt's $\sqrt{3}$ scheme [66] where a triangular grid becomes denser by the insertion of a new point in the middle of each triangle: the old edges disappear, being replaced by new ones joining each new point to the corners of its triangle and to the neighbouring new points. This scheme was not a box spline – its basis function is a fractal – and its publication led to an upsurge of interest in the analysis of smoothness¹ and in the determination of the support² of a scheme. In fact there is a box-spline with the same topology but a slightly larger mask. This was observed by Ron during the Dagstuhl meeting at which Kobbelt described his $\sqrt{3}$ scheme. As determined by Ron [127], over the regular grid it consists of quartic pieces and has the high continuity (C^3 in this case) expected from a box-spline scheme with many distinct generator directions. However, the translates of the basis function are not linearly independent, and so it is not possible to compute the control points from a set of points, similarly connected, which are to be interpolated. This topology of refinement of a triangular mesh was also explored by Guskov [53], who identified a bivariate family of schemes, one of which is an interpolating scheme.

The same theme led to Velho's 4-8 scheme [77, 78], a box-spline over the 4-direction grid (the quad grid with diagonals), which appears to have many nice properties, particularly in terms of being well behaved when there are diagonal features in the surface being designed. Yet another box-spline over this grid forms the quad part of Peters' 4-3 scheme [139].

Alexa [87] showed that the rotation can be even more general, the arity being described by a pair of integers³. This insight led to the exploration of skew schemes by Ivrissimtzis [129] and by Dodgson [111]. Direct exploration of

¹ the number of continuous derivatives.

² The support of a scheme is the non zero domain of the basis function of a scheme.

This was analysed fully by Ivrissimtzis [137] who showed exactly when the boundary of the support would follow lines of the grid, and when it would be fractal.

³ Another view [113, 124] is that it is a dilation matrix, a 2×2 matrix with integer entries and eigenvalues greater than or equal to 1 and its spectral radius is greater than 1, and with certain symmetries [124] corresponding to those of the topology of refinement.

ternary univariate schemes by Hassan and Ivrissimtzis [88, 114] was also happening at about the same time. This thread was later followed by Loop [115], who explored ternary subdivision of the 3-direction box-spline and found an appropriate set of coefficients for the extraordinary point case. A ternary triangulation interpolating scheme was explored, so far unsuccessfully, by Dodgson et al. in [105].

2.2 Classification

We have also been able to see more structure in the set of known surface subdivision schemes. Whereas before we had half a dozen or so known patterns, with the tensor product box-splines providing one coherent family, we can now see a framework into which all the schemes likely to be of interest will probably fit. In fact there are a few variants on this but they do not differ greatly, and [128] is a good starting point. Another is [113].

Such a classification has a number of levels, the top level being the plane tessellation which appears in the regular case. Then whether the scheme is primal, dual, both or neither, and then the arity.

Once these attributes are defined, the topology of the refinement of the polyhedron from step to step is fully determined. We then identify the *footprint*, the set of non-zero coefficients in the mask. From this and the arity the region of support can be determined, and, most important, the question of whether the scheme gives a fractal limit surface or possibly a piecewise polynomial. The support boundary [137] gives a first clue to this.

The shapes of the stencils are also determined from the footprint.

The next step is to specify the actual coefficients in the regular grid case, and finally, schemes can differ by the special rules applied at extraordinary points, although two schemes, with differences in only this respect, should be regarded as variants rather than as distinct schemes.

This classification leads to a notation [128] for the description of almost any uniform linear stationary scheme, in which two letters code the regular grid type and the primality or duality of the scheme. These are followed by two integers defining the arity and then a list of lists of independent coefficients in the mask. This is a formal notation, (although ad hoc rather than following a more general structure such as L-systems [110]). In principle it should be possible to compile an implementation of a new scheme from this notation. The extraordinary point rules are covered by letting the coefficients be functions of the valency of the extraordinary point.

Curve schemes do not have, or need, quite such a rich classification. A sufficient analogous categorisation can use the primal/dual aspect as the first level, then the arity and the mask width, followed by the approximating/interpolating issue, and finally by the detail of the values in the mask. The formal notation has the primal/dual letter, the arity, and then a list of the independent coefficients in the mask. Everything in a linear stationary scheme is derivable from this information.

3 New Domains and New Ranges

3.1 Trivariate Data

Recently first attempts were made to extend subdivision schemes to volumetric data. The simplest case is that of refining trivariate function values given at the vertices of a cube-type lattice (hexahedral lattice), which is topologically equivalent to the integer lattice in \mathbb{R}^3 , namely to \mathbb{Z}^3 .

This was explored by Peters and Wittman [44], using a 7-direction box-spline and by Barthe et al. [89], using the trivariate analogue of Chaikin, as a convenient way of managing smooth data without excessive grid density for the definition of implicit free-form surfaces. Weimer and Warren [59] use a regular trivariate domain for the solution of flow fields. Chang et al. [107] derived a C^5 subdivision scheme based on box-splines, that must be applied over hybrid tetrahedral/octahedral meshes.

MacCracken and Joy [39] used 3D subdivision as a way of defining displacement fields for free-form deformations. Their scheme is the tensor product tri-cubic B-spline scheme over the cube-type lattice, and has an extension to similar lattices with arbitrary connectivities.

Bajaj et al. [95] proposed an alternative treatment of extraordinary edges and points to that presented in [39]. Their scheme also generates tri-cubic B-spline volumes at the regular parts of the lattice. In this tensor-product setting, an extraordinary edge is essentially just the tensor product of a bivariate extraordinary point with the curve associated with the sequence of extraordinary edges. This brings little new insight except that, because the subdominant eigenvalue associated with the curve is always $1/2$, in order to keep the aspect ratios reasonable during refinement, that of the extraordinary point also needs to be $1/2$.

The analysis of points where extraordinarities meet is yet to be developed. Due to the many possible combinations of the valency of the point and the valencies of the edges meeting at the point, such analysis is expected to be highly complicated. It appears to be enumerable only if we limit the extraordinary edges to have a valency at most one more or one less than the normal, when the number of possible interesting configurations (for a hexahedral grid) is only eight.

In contrast to triangulations in the 2D case, the tessellation of the integer lattice in 3D by a tetrahedralisation is more complicated, because there are no regular tetrahedralisations, only semi-regular⁴, and there is not one obviously preferable choice. It is common in the Finite Element literature to use two different types of tetrahedra to cover the lattice.

⁴ In fact there is only one regular grid in the trivariate case, the ‘hexahedral’ grid (a trivariate grid topologically equivalent to the regular cubical grid). All ‘regular’ tetrahedralisations are in fact only semiregular, with the underlying rotation group of the hexahedral grid.

There is one lattice of similar tetrahedra which tessellates \mathbb{R}^3 . This is not a tessellation of \mathbb{Z}^3 , but of the lattice generated by it together with the midpoints of the cubes of \mathbb{Z}^3 [22], but even this is semiregular in that some edges are 6-valent while others are 8-valent.

In work in progress, Dyn, Greiner and Duchamp study a subdivision scheme, based on local averaging steps, which generates C^1 functions over this lattice. This scheme is well defined over an extension of the above lattice to arbitrary connectivities, but at this stage the analysis of continuity and smoothness near the extraordinary edges and vertices is not satisfactory. For example, the situation round an extraordinary edge is not straightforwardly described in terms of matrices for grids of tetrahedra.

3.2 Higher Dimensions

It is quite clear that direct analogues exist for manifolds of higher dimensions. However, what problems will emerge as the dimension rises still further is not yet evident.

3.3 Associated Fields

There is no reason why the subdivision process need be carried out only on the coordinates of polyhedron points. DeRose [49, 50] uses additional data to carry coordinates for applying texture to subdivision surfaces, an important issue in Computer Graphics. Each vertex has two additional values, as well as the three coordinates, and all five values are refined as if the surface lay in a space of five dimensions. Such additional values can be used for any other properties for which we require a field of values to vary smoothly over the limit surface. All that is necessary is to think of the subdivision as being carried out in a space of many dimensions.

3.4 Field Analysis

In particular, a subdivision surface or volume can support an analogue of the finite element method by using the subdivision basis functions instead of the Lagrange functions conventionally used in stress-analysis elements. Subdivision is now logically carried out in the space of 3 position coordinates and 3 components of displacement, although the latter are treated as algebraic variables solved for by an energy minimisation process. This gives an extremely powerful method, because the subdivision process gives adaptive basis functions, strongly analogous to standard FE practice, but performing much better, in that they span a space in which the first derivative is continuous everywhere (like the actual field in an elliptic problem such as elasticity), so that freedoms are not wasted on modelling possible discontinuities of derivative.

This has been described by Cirak, Ortiz and Schröder [57, 91] in the surface case and by Weimer and Warren [59] in the solid case. A key result for surface analysis was obtained by Reif and Schröder in [74], which showed that it was not necessary to have bounded curvature for the bending energy of a subdivision surface to be well-defined. This meant that easily accessible un-tuned versions of Loop and Catmull-Clark could be used for this purpose.

The use of subdivision gives both good basis functions, capable of giving at least one order of magnitude improvement over conventional finite elements in speed for a given accuracy, and a very convenient h -refinement mechanism. This means that we can expect to see enormous benefits in systems which give almost real time stiffness and strength analysis during the interactive modification of not-too-complicated CAD models.

3.5 Compact Sets

In a series of papers, Dyn and Farkhi investigated spline subdivision schemes, with compact sets taking the rôle of the control points [71, 82, 86]. They found that for data consisting of convex, compact sets and with the Minkowski sum of sets replacing the usual addition of points, the limits generated by spline subdivision schemes approximate set-valued functions with convex images in a “shape preserving” way [71]. For set-valued functions with non-convex images these schemes converge to set-valued functions with convex images [140]. So no approximation can be expected.

In [82] the authors suggested performing spline subdivision by repeated averaging and replacing the averaging operation between two control points by the binary operation between two compact sets, called the “metric average”. They proved that the resulting subdivision schemes are convergent. They also showed that any such scheme operating on data sampled from a Lipschitz continuous set-valued function, generates a limit set-valued function approximating the sampled set-valued function. The smoothness properties of the limit set-valued functions have yet to be explored.

3.6 Face-valued Subdivision

It is also possible to associate values with faces rather than with vertices and to have these values propagated through the refinement process. A natural quantity to attach to a face is the average of some function f over that face. In tensor-product schemes averages over the faces in the parametric domain can be regarded as point values of the primitive function of f , obtained by integrating f in the two parameters. This was observed by Donoho [27] for averages of univariate functions. In triangular grids refinement of averages over the triangles gives rise to new families of schemes.

Cohen et al. [72] present and analyse an “average interpolating” scheme. In such a scheme each triangle is refined, in the parametric domain, into four

similar ones such that the averages assigned to the refined triangles generated from triangle T , sum to four times the average assigned to T . Cohen et al's scheme generates C^0 limit functions and is exact for linear bivariate polynomials.

A family of “approximating” face-value schemes, which are obtained by the elementary operation of repeated averaging of the face-values (in analogy to Box-splines schemes which can be described by simple repeated averaging of point values) are studied by Dyn et al. [112], where their smoothness is derived from convolution relations. Such schemes, also called “dual schemes” on triangulations, are investigated by Oswald and Schröder [122], as a continuation of the work of Zorin and Schröder [83] on alternating quad grids.

4 New Issues

4.1 Compatibility with Existing Parametric Surface Software

A requirement which has come from the move to incorporate subdivision surfaces into existing major commercial software systems already dealing with surface geometry held in other forms, is that a method is needed for the interrogation of these surfaces in a way which is compatible at a low level with the existing parametric surfaces. This is needed so that the breadth of functionality of those systems can be maintained without a complete recoding.

A large step in this direction can be taken by allowing a subdivision surface to be perceived as a collection of a small number of rectangular parametric surfaces. However, just using the regular regions of the surface and the rings around extraordinary points is not sufficient, because there are too many rings.

The alternative, developed by Stam [52], is to split the surface so that extraordinary points appear at corners of rectangular carpets, and have a special evaluator.

Whenever an evaluation request is received relating to a point in the region influenced by the extraordinary point, this evaluator does the subdivision to a sufficient level on the fly and then peeks at the right ring.

Although effective, this is inelegant, because the computation required depends on the closeness of the requested point to the extraordinary point. It also has a problem that very close to the extraordinary point, the mapping of a corner of the parameter square on to one sector of the subdivision surface has extremely large second derivatives. This means that the calling algorithms must be extremely robust to work effectively.

Zorin and Kristjansson [98] suggested that instead, in the region influenced by the extraordinary point, the surface should be examined in terms of the finite number of eigencomponents and evaluated direct from there.

It could be argued that divide and conquer algorithms, based on enclosures [93], would be a much more robust approach to interrogation, but the

suppliers of commercial software have limited budgets for such complete reimplementations, and do have the requirement to be able to support any new surface type with the full range of functionality of the system. Being able to plug the new surfaces in at the evaluation level is a very practical way of achieving this in the short term.

4.2 Artifacts

Until about 2001 the only property of subdivision surfaces that anybody bothered with was the level of smoothness at extraordinary points, and at all points within schemes which did not have the box-spline underpinning which allows the level of continuity of regular regions to be determined *a priori*. Once subdivision surfaces started to be used for real tasks, it became apparent that there were other aspects equally important or more so.

One problem which was brought to light is that if an extraordinary point has a very high valency, the polyhedron after a few steps has what appears to be a cone centred at that point. This appears to be due to the fact that the subdominant eigenvalues tend to increase with valency, so that the rate of shrinkage towards such points is much slower than the average over the surface.

In particular, if the subdominant eigenvalue of an extraordinary point is significantly greater than $1/2$ in a binary scheme, the rings shrink more slowly towards that point than to normal points. After a few iterations this results in a ‘polar cap’ of long thin faces.

Because the distance of a control point from its limit position varies quadratically with the size of the local faces, the extraordinary control point remains far from the limit surfaces after all the other points have converged closely, and so if an image is drawn without projection of the vertices to their limit positions, it appears as if there is a cone-like failure of tangent continuity.

An even more serious problem is that in extremely extraordinary situations, where a high-valency extraordinary vertex is surrounded by low-valency ones, the shape of the limit surface can be significantly qualitatively different from that of the original polyhedron, showing features whose spatial frequencies are as high as the valency of the central extraordinarity. So far we have identified three possible causes for these:

1. the natural configuration of the high-valency point
2. inequality of the cup- and saddle-eigenvalues
3. plain interaction at the first level of refinement.

This is a high priority target for more understanding.

What is common to these effects is that features appear in the limit surface which have a spatial frequency too high, relative to the density of the original mesh, to be corrected by the careful placing of the control points. Once this is observed, we can identify other effects which also have this aspect in common.

In particular, all of our stationary curve schemes do in fact contain, in the limit curve, spatial frequencies of twice the Shannon limit. To avoid this would require that the basis function should be $\sin(x)/x$ which would in turn require a subdivision mask of infinite width. Indeed, the higher the degree, the smaller this artifact, and the benefit from looking on subdivision as being made out of sampling and smoothing occurs precisely because the smoothing reduces high frequencies more than lower.

A more interesting effect is that aliasing effects can also appear in directions where the original polyhedron has no features. A simple extruded ridge in the polyhedron can result in a limit surface with a ‘dinosaur back’, with one hump per original control point.

This is now fully understood. Every scheme misbehaves in this way if the original polyhedron has its vertices sampled from an extruded surface, except when the direction of extrusion is one of a small number of special ones – the directions in which the symbol of the scheme has at least one factor of $(1 - z^a)/(1 - z)$. Some schemes, such as Kobbelt’s $\sqrt{3}$ [66] have no such directions. Catmull-Clark [6] has two, the directions of the edges of the mesh; Loop [13] has three; the 4-8 [77, 78] and 4-3 [139] schemes, despite being quad-based, have four, and the Dagstuhl scheme [127] has six. It turns out that there are also advantages of having more than one $(1 - z^a)/(1 - z)$ factor in a direction, as then a polyhedron formed by linearly varying extrusion will result in a limit surface with the same property.

The final effect, not strictly an aliasing one, is that the way that edges are handled in the raw versions of the standard subdivision schemes has a serious deficiency, in that the local curvature across the boundary is determined not by the shape of the polyhedron measured across the boundary but by the curvature along the local boundary. Indeed, in the Catmull-Clark scheme, the isoparametric lines crossing a boundary have zero curvature where they meet it.

These effects are described more fully, with their explanations, where they are known, by Sabin and Barthe in [116].

4.3 Street Wisdom

In the NURBS generation of surface descriptions, there seemed to be rather little know-how about how to choose control points to define the surface that you really wanted. Perhaps this was because there was little freedom to choose the topology and so recourse to just making a dense control polyhedron was frequently required.

In the subdivision surface case it is possible to use the extraordinary points to achieve a lot, and the need for ‘street wisdom’ is larger, because the opportunity to use it to reduce the density of the original mesh and thus give smoother surfaces easily is much higher. There seem to be two important rules.

1. Extraordinary points have two purposes:
 - a) to allow the mesh to follow features even when those features do not form a regular grid. ‘Features’ in this context includes the edges of faces, and so if this principle is followed, the need for trimmed surfaces is significantly reduced.
 - b) to match the local total curvatures, and thereby allow the mesh to remain more or less orthogonal everywhere. An extraordinary point of valency 3 in a quad grid provides enough positive discrete total curvature to satisfy an octant of a sphere. One of valency 5 provides the same amount of negative discrete total curvature.

So the first rule is to *choose the topology of the initial polyhedron so that the only extraordinary vertices are those needed for these purposes*.

In both cases the need for extremes of extraordinarity is very unusual. In almost all cases the variation of valency from the regular case need only be plus or minus one. Better results will almost always be obtained in a quad grid by using two separate 5-valent points rather than one 6-valent.

2. The second rule is to *keep the mesh as sparse as you can*. The filling in between the extraordinary points is exactly what the subdivision process does, and initial design can be carried out by providing a mesh with very few points except the necessary extraordinary ones (each such point being required either by 1a or by 1b above), carrying out one step at a time of subdivision and correcting the mesh to suit shorter wavelength features of the required shape at each step.

These rules are primitive. We hope that significant work will go into refining them into practical tools for surface designers to apply when using subdivision surfaces. The structures used by Skaria et al. [80], although derived in a different way, appear to be consistent with these guidelines.

4.4 Compression

Khodakovskiy et al. [68] showed that this last idea can be exploited to give an effective compression of smooth surfaces for transmission over the web. The coarse mesh is transmitted (possibly coded, although it should be so sparse that coding does not save many bytes) together with the corrections by which the control points computed at each refinement level need to be moved so that the limit surface eventually matches the desired surface.

If the original surface is smooth and the subdivision scheme is a good one, the corrections drop dramatically in amplitude as refinement proceeds, so that the total information sent can be a very small fraction of the total mesh. Indeed it is likely to be independent of the original density of the mesh.

Guskov’s idea of normal subdivision [65], where the corrections are applied only in a direction perpendicular to the surface, saves a further factor of 3 on the corrections, although it means that a given tessellation is not necessarily exactly matched. It was pointed out by a referee that entropy coding would normally exploit this redundancy almost as effectively as normal subdivision.

4.5 Tuning

The response to the artifacts which are associated with extraordinary points has to be to try to tune the behaviour round the extraordinary points by altering the local coefficients. There are two approaches to this: we can either think of the stencils as the primary representation of the scheme, or else the masks.

In the stencil-oriented approach we modify the values in those stencils which contain a reference to an extraordinary point. There are clear stages in which we can do this. The first stage is to modify only the stencil of the new extraordinary point itself. This was done in a half-hearted way in the original Catmull-Clark paper [6] as a response to the Doo-Sabin analysis [7] of the behaviour at the extraordinary point. It was done properly for the Catmull-Clark scheme by Ball and Storry [9]. Loop's scheme was explicitly designed with exactly this tuning as a design principle [13].

However, this can only control the continuity of cup-like curvatures. In order to control the saddle-like curvatures it is necessary also to modify the stencils of the 1-ring of new vertices. This was done for the Catmull-Clark scheme by Sabin [18], and for the Loop scheme by Holt [40], in a way which did not alter the footprints of the stencils. No new influences were introduced; only the already non-zero coefficients were altered.

More recent tunings, by Zorin of the butterfly scheme [38], by Prautzsch and Umlauf of the Loop and Butterfly schemes [46], by Loop of the Loop scheme [81, 99], and by Barthe of the Catmull-Clark scheme [133] have taken the attitude that it is OK to increase the sizes of the stencils if by doing so other properties, such as positivity, can be maintained as well as the required eigenratios. The tuning of the 4-3 scheme [139] however, maintains the footprints of the stencils.

The mask-oriented approach, on the other hand, says that as a first stage of tuning we will alter only the mask associated with the extraordinary point itself. The entries in the mask become expressions which depend on the extraordinary valency, and we choose them so that certain desired properties of the scheme (for example: good subdominant eigenvalue, bounded curvature, good characteristic maps) are achieved.

When this approach is applied, the stencil of each of the new vertices covered by the mask of the extraordinary vertex has to be renormalised so that the entries sum to unity, but this is straightforward, and covers the situation automatically where there are vertices which lie within the masks of more than one extraordinary point.

We now see the stages of stencil-oriented tuning appearing again in terms of which coefficients in the mask vary with valency. The central value controls the cup-curvature eigencomponent; the 1-ring value controls the ratio of the saddle-curvature and tangent plane eigencomponents; the rings further out control the eigenvectors, so that we can choose a neat natural configuration leading to a well-behaved characteristic map.

This approach is more limited, in that the influences of regular vertices are always regular, but it does ensure that the footprints of vertices near the extraordinary vertex remain standard, so that the support analysis remains valid in that region. It is argued here that this should be the approach of first resort, and that the larger number of freedoms available through stencil-based tuning should not be invoked until it is certain that the objectives cannot be met by mask-based tuning.

Both views bring their own insights and both need to be explored a lot more in the near future.

5 New Ideas

The theory described above has almost entirely focused on the uniform stationary case, in which the refinement rules are the same throughout the polyhedron and are also the same at every step, but we do not have to be limited in this way. We can consider non-uniform schemes, in which the rules vary from place to place, and non-stationary ones in which the rules vary with the refinement level.

5.1 Non-uniform Schemes

In a non-uniform scheme, the rules do not need to be the same at all parts of the grid.

Boundaries

In fact we have always had some measure of non-uniformity, in the sense that any practical scheme needed to have appropriate rules at the boundaries.

As was pointed out above, the boundary rules actually used turn out to be a little unfortunate, and Levin [55, 56] suggested the use of what he called “combined schemes” which defined a bounded piece of surface by both a polyhedron and a set of surface normals to be matched at the boundary. This effectively gives an analogue of the Bézier edge conditions normally used for NURBS surfaces.

Nasri and Sabin [109] surveyed and classified the ways in which constraints on the original polyhedron could be used to achieve interpolation of both boundaries and internal feature curves, with and without also controlling the variation of surface normal along those curves. Sabin and Bejancu [126] looked for an analogue of the “not-a-knot” condition found so valuable in the old days of interpolating cubic splines, and found that this could be expressed in terms of fourth differences over the polyhedron for both Catmull-Clark and Loop schemes.

Unequal-intervals

One of the potential barriers to adoption of subdivision surfaces as a successor to NURBS in the representation of engineering objects is the problem of compatibility. The possibility of having unequal spacing of knots in NURBS is frequently used, while all current subdivision schemes are implicitly the equivalent of equal interval splines.

Dyn, Gregory and Levin [31] made first steps in this by considering the quasi-uniform case, where all the intervals on one side of a central point have one size, all those on the other a different size. After one refinement all intervals were halved. Within this restricted variability the tools of eigenanalysis can still be applied, and they showed that all the important properties could be maintained. This work was continued by Dyn and Levin in [62] to the general unequal interval case.

The univariate case of B-spline unequal interval subdivision was explored by Qu and Gregory [24, 36] and by Warren [32], who showed that knot insertion in an unequal interval context could give a corresponding subdivision scheme, with all the expected properties.

Sederberg et al. [51] explored the possibility of labelling every edge in a polyhedron with a “knot interval”. A variant of Catmull-Clark was then used which produced a similarly-labelled refined polyhedron. This scheme contained all the unequal interval bicubic B-spline surfaces as a particular case when the grid was completely regular and the labels were equal on opposite edges of every quad. It also contained all Catmull-Clark surfaces as the particular case when all labels were equal.

It had two disadvantages. The first was the need to specify all the labels: some preprocess determining labels from some terser specification would be necessary in practice. The second was that the spectrum at extraordinary points had the subdominant eigenvalues split, so that proving smoothness was extremely hard. In fact slightly different rules for the new labellings could have solved this.

It seems that new work in this direction would be valuable, but it is now clear that the ambition of having all edges separately labellable is not in fact necessary to cover the regular non-uniform case. It is also clear that the rules for defining the new labellings should be derived from some logical argument about where the new “knots” should be. There is no reason why new knots should be either rigidly at the middles of old intervals or at wherever some convenient formula happens to put them.

Mixed Grid Schemes

Where a desired surface has a clear warp and weft, indicated by its very distinct principal curvatures, a quad grid is the natural choice, avoiding lateral artifacts by running the flow of the grid along the flow of the surface. However, for surfaces where there is no such bias, containing bumps, hollows and saddles

(think of mild terrain), the natural way to define a polyhedron is to put a vertex at each maximum and each minimum and maybe at each saddle as well. In such a case it makes sense to use either a triangulation, or else a mixed grid containing both triangles and quads.

The special case of such a mixed grid, consisting of a triangular mesh on one side of an “extraordinary line” and a quad grid on the other was first raised by Loop at a Dagstuhl meeting, and he and Stam [118] produced an initial scheme. This case has been fully analysed by Levin and Levin [123] and an alternative scheme proposed.

The fuller case where quads and triangles can be mixed as appropriate for the surface being defined is the subject of a paper by Peters and Shiue [139]. In this case the quad part of the scheme, applied wherever a sufficiently large piece of the mesh consists entirely of quads is a four-direction box-spline, rather than Catmull-Clark, while the triangle part is a bounded-curvature variant of Loop.

The extraordinary line analysis is still an important part of the tool-box for such schemes, because as refinement proceeds, the interiors of original triangles become regions of triangle mesh and the interiors of original quads, quad mesh, with extraordinary lines between them. However, the corners, which can be of several different kinds, each need their own analysis, which is addressed by Levin in [121].

5.2 Non-stationary Schemes

In a non-stationary scheme the rules can change from level to level. In the simplest, uniform, form of non-stationary scheme the rules at each level of subdivision are the same everywhere.

There is a very interesting family of univariate non-stationary schemes generating Exponential B-splines (see [8] about this notion), and in particular trigonometric splines [30]. Any such scheme is similar to a corresponding B-spline scheme, has the same support of its mask, and its refinement rules converge to those of the corresponding B-spline scheme as the refinement level increases. In particular it is possible to construct a univariate scheme generating circles. (In several ways, the one of smallest support has refinement rules converging to those of the Chaikin’s scheme). In [23] Dyn and Levin constructed the scheme of smallest support, which, when applied to the vertices of a regular n -gon, generates a circle.

Recently Morin, Warren and Weimer [79] constructed a circle-generating scheme, with refinement rules converging to the refinement rules of a cubic B-spline scheme, and then constructed its tensor-product with a stationary scheme, to obtain a subdivision scheme generating surfaces of revolution.

The analysis of convergence and smoothness of non-stationary scheme cannot use the analysis tools for stationary schemes. One method for studying the convergence is by comparison to a convergent stationary scheme [30, 94]. Recently Dyn, Levin and Luzzatto analysed with this method a family of

non-stationary interpolatory schemes reproducing exponential functions [125]. These schemes are applicable to the processing of highly oscillatory signals.

Sabin [141] shows an analogue of the four-point scheme, in which the computation of the new vertices is not expressed in terms of linear combinations, but as a non-linear geometric construction which guarantees the preservation of circles. The non-stationarity is present in the non-linearity, but is hidden under an apparently constant construction.

5.3 Geometry Driven Schemes

This last scheme is non-uniform as well as non-stationary, and so is better described as a geometry-driven scheme. The rules both ensure that any set of points which lie in sequence on a circle, whatever their spacing, will give a circular limit curve, but the spacing also converges towards local uniformity, so that the results of Dyn and Levin [30] can then be invoked to determine the smoothness of the scheme.

In that sense it may well indicate a useful direction for other explorations. We define here the term “geometry-driven” as the name for a class of schemes in which the new polygon or polyhedron has its local shape determined geometrically from the locality of the old one. It is likely that such schemes will lose affine-invariance, and the actual value of affine invariance as against invariance under solid body, mirror and scaling transforms should be a matter for debate in the community.

Another example of a family of geometry driven schemes is described by Marinov [132] elsewhere in this book. It is derived from the “classical” linear 4-point scheme, by allowing a variable tension parameter instead of the fixed tension parameter. The tension parameter is adapted locally in various ways, according to the geometry of the control polygon within the 4-point stencil. With this change the new schemes retain the locality of the 4-point scheme, and at the same time achieve important shape-preservation properties, such as artifact elimination and co-convexity preservation. The proposed schemes are robust and have the special features of “double-knot” edges corresponding to continuity without smoothness in the limit curves (for artifact elimination), and inflection edges for co-convexity preservation.

5.4 Hermite Subdivision

In a Hermite scheme, not only point positions, but also derivatives or normals are carried forward from each stage to the next. This is not a totally new idea, but there have been significant developments recently.

Hermite schemes were first designed mainly for functional data. Merrien [25, 29] introduced interpolatory Hermite schemes of extremely local support. His univariate schemes are two-point schemes, namely the values of the function and a fixed number of its derivatives in the mid-point of an

interval is determined by the values of the function and the same number of its derivatives at the two boundary points of the interval.

Similarly in his Hermite schemes on triangulations, the values of the function and its derivatives attached to a mid-point of an edge of a triangle in the refinement step, depend only on the values of the function and its derivatives attached to the boundary points of that edge. Merrien analysed the convergence and smoothness of his schemes. Dyn and Levin provided in [63] analysis tools for univariate interpolatory Hermite subdivision schemes, for any number of derivatives attached to the grid points and any support size of the scheme. Dyn and Lyche designed a specific bivariate Hermite subdivision scheme generating the Powell-Sabin twelve-split quadratic spline [64].

Recently new Hermite-type schemes were developed. Jüttler and Schwannecke in [97] represented Hermite schemes for curves corresponding to data of position and tangent vectors, in terms of spline curves with control points. They provided an analysis method based on their representation, and designed new schemes, interpolatory and non-interpolatory. van Damme [42] proposed several schemes. For bivariate functions he gave proofs for the convergence and smoothness of the corresponding schemes, while for surfaces he gave only experimental results. Han et al. [138] defined the notion of non-interpolatory Hermite subdivision schemes, and designed a family of such schemes for surface generation, whose refinement is done with a dilation matrix. They analyse the smoothness of their schemes which have certain symmetries of relevance to geometric modelling.

5.5 Non-linear Schemes for the Functional Case

In analogy to the geometry-driven extensions of the linear 4-point scheme, new non-linear 4-point schemes for interpolating given equidistant data, sampled from a piecewise smooth function, were designed by Cohen, Dyn and Matei [136]. These schemes are based on the ENO (essentially non-oscillatory) interpolation technique, and use a 4-point stencil for the insertion of a new point, which is either centred relative to the inserted point or moved by one point to one of the two sides. The choice of the stencil is data dependent and aims at using data from the smooth pieces of the approximated function. These schemes converge to Hölder continuous limit functions, and reproduce cubic polynomials.

5.6 Reverse Subdivision

Reverse subdivision is the process of taking a dense grid and determining a coarse one which, when refined by subdivision, gives a good approximation to the original. It is closely related to wavelets and to compression.

Although the idea of looking at subdivision through the insights of wavelet mathematics can be traced back to [28, 21] or even earlier, Warren [43] provides a very readable introduction to the idea of applying wavelet techniques

to subdivision schemes, using the cubic B-spline univariate subdivision as a concrete example throughout.

In [60] and [84], Bartels and Samavati provide a much fuller description of the links between wavelets, least squares fitting and the reversal of specific subdivision schemes. Although the tensor product case is covered, the ideas are essentially univariate, but in [85] and [108] they apply these ideas to surfaces, reversing Loop and Butterfly, and the original Doo subdivision [5] respectively.

Hassan and Dodgson [130] show a simpler approach based on Chaikin subdivision with a purely local determination of the new (coarser) vertices and the error terms.

Suzuki, Takeuchi and Kanai [61] fit approximate subdivision surfaces by using an interactively defined initial polyhedron to indicate which parts of the given dense surface should be converged to from each original vertex, and then solve the linear system implied by the limit positions of each vertex to solve for a good set of coarse control points. Ma and Zhou [75] gain their initial polyhedron from a consideration of the boundary of the given dense data and carry out a similar operation. Ma et al. [92] derive their coarse initial polyhedron for Loop subdivision by using strong decimation and re-triangulation on the grid, and use least squares fitting rather than collocation. This fitting is done in a single swoop direct from a really coarse mesh, whereas that of the wavelet-based approaches is done one step at a time. It also has the effect that subsequent resubdivision will seldom give the same triangulation connectivity as the original. Taubin [96] observed that loss of the original connectivity can often be avoided if the original mesh had few extraordinary vertices, and those vertices are joined by chains of edges passing through normal vertices without turning. He identified a parallelisable algorithm for detecting such situations in both triangular and quad meshes.

Surazhsky and Gotsman [119] observe that in general dense triangular meshes can benefit from a process of remeshing, wherein the number of extraordinary vertices is reduced significantly (and the number of extremely extraordinary vertices reduced very significantly). Their resulting meshes are dramatically smoother than the originals, and are likely to provide a better start for recapturing of a coarse subdivision surface. Within their technology is a technique for migrating extraordinary vertices within a mesh, so that extraordinarities of opposite sign can be brought together to annihilate each other.

Alkalai and Dyn [131] also carry out an optimisation of the connectivity of a given triangulation, but at the coarse level. This improves the convexity of the initial polyhedron, minimising the discrete curvature merely by applying 2:2 swaps. The vertex coordinates are not changed.

Acknowledgements

This paper was partially supported by the European Union research project ‘Multiresolution in Geometric Modelling (MINGLE)’ under grant HPRN-CT-1999-00117. Thanks are also due to Nira Dyn, who provided valuable critical comment and drafts of some sections of the paper.

References

before 1980

1. G. de Rham. Un peu de mathématique à propos d'une courbe plane *Elemente der Mathematik* 2, 73–76, 89–97, 1947.
2. G. Chaikin. An algorithm for high speed curve generation *Computer Graphics & Image Processing* 3, 346–349, 1974.
3. A. R. Forrest. Notes on Chaikin's algorithm. University of East Anglia Computational Geometry Project Memo, CGP74/1, 1974.
4. R. Riesenfeld. On Chaikin's algorithm. *Computer Graphics & Image Processing* 4, 304–310, 1975.
5. D. Doo. A subdivision algorithm for smoothing down irregularly shaped polyhedrons. *Proc. Int'l Conf. on Interactive Techniques in Computer Aided Design*, IEEE Computer Soc., 157–165, 1978.
6. E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design* 10, 350–355, 1978.
7. D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design* 10, 356–360, 1978.

1980–1984

8. L. L. Schumaker. *Spline Functions*, John Wiley, New York, 1981.
9. A. Ball and D. Storry. Recursively generated B-spline surfaces. *Proc. CAD84*, 112–119, 1984.
10. W. Dahmen and C. Micchelli. Subdivision algorithms for the generation of box-spline surfaces. *Computer Aided Geometric Design* 1, 115–129, 1984.
11. D. Storry. B-spline surfaces over an irregular topology by recursive subdivision. Ph.D. Thesis, Loughborough University, 1984.

1985–1989

12. A. Ball and D. Storry. A matrix approach to the analysis of recursively generated B-spline surfaces. *Computer Aided Design* 18(8), 437–442, 1986.
13. C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah, 1987.
14. N. Dyn, D. Levin, and J. A. Gregory. A four-point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design* 4, 257–268, 1987.
15. A. Ball and D. Storry. Conditions for tangent plane continuity over recursively generated B-spline surfaces. *ACM Transactions on Graphics* 7(2), 83–108, 1988.
16. G. Deslauriers and S. Dubuc. Symmetric iterative interpolation processes *Constructive Approximation* 5, 49–68, 1989.

1990–1994

17. N. Dyn, J. Gregory, and D. Levin. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics* 9, 160–169, 1990.
18. M. Sabin. Cubic recursive division with bounded curvature. *Curves and Surfaces*, L. L. Schumaker, J.-P. Laurent, and A. Le Méhauté (eds.), Academic Press, 411–414, 1991.
19. A. Cavaretta, W. Dahmen, and C. Micchelli. Stationary subdivision *Memoirs of the AMS*, vol 453, 1991.
20. N. Dyn, J. Gregory, and D. Levin. Analysis of uniform binary subdivision schemes for curve design *Constructive Approximation* 7, 127–147, 1991.
21. T. DeRose, M. Lounsbery, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics* 11, 34–73, 1992.
22. D. Moore. Simplicial mesh generation with applications. Ph.D. thesis, Cornell University, 1992.
23. N. Dyn and D. Levin. Stationary and non-stationary binary subdivision schemes. *Mathematical Methods in Computer Aided Geometric Design II*, T. Lyche, and L. L. Schumaker (eds.), Academic Press, 209–216, 1992.
24. R. Qu and J. Gregory. A subdivision algorithm for non-uniform B-splines *Approximation Theory, Spline Functions and Applications*, Singh (ed.), 423–436, 1992.
25. J.-L. Merrien. A family of Hermite interpolants by bisection algorithms. *Numerical Algorithms* 2, 187–200, 1992.
26. N. Dyn. Subdivision schemes in computer aided geometric design. *Advances in Numerical Analysis – Volume II, Wavelets, Subdivision Algorithms and Radial Basis Functions*, W. Light (ed) Clarendon Press, Oxford, 36–104, 1992.
27. D. L. Donoho. Smooth wavelet decompositions with blocky coefficient kernels. *Recent Advances in Wavelet Analysis*, L. L. Schumaker and G. Webb (eds.), Academic Press, Boston, 259–308, 1993.
28. H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stützle. Piecewise smooth surface reconstruction *Proc. ACM SIGGRAPH '94*, 295–302, 1994.
29. J.-L. Merrien. Dyadic Hermite interpolation on a triangulation. *Numerical Algorithms* 7, 391–410, 1994.

1995

30. N. Dyn and D. Levin. Analysis of asymptotically equivalent binary subdivision schemes. *Journal of Math. Anal. Appl.* 193, 594–621, 1995.
31. N. Dyn, J. Gregory, and D. Levin. Piecewise uniform subdivision schemes. *Mathematical Methods for Curves and Surfaces [33]*, 111–119, 1995.
32. J. Warren. Binary subdivision schemes for functions of irregular knot sequences. *Mathematical Methods for Curves and Surfaces [33]*, 543–562, 1995.
33. M. Daehlen, T. Lyche, and L. Schumaker (eds.). *Mathematical Methods for Curves and Surfaces*, Vanderbilt University Press, ISBN 8265-1268-2, 1995.
34. U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design* 12, 2, 153–174, 1995.

1996

35. U. Reif. A degree estimate for subdivision surfaces of higher regularity. *Proc AMS* 124(7), 2167–2174, 1996.
36. J. Gregory and R. Qu. Non-uniform corner cutting. *Computer Aided Geometric Design* 13, 763–772, 1996.
37. L. Kobbelt. Interpolatory subdivision on open quadrilateral nets with arbitrary topology. *Computer Graphics Forum* 15, 409–420, 1996.
38. D. Zorin, P. Schröder, and W. Sweldens. Interpolatory subdivision for meshes with arbitrary topology. *Proc. ACM SIGGRAPH '96*, 189–192, 1996.
39. R. MacCracken and K. Joy. Free-form deformations with lattices of arbitrary topology *Proc. ACM SIGGRAPH '96*, 181–188, 1996.
40. F. Holt. Toward a curvature continuous stationary subdivision algorithm. *ZAMM 1996 S1 (Proc GAMM)*, 423–424, 1996.

1997

41. J. Peters and U. Reif. The simplest subdivision scheme for smoothing polyhedra. *ACM Transactions on Graphics* 16, 420–431, 1997.
42. R. van Damme. Bivariate Hermite subdivision. *Computer Aided Geometric Design* 14, 847–875, 1997.
43. J. Warren. Sparse filter banks for binary subdivision schemes. *Proc. Mathematics of Surfaces VII [45]*, 427–438, 1997.
44. J. Peters and M. Wittman. Smooth blending of basic surfaces using trivariate box splines. *Proc. Mathematics of Surfaces VII [45]*, 409–426, 1997.
45. T. Goodman and R. Martin (eds.. *Proc. Mathematics of Surfaces VII*, Information Geometers, ISBN 1-874728-12-7, 1997.

1998

46. H. Prautzsch and G. Umlauf. Improved triangular subdivision schemes. *Proc. Computer Graphics International*, 626–632, 1998.
47. H. Prautzsch and G. Umlauf. A G^2 subdivision algorithm. *Computing Supplements* 13, Springer Verlag, 217–224, 1998.
48. H. Prautzsch. Smoothness of subdivision surfaces at extraordinary points. *Adv. Comput. Math.* 9, 377–389, 1998.
49. T. DeRose et al. Texture mapping and other uses of scalar fields on subdivision surfaces in computer graphics and animation US Patent 6,037,949, 1998.
50. T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. *Proc. ACM SIGGRAPH '98*, 85–94, 1998.
51. T. Sederberg, D. Sewell, and M. Sabin. Non-uniform recursive subdivision surfaces. *Proc. ACM SIGGRAPH '98*, 387–394, 1998.
52. J. Stam. Exact Evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. *Proc. ACM SIGGRAPH '98*, 395–404, 1998.
53. I. Guskov. Multivariate subdivision schemes and divided differences. Princeton University preprint, <http://www.cs.caltech.edu/~ivguskov/two.ps.gz>, 1998.
54. J. Peters and U. Reif. Analysis of algorithms generalizing B-spline subdivision. *SIAM J. Numerical Analysis* 35, 728–748, 1998.

1999

55. A. Levin. Combined subdivision schemes. Ph.D. thesis, Tel-Aviv University, 1999.
56. A. Levin. Combined subdivision schemes for the design of surfaces satisfying boundary conditions. *Computer Aided Geometric Design* 16, 345–354, 1999.
57. F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: A new paradigm for thin-shell finite element analysis. Technical report <http://www.multires.caltech.edu/pubs/>, 1999.
58. K. Qin and H. Wang. Eigenanalysis and continuity of non-uniform Doo-Sabin surfaces. *Proc. Pacific Graphics* 99, 179–186, 1999.
59. H. Weimer and J. Warren. Subdivision schemes for fluid flow. *Proc. ACM SIGGRAPH '99*, 111–120, 1999.
60. F. Samavati and R. Bartels. Multiresolution curve and surface representation: reversing subdivision rules by least-squares data fitting. *Computer Graphics Forum* 18, 97–119, 1999.
61. H. Suzuki, S. Takeuchi, and T. Kanai. Subdivision surface fitting to a range of points. *Pacific Graphics* 99, 158–167, 1999.
62. N. Dyn. Using Laurent polynomial representation for the analysis of non-uniform binary subdivision schemes. *Adv. Comput. Math.* 11, 41–54, 1999.
63. N. Dyn and D. Levin. Analysis of Hermite interpolatory subdivision schemes. *Spline Functions and the Theory of Wavelets*, S. Dubuc (ed.), AMS series CRM Proceedings and Lecture Notes 18, 105–113, 1999.
64. N. Dyn and T. Lyche. Hermite subdivision scheme for the evaluation of the Powell-Sabin 12-split element. *Approximation Theory IX*, C. Chui and L. L. Schumaker (eds.), Vanderbilt University Press, 1–6, 1999.

2000

65. I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal meshes. *Proc. ACM SIGGRAPH 2000*, 95–102, 2000.
66. L. Kobbelt. $\sqrt{3}$ subdivision *Proc. ACM SIGGRAPH 2000*, 103–112, 2000.
67. H. Biermann, A. Levin, and D. Zorin. Piecewise smooth subdivision surfaces with normal control *Proc. ACM SIGGRAPH 2000*, 113–120, 2000.
68. A. Khodakovskiy, P. Schröder, and W. Sweldens. Progressive geometry compression. *Proc. ACM SIGGRAPH 2000*, 271–278, 2000.
69. J. Peters and G. Umlauf. Gaussian and mean curvature of subdivision surfaces. *The Mathematics of Surfaces IX*, Cipolla and Martin (eds.), Springer, ISBN 1-85233-358-8, pages 59–69, 2000.
70. H. Prautzsch and G. Umlauf. A G^1 and a G^2 subdivision scheme for triangular nets. *Journal of Shape Modeling* 6, 21–35, 2000.
71. N. Dyn and E. Farkhi. Spline subdivision schemes for convex compact sets. *Journal of Computational and Applied Mathematics* 119, 133–144, 2000.
72. A. Cohen, N. Dyn, K. Kaber, and M. Postel. Multiresolution schemes on triangles for scalar conservation laws. *Journal of Computational Physics* 161, 264–286, 2000.
73. J. Peters. Patching Catmull-Clark meshes. *Proc. ACM SIGGRAPH 2000*, pp255–258, 2000.

74. U. Reif and P. Schröder. Curvature integrability of subdivision surfaces. *Adv. Comp. Math.* 12, 1–18, 2000.
75. W. Ma and N. Zhao. Catmull-Clark surface fitting for reverse engineering applications. *Proc. Geometric Modeling and Processing 2000*, IEEE, 274–283, 2000.

2001

76. A. Nasri, K. van Overfeld, and B. Wyvill. A recursive subdivision algorithm for piecewise circular spline. *Computer Graphics Forum* 20(1), 35–45, 2001.
77. L. Velho. Quasi 4-8 subdivision. *Computer Aided Geometric Design* 18(4), 345–358, 2001.
78. L. Velho and D. Zorin. 4-8 subdivision. *Computer Aided Geometric Design* 18(5), 397–427, 2001.
79. G. Morin, J. Warren, and H. Weimer. A subdivision scheme for surfaces of revolution. *Computer Aided Geometric Design* 18(5), 483–502, 2001.
80. S. Skaria, E. Akleman, and F. Parke. Modeling subdivision control meshes for creating cartoon faces. *Proc. Shape Modeling and Applications 2001*, 216–225, 2001.
81. C. Loop. Triangle mesh subdivision with bounded curvature and the convex hull property. Technical report MSR-TR-2001-24, Microsoft Research, 2001.
82. N. Dyn and E. Farkhi. Spline subdivision schemes for compact sets with metric averages. *Trends in Approximation Theory*, K. Kopotun, T. Lyche and M. Neamtu (eds.), Vanderbilt University Press. Nashville, TN, 93–102, 2001.
83. D. Zorin and P. Schröder. A unified framework for primal/dual quadrilateral subdivision schemes. *Computer Aided Geometric Design* 18, 429–454, 2001.
84. R. Bartels and F. Samavati. Reversing subdivision rules: local linear conditions and observations on inner products. *J. Comp and Appl. Math.*, 119,(1–2), 29–67, 2001.
85. F. Samavati and R. Bartels. Reversing Subdivision using local linear conditions: generating multiresolutions on regular triangular meshes. preprint, <http://www.cgl.uwaterloo.ca/~rhabartel/Papers/TriMesh.pdf>, 2001.

2002

86. N. Dyn and E. Farkhi. Spline subdivision schemes for compact sets – a survey. *Serdica Math. J.* Vol.28(4), 349–360, 2002.
87. M. Alexa. Refinement operators for triangle meshes. *Computer Aided Geometric Design* 19(4), 169–172, 2002.
88. M. Hassan, I. P. Ivrissimtzis, N. A. Dodgson, and M. A. Sabin. An interpolating 4-point C² ternary stationary subdivision scheme. *Computer Aided Geometric Design* 19, 1–18, 2002.
89. L. Barthe, B. More, N. A. Dodgson, and M. A. Sabin. Triquadratic reconstruction for interactive modelling of potential fields. *Proc. Shape Modeling and Applications 2002*, 145–153, 2002.
90. J. Warren and H. Weimer. *Subdivision Methods for Geometric design.*, Morgan Kaufmann, 2002.

91. F. Cirak, M. Scott, E. Antonsson, M. Ortiz, and P. Schröder. Integrated modeling, finite element analysis and engineering design for thin-shell structures using subdivision. *Computer Aided Design* 34, 137–148, 2002.
92. W. Ma, X. Ma, S-K. Tso, and Z. Pan. Subdivision surface fitting from a dense triangle mesh. *Proc. Geometric Modeling and Processing* 2002, 94–103, 2002.
93. M. Sabin. Interrogation of subdivision surfaces. *Handbook of Computer Aided Design*, Chap. 12, 327–341, 2002.
94. N. Dyn and D. Levin. Subdivision schemes in geometric modelling. *Acta Numerica*, 73–144, 2002.
95. C. Bajaj, S. Schaefer, J. Warren, and G. Xu. A subdivision scheme for hexahedral meshes. *The Visual Computer* 18, 343–356, 2002.
96. G. Taubin. Detecting and reconstructing subdivision connectivity. *The Visual Computer* 18, 357–367, 2002.
97. B. Jüttler, U. Schwanecke. Analysis and design of Hermite subdivision schemes. *The Visual Computer* 18, 326–342, 2002.
98. D. Zorin, D. Kristjansson. Evaluation of piecewise smooth subdivision surfaces. *The Visual Computer* 18, 299–315, 2002.
99. C. Loop. Bounded curvature triangle mesh subdivision with the convex hull property. *The Visual Computer* 18, 316–325, 2002.
100. M. Sabin. Subdivision of box-splines. *Tutorials on Multiresolution in Geometric Modelling* [104], 3–23, 2002.
101. N. Dyn. Interpolatory subdivision schemes. *Tutorials on Multiresolution in Geometric Modelling* [104], 25–50, 2002.
102. N. Dyn. Analysis of convergence and smoothness by the formalism of Laurent polynomials. *Tutorials on Multiresolution in Geometric Modelling* [104], 51–68, 2002.
103. M. Sabin. Eigenanalysis and artifacts of subdivision curves and surfaces. *Tutorials on Multiresolution in Geometric Modelling* [104], 69–92, 2002.
104. A. Iske, E. Quak, and M. S. Floater (eds. *Tutorials on Multiresolution in Geometric Modelling*, Springer, ISBN 3-540-43639-1, 2002.
105. N. A. Dodgson, M. A. Sabin, L. Barthe, and M. F. Hassan. Towards a ternary interpolating scheme for the triangular mesh. Technical Report UCAM-CL-TR-539, Computer Laboratory, University of Cambridge, 2002.
106. I. P. Ivrissimtzis, N. A. Dodgson, and M. A. Sabin. A generative classification of mesh refinement rules with lattice transformations. Preprint of [128], Technical Report UCAM-CL-TR-542, Computer Laboratory, University of Cambridge, 2002.
107. Y. Chang, K. McDonnell, and H. Qin. A new solid subdivision scheme based on box splines. *Proc. of the Seventh ACM Symposium on Solid Modeling and Applications*, 226–233, 2002.
108. F. Samavati, N. Mahdavi-Amiri, and R. Bartels. Multiresolution surfaces having arbitrary topologies by a reverse Doo subdivision method. *Computer Graphics Forum* 21, 121–136, 2002.
109. A. Nasri and M. Sabin. A taxonomy of interpolation constraints on recursive subdivision surfaces. *The Visual Computer* 18, 382–403, 2002.

2003

110. P. Prusinkiewicz, F. Samavati, C. Smith, and R. Karwowski. L-system description of subdivision curves. *International Journal of Shape Modeling* 9, 41–59, 2003.
111. N. A. Dodgson, I. P. Ivrissimtzis, and M. A. Sabin. Characteristics of dual triangular $\sqrt{3}$ subdivision. *Curve and Surface Fitting: Saint-Malo 2002* [117], 119–128, 2003.
112. N. Dyn, D. Levin, and J. Simoens. Face value subdivision schemes on triangulations by repeated averaging. *Curve and Surface Fitting: Saint-Malo 2002* [117], 129–138, 2003.
113. B. Han. Classification and construction of bivariate subdivision schemes. *Curve and Surface Fitting: Saint-Malo 2002* [117], 187–198, 2003.
114. M. F. Hassan and N. A. Dodgson. Ternary and Three-point univariate subdivision schemes. *Curve and Surface Fitting: Saint-Malo 2002* [117], 199–208, 2003.
115. C. Loop. Smooth ternary subdivision of triangle meshes. *Curve and Surface Fitting: Saint-Malo 2002* [117], 295–302, 2003.
116. M. Sabin and L. Barthe. Artifacts in recursive subdivision surfaces. *Curve and Surface Fitting: Saint-Malo 2002* [117], 353–362, 2003.
117. A. Cohen, J-L. Merrien, and L. L. Schumaker (eds.). *Curve and Surface Fitting: Saint-Malo 2002*, Nashboro Press, Brentwood, TN, ISBN 0-9728482-1-5, 2003.
118. J. Stam and C. Loop. Quad/triangle subdivision. *Computer Graphics Forum* 22(1), 79–85, 2003.
119. V. Surazhsky and C. Gotsman. Explicit surface remeshing. *Eurographics symposium on geometry processing* [120], 17–27, 2003.
120. L. Kobbelt, P. Schröder, and H. Hoppe (eds. *Eurographics symposium on geometry processing*., Eurographics Association, 2003.
121. A. Levin. Polynomial generation and quasi-interpolation in stationary non-uniform subdivision. *Computer Aided Geometric Design* 20, 41–60, 2003.
122. P. Oswald and P. Schröder. Composite primal/dual sqrt(3) subdivision schemes. *Computer Aided Geometric Design* 20, 135–164, 2003.
123. A. Levin and D. Levin. Analysis of quasi-uniform subdivision. *Applied and Computational Harmonic Analysis* 15, 18–32, 2003.
124. B. Han. Computing the smoothness exponent of a symmetric multivariate refinable function. *SIAM Journal on Matrix Analysis and its Applications*, 24, 693–714, 2003.
125. N. Dyn, D. Levin, and A. Luzzatto. Non-stationary interpolatory subdivision schemes reproducing spaces of exponential polynomials. *Found. Comput. Math.*, 187–206, 2003.
126. M. Sabin and A. Bejancu. Boundary conditions for the 3-direction box-spline *Maths of Surfaces X*, Springer, 2003.
127. A. Ron. Private communication, 2003.

2004

128. I. P. Ivrissimtzis, N. A. Dodgson, and M. A. Sabin. A generative classification of mesh refinement rules with lattice transformations. *Computer Aided Geometric Design* vol 21(1), 99–109, 2004.
129. I. P. Ivrissimtzis, N. A. Dodgson, and M. A. Sabin. $\sqrt{5}$ Subdivision. *Advances in Multiresolution for Geometric Modelling* [135], pages 285–299 (this book), 2004.
130. M. F. Hassan and N. A. Dodgson. Reverse Subdivision. *Advances in Multiresolution for Geometric Modelling* [135], pages 271–283 (this book), 2004.
131. N. Alkalai and N. Dyn. Optimizing 3D triangulations for improving the initial triangulation for the butterfly subdivision scheme. *Advances in Multiresolution for Geometric Modelling* [135], pages 231–244 (this book), 2004.
132. N. Dyn, D. Levin, and M. Marinov. Geometrical interpolation shape-preserving 4-point schemes. *Advances in Multiresolution for Geometric Modelling* [135], pages 301–315 (this book), 2004.
133. L. Barthe, C. Gérot, M. A. Sabin, and L. Kobbelt. Simple computation of the eigencomponents of a subdivision matrix in the frequency domain. *Advances in Multiresolution for Geometric Modelling* [135], pages 245–257 (this book), 2004.
134. C. Gérot, L. Barthe, N. A. Dodgson, and M. A. Sabin. Subdivision as a sequence of sampled C^p surfaces. *Advances in Multiresolution for Geometric Modelling* [135], pages 259–270 (this book), 2004.
135. N. A. Dodgson, M. S. Floater, and M. A. Sabin (eds.). *Advances in Multiresolution for Geometric Modelling*, Springer-Verlag (this book), 2004.

In progress

136. A. Cohen, N. Dyn, and B. Matei. Quasilinear subdivision schemes with applications to ENO interpolation. *Applied and Computational Harmonic Analysis*, to appear.
137. I. P. Ivrissimtzis, N. A. Dodgson, and M. A. Sabin. The support of recursive subdivision surfaces. *ACM Transactions on Graphics*, to appear.
138. B. Han, T. Yu, and Yong-GangXue. Non-interpolatory Hermite subdivision schemes. Preprint.
139. J. Peters and L-J. Shiue. 4-3 Directionally Ripple-free Subdivision. Submitted to *ACM Transactions on Graphics*.
140. N. Dyn and E. Farkhi. Convexification rates in Minkowski averaging processes. In preparation.
141. M. Sabin. A circle-preserving interpolatory subdivision scheme. In preparation.

Optimising 3D Triangulations: Improving the Initial Triangulation for the Butterfly Subdivision Scheme

Nurit Alkalai and Nira Dyn

School of Mathematical Sciences, Tel Aviv University, Israel
`{nalkalai|niradyn}@post.tau.ac.il`

Summary. This work is concerned with the construction of a “good” 3D triangulation of a given set of points in 3D, to serve as an initial triangulation for the generation of a well shaped surface by the butterfly scheme. The optimisation method is applied to manifold meshes, and conserves the topology of the triangulations. The constructed triangulation is “optimal” in the sense that it locally minimises a cost function. The algorithm for obtaining a locally-optimal triangulation is an extension of Lawson’s *Local Optimisation Procedure* (LOP) algorithm to 3D, combined with a priority queue. The first cost function designed in this work measures an approximation of the discrete curvature of the surface generated by the butterfly scheme, based on the normals to this surface at the given 3D vertices. These normals can be expressed explicitly in terms of the vertices and the connectivity between them in the initial mesh. The second cost function measures the deviations of given normals at the given vertices from averages of normals to the surface generated by the butterfly scheme in neighbourhoods of the corresponding vertices. It is observed from numerical simulations that our optimisation procedure leads to good results for vertices sampled from analytic objects. The first cost function is appropriate for analytic surfaces with a large proportion of convex vertices. Furthermore, the optimisation with this cost function improves convex regions in non-convex complex models. The results of optimisation with respect to the second cost function are satisfactory even when all the vertices are non-convex, but this requires additional initial information which is obtainable easily only from analytic surfaces.

1 Introduction

Triangle meshes (triangulations) are commonly used for representing 3D surfaces. Given a set of points sampled from a smooth surface, the triangle mesh with these points as vertices serves as a representation (approximation) of the sampled surface. This representation depends on the choice of the connections between the vertices.

In this work we investigate how to choose a “*good*” triangulation for a fixed set of vertices, to serve as the initial triangulation for the butterfly subdivision scheme, under the assumption that the sampled surface is smooth.

Our procedure supplies a way to obtain a “*good*” triangulation of a given set of vertices from a given triangulation of these vertices. Starting with an initial manifold triangulation, with no boundaries, whose vertices are the 3D points, we use an extension of Lawson’s algorithm [10], which is a local swapping algorithm, to improve the triangulation at each edge swap. For example the top right meshes in Figs. 4 and 5 are improvements on the top left meshes. A similar approach in the case of 2D triangulations is taken in [8].

The choice is of a *locally optimal triangulation* relative to a cost function. We use two cost functions: the first measures a quantity approximating the curvature of the surface generated by the butterfly scheme (*butterfly surface*) based on the normals to the butterfly surface. The second cost function measures the deviations of averages of normals to the butterfly surface near the given vertices, from given normals at these vertices. The normals to the butterfly surface can be computed from the set of given vertices and their triangulation.

Alboul and van Damme, in a series of papers concerned with the generation of a mesh from a cloud of 3D points, consider a cost function which is a discrete analog of the L_1 -norm of the *Gaussian curvature* over the triangular mesh [2]. Although this cost function requires heavy computation, it has an important property. As proved in [2], for data sampled from a convex surface (*convex data*), swapping edges with this cost function leads to the unique global minimum of this cost function which corresponds to the unique convex triangulation of the convex data up to flat areas.

Dyn et al. [7] introduce three cost functions, which measure different kinds of discrete curvature. The computation of these cost functions is much simpler as compared with the cost function in [2] and the results obtained are satisfactory. We use, as in [7], a greedy algorithm of edge swaps, based on a priority queue, so as to maximally reduce the cost function at each step. This algorithm terminates at a local minimum.

We have made numerical experiments with the two cost functions described in Sect. 3 below. From our simulations we conclude that we have at hand a tool which improves significantly the visual appearance and also several quality measurements of the butterfly surface generated from the locally optimal triangulation. The first cost function is appropriate for analytic surfaces with a large proportion of convex vertices. Furthermore, the optimisation with this cost function improves convex regions in non-convex complex models. The results based on the second cost function are satisfactory also for a surface consisting of non-convex vertices only, but it requires additional initial information which is easily obtained only from analytic surfaces.

The outline of this paper is as follows: Sect. 2 presents the theoretical background of our work. In Sect. 3 the cost functions are presented and Sect. 4 summarises our simulations and conclusions.

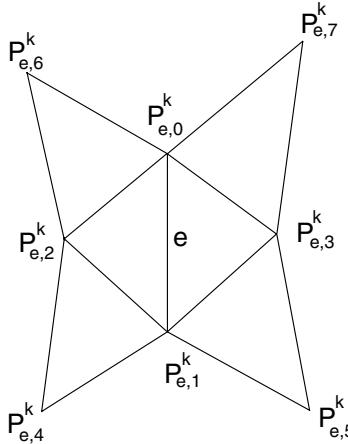


Fig. 1. The points participating in the butterfly insertion rule given by equation (1).

2 Preliminaries

2.1 The Butterfly Subdivision Scheme

The *butterfly* scheme is an interpolatory subdivision scheme for general triangulations of control points [6]. On regular triangulations (where every vertex has valency 6) it generates C^1 limit functions/surfaces, and it is the interpolatory scheme with smallest support of its mask with this property. The differential properties of the surfaces produced by this scheme, in the neighbourhood of the vertices of the initial triangulation, depend on the valency of these vertices.

The butterfly scheme repeatedly refines triangulations, starting from an initial triangulation T^0 and generating a sequence of ever more refined triangulations $\{T^k\}_{k \geq 0}$. Given a set of control points $\{\mathbf{p}_i^k\}$ which comprise the vertices of a triangulation T^k , the scheme associates with every edge $e \in T^k$ a new point \mathbf{q}_e^k defined according to the rule

$$\mathbf{q}_e^k = \frac{1}{2}(\mathbf{p}_{e,0}^k + \mathbf{p}_{e,1}^k) + 2w(\mathbf{p}_{e,2}^k + \mathbf{p}_{e,3}^k) - w \sum_{j=4}^7 \mathbf{p}_{e,j}^k \quad (1)$$

where the locations of the points $\mathbf{p}_{e,j}^k$ relative to the edge e in T^k are depicted in Fig. 1. The parameter w serves as a tension parameter in the sense that decreasing its value to zero is equivalent to tightening the butterfly surface towards the piecewise linear surface T^0 . The butterfly scheme defines the control points at stage $k+1$ (the vertices of T^{k+1}) as

$$\{\mathbf{p}_i^{k+1}\} = \{\mathbf{p}_i^k\} \cup \{\mathbf{q}_e^k : e \in T^k\},$$

and the triangulation T^{k+1} as the collection of edges

$$\{(\mathbf{q}_e^k, \mathbf{p}_{e,j}^k), j = 0, 1, (\mathbf{q}_e^k, \mathbf{q}_{e_{i,j}}^k), i = 0, 1, j = 2, 3 : e \in T^k\},$$

where $e_{i,j} = (\mathbf{p}_{e,i}^k, \mathbf{p}_{e,j}^k)$. With this construction of T^{k+1} , the number of edges having \mathbf{p}_i^k as a vertex in T^{k+1} is the same as in T^k , while each new vertex is regular, namely a vertex with six edges in T^{k+1} . Therefore, with the exclusion of the irregular points in T^0 , all the vertices of T^k are regular.

Necessary conditions for C^1 limit subdivision surfaces near an irregular point are given in [3, 4]. The ranges of the tension parameter w , in which these conditions hold for butterfly surfaces, are numerically computed in [9]. These ranges are presented in Table 1, for vertices with valency between 4 and 7.

Table 1. Ranges of w where necessary conditions for C^1 at an irregular point hold.

| valency | 4 | 5 | 6 | 7 |
|--------------|-----------|-----------|-----------|-----------|
| range of w | (0, 0.08] | (0, 0.10] | (0, 0.12] | (0, 0.14] |

Remark 1. The value $w = \frac{1}{16}$ is included in all four ranges in Table 1. This value of w is the best in terms of approximation properties of the butterfly surface [5]. Yet, at irregular points with valency greater than 7, $w = \frac{1}{16}$ is not included in the corresponding ranges.

Sufficient conditions for C^1 limit surfaces of a subdivision scheme near an irregular point are given in [11]. These conditions hold for the butterfly scheme, as is checked numerically in [12]. A modification of the butterfly insertion rule near an irregular point of valency n is given in [14]. This modification improves the resulting surface in the neighbourhoods of the irregular points.

The normals of the butterfly surface can be calculated at any vertex of any triangulation T^k $k \geq 0$, [12, 13]. The formula for the normal, $\mathbf{n}(\mathbf{p}^m; T^m)$, at a vertex \mathbf{p}^m of T^m of valency n , requires n regular adjacent vertices to \mathbf{p}^m . The normal at \mathbf{p}^m depends on the vertices in the first two rings of vertices around \mathbf{p}^m . To achieve n regular vertices at the first ring of neighbouring vertices it is sufficient to perform locally one subdivision step near \mathbf{p}^m , see Fig. 2. This step depends only on the first two rings of neighbouring vertices around \mathbf{p}^m .

2.2 Optimising Triangulations

In this work we search for a “good” triangulation to serve as the initial triangulation for the butterfly subdivision scheme. Optimising triangulations is a procedure for selecting a “good” triangulation, T^* , according to a certain criterion, from the triangulations that have the same set of vertices, V .

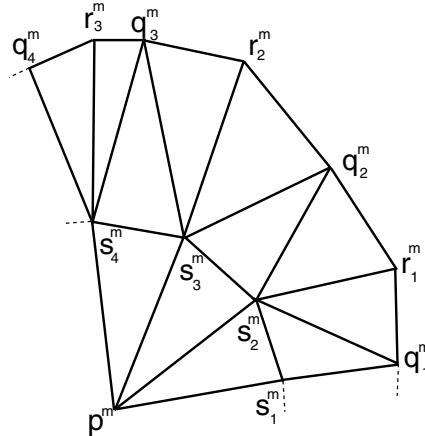


Fig. 2. Control points in the neighbourhood of p^m after one subdivision step. $\{q_i^m\}_{i=1}^n$ are the points of the outer ring that were generated at stage m ; $\{r_i^m\}_{i=1}^n$ are the points of the outer ring that belong to the inner ring of stage $m-1$; $\{s_i^m\}_{i=1}^n$ are the points of the inner ring that were generated at stage m . All these points are regular.

Definition 1. Let V be a set of points in \mathbb{R}^3 . An optimal triangulation, with respect to a given cost function (criterion) $F(T)$, is a triangulation T^* of V such that

$$F(T^*) \leq F(T)$$

for every triangulation T of V .

An optimal triangulation of V always exists since there are finitely many triangulations of V , yet it might be difficult to attain this optimal triangulation in practice. In most cases, a *locally* optimal triangulation can be computed with reasonable effort. Such a triangulation is defined below.

Let T be a triangulation of V , e an internal edge of T , $Q(e; T)$ the four vertices of the two triangles having e as a common edge, and let e^s be the edge connecting the two vertices of $Q(e; T)$ which are not on e . There are two possible ways to triangulate a non-planar $Q(e; T)$ either by e or by e^s .

Definition 2. An edge e is called locally optimal if at least one of the following conditions holds:

1. $F(T) \leq F(\hat{T})$, where \hat{T} is obtained from T by replacing e by e^s .
2. The edge e emanates from a vertex of valency 3, see Fig. 3(b).
3. $Q(e; T)$ is planar and is not strictly convex, see Fig. 3(a).
4. At least one of the two normals to the two triangles of $Q(e^s; T)$ has a different orientation relative to the orientation of the two normals to the two triangles of $Q(e; T)$.

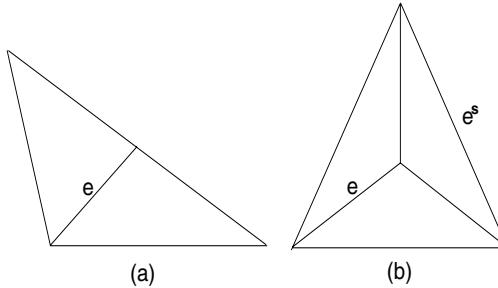


Fig. 3. Geometric situations where an edge e cannot be replaced by e^s .

Definition 3. A locally optimal triangulation is a triangulation T^* in which all edges of T^* are locally optimal.

We use a method for constructing a locally optimal triangulation, which is an extension of Lawson's *LOP* algorithm [10] to 3D manifold meshes. We replace e by e^s , if e is not locally optimal. This is called an *edge swap*.

In view of Definition 2, the locally optimal triangulation obtained by the extended *LOP* (*ELOP*) algorithm is a manifold mesh with the same topology as the initial triangulation.

The *ELOP* Algorithm

Given an initial triangulation T^0 , which is a manifold mesh without boundary:

1. Set $T \leftarrow T^0$.
2. If T is locally optimal, $T^* \leftarrow T$ and end the procedure.
3. Let e be an edge of T which is not locally optimal. Replace e by e^s and go to step 2.

Each time an edge swap occurs, the resulting triangulation has a strictly smaller value of the cost function than the previous one. Since the number of triangulations is finite, *ELOP* converges, after a finite number of edge swaps, to a locally optimal triangulation. The locally optimal triangulation obtained by *ELOP* depends on the specific order in which edges are swapped.

In two cases *LOP* attains the global minimum in \mathbb{R}^2 . These are the *Delaunay triangulation* and the unique *convex triangulation* of convex data. In \mathbb{R}^3 an adapted version of *LOP* [2] attains the *tight triangulation* of vertices sampled from a convex surface.

To accelerate the convergence of the *ELOP* algorithm, at each step we make that swap which maximally reduces the cost function. For that we employ a priority queue as in [7].

A Priority Queue of the Edges of a Triangulation.

Let E be the set of edges of a triangulation T and let $s(e)$, the swap value of an edge e , be defined as the difference between the value of the cost function before and after swapping e with e^s . The priority queue P is a permutation of the set of integers $\{1, 2, \dots, \#E\}$, such that $s(e_{P(i)}) \geq s(e_{P(j)})$ for all $1 \leq i < j \leq \#E$. The main advantages of using such a priority queue are that $P(1)$ is the index of the edge with the largest swap value, and that the condition $s(e_{P(1)}) \leq 0$, indicates that the cost function can not be further reduced by swapping one of the edges. When an edge swap occurs, only a local neighbourhood of edges of the swapped edge change their swap values. The locations of these edges in the priority queue have to be updated.

ELOP and the Butterfly Scheme

In this work we restrict the discussion to finding a good initial triangulation for the butterfly scheme with $w = \frac{1}{16}$. By Remark 1, we allow edge swaps which generate vertices with valencies 4, 5, 6 and 7 only. If in the initial triangulation there is a vertex of valency 3 or of valency greater than 7, the normal to the butterfly surface in such a vertex does not exist, and we use instead the average of the normals to the triangles that belong to the cell of the vertex in the triangulation obtained after two iterations of the butterfly scheme with $w = \frac{1}{16}$.

3 The Cost Functions

The cost functions we employ in the process of optimising the initial triangulation for the butterfly scheme are based on the normals to the butterfly surface, generated from the initial triangulation.

We introduce two cost functions. The first depends on the set of vertices and the current triangulation, while the second depends also on given normals at the vertices $\{\mathbf{n}(v) : v \in V\}$.

To introduce the cost functions we need more notation:

- V is the set of given vertices.
- $T = T(V)$ is a triangulation of V , $T(V)$, consisting of vertices and connectivity.
- $S^k(T)$ is the triangulation after k refinement steps of the butterfly scheme acting on T .
- $C(v; T)$ is the cell of the vertex v in T , i.e. the set of all triangles in T sharing v .
- $|C(v; T)|$ is the number of triangles in the cell $C(v; T)$. i.e. $|C(v; T)|$ is the valency of v in T .
- $\mathbf{n}(\tau)$ is the normal to the triangle τ .

- $S^\infty(T)$ is the butterfly surface generated from the initial triangulation T .
- $\mathbf{n}(v; T)$ is the normal to $S^\infty(T)$ at the vertex $v \in T$.

With this notation the cost functions are

$$F_1(T) = \sum_{v \in V} \frac{1}{|C(v; T)|} \sum_{\tau \in C(v; S^2(T))} \| \mathbf{n}(v; T) - \mathbf{n}(\tau) \|_2,$$

$$F_2(T) = \sum_{e \in E} \sum_{v \in Q(e; T)} \left\| \mathbf{n}(v) - \frac{1}{|C(v; T)|} \sum_{u \in C(v; S^2(T))} \mathbf{n}(u; S^2(T)) \right\|_2.$$

The use of two refinement steps of the scheme in both cost functions is due to the fact that $S^2(T)$ is an adequate approximation for our purposes to $S^\infty(T)$. In fact, in our simulations we represent the butterfly surface by $S^2(T)$.

Minimising $F_1(T)$ flattens the cells of convex vertices, thus generating a smoother surface in the vicinity of convex vertices. Minimising $F_2(T)$ corresponds to approximating the given normals by averages of the normals to $S^\infty(T)$ at points in the neighbourhood of the corresponding vertices. For each edge in the current triangulation, $F_2(T)$ measures the deviations between the given normals in a neighbourhood of the edge and the normals to $S^\infty(T)$ in a similar, but refined, neighbourhood.

4 Results

We have tested our optimisation method on several data sets, both small sets sampled from analytic objects (e.g. torus, sphere) and non-analytic objects given by triangular meshes. In case of large and fine triangular meshes, we used a simplification method to get a simplified model, T^0 , of the object, regarding the given fine mesh as the surface to be approximated by the butterfly surface.

4.1 Analytic Models

The first class of data sets we investigated consists of analytic models, convex and non-convex. Application of F_1 and F_2 in the optimisation procedure in the case of a convex model results in a triangulation which is close to the unique convex triangulation of the set of vertices. Since the unique convex triangulation is the best among all possible triangulations, our results in this case are satisfactory.

As an example of an analytic convex surface we take the sphere. Our optimisation procedure, with either cost function, yields a well shaped butterfly surface. Fig. 4 shows the butterfly surface for this example, generated with F_1 . The butterfly surface generated with F_2 is similar.

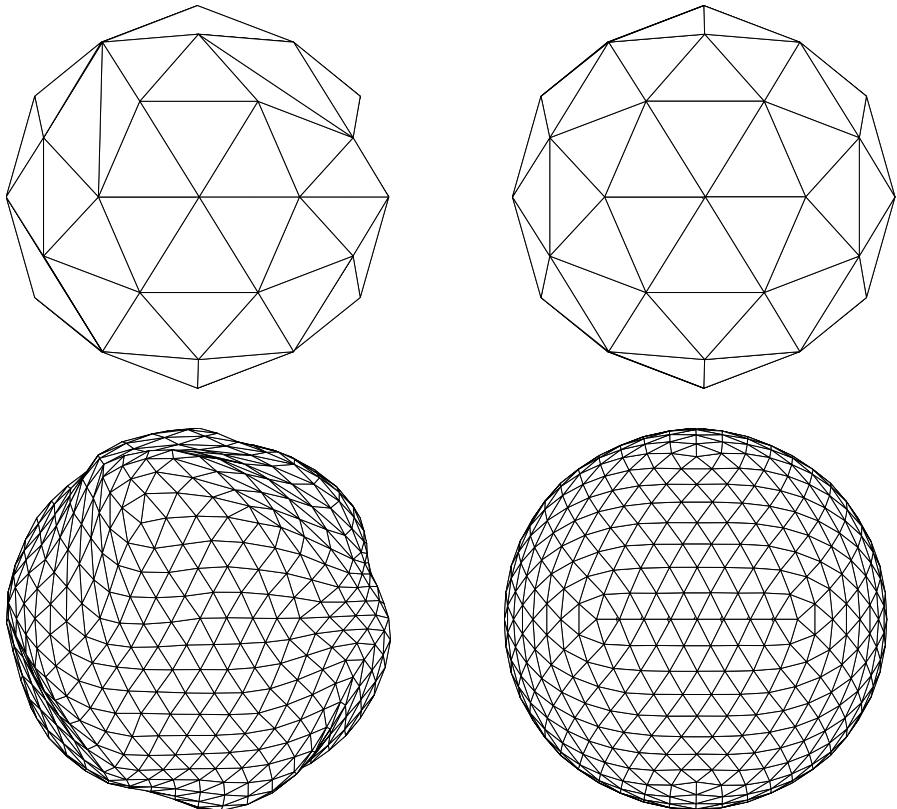


Fig. 4. Example 1 – sphere. Top: original meshes; left T^0 , right T^* . Bottom: approximations to the butterfly surfaces with F_1 as a cost function; left $S^2(T^0)$, right $S^2(T^*)$.

The optimisation of non-convex models with F_1 as the cost function has different qualities (see Figs. 5 and 6). While for the torus (Fig. 5), which consists of many convex and few non-convex vertices, the results are satisfactory, the results for the hyperboloid-like surface (Fig. 6), which consists of non-convex vertices only, are disappointing. This is due to the fact that F_1 is the sum, over all vertices, of a type of a discrete curvature. For convex vertices a decrease in this curvature corresponds to an increase in local smoothness. This is not the case for non-convex vertices. The use of F_2 as the cost function improves the performance of our procedure for Example 3 (compare the right hand meshes in Figs. 6 and 7). The given normals at the given vertices supply additional information on the surface to be approximated by the butterfly surface. Our procedure with F_2 does not perform any edge swaps when the given triangulation T^0 is well shaped, even if non-convex vertices are involved, as in the case of Example 3. This is in contrast to our procedure with

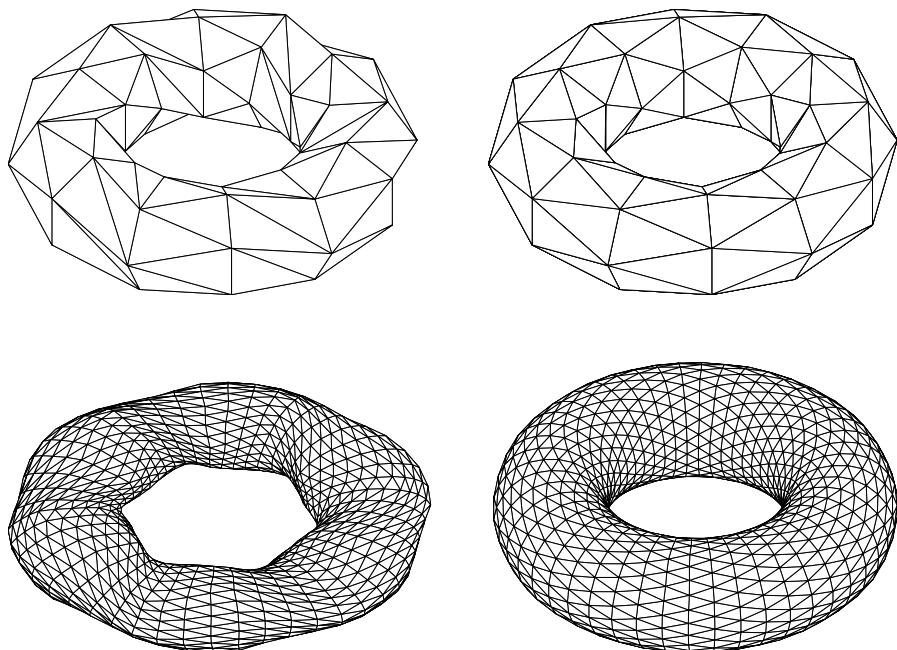


Fig. 5. Example 2 – torus. Top: original meshes; left T^0 , right T^* . Bottom: approximations to the butterfly surfaces with F_1 as a cost function; left $S^2(T^0)$, right $S^2(T^*)$.

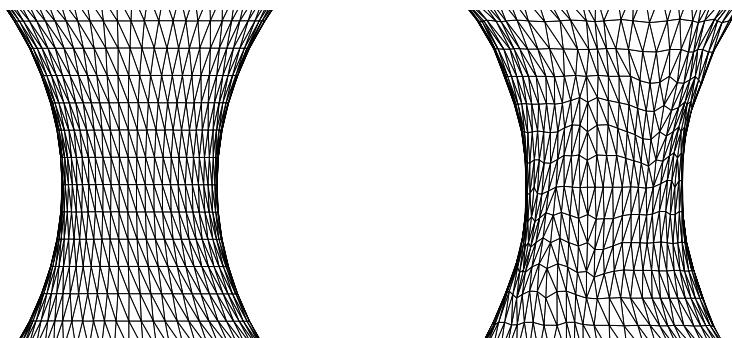


Fig. 6. Example 3 – a hyperboloid-like surface. Approximations to the butterfly surfaces with F_1 as a cost function; left $S^2(T^0)$, right $S^2(T^*)$.

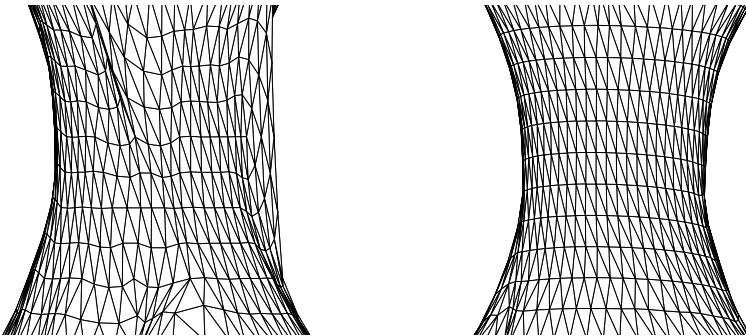


Fig. 7. In this case, T^0 is a randomly perturbed T^0 of Example 3 (Fig. 6). Approximations to the butterfly surfaces with F_2 as a cost function; left $S^2(T^0)$, right $S^2(T^*)$.

F_1 which does not perform any edge swaps only if the triangulation is convex. To see that our procedure with F_2 can improve a “bad” triangulation with non-convex vertices, we randomly changed about 25% of the edges in the initial triangulation of Example 3 and applied our procedure with F_2 . The resulting butterfly surface is significantly improved away from the artificial planar boundaries, as shown in Fig. 7.

4.2 Complex Models

We show one example of a complex model, a head-like mesh. With a triangular mesh of small size as input to our optimisation procedure, we can use only F_1 as the cost function, since normals at the given vertices cannot be estimated. The optimisation procedure succeeds with F_1 in convex regions of the model and fails in non-convex regions, see Fig. 8.

4.3 Measurements

To confirm our observations from the figures, we computed three quantities which measure the quality either of the initial triangulation or of the triangulation obtained from it by two refinement steps with the butterfly scheme. We compare T^* with T^0 or $S^2(T^*)$ with $S^2(T^0)$. The measures are:

- percentage of regular vertices in T^0 and in T^* ;
- Gaussian curvature of $S^2(T^0)$ and of $S^2(T^*)$;
- discrete mean curvature of $S^2(T^0)$ and of $S^2(T^*)$.

Tables 2–4 show these measures. They support our conclusions derived from the visual quality of the surfaces.

Table 2 shows the regularity ratio of a triangulation, which is the ratio between the number of regular vertices and the total number of vertices.

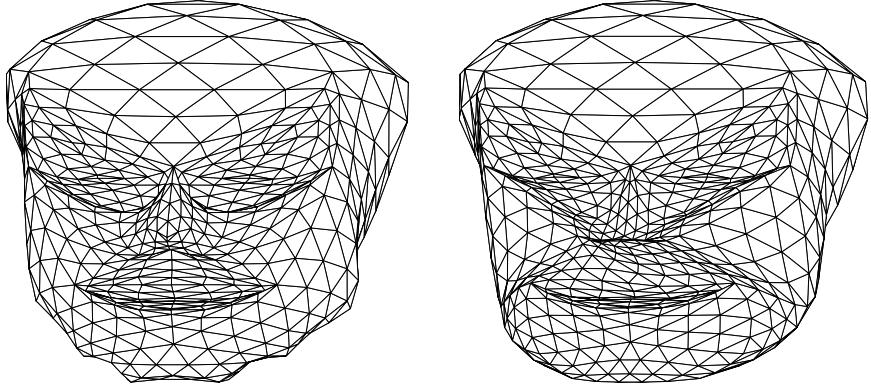


Fig. 8. Example 4 – a head-like mesh. Approximations to the butterfly surfaces with F_1 as a cost function; left $S^2(T^0)$, right $S^2(T^*)$.

Table 2. The regularity ratio as percentage, for T^0 and T^* .

| Mesh | T^0 | F_1 | F_2 |
|------------------|-------|-------|-------|
| | | T^* | T^* |
| Torus | 100 | 100 | 100 |
| Sphere | 36 | 71 | 100 |
| Hyperboloid-like | 69 | 32 | 54 |
| Head-like | 10 | 27 | — |

This demonstrates that a high regularity ratio is in accordance with the visual quality.

Table 3 shows the discrete Gaussian curvature of a triangulation. This is calculated as $G(T) = \sqrt{\sum_{v \in V} [2\pi - \theta(v)]^2}$, where $\theta(v)$ is the sum of all angles between two adjacent edges emanating from v [2].

Table 3. Discrete Gaussian curvature of the butterfly surfaces.

| Mesh | $S^2(T^0)$ | F_1 | F_2 |
|------------------|------------|------------|------------|
| | $S^2(T^*)$ | $S^2(T^*)$ | $S^2(T^*)$ |
| Torus | 1.7002 | 0.8330 | 0.8330 |
| Sphere | 1.0096 | 0.5127 | 0.5127 |
| Hyperboloid-like | 9.8641 | 4.7348 | 2.0178 |
| Head-like | 4.9720 | 3.5796 | — |

Note that our procedure with F_1 roughly halves the discrete Gaussian curvature for all analytic surfaces, while F_2 reduces it by at least as much. So these measures are not appropriate for identifying visually pleasing surfaces.

The discrete mean curvature of a triangulation is calculated as $H(T) = \sqrt{\sum_{v \in V} |\frac{\bar{H}_v}{S_v}|^2}$, where \bar{H}_v is an approximation of the discrete integral mean

curvature at a vertex, suggested by [2], and S_v is the area of the intersection of the Voronoi cell of v relative to $V(T)$ with $C(v; T)$, as suggested by [7]. Table 4 shows this for each of the examples.

Table 4. Discrete mean curvature of the butterfly surfaces.

| Mesh | $S^2(T^0)$ | F_1 $S^2(T^*)$ | F_2 $S^2(T^*)$ |
|------------------|------------|---------------------|---------------------|
| Torus | 23.8434 | 16.4170 | 16.4170 |
| Sphere | 4.5772 | 2.0178 | 2.0178 |
| Hyperboloid-like | 222.6325 | 398.0568 | 102.8953 |
| Head-like | 39.0294 | 38.7954 | — |

The discrete mean curvature changes significantly with the initial triangulation and is in accordance with the visual quality.

Remark 2. All three measurements obtained using the second cost function for Example 3 (the hyperboloid-like surface) are significantly better than those using the first cost function.

5 Conclusion

T^* , obtained by the optimisation procedure from T^0 , can be regarded as an improved triangulation of the given vertices relative to T^0 . Thus our procedure generates a “good” 3D triangular mesh of a set of vertices from a given initial manifold mesh without boundaries.

Acknowledgements

The authors wish to thank David Levin for his help during this research and the editors for their help in improving the paper. This work was supported in part by a grant from the Israeli Academy of Sciences (Centre of Excellence program) and by the European Union research project “Multiresolution in Geometric Modelling (MINGLE)” under grant HPRN-CT-1999-00117.

References

1. Alkalai, N.: Optimizing 3D Triangulations for improving the initial triangulation for the butterfly subdivision scheme. M.Sc. thesis, Tel-Aviv University (2002).
2. Alboul, L. and R. van Damme: Polyhedral metrics in surface reconstruction: tight triangulation. *The Mathematics of Surfaces VII*, T. Goodman and R. Martin (eds), Clarendon Press, Oxford (1997), 309–336.

3. Ball, A. A. and D. J. T. Storry: Conditions for tangent plane continuity over recursively generated B-spline surfaces. *ACM Transaction on Graphics* **7** (1988), 82–102.
4. Doo D. and M. Sabin: Behavior of recursive subdivision surfaces near extraordinary points. *Computer-Aided Design* **10** (1978), 356–360.
5. Dyn N.: Interpolatory subdivision schemes, in *Tutorials on Multiresolution in Geometric Modelling*, A. Iske, E. Quak, and M. S. Floater (eds.), Springer-Verlag, Heidelberg (2000), 25–50.
6. Dyn, N., J. A. Gregory, and D. Levin: A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics* **9** (1990), 160–169.
7. Dyn, N., K. Hormann, S.-J. Kim and D. Levin: Optimizing 3D triangulations using discrete curvature analysis. *Mathematical Methods for Curves and Surfaces*, Oslo 2000 (2001), 135–146. Vanderbilt University Press.
8. Dyn, N., D. Levin and S. Rippa: Data dependent triangulation for piecewise linear interpolation. *IMA J. Numer. Anal.* **10** (1990), 137–154.
9. Hed, S.: Analysis of subdivision schemes for surface generation. M.Sc. thesis, Tel-Aviv University (1992).
10. Lawson, C. L.: Software for C^1 interpolation. *Mathematical Software III*, J. R. Rice (ed.). Academic Press, New York (1977), 161–194.
11. Reif, U.: A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design* **12** (1995), 153–174.
12. Shenkman, P.: Computing normals and offsets of curves and surfaces generated by subdivision schemes. M.Sc. thesis, Tel-Aviv University, (1996).
13. Shenkman, P., N. Dyn and D. Levin: Normals of the butterfly subdivision scheme surfaces and their applications. *Computational and Applied Mathematics* **102** (1999), 157–180.
14. Zorin, D., P. Schröder and W. Sweldens: Interpolating subdivision for meshes with arbitrary topology. *Proc. ACM SIGGRAPH '96*, (1996), 189–192.

Simple Computation of the Eigencomponents of a Subdivision Matrix in the Fourier Domain

Loïc Barthe¹, Cédric Gérot², Malcolm Sabin³, and Leif Kobbelt⁴

¹ Computer Graphics Group, IRIT/UPS, Toulouse, France

lbarthe@irit.fr

² Laboratoire des Images et des Signaux, Domaine Universitaire, Grenoble, France

Cedric.Gerot@lis.inpg.fr

³ Numerical Geometry Ltd., Cambridge, UK

malcolm@geometry.demon.co.uk

⁴ Computer Graphics Group, RWTH Aachen, Germany

kobbelt@cs.rwth-aachen.de

Summary. After demonstrating the necessity and the advantage of decomposing the subdivision matrix in the frequency domain when analysing a subdivision scheme, we present a general framework based on a method, introduced by Ball and Storry, which computes the Discrete Fourier Transform of a subdivision matrix. The efficacy of the technique is illustrated by performing the analysis of Kobbelt's $\sqrt{3}$ scheme in a very simple manner.

1 Introduction

Subdivision surfaces have become a standard technique for both animation and freeform shape modelling [21]. After *one* step of subdivision, a coarse mesh is refined to a finer one and several iterations generate a sequence of incrementally refined meshes which converge to a smooth surface. The main advantage of subdivision surfaces over other freeform representations such as splines [9] is that they are defined by control meshes with arbitrary connectivity while generating smooth surfaces with arbitrary manifold topology. One of the most important stages in subdivision scheme analysis is the evaluation of the scheme's smoothness properties. This is done in *two* steps.

First, one has to study the continuity properties of the scheme in a regular lattice (composed of valence 6 vertices for triangles meshes and valence 4 vertices for quadrilateral meshes). Often the scheme is derived from the uniform knot-insertion operator of some Box-spline surface [2] which leads us to a trivial analysis: by construction the refined meshes converge to piecewise polynomial surfaces with a known degree of smoothness between the patches [3, 5, 11]. On the other hand, the scheme can be non-polynomial [8, 22, 10], i.e. it is not derived from any known surface repre-

sentation and the continuity of the limit surface is analysed using sufficient conditions based on z -transforms [6, 10, 7].

As the second step in the analysis, one has to analyse the scheme's smoothness in the vicinity of extraordinary vertices (EVs). The z -transform is not applicable at EVs, and even though we can prove C^1 continuity for schemes derived from Box-splines by showing that the characteristic map is regular and injective [15, 12, 20], the complete analysis is performed using necessary conditions based on the eigenstructure of the subdivision matrix [5, 1, 16]. In fact, the convergence behaviour of a subdivision scheme at an EV is completely determined by the eigencomponents of its subdivision matrix. The analysis of a subdivision scheme [5, 11, 19, 14, 10] in the vicinity of such irregularities of the control mesh hence requires a simple technique for identifying and computing the various eigencomponents. The standard method exploits the scheme's rotational symmetries through the use of the Fourier transform. This partitions the subdivision matrix, whose size varies linearly with the valence of the EV, into a block diagonal matrix. Although the number of blocks depends on the valence, the blocks are of fixed size, and so it becomes possible to determine the eigencomponents for all valencies with a single algebraic computation.

In this paper, we first illustrate by a practical example the importance of the frequency analysis and we emphasise the necessity of identifying the eigenvalues with respect to their rotational frequency. We then present the general form of Ball and Storry's method [1] which performs fast computation of the different frequency blocks. This approach is illustrated on Kobbelt's $\sqrt{3}$ scheme [10] and we show how very simple computations performed on a single subdivision iteration rather than on the square of the subdivision matrix allow us to deal with the scheme's rotation property and to find the specific subdivision rules for the EVs.

Another method for computing the eigencomponents of a subdivision matrix is based on z -transforms and it exploits the circulant structure of the subdivision matrix's blocks. This technique also leads to very simple computations and all details can be found in [12, 18]. Both methods are equivalent in terms of complexity, however our method computes the eigencomponents in the Fourier domain while the use of z -transforms provides the eigencomponents in the spatial domain. Depending on the application, one or the other method may be preferable.

2 Frequency of the Different Eigenvalues

The operator which maps a central EV of valence v and its r -ring neighbourhood \mathbf{P} to the same topological configuration \mathbf{p} after *one* step of subdivision is called the subdivision matrix S . The vectors of old vertices \mathbf{P} and new vertices \mathbf{p} are linked by the relation $\mathbf{p} = S\mathbf{P}$. The matrix S is square and each of its rows contains the coefficients of an affine combination of the old vertices \mathbf{P} which computes *one* new vertex of \mathbf{p} . The convergence behaviour of the

subdivision scheme at the central EV is completely defined by the eigencomponents (eigenvalues and eigenvectors) of the matrix S . The matrix S is then decomposed into $S = M \Lambda M^{-1}$ where Λ is a diagonal matrix of eigenvalues $\{\lambda_j\}$ and M is a square matrix whose columns are the (*right*) eigenvectors \mathbf{m}_j . This can be well understood if we interpret the eigencomponents as a local Taylor expansion. Indeed, this interpretation allows us to associate the different geometric configurations (position, tangent plane, curvature) with the eigencomponents by which they are defined. The smoothness analysis then relies on necessary conditions for the eigencomponents of the different geometric configurations.

As we will see in Sect. 2.1, given just the eigendecomposition of a subdivision matrix, we cannot directly deduce which eigenvalue corresponds to which geometric configuration so that we do not know how to apply the conditions for the scheme's smoothness analysis. In Sect. 2.2 we show how the decomposition of the subdivision matrix in the Fourier domain resolves this problem.

2.1 Geometric Configurations and their Eigencomponents

The Taylor expansion of a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ can be written as follows:

$$\begin{aligned} f(x, y) = & f + f_x x + f_y y + (f_{xx} + f_{yy}) \left(\frac{x^2}{4} + \frac{y^2}{4} \right) \\ & + (f_{xx} - f_{yy}) \left(\frac{x^2}{4} - \frac{y^2}{4} \right) + f_{xy} xy + \dots , \end{aligned} \quad (1)$$

where each function expression on the right hand side is evaluated at $(0, 0)$. The point f is a position, the *two* first order partial derivatives f_x and f_y are the coefficients of x and y defining the tangent plane and the *three* second order partial derivatives f_{xx} , f_{yy} and f_{xy} are the coefficients of three quadratic configurations defining the curvature: an elliptic configuration $x^2 + y^2$, which we denote as *cup*, and *two* rotationally symmetric hyperbolic configurations $x^2 - y^2$ and xy which we denote as *saddle*.

On the other hand, the vector of new vertices \mathbf{p} is expressed as a local Taylor expansion when it is computed as $\mathbf{p} = S \mathbf{P} = M \Lambda M^{-1} \mathbf{P} = M \Lambda \mathbf{l}$ with $\mathbf{l} = M^{-1} \mathbf{P}$ [13]. We then have:

$$\mathbf{p} = \mathbf{m}_0 \lambda_0 l_0 + \mathbf{m}_1 \lambda_1 l_1 + \mathbf{m}_2 \lambda_2 l_2 + \mathbf{m}_3 \lambda_3 l_3 + \mathbf{m}_4 \lambda_4 l_4 + \mathbf{m}_5 \lambda_5 l_5 + \dots , \quad (2)$$

where the $l_j \in \mathbb{R}^3$ are the approximations of the Taylor coefficients, i.e. the successive partial derivatives (l_0 is a position, l_1 and l_2 approximate the first order derivatives, etc), the \mathbf{m}_j 's correspond to the polynomials in Eq. (1), whose function values scale with the factors λ_j . Eq. (1) and (2) both behave like local Taylor expansions applied in different contexts. They have identical geometric interpretation and in Eq. (2), the components \mathbf{m}_j , λ_j and l_j with

index $j = 0$ are responsible for the central EV's position; components with indices $j = 1, 2$ are responsible for the tangent plane; and those with indices $j = 3, 4, 5$ are responsible for the curvature: $j = 3$ for the cup and $j = 4, 5$ for the *two* saddles.

The study of the subdivision scheme's smoothness at the EV is based on necessary conditions affecting the different eigencomponents $\{\lambda_j\}$ and $\{\mathbf{m}_j\}$. For instance, the condition $1 = |\lambda_0| > |\lambda_j|$ for all $j > 0$ is necessary for convergence of the scheme, the additional conditions $1 > |\lambda_1|$, $1 > |\lambda_2|$ and $\min(|\lambda_1|, |\lambda_2|) > |\lambda_j|$ for $j > 2$ are necessary for C^1 continuity. However, in practice, useful schemes are images under rotation and therefore we restrict our analysis to this special case. This yields the additional properties : $|\lambda_1| = |\lambda_2|$ and $|\lambda_4| = |\lambda_5|$. Properties like bounded curvature ($|\lambda_2|^2 \geq |\lambda_3|$, $|\lambda_2|^2 \geq |\lambda_4| = |\lambda_5|$ and $\min(|\lambda_3|, |\lambda_4|) > |\lambda_j|$ for $j > 5$) are necessary for C^2 continuity. If $|\lambda_2|^2 = |\lambda_3| = |\lambda_4| = |\lambda_5|$, the scheme has a *non-zero* bounded curvature (it has not got a flat spot at the EV) and if $|\lambda_2|^2 > |\lambda_j|$, $j = 3, 4, 5$, the scheme has *zero* curvature generating a flat spot at the EV.

A critical point in the analysis after eigendecomposition of the subdivision matrix is then to identify which index (or configuration) corresponds to which eigencomponents. Since we know which eigenvalue corresponds to which eigenvector, the task is reduced to the identification of the different eigenvalues. This is illustrated by the following example.

Let us consider a variant of Loop's subdivision scheme [11] having its n -tuple of eigenvalues $(\lambda_0, \dots, \lambda_7)$ at a valence 7 EV sorted by geometric configuration (following our Taylor notation (2)) and having their value in the set:

$$\left\{ 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \lambda_j < \frac{1}{4} \right\} \quad (3)$$

We emphasise that the set of eigenvalues is not sorted from the greatest to the smallest as it is usually done, but by geometric configuration. The question is: how can we know which one of the eigenvalues in (3) is $\lambda_0, \lambda_1, \lambda_2, \dots$? Indeed, following (2), each order satisfies different properties. For instance, the eigenvalues can have the following values:

$$(\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7) = \left(1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, < \frac{1}{4}, < \frac{1}{4} \right), \quad (4)$$

and hence one can deduce that the scheme is certainly C^1 and that it has bounded curvature. If indeed the analysis of the characteristic map proves C^1 continuity, the scheme is C^1 at the EV and it has bounded curvature without flat spot (first row in Fig. 1). However if the eigenvalues are actually:

$$(\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7) = \left(1, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, < \frac{1}{4}, < \frac{1}{4} \right), \quad (5)$$

the necessary condition for C^1 continuity is not satisfied ($|\lambda_2| < |\lambda_4|$) so that the scheme is not C^1 and it is not even necessary to analyse the characteristic

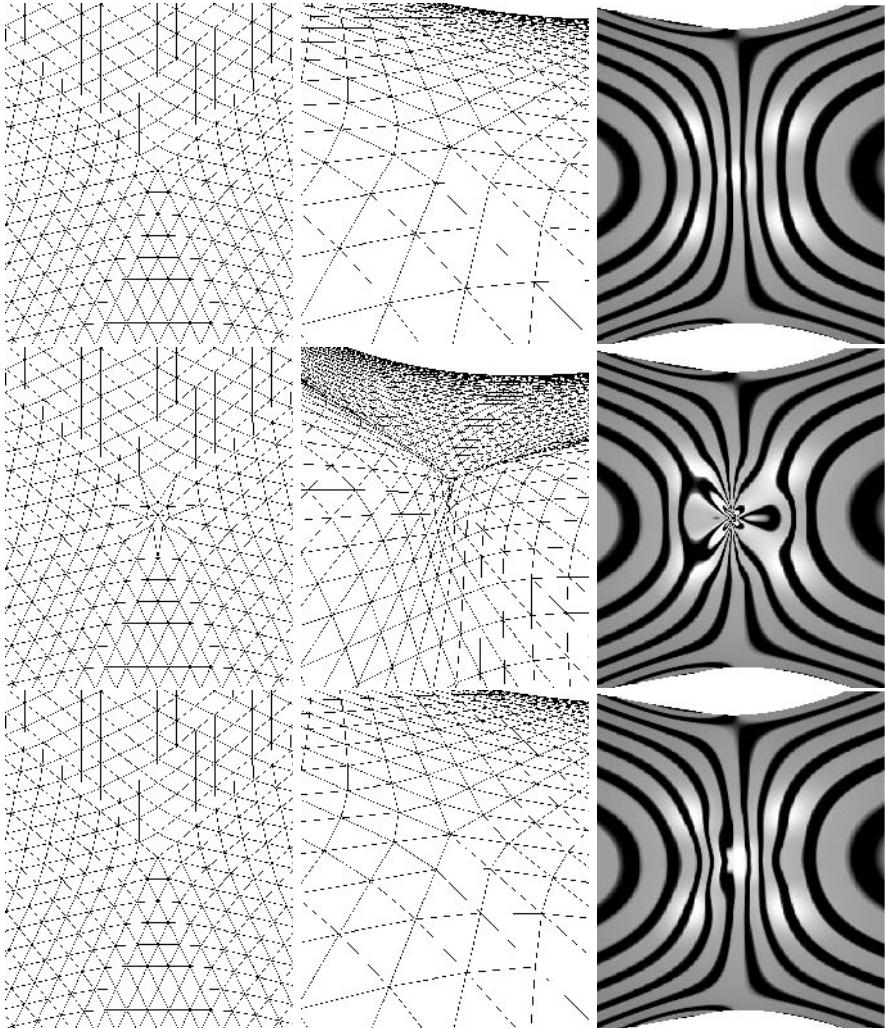


Fig. 1. Different versions of Loop’s scheme at a valence 7 EV. The first column shows the subdivision of a semi-regular planar mesh with the EV in its centre. The second column shows a subdivided saddle mesh with the EV in its centre and the third column illustrates the curvature behaviour using reflection lines on the same saddle mesh. In the first row, the scheme has the eigenspectrum (4) and it is C^1 continuous with non-zero bounded curvature (no flat spot). In the second row, it has the eigenspectrum (5). We clearly see the shrinking factor of $\lambda_1 = \lambda_2 = \frac{1}{4}$ at the EV in the planar configuration and the C^1 discontinuity in the saddle mesh. In the third row it has the eigenspectrum (6) and the scheme is C^1 continuous with bounded curvature (flat spot in the saddle configuration). The misbehaviour of the curvature is illustrated by the reflection lines in the saddle mesh.

map (second row in Fig. 1). Finally, the situation where :

$$(\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7) = \left(1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \lambda_4 < \frac{1}{4}, \lambda_5 < \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right), \quad (6)$$

is more problematic because if the analysis of the characteristic map proves that the scheme is C^1 , one could conclude from the eigenspectrum (3) that it has also bounded curvature without flat spot at the EV ($|\lambda_1|^2 = |\lambda_2|^2 = |\lambda_3| = |\lambda_4| = |\lambda_5|$) while this is not the case. In the saddle configuration, the curvature is bounded with a flat spot, and some curvature misbehaviour can be introduced in the limit surface by the eigenvectors corresponding to the eigenvalues λ_6 and λ_7 (third row in Fig. 1).

The decomposition of the subdivision matrix in the Fourier domain will allow us to determine if we are in the situation (4), (5), (6) or in a situation where the eigenvalues are sorted in a different manner.

2.2 Identification of the Eigenvalues in the Fourier Domain

The identification of the eigenvalues is based on the decomposition of the subdivision matrix S in the Fourier domain. The block circulant subdivision matrix S with n blocks $S_{i,j}$ is transformed into a block diagonal matrix \tilde{S} having v blocks \tilde{S}_ω which correspond to the *rotational frequencies* $\omega = \{0, \dots, v-1\}$ ordered in frequency in \tilde{S}_ω (as illustrated in Equation (7)). We define *rotational frequency* below. The eigenvalues of the frequency blocks \tilde{S}_ω are amplitudes of the eigenvalues of the matrix S [1, 17] hence, if we know which frequency represents each configuration (position, tangent plane, cup and saddle), we can identify the eigenvalues from the frequency block in which they are computed.

$$S = \begin{bmatrix} & & & \\ \hline & S_{0,0} & \cdots & S_{0,n} \\ \hline & \vdots & \ddots & \vdots \\ \hline & \vdots & \ddots & \vdots \\ \hline & S_{n,0} & \cdots & S_{n,n} \end{bmatrix} \longrightarrow \tilde{S} = \begin{bmatrix} [\tilde{S}_0] & & & 0 \\ & [\tilde{S}_1] & & \\ & & \ddots & \\ 0 & & & [\tilde{S}_{v-1}] \end{bmatrix} \quad (7)$$

In the Taylor expansion (1), the constant term refers to a position, the terms for x and y define the tangent plane and terms in $x^2 + y^2$, $x^2 - y^2$ and xy define the different curvature configurations (cup and two saddles). The expression of these configurations in a cylindrical coordinate system (Eq. (8)) as a periodic function $g(\theta) = \rho \cos(\omega\theta + \varphi)$ (where ρ is the amplitude, ω is the frequency and φ is the phase) directly provides the rotational frequency ω associated to each configuration. For instance, $x^2 - y^2 = r^2 \cos(2\theta)$ and hence this saddle configuration has a frequency component $\omega = 2$. Note that,

due to rotational symmetry, $\tilde{S}_\omega = \tilde{S}_{v-\omega}$ so that it is enough to consider the frequencies $\omega = \{0, \dots, \frac{v}{2}\}$.

$$(x, y, z) \rightarrow (r, \theta, z) \quad \text{with} \quad \begin{cases} x = r \cos(\theta) \\ y = r \sin(\theta) \\ z = z \end{cases} \quad (8)$$

This tells us that the position configuration has the frequency $\omega = 0$ and hence the eigenvalue $|\lambda_0|$ is the dominant eigenvalue of the frequency block \tilde{S}_0 , the tangent plane configuration has the frequency $\omega = 1$ and $|\lambda_1| = |\lambda_2|$ equals the dominant eigenvalue $\tilde{\mu}_1$ of the frequency block \tilde{S}_1 , the cup configuration has the frequency $\omega = 0$ and $|\lambda_3|$ equals the subdominant eigenvalue $\tilde{\mu}_0$ of the frequency block \tilde{S}_0 and finally the saddle configurations have the frequency $\omega = 2$ and $|\lambda_4| = |\lambda_5|$ equals the dominant eigenvalue $\tilde{\mu}_2$ of the frequency block \tilde{S}_2 . This relation between the different eigencomponents and their rotational frequencies is presented in [5].

When a scheme is convergent, its eigenvalue λ_0 equals 1 (Sect. 2.1) and since $|\lambda_0|$ is the dominant eigenvalue of the frequency block \tilde{S}_0 , it can be written as:

$$\tilde{S}_0 = \begin{bmatrix} 1 & \cdots \\ 0 & [\tilde{S}'_0] \end{bmatrix}. \quad (9)$$

In this paper, we only consider convergent subdivision schemes, and so $\lambda_0 = 1$ and the cup eigenvalue $|\lambda_3|$ is the dominant eigenvalue $\tilde{\mu}_0$ of the block \tilde{S}'_0 .

3 Computation of the Frequency Blocks

In this section, we present a general framework, used in [1] on the Catmull-Clark scheme [3], which computes the eigencomponents in the frequency domain. We present the procedure on a triangular lattice with a 2-ring neighbourhood around the central EV and a standard dyadic refinement (see Fig. 2(a)). The adaptation to a single 1-ring neighbourhood or to quadrilateral meshes [1] is straightforward. We notice that this method may not be suitable for analysing non-rotationally symmetric schemes, however, as pointed out in [4], these schemes are mainly of theoretical interest and all the schemes used in practical applications are rotationally symmetric.

The set of old vertices \mathbf{P} is defined as follows:

$$\mathbf{P} = \{A, B_0, \dots, B_{v-1}, C_{\frac{1}{2}}, \dots, C_{v-\frac{1}{2}}, D_0, \dots, D_{v-1}\},$$

where the different letters $X = \{B, C, D\}$ denote the different sets of rotationally symmetric vertices around the central EV A and the indices give the rotational position of each vertex: vertex X_j is at an angle of $\frac{j2\pi}{v}$ (where v is the valence of the EV) from the axis of origin. The set of new vertices \mathbf{p} :

$$\mathbf{p} = \{a, b_0, \dots, b_{v-1}, c_{\frac{1}{2}}, \dots, c_{v-\frac{1}{2}}, d_0, \dots, d_{v-1}\},$$

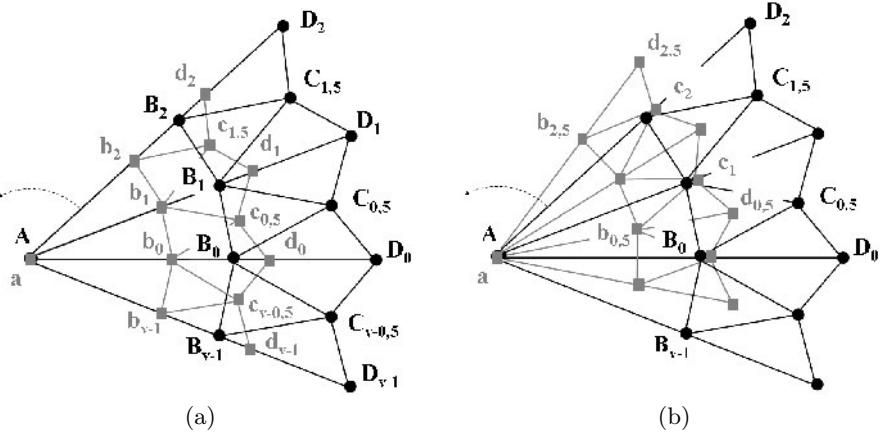


Fig. 2. Subdivision with (a) a dyadic refinement and (b) a $\sqrt{3}$ refinement, of a 2-ring configuration around a central EV A of valence v . Capital letters denote the set of old vertices \mathbf{P} and small letters the set of new vertices \mathbf{p} .

represents the same topological configuration, but after *one* step of subdivision and around the central EV a . The new vertices of \mathbf{p} are computed using affine combinations g_k^j of the old vertices of \mathbf{P} :

$$a = g^0(\mathbf{P}), \quad b_k = g_k^1(\mathbf{P}), \quad c_{k+\frac{1}{2}} = g_k^2(\mathbf{P}), \quad d_k = g_k^3(\mathbf{P}), \quad k = 0, \dots, v-1,$$

where the affine combinations g_k^j are the rows of the subdivision matrix S .

Using the Discrete Fourier Transform (DFT):

$$\tilde{x}_\omega = \frac{1}{v} \sum_{l=0}^{v-1} x_l \exp(-2\pi i \omega l/n), \quad x = \{a, b, c, d\}, \quad (10)$$

we express the rotational frequencies $\{\tilde{a}_\omega, \tilde{b}_\omega, \tilde{c}_\omega, \tilde{d}_\omega\}$ of each set of rotationally symmetric new vertices of \mathbf{p} in terms of the rotational frequencies $\tilde{\mathbf{P}}_\omega = \{\tilde{A}_\omega, \tilde{B}_\omega, \tilde{C}_\omega, \tilde{D}_\omega\}$ of the old vertices of \mathbf{P} :

$$\tilde{a}_\omega = \tilde{g}_\omega^0(\tilde{\mathbf{P}}_\omega), \quad \tilde{b}_\omega = \tilde{g}_\omega^1(\tilde{\mathbf{P}}_\omega), \quad \tilde{c}_\omega = \tilde{g}_\omega^2(\tilde{\mathbf{P}}_\omega), \quad \tilde{d}_\omega = \tilde{g}_\omega^3(\tilde{\mathbf{P}}_\omega),$$

where the affine combinations \tilde{g}_ω^j are the rows of the frequency blocks \tilde{S}_ω of the discrete Fourier transform \tilde{S} of the subdivision matrix S . In vertices A and a , the only frequency is the zero frequency, hence the terms in $\exp(-2\pi i \omega l/v)$ vanish in the expression of \tilde{a}_ω and we have: $A = \tilde{A}_0 = \tilde{A}$, $a = \tilde{a}_0 = \tilde{a}$ and $\forall \omega > 0$, $\tilde{A}_\omega = \tilde{a}_\omega = 0$. Furthermore, in order to express the frequency block \tilde{S}_0 as in Equation (9), we have to centre the analysis at the central EV so that:

$$\begin{bmatrix} \tilde{a}_0 \\ \tilde{b}_0 - \tilde{a}_0 \\ \tilde{c}_0 - \tilde{a}_0 \\ \tilde{d}_0 - \tilde{a}_0 \end{bmatrix} = \begin{bmatrix} & \tilde{S}_0 & \end{bmatrix} \begin{bmatrix} \tilde{A}_0 \\ \tilde{B}_0 - \tilde{A}_0 \\ \tilde{C}_0 - \tilde{A}_0 \\ \tilde{D}_0 - \tilde{A}_0 \end{bmatrix} \text{ with } \begin{bmatrix} \tilde{S}_0 \\ \tilde{S}'_0 \end{bmatrix} = \begin{bmatrix} 1 & \cdots \\ 0 & \begin{bmatrix} \tilde{S}'_0 \end{bmatrix} \end{bmatrix},$$

hence

$$\begin{bmatrix} \tilde{b}_0 - \tilde{a}_0 \\ \tilde{c}_0 - \tilde{a}_0 \\ \tilde{d}_0 - \tilde{a}_0 \end{bmatrix} = \begin{bmatrix} & \tilde{S}'_0 & \end{bmatrix} \begin{bmatrix} \tilde{B}_0 - \tilde{A}_0 \\ \tilde{C}_0 - \tilde{A}_0 \\ \tilde{D}_0 - \tilde{A}_0 \end{bmatrix},$$

and because for $\omega > 0$: $\tilde{A}_\omega = \tilde{a}_\omega = 0, \forall \omega > 0$ we have:

$$\begin{bmatrix} \tilde{b}_\omega \\ \tilde{c}_\omega \\ \tilde{d}_\omega \end{bmatrix} = \begin{bmatrix} & \tilde{S}_\omega & \end{bmatrix} \begin{bmatrix} \tilde{B}_\omega \\ \tilde{C}_\omega \\ \tilde{D}_\omega \end{bmatrix}.$$

The size of the frequency blocks is equal to the number of sets of rotationally symmetric vertices in the neighbourhood of the central EV. Hence, these blocks are of fixed size (here they are 3×3 matrices) and they can be expressed as a function of the valence v of the central EV. The original problem of computing the eigencomponents of large $(3v+1) \times (3v+1)$ matrices (*in the spatial domain*) for each value of the valence v is reduced to a single eigendecomposition of small 3×3 matrices (*in the Fourier domain*), producing the different eigencomponents expressed in terms of the valence v . Depending on their complexity, the frequency blocks can be either decomposed by hand computations or using the symbolic toolbox of any mathematical software.

We note that the right eigenvectors $\tilde{\mathbf{m}}_0$, $\tilde{\mathbf{m}}_1$ and $\tilde{\mathbf{m}}_2$ associated respectively with the eigenvalues $\tilde{\mu}_0$, $\tilde{\mu}_1$ and $\tilde{\mu}_2$, can be interpreted as amplitudes of the eigenvectors $\mathbf{m}_0, \dots, \mathbf{m}_5$ of the spatial domain, e.g. the tangent plane eigenvector $\tilde{\mathbf{m}}_1 = [r_B, r_C, r_D]$ gives the radii r_X of vertices B_k , $C_{k+\frac{1}{2}}$ and D_k from the central EV A ($X \in \{B, C, D\}$).

4 Example Using Kobbelt's $\sqrt{3}$ Scheme

We illustrate our method using the practical example of Kobbelt's $\sqrt{3}$ subdivision scheme [10], the application of the general procedure presented in Sect. 3. This scheme rotates the lattice after *one* step of subdivision due to the insertion of new vertices in the middle of the old faces (as we can see in Fig. 2(b)). If the eigendecomposition is performed on the subdivision matrix S , we obtain complex eigencomponents generated by the scheme's rotation property. To overcome this difficulty in [10] the scheme is analysed after *two* steps of subdivision so that the lattice is aligned with the *two* steps older one but rotated by *one* sector. The lattice is then rotated back using a permutation matrix R , and the matrix studied finally is $\hat{S} = R S^2$.

We first compute the frequency blocks on the 2-ring configuration shown in Fig. 2(b). The computation is performed on a single subdivision step without any back-rotation. We then reproduce the results presented in [10] using the eigencomponents in the Fourier domain. We will see that the computations are so simple that they can be done quickly and easily by hand.

Kobbelt's $\sqrt{3}$ scheme is composed of *two* refinement rules (*stencils*): one which displaces an old vertex (Eq. (11)) and one which computes a new vertex in the centre of a triangle (Eq. (12)). They are defined by the following formulae:

$$a = (1 - \alpha_v)A + \frac{\alpha_v}{v} \sum_{j=0}^{v-1} B_j \quad (11)$$

$$b_{k+\frac{1}{2}} = \frac{1}{3}(A + B_k + B_{k+1}), \quad (12)$$

where α_v is a parameter which can be used to improve the surface smoothness at the EV for different valencies v . The new vertices c_k are computed using the regular relaxation rule (valence 6 vertex) and the new vertices $d_{k+\frac{1}{2}}$ using the insertion rule as follows:

$$\begin{aligned} c_k &= \frac{2}{3}B_k + \frac{1}{18}(A + B_{k-1} + B_{k+1} + C_{k-\frac{1}{2}} + C_{k+\frac{1}{2}} + D_k) \\ d_{k+\frac{1}{2}} &= \frac{1}{3}(B_k + B_{k+1} + C_{k+\frac{1}{2}}). \end{aligned}$$

The DFT (Eq. (10)) is then used to derive the rotational frequencies of the different sets of rotationally symmetric vertices:

$$\begin{aligned} \tilde{a}_0 &= 1 \cdot \sum_{j=0}^0 a e^0 = (1 - \alpha_v)A + \frac{\alpha_v}{v} \sum_{j=0}^v B_j \quad \text{using the DFT and Eq. (11)} \\ &= (1 - \alpha_v)\tilde{A}_0 + \alpha_v\tilde{B}_0 \quad \text{because the only frequency in } a \text{ is } \omega = 0 \end{aligned}$$

$$\begin{aligned}
\tilde{b}_\omega &= \frac{1}{v} \sum_{j=0}^{v-1} b_j e^{-2\pi i \omega j/v} \quad \text{DFT} \\
&= \frac{1}{v} \sum_{j=0}^{v-1} \left[\frac{1}{3} (A + B_{j-\frac{1}{2}} + B_{j+\frac{1}{2}}) \right] e^{-2\pi i \omega j/v} \quad \text{using Eq. (12)} \\
&= \frac{1}{3} A + \frac{1}{3v} \sum_{l=-\frac{1}{2}}^{v-\frac{3}{2}} B_l e^{-2\pi i \omega (l+\frac{1}{2})/v} + \frac{1}{3v} \sum_{l=\frac{1}{2}}^{v-\frac{1}{2}} B_l e^{-2\pi i \omega (l-\frac{1}{2})/v} \\
&= \frac{1}{3} A + \frac{1}{3v} e^{-2\pi i \omega / 2v} \sum_{l=-\frac{1}{2}}^{v-\frac{3}{2}} B_l e^{-2\pi i \omega l/v} + \frac{1}{3v} e^{2\pi i \omega / 2v} \sum_{l=\frac{1}{2}}^{v-\frac{1}{2}} B_l e^{-2\pi i \omega l/v} \\
&= \frac{1}{3} \tilde{A}_0 + \frac{1}{3} (e^{-\pi i \omega / v} + e^{\pi i \omega / v}) \tilde{B}_\omega \\
&= \frac{1}{3} \tilde{A}_0 + \frac{2}{3} k_\omega \tilde{B}_\omega \quad \text{with } k_\omega = \cos\left(\frac{\pi \omega}{v}\right) \text{ and because } e^{-i\theta} + e^{i\theta} = 2 \cos \theta
\end{aligned}$$

hence

$$\begin{cases} \tilde{b}_0 - \tilde{a}_0 = \left(\frac{2}{3} - \alpha_v\right) (\tilde{B}_0 - \tilde{A}_0) & \text{if } \omega = 0 \\ \tilde{b}_\omega = \frac{2}{3} k_\omega \tilde{B}_\omega & \text{otherwise.} \end{cases}$$

Using similar computations, we obtain:

$$\begin{cases} \tilde{c}_0 - \tilde{a}_0 = \left(\frac{7}{9} - \alpha_v\right) (\tilde{B}_0 - \tilde{A}_0) + \frac{1}{9} (\tilde{C}_0 - \tilde{A}_0) + \frac{1}{18} (\tilde{D}_0 - \tilde{A}_0) & \text{if } \omega = 0 \\ \tilde{c}_\omega = \left(\frac{2}{3} + \frac{1}{9} k_{2\omega}\right) \tilde{B}_\omega + \frac{1}{9} k_\omega \tilde{C}_\omega + \frac{1}{18} \tilde{D}_\omega & \text{otherwise} \end{cases}$$

$$\begin{cases} \tilde{d}_0 - \tilde{a}_0 = \left(\frac{2}{3} - \alpha_v\right) (\tilde{B}_0 - \tilde{A}_0) + \frac{1}{3} (\tilde{C}_0 - \tilde{A}_0) & \text{if } \omega = 0 \\ \tilde{d}_\omega = \frac{2}{3} k_\omega \tilde{B}_\omega + \frac{1}{3} \tilde{C}_\omega & \text{otherwise.} \end{cases}$$

From these expressions, we directly deduce the frequency blocks:

$$\tilde{S}'_0 = \begin{bmatrix} \frac{2}{3} - \alpha_v & 0 & 0 \\ \frac{7}{9} - \alpha_v & \frac{1}{9} & \frac{1}{18} \\ \frac{2}{3} - \alpha_v & \frac{1}{3} & 0 \end{bmatrix} \quad \text{and if } \omega > 0 \quad \tilde{S}_\omega = \begin{bmatrix} \frac{2}{3} k_\omega & 0 & 0 \\ \frac{2}{3} + \frac{1}{9} k_{2\omega} & \frac{1}{9} k_\omega & \frac{1}{18} \\ \frac{2}{3} k_\omega & \frac{1}{3} & 0 \end{bmatrix}.$$

Since the only non-zero coefficient of the first rows of the matrices \tilde{S}'_0 and \tilde{S}_ω is the one in the left-hand corner, we directly obtain the dominant eigenvalues $\tilde{\mu}_0$, $\tilde{\mu}_1$ and $\tilde{\mu}_2$ as:

$$\tilde{\mu}_0 = \frac{2}{3} - \alpha_v, \quad \tilde{\mu}_1 = \frac{2}{3} k_1, \quad \tilde{\mu}_2 = \frac{2}{3} k_2. \quad (13)$$

As we see, the free parameter α_v only appears in the cup eigenvalue, hence the scheme's behaviour is improved at the EV by bounding the curvature in the cup configuration using the condition: $\tilde{\mu}_0 = \tilde{\mu}_1^2$. The value of parameter α_v in Eq. (11) is then derived as follows:

$$\tilde{\mu}_0 = \tilde{\mu}_1^2 \iff \frac{2}{3} - \alpha_v = \left(\frac{2}{3} k_1 \right)^2 \iff \alpha_v = \frac{2}{9} \left(2 - \cos \left(\frac{2\pi}{v} \right) \right).$$

The eigenvalues $\tilde{\mu}_i$ ($i = 1, 2, 3$) could have been computed in an even simpler manner by only considering a 1-ring neighbourhood around the central EV. However, the choice of a 2-ring neighbourhood is based on our wish of giving an example allowing a more complete analysis based on a larger neighbourhood. It also allows us to check that more of the eigenvalues are adequately sorted (see the necessary conditions on the eigenvalues in Sect. 2.1).

5 Conclusion

In this paper, we have emphasised the importance of the analysis of the subdivision matrix in the Fourier domain: the analysis of large matrices in the spatial domain is reduced to the analysis of small matrices in the Fourier domain so that it becomes easier to compute the different eigencomponents. Moreover, we can determine which geometric configuration is defined by which eigencomponents. We have presented a general framework computing the subdivision matrix in the frequency domain and it has been illustrated on the practical example of Kobbelt's $\sqrt{3}$ scheme. We have shown how this computation technique allows us to analyse the scheme in a very simple manner.

There are limitations for this computation technique when the rotational position of a set of rotationally symmetric vertices with respect to the axis of origin is unknown. More investigations have to be carried out to solve this problem while keeping the computations as simple as possible. As we have demonstrated however, this approach performs well on 2-ring neighbourhood configurations and hence it is very well suited to analyse all of the standard subdivision schemes or any new rotationally symmetric scheme.

Acknowledgement

This work was supported in part by the European Union research project "Multiresolution in Geometric Modelling (MINGLE)" under grant HPRN-CT-1999-00117.

References

1. Ball, A.A., Storry, D.J.T.: Conditions for tangent plane continuity over recursively generated B-spline surfaces. *ACM Trans. Graphics*, **7**(2):83–102 (1988)

2. de Boor, C., Hollig, D., Riemenschneider, S.: *Box Splines*. Springer-Verlag, New York (1994).
3. Catmull, E., Clark, J.: Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, **10**(6):350–355 (1978)
4. Dodgson N.A., Ivrissimtzis, I.P., Sabin, M.A.: Characteristics of dual triangular $\sqrt{3}$ subdivision. *Curve and Surface Fitting: Saint-Malo 2002*, Nashboro Press, Brentwood, 119–128 (2003)
5. Doo, D., Sabin, M.A.: Analysis of the behaviour of recursive subdivision surfaces near extraordinary points. *Computer Aided Design*, **10**(6):356–360 (1978)
6. Dyn, N.: Subdivision schemes in Computer-Aided Geometric Design. *Advances in Numerical Analysis II: Wavelets, Subdivision Algorithms and Radial Basis Functions*, W. Light (ed.), Clarendon Press, Oxford, 36–104 (1992)
7. Dyn, N.: Analysis of convergence and smoothness by the formalism of Laurent polynomials. *Tutorials on Multiresolution in Geometric Modelling*, A. Iske, E. Quak and M. Floater (eds.), Springer, 51–68 (2002)
8. Dyn, N., Levin, D., Gregory, J.: A butterfly subdivision scheme for surface interpolation with tension control. *ACM Trans. Graphics*, **9**(2):160–169 (1990)
9. Farin, G.: *Curves and Surfaces for CAGD. 5th Edition*, Academic Press (2002)
10. Kobbelt, L.: $\sqrt{3}$ -Subdivision. *Proc. ACM SIGGRAPH 2000*, 103–112 (2000)
11. Loop, C.: Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah (1987)
12. Peters, J., Reif, U.: Analysis of algorithms generalizing B-spline subdivision. *SIAM Journal of Num. Anal.*, **35**(2):728–748 (1998)
13. Prautzsch, H.: Analysis of C_k -Subdivision surfaces at extraordinary points. Technical Report 98/4, Fakultät für Informatik, University of Karlsruhe, Germany (1998)
14. Prautzsch, H., Umlauf, G.: A G^2 -subdivision algorithm. *Geometric Modelling, Dagstuhl 1996, Computing supplement 13*, Springer-Verlag, 217–224 (1998)
15. Reif, U.: A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Design*, **12**:153–174 (1995)
16. Sabin, M.A.: Eigenanalysis and artifacts of subdivision curves and surfaces. *Tutorials on Multiresolution in Geometric Modelling*, A. Iske, E. Quak and M. Floater (ed.), Springer, 69–97 (2002)
17. Stam, J.: Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. *Proc. ACM SIGGRAPH '98*, 395–404 (1998)
18. Warren, J., Weimer, H.: *Subdivision methods for geometric design: A constructive approach*. San Francisco: Morgan Kaufman, (2002)
19. Zorin, D.: Stationary subdivision and multiresolution surface representations. PhD thesis, Caltech, Pasadena, California (1997)
20. Zorin, D.: A method for analysis of C^1 -continuity of subdivision surfaces. *SIAM Journal of Num. Anal.*, **35**(5):1677–1708 (2000)
21. Zorin, D., Schröder, P.: Subdivision for modeling and animation. *SIGGRAPH 2000 Course Notes*. (2000)
22. Zorin, D., Schröder, P., Sweldens, W.: Interpolating subdivision for meshes with arbitrary topology. *Proc. ACM SIGGRAPH '97*, 189–192 (1996)

Subdivision as a Sequence of Sampled C^p Surfaces

Cédric Gérot¹, Loïc Barthe², Neil A. Dodgson³, and Malcolm Sabin⁴

¹ Laboratoire des Images et des Signaux, Domaine Universitaire, Grenoble, France
`Cedric.Gerot@lis.inpg.fr`

² Computer Graphics Group, IRIT/UPS, Toulouse, France
`lbarthe@irit.fr`

³ Computer Laboratory, University of Cambridge, UK
`nad@ccl.cam.ac.uk`

⁴ Numerical Geometry Ltd., Cambridge, UK
`malcolm@geometry.demon.co.uk`

Summary. This article deals with practical conditions for tuning a subdivision scheme in order to control its artifacts in the vicinity of a mark point. To do so, we look for good behaviour of the limit vertices rather than good mathematical properties of the limit surface. The good behaviour of the limit vertices is characterised with the definition of C^2 -convergence of a scheme. We propose necessary explicit conditions for C^2 -convergence of a scheme in the vicinity of any mark point being a vertex of valency greater or equal to three.

1 Introduction

A bivariate subdivision scheme defines a sequence of polygonal meshes each of whose vertices is a linear combination of vertices belonging to the previous mesh in the sequence. Such a scheme is interesting if the sequence converges to a surface which is as regular as possible. Tuning is the choosing of coefficients for the linear combinations used in constructing new points. This article deals with conditions which may be used for tuning a scheme in order to get such a sequence.

Some schemes (Loop [7], Catmull-Clark [4], Doo-Sabin [5],...) are defined so that each polygonal mesh is the control polyhedron of a Box-Spline surface which is the limit surface of the sequence. In this case the convergence and regularity problems are solved by definition, except around extraordinary vertices. An extraordinary vertex is a vertex of the mesh whose valency is not equal to six if the mesh faces are triangles, or not equal to four if the mesh faces are quadrilaterals. For extraordinary vertices in a Box-Spline based scheme and for all vertices in other schemes, the convergence of the scheme and the regularity of its limit surface in the vicinity of a vertex need to be

analysed. This analysis may lead to a tuning of the scheme. In most cases, the coefficients of the linear combinations depend only on the local topology of the mesh, and not on its geometry. Moreover, we assume the scheme to be stationary: the coefficients remain the same through the sequence of polygonal meshes.

The first analysis of the behaviour of the limit surface around an extraordinary vertex was by Doo and Sabin [5]. They give necessary conditions for a scheme being convergent towards a C^2 -continuous limit surface. These conditions are derived from estimates of its first and second derivatives around the extraordinary vertex. Subsequently, most researchers have interpreted this question as follows: the mesh around but excluding the extraordinary vertex is the control polyhedron of continuous patches. At each subdivision step, a new ring of such patches fills in a part of the n -sided hole created by the virtual removal of the extraordinary vertex. The researchers analyse how this iterative insertion of new rings converges and completely fills in this hole. Ball and Storry [1] give sufficient conditions for the surface being tangent plane continuous. Reif [11] remarks that, in terms of differential geometry, a surface is C^p -continuous if there exists a C^p function which parameterizes it. If the limit surface is tangent plane continuous, the surface can be parameterized over a characteristic map of this tangent plane around the extraordinary vertex. Reif gives necessary and sufficient conditions for any stationary scheme to be convergent towards a C^1 -continuous surface. Independently, Prautzsch [9] and Zorin [16] proposed in the late 90's necessary and sufficient conditions for a scheme to be convergent towards a C^p -continuous surface. They both use more or less the parameterization over the characteristic map proposed by Reif.

Most of the prior work on tuning subdivision schemes alters local coefficients in order to fulfil the aforementioned necessary and sufficient conditions [13]. But mathematical C^p -continuity of the limit surface is perhaps not the best target to aim for. We may look for good behaviour of the limit vertices rather than good mathematical properties of the limit surface. Good behaviour of the limit vertices may mean fewer artifacts on the limit surface [14]. For instance, Prautzsch and Umlauf tuned the Loop and Butterfly schemes in order to make them C^1 and C^2 -continuous around an extraordinary vertex by creating a flat spot [10]; but a flat spot may be considered as an artifact. Furthermore, the necessary and sufficient conditions for C^2 -continuity of the limit surface are not explicit if the scheme is not Box-Spline based.

In this paper, we characterise the good behaviour of the limit vertices with the definition of C^2 -convergence of a scheme. This definition is based on the interpretation proposed implicitly by Doo and Sabin [5]. Each control mesh is viewed not as the control polyhedron of a Box-Spline surface but as the sampling of a continuous surface. Thus the sequence of meshes are samplings of a sequence of continuous surfaces which converges uniformly towards the limit surface. Naturally, C^2 -convergence of a scheme is related

to the C^2 -continuity of the limit surface: it is a sufficient condition for it. And because the definition of C^2 -convergence of a scheme is *theoretical* and formal, we propose in this paper explicit but only necessary conditions for C^2 -convergence.

In the following section, we present the theoretical tools we use in Sect. 3 to establish the necessary conditions for a scheme to C^2 -converge.

2 Theoretical Tools

We first describe our notation and then we propose the definition for the C^p -convergence of a scheme. From this definition, we derive a description of the limit points. Finally the eigenanalysis of the Fourier transformed subdivision matrix gives a description of the limit frequencies.

2.1 Notation

We study the convergence of a subdivision scheme towards a regular surface in the vicinity of a vertex which is a mark point. A mark point is a point of a mesh whose vicinity keeps the same topology throughout subdivision. For instance, mesh vertices are mark points in the case of Loop or Catmull-Clark subdivision schemes, and face centres but not mesh vertices are mark points in the case of Doo-Sabin refinement. As a consequence, our analysis does not apply to Doo-Sabin nor to other schemes where the vertices are not mark points. The generalisation of this analysis to any mark point being a vertex or a face centre can be found in a technical report by the same authors [6] which also contains detailed proofs of the results presented here.

Let A be the mark point, and n its valency (number of outgoing edges from A). We assume that the vicinity of A is made up of ordinary vertices. This hypothesis is relevant because after a subdivision step, the vertices of the mesh map to vertices with the same valency, and new vertices are created which are all ordinary. Thus, after several subdivision steps, every extraordinary vertex is surrounded by a sea of ordinary vertices. As a consequence, the vicinity of A may be divided into n topologically equivalent sectors. In the j th sector, let $B_j, C_j, D_j \dots$ be an infinite number of vertices sorted from the topologically nearest vertex from A to the farthest. If there exist two vertices in one sector on the same ring which are in complementary positions then they are labelled with the same letter, but with a prime put on the vertex which is further anticlockwise from the positive x -axis. An example is E and E' in Fig. 1. However, if the points are not in complementary positions, then they are given distinct letters.

Let $A^{(k)}$ be the mark point and $\{B_j^{(k)}, C_j^{(k)}, D_j^{(k)} \dots\}_{j \in 1 \dots n}$ its vicinity after k subdivision steps. All these vertices are put into an infinite vector

$$\mathbf{P}^{(k)} := \left[A^{(k)} B_1^{(k)} \dots B_n^{(k)} C_1^{(k)} \dots C_n^{(k)} D_1^{(k)} \dots D_n^{(k)} \dots \right]^T$$

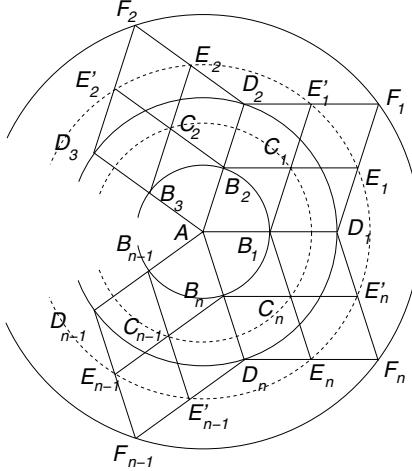


Fig. 1. Labelling of the vicinity of a mark point

Finally, a surface is C^p -continuous if there exists a C^p -diffeomorphic parameterization of it from a subset of \mathbb{R}^2 . We define a parameterization domain by projecting onto \mathbb{R}^2 without folding the polygonal mesh around the mark point. $A^{(k)}$ is projected onto $(0, 0)$, and $\forall X \in \{B, C, D, \dots\}$, $\forall j \in \{1, \dots, n\}$, $X_j^{(k)}$ is projected onto $(x_j^{(k)}, y_j^{(k)})$. For simplicity, we ask $(x_j^{(k)}, y_j^{(k)})$ to lie on the same circle for given k and X , and to lie on the same radial axis for given j and X :

$$(x_j^{(k)}, y_j^{(k)}) := (\varrho_X^{(k)} \cos(\theta_{(X,j)}), \varrho_X^{(k)} \sin(\theta_{(X,j)})) ,$$

where

$$\theta_{(X,j)} := \frac{2\pi}{n}(j + \alpha_X) ,$$

Furthermore, because the vertices $X_j^{(k)}$ converge to the limit mark point if the scheme converges [11], we ask that $\lim_{k \rightarrow \infty} (\varrho_X^{(k)}) = 0$. The choice of the phases α_X and the radii $\varrho_X^{(k)}$ remains free. These degrees of freedom will be used in the characterisation of C^1 -convergence in Sect. 3.

2.2 C^p -Convergence and Behaviour of the Limit Points

We propose the following definition for the C^p -convergence of a scheme. The scheme C^p -converges in the vicinity of A if

- for every X in the infinite vicinity $\{B, C, D, \dots\}$ of A , there exist phases α_X and, for all j in $\{1, \dots, n\}$, for every k , radii $\varrho_X^{(k)}$ and a C^p -continuous function $\mathcal{F}^{(k)}(x, y)$ such that

$$\begin{aligned} A^{(k)} &= \mathcal{F}^{(k)}(0, 0) , \\ X_j^{(k)} &= \mathcal{F}^{(k)}(\varrho_X^{(k)} \cos(\theta_{(X,j)}), \varrho_X^{(k)} \sin(\theta_{(X,j)})) . \end{aligned}$$

- Furthermore, the sequence of p th differentials $(d^p \mathcal{F}^{(k)})_k$ converges uniformly onto $d^p \mathcal{F}$ which is the p th differential of a C^p -continuous parameterization $\mathcal{F}(x, y)$ of the limit surface in the vicinity of the limit mark point.
- Finally, for all $q \in 0, \dots, p-1$, the sequence $(d^q \mathcal{F}^{(k)}(0, 0))_k$ converges onto $d^q \mathcal{F}(0, 0)$.

In this definition, an infinite vicinity $\{B, C, D, \dots\}$ is taken into account. In any practical application, we will consider only a finite number of vertices. More precisely, we choose the set of vertices which will influence the limit position of the mark point and its neighbourhood. This practical restriction is not inconsistent with finding only necessary conditions for C^p -convergence. From the definition, we see that if the scheme C^p -converges in the vicinity of A , then the sequence of meshes converges towards a C^p -continuous surface around the limit mark point. But the converse is not true: a scheme, which converges towards a C^p -continuous surface is not *necessarily* C^p -convergent. Note also that the definition domain of $\mathcal{F}^{(k)}$ shrinks as k grows since $\lim_{k \rightarrow \infty} (\varrho_X^{(k)}) = 0$ from Sect. 2.1.

In Sect. 3 we will consider the necessary conditions for C^2 -convergence. Therefore, consider a scheme which C^2 -converges in the vicinity of the mark point A . The parameterization $\mathcal{F}(x, y)$ is C^2 -continuous. From its Taylor expansion around $(0, 0)$, we may describe the behaviour of the limit points in the vicinity of A . In the following lines, we detail this behaviour with derivatives of the limit function and according to the regularity of the scheme convergence. If the scheme C^0 -converges then $\forall X \in \{B, C, D, \dots\}, \forall j \in \{1, \dots, n\}$,

$$\lim_{k \rightarrow \infty} (A^{(k)}) = \mathcal{F}(0, 0) , \quad \text{and} \quad \lim_{k \rightarrow \infty} (X_j^{(k)}) = \mathcal{F}(0, 0) . \quad (1)$$

If the scheme C^1 -converges then $\forall X \in \{B, C, D, \dots\}, \forall j \in \{1, \dots, n\}$,

$$\lim_{k \rightarrow \infty} \left(\frac{X_j^{(k)} - \mathcal{F}^{(k)}(0, 0)}{\varrho_X^{(k)}} \right) = \cos(\theta_{(X, j)}) \frac{\partial \mathcal{F}}{\partial x}(0, 0) + \sin(\theta_{(X, j)}) \frac{\partial \mathcal{F}}{\partial y}(0, 0) . \quad (2)$$

If the scheme C^2 -converges then $\forall X \in \{B, C, D, \dots\}, \forall j \in \{1, \dots, n\}$,

$$\begin{aligned} \lim_{k \rightarrow \infty} \left(\frac{\Delta_{X, j}^{(k)}}{\varrho_X^{(k)2}} \right) &= \left(\frac{\partial^2 \mathcal{F}}{\partial x^2}(0, 0) + \frac{\partial^2 \mathcal{F}}{\partial y^2}(0, 0) \right) \frac{1}{4} + \frac{\partial^2 \mathcal{F}}{\partial x \partial y}(0, 0) \frac{\sin(2\theta_{(X, j)})}{2} \\ &\quad + \left(\frac{\partial^2 \mathcal{F}}{\partial x^2}(0, 0) - \frac{\partial^2 \mathcal{F}}{\partial y^2}(0, 0) \right) \frac{\cos(2\theta_{(X, j)})}{4} . \end{aligned} \quad (3)$$

with

$$\Delta_{X, j}^{(k)} := X_j^{(k)} - \mathcal{F}^{(k)}(0, 0) - \varrho_X^{(k)} (\cos(\theta_{(X, j)}) \frac{\partial \mathcal{F}^{(k)}}{\partial x}(0, 0) + \sin(\theta_{(X, j)}) \frac{\partial \mathcal{F}^{(k)}}{\partial y}(0, 0))$$

2.3 Eigenanalysis of the Transformed Subdivision Matrix

Consideration of the relationship between the spatial and frequency domains allows us to produce necessary conditions for C^2 -convergence. In this section we introduce the necessary notation for the subdivision matrix transformed into the frequency domain. We may write the discrete rotational frequencies $\tilde{X}^{(k)}(\omega)$ of each set of vertices $\{X_j^{(k)}\}_{j \in 1 \dots n}$ by applying a Discrete Fourier Transform. It is well-known [1] that there exists a matrix $\tilde{\mathbf{M}}(\omega)$ such that for all ω in $\{-\lfloor \frac{n-1}{2} \rfloor, \dots, \lfloor \frac{n}{2} \rfloor\}$,

$$\tilde{\mathbf{P}}^{(\mathbf{k+1})}(\omega) = \tilde{\mathbf{M}}(\omega) \tilde{\mathbf{P}}^{(\mathbf{k})}(\omega)$$

where, if $\omega \neq 0$,

$$\tilde{\mathbf{P}}^{(\mathbf{k})}(\omega) := \left[\tilde{B}^{(k)}(\omega) \tilde{C}^{(k)}(\omega) \tilde{D}^{(k)}(\omega) \dots \right]^T$$

and otherwise

$$\tilde{\mathbf{P}}^{(\mathbf{k})}(0) := \left[\tilde{A}^{(k)}(0) \tilde{B}^{(k)}(0) \tilde{C}^{(k)}(0) \tilde{D}^{(k)}(0) \dots \right]^T.$$

For every discrete rotational frequency ω , the matrix $\tilde{\mathbf{M}}(\omega)$ is supposed to be non defective (otherwise we should use the canonical Jordan form)

$$\tilde{\mathbf{M}}(\omega) = \tilde{\mathbf{V}}(\omega)^{-1} \tilde{\Lambda}(\omega) \tilde{\mathbf{V}}(\omega)$$

where the columns $\tilde{\mathbf{v}}_l(\omega)$ of $\tilde{\mathbf{V}}(\omega)^{-1}$ are the right eigenvectors of $\tilde{\mathbf{M}}(\omega)$, the rows $\tilde{\mathbf{u}}_l^T(\omega)$ of $\tilde{\mathbf{V}}(\omega)$ are the left eigenvectors of $\tilde{\mathbf{M}}(\omega)$, and $\tilde{\Lambda}(\omega)$ is diagonal whose diagonal components $\tilde{\lambda}_l(\omega)$ are the eigenvalues of $\tilde{\mathbf{M}}(\omega)$, with $l \geq 1$. Let $L_l^-(\omega)$, $L_l(\omega)$, and $L_l^+(\omega)$ be sets of indices such that

$$\begin{aligned} & \text{if } q \in L_l^-(\omega) \text{ then } |\tilde{\lambda}_q(\omega)| < |\tilde{\lambda}_l(\omega)|, \\ & \text{if } q \in L_l(\omega) \text{ then } |\tilde{\lambda}_q(\omega)| = |\tilde{\lambda}_l(\omega)|, \\ & \text{if } q \in L_l^+(\omega) \text{ then } |\tilde{\lambda}_q(\omega)| > |\tilde{\lambda}_l(\omega)|, \end{aligned}$$

where $|\tilde{\lambda}|$ is the modulus of the complex number $\tilde{\lambda}$. Then, with $\mathcal{P}(q, \omega) = \tilde{u}_q(\omega)^T \tilde{\mathbf{P}}^{(0)}(\omega)$, we get for every l ,

$$\begin{aligned} \tilde{\mathbf{P}}^{(\mathbf{k})}(\omega) &= \sum_{q \in L_l^+(\omega)} \tilde{\lambda}_q(\omega)^k \mathcal{P}(q, \omega) \tilde{\mathbf{v}}_q(\omega) \\ &= \tilde{\lambda}_l(\omega)^k \left(\sum_{q \in L_l(\omega)} \mathcal{P}(q, \omega) \tilde{\mathbf{v}}_q(\omega) + \sum_{q \in L_l^-(\omega)} \left(\frac{\tilde{\lambda}_q(\omega)}{\tilde{\lambda}_l(\omega)} \right)^k \mathcal{P}(q, \omega) \tilde{\mathbf{v}}_q(\omega) \right). \end{aligned} \tag{4}$$

Thus, as k grows to infinity,

$$\tilde{\lambda}_l(\omega)^k \sum_{q \in L_l(\omega)} \mathcal{P}(q, \omega) \tilde{\mathbf{v}}_q(\omega)$$

is a good estimate of the frequency

$$\tilde{\mathbf{P}}^{(k)}(\omega) = \sum_{q \in L_l^+(\omega)} \tilde{\lambda}_q(\omega)^k \mathcal{P}(q, \omega) \tilde{\mathbf{v}}_q(\omega)$$

in the same way that

$$\sum_{a+b=l} \frac{x^a y^b}{l!} \frac{\partial^l \mathcal{F}}{\partial x^a \partial y^b}(0, 0)$$

is a good estimate of the function

$$\mathcal{F}(x, y) = \sum_{a+b < l} \frac{x^a y^b}{l!} \frac{\partial^l \mathcal{F}}{\partial x^a \partial y^b}(0, 0)$$

as (x, y) converges to $(0, 0)$.

3 Necessary Conditions for C^2 -Convergence and Derivatives of the Limit Surface

Equations (1), (2) and (3) describe the behaviour of the limit points. Applying the Discrete Fourier Transform on these equations gives a description of the limit frequencies. The consistency between this description and the one given by equation (4) implies necessary conditions for the C^2 -convergence of the scheme. It also gives the partial derivatives of the limit surface in the mark point. As a notation, if $\tilde{X}^{(k)}(\omega)$ is the m th component of $\tilde{\mathbf{P}}^{(k)}(\omega)$, then $(\tilde{v}_l(\omega))_X$ is the m th component of $\tilde{\mathbf{v}}_l(\omega)$. We assume also without any restriction that for every fixed ω , $\tilde{\lambda}_2(\omega)$ is the eigenvalue of $\tilde{\mathbf{M}}(\omega)$ with the greatest modulus after $\tilde{\lambda}_1(\omega)$ and any other eigenvalues with same modulus as $\tilde{\lambda}_1(\omega)$: for all ω , $L_1(\omega) = L_2^+(\omega)$.

3.1 C^0 -Convergence

If the scheme C^0 -converges, then

$$\begin{cases} \tilde{\lambda}_1(0) = 1 & , \\ |\tilde{\lambda}_1(\omega)| < 1 & \text{if } \omega \neq 0, \end{cases}$$

and if $L_1(0) = \{1\}$,

$$(\tilde{v}_1(0))_X = \nu_0$$

with ν_0 being a constant, and

$$\mathcal{F}(0, 0) = \frac{\mathcal{P}(1, 0)}{n} (\tilde{v}_1(0))_X .$$

Not only do we get necessary conditions on eigenvalues and eigenvectors of $\tilde{\mathbf{M}}(\omega)$, but we get also the value of $\mathcal{F}(0, 0)$, that is the limit mark point.

3.2 C^1 -Convergence

If the scheme C^1 -converges and the mark point is a vertex, then

$$|\tilde{\lambda}_2(0)| < |\tilde{\lambda}_1(\pm 1)| \text{ and } |\tilde{\lambda}_1(\omega)| < |\tilde{\lambda}_1(\pm 1)| ,$$

with $\omega \notin \{-1, 0, 1\}$.

Furthermore, when k is *large*, if $L_1(1) = L_1(-1) = \{1\}$, the moduli of the eigencomponents $|(\tilde{v}_1(1))_X|$ and $|(\tilde{v}_1(-1))_X|$ are sorted like the radii, ϱ_X , of the rings.

If the scheme is rotationally invariant, the modulus of the eigenvalue $|\tilde{\lambda}_1(1)| = |\tilde{\lambda}_1(-1)|$ gives the speed of the parameters' shrinkage during the subdivision process. The freedom we had in the choice of the parameters $\varrho_X^{(k)}$ and the phase α_k is restricted. But there remains enough freedom to write things quite simply. For simplicity, we can define the radii $\varrho_X^{(k)}$ as follows:

$$\varrho_X^{(k)} = \left| \tilde{\lambda}_1(1) \right|^k |(\tilde{v}_1(1))_X| = \left| \tilde{\lambda}_1(-1) \right|^k |(\tilde{v}_1(-1))_X| .$$

Furthermore, if we define α_X as

$$\alpha_X = \frac{n}{2\pi} \varphi_{(\tilde{v}_1(-1))_X}$$

with $\varphi_{(\tilde{v}_1(-1))_X}$ being the phase of $(\tilde{v}_1(-1))_X$, and if

$$\frac{\partial \mathcal{F}}{\partial x}(0, 0) \pm i \frac{\partial \mathcal{F}}{\partial y}(0, 0) \neq 0 ,$$

then

$$\begin{cases} \frac{\partial \mathcal{F}}{\partial x}(0, 0) = \frac{2}{n} \Re(\mathcal{P}(1, 1)) = \frac{2}{n} \Re(\mathcal{P}(1, -1)) , \\ \frac{\partial \mathcal{F}}{\partial y}(0, 0) = \frac{2}{n} \Im(\mathcal{P}(1, 1)) = -\frac{2}{n} \Im(\mathcal{P}(1, -1)) , \end{cases}$$

with $\Re(\mathcal{P}(1, 1))$ and $\Im(\mathcal{P}(1, 1))$ being respectively the real and the imaginary parts of $\mathcal{P}(1, 1)$.

In conclusion, the necessary conditions for C^1 -convergence are dominance of the main eigenvalues $\tilde{\mathbf{M}}(1)$ and $\tilde{\mathbf{M}}(-1)$, after the main eigenvalue from $\tilde{\mathbf{M}}(0)$, and a configuration of the elements of the associated eigenvectors which defines the vertices' coordinates in an injective parametric space. Furthermore, these conditions give us the values of $\frac{\partial \mathcal{F}}{\partial x}(0, 0)$ and $\frac{\partial \mathcal{F}}{\partial y}(0, 0)$.

3.3 C^2 -Convergence

If the scheme C^2 -converges and the mark point is a vertex, then

$$\tilde{\lambda}_2(0) = \left| \tilde{\lambda}_1(\pm 2) \right| = \left| \tilde{\lambda}_1(\pm 1) \right|^2, \text{ and } \left| \tilde{\lambda}_1(\omega) \right| < \left| \tilde{\lambda}_1(\pm 1) \right|^2,$$

with $\omega \notin \{-2, -1, 0, 1, 2\}$. Furthermore,

$$\left| \tilde{\lambda}_2(\pm 1) \right| < \left| \tilde{\lambda}_1(\pm 1) \right|^2$$

if and only if

$$\lim_{k \rightarrow \infty} \left(\frac{\left| \frac{\partial \mathcal{F}^{(k)}}{\partial x}(0, 0) \mp i \frac{\partial \mathcal{F}^{(k)}}{\partial y}(0, 0) \right| - \left| \frac{\partial \mathcal{F}}{\partial x}(0, 0) \mp i \frac{\partial \mathcal{F}}{\partial y}(0, 0) \right|}{\left| \tilde{\lambda}_1(\pm 1) \right|^k} \right) = 0.$$

Furthermore, if $L_2(0) = \{2\}$, then

$$\frac{(\tilde{v}_2(0))_X - (\tilde{v}_2(0))_A}{|(\tilde{v}_1(1))_X|^2} \quad \text{and} \quad \frac{(\tilde{v}_2(0))_X - (\tilde{v}_2(0))_A}{|(\tilde{v}_1(-1))_X|^2}$$

depend neither on X nor on k , and if $L_1(2) = L_1(-2) = \{1\}$, then the ratios

$$\frac{|(\tilde{v}_1(2))_X|}{|(\tilde{v}_1(1))_X|^2}, \quad \frac{|(\tilde{v}_1(2))_X|}{|(\tilde{v}_1(-1))_X|^2}, \quad \frac{|(\tilde{v}_1(-2))_X|}{|(\tilde{v}_1(1))_X|^2}, \quad \frac{|(\tilde{v}_1(-2))_X|}{|(\tilde{v}_1(-1))_X|^2}$$

and the differences

$$\varphi_{(\tilde{v}_1(2))_X} - \varphi_{(\tilde{v}_1(1))_X} \quad \text{and} \quad \varphi_{(\tilde{v}_1(-2))_X} - \varphi_{(\tilde{v}_1(-1))_X}$$

do not depend on X .

Furthermore, if we define $\varrho_X^{(k)}$ and α_X as proposed in Sect. 3.2 for a rotationally invariant scheme, then we can scale the eigenvectors such that

$$(\tilde{v}_1(2))_X = (\tilde{v}_1(1))_X^2, \quad (\tilde{v}_1(-2))_X = (\tilde{v}_1(-1))_X^2,$$

and, if

$$\frac{\partial^2 \mathcal{F}}{\partial x^2}(0, 0) - \frac{\partial^2 \mathcal{F}}{\partial y^2}(0, 0) \mp 2i \frac{\partial^2 \mathcal{F}}{\partial x \partial y}(0, 0) \neq 0,$$

then

$$\frac{\partial^2 \mathcal{F}}{\partial x^2}(0, 0) + \frac{\partial^2 \mathcal{F}}{\partial y^2}(0, 0) = 4 \frac{\mathcal{P}(2, 0)}{n},$$

$$\frac{\partial^2 \mathcal{F}}{\partial x^2}(0, 0) - \frac{\partial^2 \mathcal{F}}{\partial y^2}(0, 0) = \frac{8}{n} \Re(\mathcal{P}(1, 2)) = \frac{8}{n} \Re(\mathcal{P}(1, -2)),$$

$$\frac{\partial^2 \mathcal{F}}{\partial x \partial y}(0, 0) = -\frac{4}{n} \Im(\mathcal{P}(1, 2)) = \frac{4}{n} \Im(\mathcal{P}(1, -2)).$$

In conclusion, the necessary condition for C^2 -convergence is that among all the eigenvalues of all $\tilde{M}(\omega)$, the global subsubdominant eigenvalues are the subdominant eigenvalues of $\tilde{M}(0)$ and the dominant eigenvalue of $\tilde{M}(2)$ and $\tilde{M}(-2)$. These global subsubdominant eigenvalues are equal to the square of $|\tilde{\lambda}_1(\pm 1)|$, the global subdominant eigenvalues. Furthermore, the elements of the associated eigenvectors are in a quadratic configuration. These conditions have been already proposed by Sabin [12] as a condition related to the C^2 -continuity of the limit surface. They also let us get the values of the partial derivatives $\frac{\partial^2 \mathcal{F}}{\partial x^2}(0, 0)$, $\frac{\partial^2 \mathcal{F}}{\partial y^2}(0, 0)$ and $\frac{\partial^2 \mathcal{F}}{\partial x \partial y}(0, 0)$. See Barthe et al. [2] (pp. 245–257 of this book) for a further discussion of the rôle of the eigenvalues.

4 Discussion

Many authors interpret a subdivision scheme as a linear map between patches which progressively fill in an n -sided hole around an extraordinary point. Prautzsch [9] and Zorin [16] proposed necessary and sufficient conditions for C^p -regularity of the limit surface, on the eigenvalues and eigenbasis functions of this linear map. In contrast, we interpret a subdivision scheme as a linear map between samplings of two successive surfaces from a sequence of C^p surfaces. If this sequence converges with sufficient regularity (C^p -converges) these samplings may be used to approximate the derivatives of the limit surface. We propose necessary conditions for the C^2 -convergence of a scheme, which is itself a sufficient condition for the C^2 -continuity of the limit surface, on the eigenvalues and eigenvectors of the transformed subdivision matrix. As already stated, a scheme which converges toward a C^2 -continuous limit surface does not necessarily C^2 -converge. But it is interesting to understand the difference between our necessary conditions for C^p -convergence, and the condition for the C^p -regularity of the limit surface proposed by Reif, Prautzsch and Zorin.

C^0 -regularity. We find the same conditions.

C^1 -regularity. Because we ask the subdominant eigenvalues to come from $\tilde{M}(1)$ and $\tilde{M}(-1)$, we ensure the orthoradial injectivity of Reif's characteristic map as described in [8]; and because we ask the components of the associated eigenvectors to be sorted like the parameters $\varrho_X^{(k)}$, we ensure good conditions for the radial injectivity of this map.

C^2 -regularity. Reif's characteristic map [11] is given by the subdominant eigenbasis functions. If the scheme is Box-Spline based, the eigenbasis functions are Box-Splines with our eigenvectors as control points (more precisely, our eigenvectors provide their radial coordinate). One of the conditions proposed by Prautzsch [9] and Zorin [16] for C^2 -regularity, is that the eigenbasis functions z associated to the subsubdominant eigenvalue should belong to $\text{span}\{x^i y^j; i + j = 2\}$ where x and y are the eigenbasis functions associated to the subdominant eigenvalue. Our condition is

the same, but with the eigenvectors instead of the eigenbasis functions. And the eigenvectors provide the altitude over the characteristic map of the control points of z . Around an ordinary vertex, we have checked that the quadratic configuration of the eigenvectors is fulfilled for Loop and Catmull-Clark schemes. Stam does so for the quadratic configuration of eigenbasis functions [15]. The possibility to get quadratic configuration of both eigenvectors and eigenbasis functions around an extraordinary vertex remains to be investigated.

5 Conclusion

We have proposed practical conditions for tuning a scheme in order to control its artifacts in the vicinity of a mark point. To do so, we look for good behaviour of the limit vertices rather than good mathematical properties of the limit surface. The good behaviour of the limit vertices is characterised with the definition of C^2 -convergence of a scheme. We propose necessary explicit conditions for C^2 -convergence of a scheme in the vicinity of any mark point being a vertex of valency greater or equal to three. We cast some light on the relationship between these conditions and the classical necessary and sufficient conditions for the C^2 -continuity of the limit surface. Even though the differences between them are not large, we stress the fact that our conditions are designed for tuning a scheme leading to fewer artifacts on the limit surface [3] rather than for tuning a scheme leading to a C^2 -continuous limit surface.

Acknowledgement

This work was supported in part by the European Union research project “Multiresolution in Geometric Modelling (MINGLE)” under grant HPRN-CT-1999-00117.

References

1. 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.
2. L. Barthe, C. Gérot, M. A. Sabin, and L. Kobbelt. Simple computation of the eigencomponents of a subdivision matrix in the fourier domain. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 245–257 (this book). Springer-Verlag, 2004.
3. L. Barthe and L. Kobbelt. Subdivision scheme tuning around extraordinary vertices. Submitted.
4. E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.

5. D. Doo and M. A. Sabin. Behaviour of recursive division surface near extraordinary points. *Computer Aided Design*, 10(6):356–360, 1978.
6. C. Gérot, L. Barthe, N. A. Dodgson, and M. A. Sabin. Subdivision as sequence of sampled C^p -surfaces and conditions for tuning schemes. Technical Report 583, University of Cambridge Computer Laboratory, Mar 2004. <http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-583.pdf>.
7. C. T. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, 1987.
8. J. Peters and U. Reif. Analysis of algorithms generalizing B-spline subdivision. *SIAM J. Numer. Anal.*, 35(2):728–748, 1998.
9. H. Prautzsch. Smoothness of subdivision surfaces at extraordinary points. *Adv. Comput. Math.*, 9:377–389, 1998.
10. H. Prautzsch and G. Umlauf. Improved triangular subdivision schemes. In *Proc. Computer Graphics International*, pages 626–632, 1998.
11. U. Reif. A unified approach to subdivision algorithm near extraordinary vertices. *Computer Geometric Aided Design*, 12:153–174, 1995.
12. M. A. Sabin. Eigenanalysis and artifacts of subdivision curves and surfaces. In A. Iske, E. Quak, and M. S. Floater, editors, *Tutorials on Multiresolution in Geometric Modelling*, pages 69–97. Springer-Verlag, 2002.
13. M. A. Sabin. Recent progress in subdivision – a survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 203–230 (this book). Springer-Verlag, 2004.
14. M. A. Sabin and L. Barthe. Artifacts in recursive subdivision surfaces. In A. Cohen, J.-L. Merrien, and L. L. Schumaker, editors, *Curve and Surface Fitting: Saint-Malo 2002*, pages 353–362. Nashboro Press, 2003.
15. J. Stam. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. *Proc. ACM SIGGRAPH '98*, pages 395–404, 1998.
16. D. Zorin. *Stationary subdivision and multiresolution surface representations*. PhD thesis, California Institute of Technology, 1997.

Reverse Subdivision

Mohamed F. Hassan and Neil A. Dodgson

Computer Laboratory, University of Cambridge, UK
`{mfh20|nad}@cl.cam.ac.uk`

Summary. We present a reverse Chaikin algorithm which generates a multiresolution representation of any line chain. It has applications in multiresolution editing and compression. We also sketch how this might be extended to the bivariate Loop subdivision algorithm.

1 Introduction

Subdivision methods for curves were introduced and mathematically analysed for the first time by de Rham in 1947. Their re-invention in 1974 by Chaikin made them available to the computer graphics community. Chaikin used them to derive a simple algorithm for the high-speed generation of curves. He first proposed the binary 2-point approximating scheme in [1], which was shown to produce the quadratic B-spline in the limit [6].

Here we present an algorithm, based on Chaikin's algorithm, to decimate a polygonal curve so that, when the uniform subdivision scheme is applied to the decimated curve, a good approximation to the original curve is achieved. The errors between the reconstructed and original polygons are stored so that we can reconstruct the original polygon exactly. By continuing this process, we can construct a hierarchy giving a multiresolution representation of the original curve. Samavati and Bartels have previously explored reverse Chaikin [7]. Also B-spline wavelets [9] solve the same problem but require the construction of the high-frequency synthesis filter matrix, \mathbf{Q} , and the solving of a sparse linear system to perform the reverse subdivision. Our approach is much simpler. The applications for this include lossy and lossless compression, multiresolution editing, and animation.

In Sects. 2–6 we present the Reverse Chaikin method; in Sect. 7 we show how this can be extended to the bivariate case, using Loop [5] as our subdivision scheme.

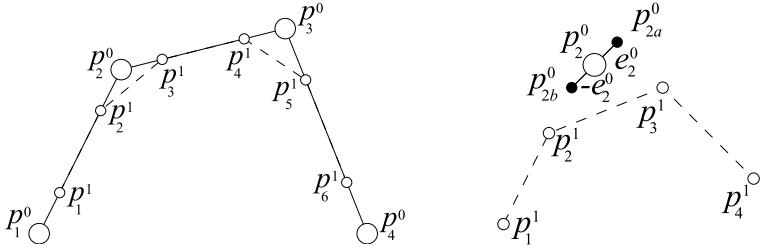


Fig. 1. Left: Chaikin's scheme – the large circles are points of the original polygon and the small circles are the points after one subdivision step. Right: Reverse Chaikin – the small circles are the original points and the large circle is the point after one reverse subdivision step. The filled circles are the two candidate points using the two different pairs of original points.

2 Reverse Chaikin

Chaikin's scheme can be defined formally by

$$\begin{aligned} p_{2i-1}^{n+1} &= \frac{3}{4}p_i^n + \frac{1}{4}p_{i+1}^n, \\ p_{2i}^{n+1} &= \frac{1}{4}p_i^n + \frac{3}{4}p_{i+1}^n, \end{aligned} \quad (1)$$

where p_m^n is the position in \mathbb{R}^3 of point m after n subdivision steps (see Fig. 1(left)).

We would like to reverse this process. Now, solving (1) for p_i^n gives

$$p_i^n = \frac{3}{2}p_{2i-1}^{n+1} - \frac{1}{2}p_{2i}^{n+1}. \quad (2)$$

However, solving (1) for p_{i+1}^n and re-indexing gives

$$p_i^n = \frac{3}{2}p_{2i-2}^{n+1} - \frac{1}{2}p_{2i-3}^{n+1}. \quad (3)$$

Geometrically, this means that there are two sets of pairs of points in the subdivided polygon that can be used to calculate the position of a single point in the original. If Chaikin's scheme was used to produce the refined polygon there is no problem, as the two positions will coincide. However, this cannot be guaranteed in the general case. Our solution is to take the average of the two positions and store the error vectors (see Fig. 1(right)).

Formally this gives

$$p_i^n = -\frac{1}{4}p_{2i-3}^{n+1} + \frac{3}{4}p_{2i-2}^{n+1} + \frac{3}{4}p_{2i-1}^{n+1} - \frac{1}{4}p_{2i}^{n+1}, \quad (4)$$

$$e_i^n = \frac{1}{4}p_{2i-3}^{n+1} - \frac{3}{4}p_{2i-2}^{n+1} + \frac{3}{4}p_{2i-1}^{n+1} - \frac{1}{4}p_{2i}^{n+1}. \quad (5)$$

The forward Chaikin step then becomes:

$$\begin{aligned} p_{2i-1}^{n+1} &= \frac{3}{4}(p_i^n + e_i^n) + \frac{1}{4}(p_{i+1}^n - e_{i+1}^n), \\ p_{2i}^{n+1} &= \frac{1}{4}(p_i^n + e_i^n) + \frac{3}{4}(p_{i+1}^n - e_{i+1}^n). \end{aligned} \quad (6)$$

2.1 Wavelet Formulation

At first sight the formulation above may appear not to be a wavelet transformation because the errors, e_i^n , are added to the points, p_i^n , before the subdivision step rather than after, as would be the case with a wavelet transformation. However, some straightforward algebraic manipulation converts it to a wavelet formulation.

If we write $\mathbf{p}^{n-1} = \mathbf{A}^n \mathbf{p}^n$ and $\mathbf{e}^{n-1} = \mathbf{B}^n \mathbf{p}^n$, where $\mathbf{p}^n = [p_0^n \ p_1^n \ p_2^n \dots]^T$ (and similarly for \mathbf{e}^n), then, from (4) and (5), the wavelet analysis filters, \mathbf{A}^n and \mathbf{B}^n are:

$$\mathbf{A}^n = \frac{1}{4} \begin{bmatrix} \ddots & & & & \\ & -1 & +3 & +3 & -1 \\ & & -1 & +3 & +3 & -1 \\ & & & -1 & +3 & +3 & -1 \\ & & & & & \ddots & \end{bmatrix}$$

$$\mathbf{B}^n = \frac{1}{4} \begin{bmatrix} \ddots & & & & \\ & +1 & -3 & +3 & -1 \\ & & +1 & -3 & +3 & -1 \\ & & & +1 & -3 & +3 & -1 \\ & & & & & \ddots & \end{bmatrix}$$

Similarly, if we write the synthesis step as $\mathbf{p}^n = \mathbf{P}^n \mathbf{p}^{n-1} + \mathbf{Q}^n \mathbf{e}^{n-1}$ then the wavelet synthesis filters, \mathbf{P}^n and \mathbf{Q}^n are derived from (6) as:

$$\mathbf{P}^n = \frac{1}{4} \begin{bmatrix} \ddots & \vdots & & & \\ & 3 & 1 & & \\ & 1 & 3 & & \\ & & 3 & 1 & \\ & & 1 & 3 & \\ & & 3 & 1 & \\ & & 1 & 3 & \\ & & & \ddots & \end{bmatrix} \quad \mathbf{Q}^n = \frac{1}{4} \begin{bmatrix} \ddots & \vdots & & & \\ & +3 & -1 & & \\ & +1 & -3 & & \\ & & +3 & -1 & \\ & & +1 & -3 & \\ & & +3 & -1 & \\ & & +1 & -3 & \\ & & & \ddots & \end{bmatrix}$$

The advantage of this reverse Chaikin method over many wavelet methods is computation speed: both the analysis and the synthesis filters are *sparse*, whereas in many wavelet methods (e.g. the B-spline wavelets [9]) the analysis filters, \mathbf{A}^n and \mathbf{B}^n , are dense, thus requiring quadratic rather than linear time to perform the analysis (reverse subdivision) step naïvely, or else requiring the solution of a linear system [9].

In addition, there are a number of subtleties in the practical implementation of reverse subdivision which make the wavelet formulation difficult, but which can be handled in the algorithmic formulation by simple modifications to the algorithm, as described in the following section.

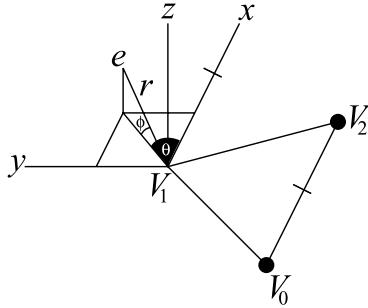


Fig. 2. Local spherical coordinate frame. The local cartesian frame for the error at V_1 is defined by the reverse subdivided points V_0 , V_1 , and V_2 : The x basis vector is parallel to $V_2 - V_0$. The z basis vector is perpendicular to the plane defined by $\{V_0, V_1, V_2\}$, and the y basis vector is given by $z \times x$. r and θ are the standard spherical coordinates, ϕ is angle from the xy -plane.

3 Subtleties

3.1 Local Coordinate Frame for Errors

If the error vectors are stored naïvely, one of the primary uses of this algorithm is lost. The reason for this is that for editing purposes we would like to edit the reverse subdivided curve and have the changes manifest in the detailed curve, without changing the error vectors. We can see immediately that if we apply a similarity transformation to the reverse subdivided curve, we will have to apply the same transformation to the error vectors in order to get the transformed version of the original curve, which is undesirable. Our solution to this problem is to use a local coordinate frame for the error vectors (see Fig. 2).

The local cartesian frame for the error at a reverse subdivided point is defined by that point and its immediate neighbours. The x unit basis vector is chosen to be parallel to the vector between the two neighbouring points. The z unit basis vector is perpendicular to the plane defined by these three points, and the y basis vector is given by $z \times x$. r and θ are the standard spherical coordinates, ϕ is the angle from the xy -plane. The x vector and the definition for ϕ were chosen such that θ and ϕ would be close to 0. Finally r can be stored as a fraction of the magnitude of the vector between the neighbouring points. A similar co-ordinate system is used by Stollnitz et al. [9].

Now, if we apply a similarity transformation to the reverse subdivided points and subdivide them using the same stored errors, we will get the same transformation of the original polygon. An example is shown in Fig. 3.

3.2 Odd Vertex Number

There is another subtlety. Suppose we have n vertices. After one Chaikin subdivision step we will have either $2n$ vertices, for closed loops, or $2n - 2$ for

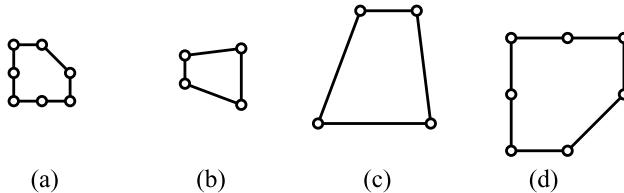


Fig. 3. An example of a similarity transformation. (a) The original outline. (b) The reverse subdivision of (a). (c) A similarity transformation of (b). The subdivision of (c) using the same errors as the original reverse subdivision. The circles indicate the position of the vertices. To perform reverse subdivision the centre vertex on the left hand side of (a) must be duplicated to give an even number.

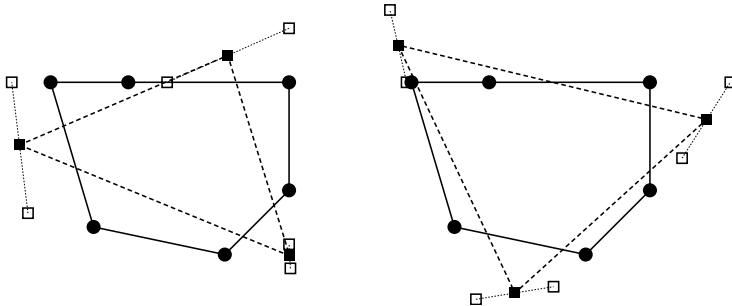


Fig. 4. One step of reverse Chaikin can produce two possible solutions. In this example, a six-vertex polygon (*solid line*) is reduced to two alternative triangles (*heavy dashed line*). The solid circles are the original vertices; the positions of the computed vertices are open squares, these are averaged to give the final position, which is shown by a solid square.

open line chains. This means that Reverse Chaikin has a problem if we have an odd number of vertices. This problem is simply overcome by duplicating the last vertex and tagging that we have done so in the error file. See Fig. 3, where the central vertex on the left hand side is duplicated.

3.3 Alternative Configurations

A problem, which is best illustrated on closed loops, is that there are multiple possible simplifications. Fig. 4 shows an example of a closed loop of six vertices and the two possible three-vertex polygons which can be produced by the reverse Chaikin algorithm. A choice between the two alternatives must be made at each level of reverse subdivision, leading to 2^k possible decimated line chains after k steps of reverse subdivision. This is not ideal, but it cannot be avoided. This is handled by generating both possible decimated line chains at each step of reverse subdivision, and then picking the one with the smallest error vectors. This is especially important for line chains which were originally

generated by Chaikin subdivision, as the correct version will have zero error vectors, while the alternative version will, in general, have non-zero error vectors.

This problem can also manifest in an open chain with an odd number of vertices, as either end can be considered to be the start of the chain, with the other end having the duplicated vertex mentioned in Sect. 3.2.

This problem is specific to the univariate case. In the bivariate case, the extraordinary vertices fix the topology of the reverse subdivided mesh, although they bring in their own problems, described in Sect. 8.

4 Examples

An example of a similarity transformation is given in Fig. 3. The original line chain is reverse subdivided, transformed, and then subdivided using the same errors as the original reverse subdivision. We can see that, after this procedure, the final line chain is the similarity transformation of the original.

An example of local editing can be seen in Fig. 5. The left side, top to bottom, is the reverse subdivided outline of England (585 vertices to 37 vertices). The right side, bottom to top, is the modified outline, subdivided using the same errors as the left side. We can see that we achieve a local editing of the original curve without loss of detail.

5 Examining Stability

It is important to examine the behaviour of the algorithm when applied multiple times; as the algorithm relies on local extrapolation, errors could tend to be magnified. Hence the distance to the original curve may grow rapidly causing it to become unstable.

To examine the stability of our algorithm qualitatively, we perform the following experiment: we construct a subdivision curve from a coarse line chain by applying Chaikin's scheme. After a fixed number of subdivisions we add white noise to each vertex. Then we smooth the curve by removing the error coefficients at the finest levels.

The results of the noise-removal experiment can be seen in Fig. 6. We see that the algorithm slowly reduces the amplitude of the noise over a number of smoothing steps and hence is stable for practical applications. This also means that the algorithm can be used for lossy compression as described in the next section.

6 Storage and Compression

The reverse Chaikin algorithm presented in Sect. 2 is suitable for multiresolution editing, provided that the error vectors are stored in the format described



Fig. 5. Example: Local editing. The left side, top to bottom, is the reverse subdivided outline of England (585 vertices to 37 vertices). The right side, bottom to top, is the modified outline, reconstructed using the same errors as the left side. The circle indicates the modified vertex.

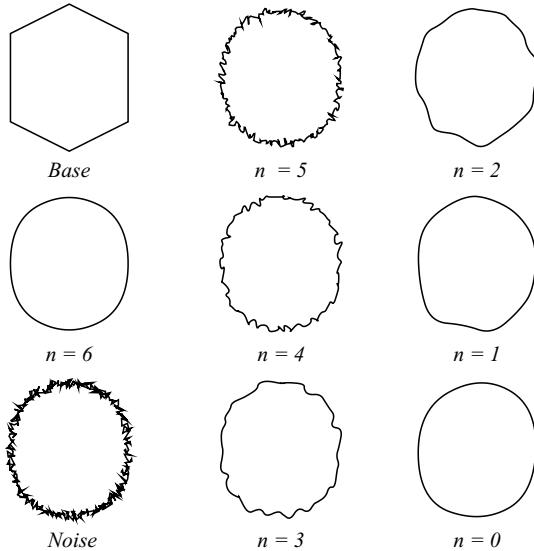


Fig. 6. Noise-removal experiment: Top-left shows the base curve. Middle-left shows the base curve subdivided 6 times (384 vertices). Bottom-left shows the curve with added noise – the average magnitude of the noise is 3 times the average distance between vertices. Middle and right show the results after the error coefficients at n and above steps have been set to zero.

in Sect. 3. The assumption is that vertex coordinates and error vectors are stored to the floating-point precision of the computer. The reverse Chaikin representation thus uses the same amount of storage space as the original set of vertices. Indeed, it can be seen as a simple wavelet transform (Sect. 2.1): transforming one set of floating-point coordinates into another, both sets containing the same number of values.

For lossless compression, it is necessary to store the reverse Chaikin error vectors in less space than is required to store vertex coordinates. The local coordinate system introduced in Sect. 3 allows us to compress error vectors more efficiently than vertex coordinates because we can expect the r , θ , and ϕ coordinate values to be close to zero with high probability. Such a distribution of coordinate values allows for efficient coding, as shorter codes can be used for the more likely values. In particular, we note that ϕ should be very close to zero in most cases, because we can expect vertices at level $(n+1)$ to lie very close to the plane through the three nearest vertices at level n . By similar reasoning θ should be close to zero.

Lossy compression can be achieved in two complementary ways. Firstly, the quantisation of vertex coordinates and error vectors can be made increasingly coarse. There is obviously some loss of resolution in the position of vertices, but the more important disadvantage is that errors in position will accumulate as coarsely-quantised error vectors are added to intermediate-level vertices

which have themselves been calculated using error vectors: the absolute error in position increasing as the subdivision level increases.

The second way of achieving lossy compression is simply to throw away error vectors. For example, throwing away the error vectors at just the highest subdivision level will remove (almost) half of the data. This means that the highest level is now approximated by the uncorrected Chaikin subdivision of the next highest level. If the original line chain is of sufficiently high resolution, then a reasonable approximation to the original may be achieved by the uncorrected Chaikin subdivision generated from several levels of reverse Chaikin without any error vectors being stored. This significantly reduces the storage required for the line chain. A related compression technique for line chains, with similar motivation, is the Douglas–Pücker algorithm [2], which replaces a line chain by a coarser line chain which approximates the original to some tolerance but which does not allow for multiresolution editing.

7 Theory of the Bivariate Case

This section provides the theory to extend these ideas to the bivariate case. Samavati et al. [8] have produced an algorithm to generate multiresolution surfaces by a reverse Doo subdivision method using a local least-squares method. Khodakovskiy et al. [4] use a wavelet transform based on Loop subdivision [5] for geometry compression. As with all wavelet methods a high-pass reconstruction filter \mathbf{Q} must be constructed, and also there is no theory for wavelet constructions around irregular vertices so this is done ad hoc. Finally computing the forward wavelet transform requires solving linear systems. Our method is simpler and is also based on Loop subdivision.

First, we have to define the constraints of the starting mesh. Here we present an algorithm which works on triangular meshes with binary subdivision connectivity. This is not such a heavy constraint as there are many algorithms that will transform an arbitrary mesh to one with subdivision connectivity (e.g. [10]). Our algorithm naturally splits into two parts.

The first part is to detect whether the given mesh has subdivision connectivity and construct the connectivity of the coarse level mesh. Two algorithms achieve this. Taubin's algorithm [11] is based on the concept of the *covering surface* in algebraic topology. The algorithm proposed by Hormann in [3] is simpler and uses the observation that most of the vertices of a mesh with subdivision connectivity are regular and all irregular vertices are guaranteed to be vertices of the coarse level mesh. Both these algorithms give a set of candidate vertices for the coarse level to which we can apply the next stage of the algorithm.

The next stage uses the inverse of the subdivision matrix of the given uniform scheme in order to construct the geometry of the coarse level vertices. There are a number of practical considerations in the implementation. One interesting issue is that the extraordinary vertices must be preserved between

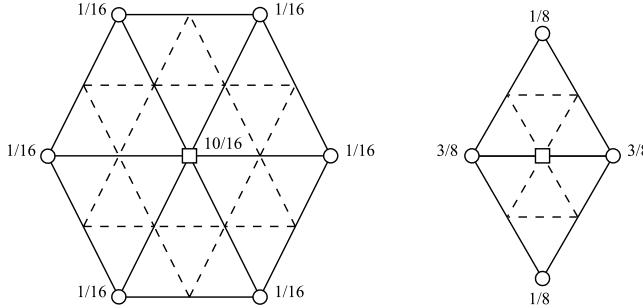


Fig. 7. Stencils for the Loop subdivision scheme for regular vertices. The square indicates the vertex being calculated.

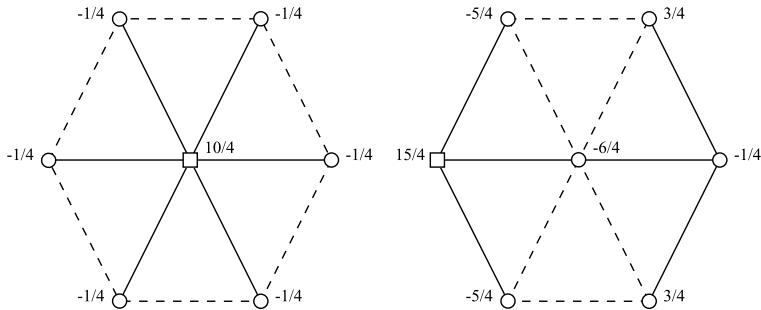


Fig. 8. Coefficients for the reverse subdivision scheme for regular vertices. Left: the stencil for one set of points: calculating the position of the central vertex. Right: the stencil for an alternative set of points: calculating the position of one of the non-central vertices. In both cases, the square indicates the vertex being calculated.

subdivision levels. This limits the form of a reverse subdivided mesh and may mean that, in some cases, there is no possible valid mesh which can be created at a coarser level. There are various potential solutions to this, including splitting the mesh into sub-meshes, which are then stitched together in some way.

7.1 Geometry of Regular Vertices

Fig. 7 shows the stencil for the standard Loop subdivision scheme in the regular case. We use this to form the subdivision matrix around a vertex. The inverse of this matrix gives us the coefficients for the reverse subdivision scheme (shown in Fig. 8(left)).

However the inverse subdivision matrix also shows that we can use other points to define the same vertex (see Fig. 8(right)). This is analogous to the univariate case. In this case there are seven candidate vertices (see Fig. 9), rather than the two in the univariate case. So now we take the Loop subdivision of the seven cases to be our actual vertex and store the errors. Again,

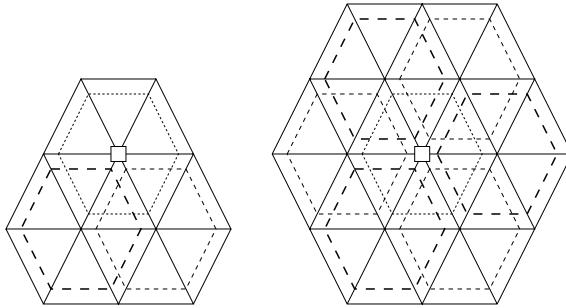


Fig. 9. Each dashed hexagon represents a set of vertices that can be used to calculate the central reverse subdivided vertex (indicated by the square). At left are three (out of the seven) sets of vertices and at right all seven are shown.

and for the same reasons, we use a local coordinate frame similar to that used in the univariate scheme.

7.2 Geometry of Extraordinary Vertices

Fig. 10(a) shows the Loop subdivision scheme for n -valent vertices. This time we cannot simply take the inverse of the subdivision matrix to calculate the coefficients of the reverse subdivision scheme as the values and size of the matrix are dependent on n , the valency. Instead we formulate the linear system as follows:

$$p'_0 = (1 - a_n)p_0 + \sum_{i=1}^n \frac{a_n}{n} p_i \quad (7)$$

$$p'_i = \frac{3}{8} (p_0 + p_i) + \frac{1}{8} (p_{i-1(n)} + p_{i+1(n)}) \quad (8)$$

where p_0 is the extraordinary vertex and $\{p_i | i = 1, \dots, n\}$ are its immediate neighbours.

Solving this system for p_0 gives

$$p_0 = (1 - b_n)p'_0 + \sum_{i=1}^n \frac{b_n}{n} p'_i \quad (9)$$

where

$$b_n = 1 + \frac{5}{8a_n - 5} \quad (10)$$

This is independent of the actual a_n used. Hence the reverse subdivided scheme is given by Fig. 10(b).

As with the regular case, these points can also be used to calculate the neighbouring points (see Fig. 10(c)). The coefficients are given by $\{c_n^i | i = 0, \dots, n\}$, where n is the valency. Unfortunately we have not found a closed

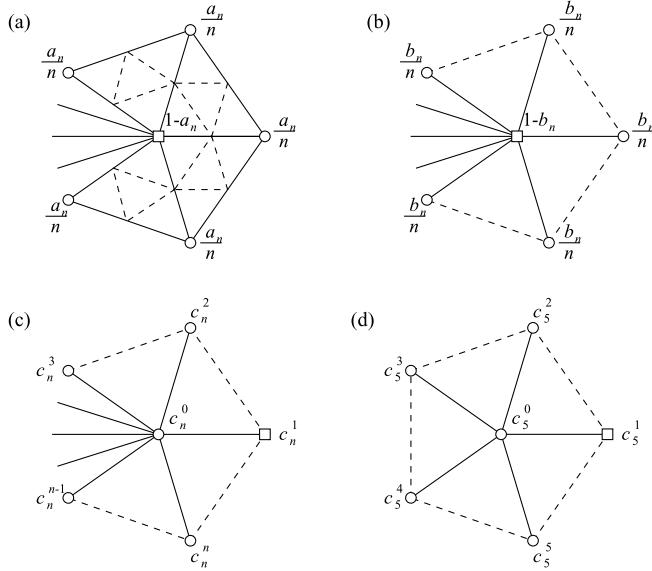


Fig. 10. Extraordinary vertices. (a) shows the standard Loop subdivision at extraordinary vertices, where usually $a_n = \frac{1}{64} \left(40 - (3 + 2 \cos \frac{2\pi}{n})^2 \right)$. (b) shows the reverse Loop subdivision at extraordinary vertices, where $b_n = 1 + \frac{5}{8a_n - 5}$ (independent of the expression for a_n). (c) shows the alternative stencil for the general n -valent vertex. (d) shows the alternative stencil for a 5-valent vertex. The square indicates the vertex being calculated.

form for the $\{c_n^i\}$ so the inverse subdivision matrix has to be calculated for each valency.

However, we find that $c_n^0 = \frac{3}{8a_n - 5}$ for all n and we can see by symmetry that $c_n^{n-i} = c_n^{i+2}$, $i = 0, \dots, n-2$. The same mechanism is used as in the regular case to handle the candidate vertices.

8 Future Work

We have yet to implement and test the bivariate algorithm given in the previous section, and this is the obvious next step. We will use Hormann's algorithm [3] as the basis for the topology step. We expect the bivariate case to find uses in the same applications as the univariate case: multiresolution editing and compression.

Acknowledgement

This work was supported in part by the European Union research project “Multiresolution in Geometric Modelling (MINGLE)” under grant HPRN-CT-1999-00117.

References

1. G.M. Chaikin. An algorithm for high-speed curve generation. *Computer Graphics and Image Processing*, 3:346–349, 1974.
2. D. Douglas and T. Pücker. Algorithms for the reduction of the number of points required to represent a digitised line or its caricature. *The Canadian Cartographer*, 10:112–122, 1973.
3. K. Hormann. An easy way of detecting subdivision connectivity in a triangle mesh. Technical Report 3, Department of Computer Science 9, University of Erlangen, May 2002.
4. Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. Progressive geometry compression. In *Proc. ACM SIGGRAPH 2000*, pages 271–278, 2000.
5. C. T. Loop. Smooth subdivision surfaces based on triangles. Master’s thesis, University of Utah, Department of Mathematics, 1987.
6. R. F. Riesenfeld. On Chaikin’s algorithm. *Computer Graphics and Image Processing*, 4:304–310, 1975.
7. F. Samavati and R. Bartels. Multiresolution curve and surface representation by reversing subdivision rules. *Computer Graphics Forum*, 18(2):97–119, 6 1999.
8. F. Samavati, N. Mahdavi-Amiri, and R. Bartels. Multiresolution surfaces having arbitrary topologies by a reverse Doo subdivision method. *Computer Graphics Forum*, 21(2):97–119, 6 2002.
9. E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. *Wavelets for computer graphics*. San Francisco; Morgan Kaufmann Publishers, 1996.
10. Vitaly Surazhsky and Craig Gotsman. Explicit surface remeshing. In *Proc. Eurographics Symposium on Geometry Processing*, pages 17–28, June 2003.
11. G. Taubin. Detecting and reconstructing subdivision connectivity. *Visual Computer*, 2002.

$\sqrt{5}$ -subdivision

Ioannis P. Ivrissimtzis¹, Neil A. Dodgson², and Malcolm Sabin³

¹ Max-Planck-Institut für Informatik, Saarbrücken, Germany
`ivrissim@mpi-sb.mpg.de`

² Computer Laboratory, University of Cambridge, UK
`nad@ccl.cam.ac.uk`

³ Numerical Geometry Ltd., Cambridge, UK
`malcolm@geometry.demon.co.uk`

Summary. Most established subdivision schemes have the refined grid at each stage aligned with the previous one. The $\sqrt{3}$ and $\sqrt{2}$ schemes alternate orientations. This paper is one of the first detailed studies of a skew scheme in which the axis directions after refinement do not either lie along or bisect those before. It raises the issue of how the analysis techniques can be applied in this new context and provides an example of how they may be thus applied.

1 Introduction

The possibility of a $\sqrt{5}$ scheme for subdivision surfaces was first discussed in [12]. There, two $\sqrt{5}$ mesh refinement rules for regular quadrilateral meshes were proposed, classified as $QP(2, 1)$ and $QP(1, 2)$, respectively. These two regular refinement rules, shown in Fig. 1(left), like the well-known $\sqrt{3}$ refinement rule for triangle meshes [17], induce a rotation of the initial mesh. In particular, the $QP(2, 1)$ refinement induces an anti-clockwise rotation by $\arctan(\frac{1}{2})$, or equivalently a clockwise rotation by $\arctan(2)$, while the $QP(1, 2)$ induces an anti-clockwise rotation by $\arctan(2)$, or equivalently a clockwise rotation by $\arctan(\frac{1}{2})$.

1.1 Related Work

Skew subdivision schemes, inducing a rotation of the grid at each iteration, were first introduced as a general class of subdivision schemes in [1]. $\sqrt{5}$ refinement was proposed in [23] as a hierarchical sampling method over a regular grid. In [22], a single step of the $\sqrt{5}$ refinement rule is described as a popular sampling method for numerical integration of two-dimensional periodic functions. In both applications the nice properties of the $\sqrt{5}$ sampling are due to fact that the five points corresponding to a square of the old grid (one old point and four new) all have different x and y coordinates, see Fig. 1(right).

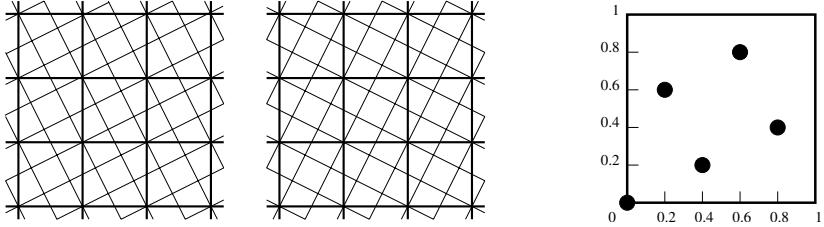


Fig. 1. Left: $QP(2, 1)$ and $QP(1, 2)$ refinement. Right: The five points corresponding to the same old quad have distinct x and y coordinates.

1.2 Overview

We propose a new subdivision scheme for quadrilateral meshes, based on the $\sqrt{5}$ refinement of a regular grid. In Sect. 2 we extend the refinement rule into irregular meshes, introducing the idea of using half-edges as the main primitive in the description of a subdivision scheme. In Sect. 3 we study the smoothness properties the $\sqrt{5}$ subdivision, tune its coefficients, and briefly discuss its support. We conclude by showing several examples of $\sqrt{5}$ subdivision surfaces.

2 $\sqrt{5}$ -refinement for Irregular Meshes.

To define a $\sqrt{5}$ subdivision scheme, first, we need an extension of the regular refinement rules shown in Fig. 1 to cover the irregular case. Despite the fact that the regular case is already relatively complicated, it turns out that there is a very simple such extension, based in the correspondence between the vertices of the new mesh and the vertices and half-edges of the old.

Fig. 2 (left) shows the correspondence between the newly introduced vertices and the half-edges of the old mesh. As, at every step of the process, we retain the old vertices, we have a correspondence between the vertices of the new mesh and the vertices and half-edges of the old. A new vertex will be called *vertex-vertex* or *halfedge-vertex* according to this correspondence.

Under the same correspondence, the faces of the new mesh can be described as 4-tuples of vertices and half-edges of the old mesh. As Fig. 2 (right) shows, there are two kind of faces on the new mesh. Those corresponding to the faces of the old mesh, and those corresponding to the half-edges of the old mesh. Fig. 3 describes the new faces in terms of the old vertices and half-edges.

Notice that descriptions of subdivision processes using half-edges as the main primitive are not common in the literature. Probably, one reason is that half-edges, lacking a direct physical interpretation, are considered an unnatural choice, and a second reason is that subdivision schemes requiring the half-edge description, like the $\sqrt{5}$ scheme proposed here, have not been studied extensively. Nevertheless, the implementation of a scheme usually involves

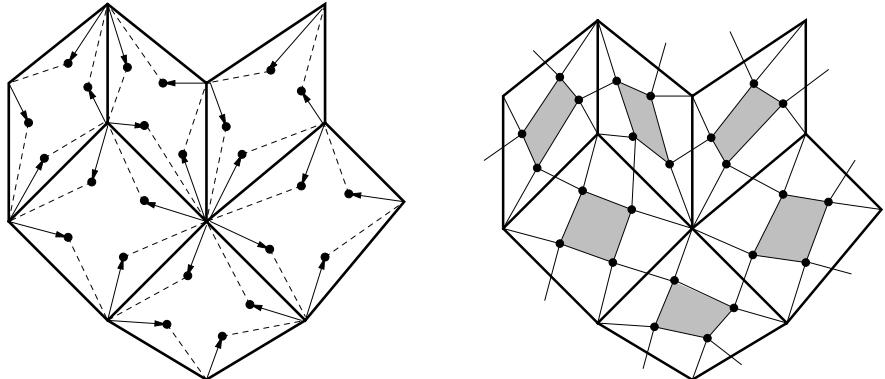


Fig. 2. Left: Every new vertex corresponds to an old half-edge. A solid and a dashed line connect the new vertex with the beginning and the end, respectively, of the corresponding old half-edge. Right: The shaded new faces correspond to old faces, the white new faces correspond to old half-edges.

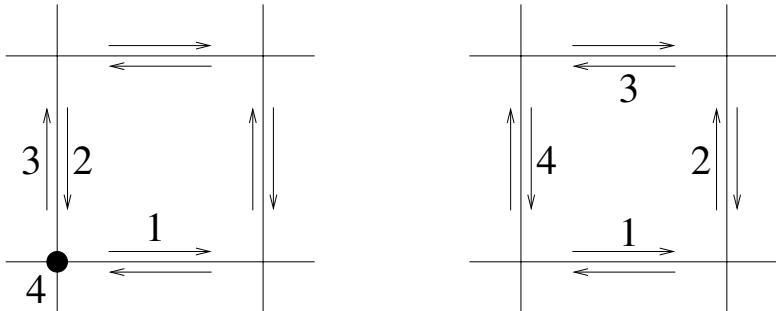


Fig. 3. Left: A new face corresponding to an old half-edge described as a 4-tuple $(1,2,3,4)$ of old vertices and half-edges. Right: A new face corresponding to an old face described as a 4-tuple of old half-edges.

an implicit half-edge description, given that the most common computer representation for meshes is the Half-Edge structure. A generalisation of this idea is presented in the Appendix at the end of the paper.

3 Stencils for the $\sqrt{5}$ -scheme.

The next step towards a definition of a subdivision scheme is to determine the point-sets of the *stencils*, that is, the set of old vertices that will be used for the calculation of the position of the new. A larger point-set, after correct tuning of the coefficients, gives smoother subdivision surfaces, while a smaller point set gives smaller support with the influence of each initial vertex better

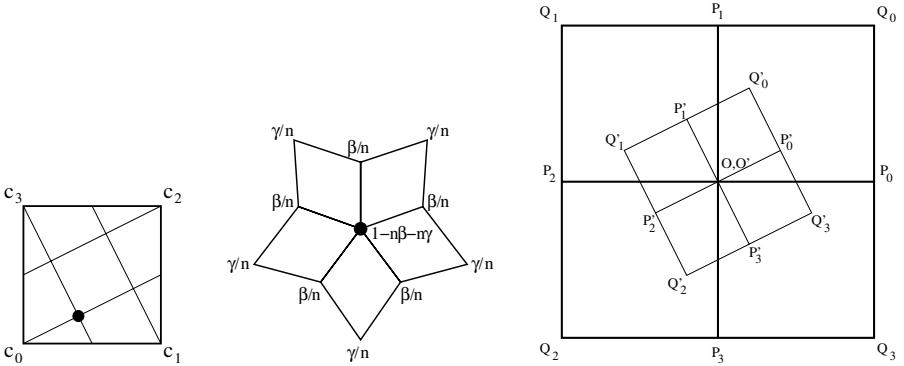


Fig. 4. Left: The stencil of the half-edge vertices. Middle: The stencil for the vertex vertices. Right: The transformation of the 1-ring of the vertex O .

localised. Here, we opt for small stencils and we use the vertices of the face of a half-edge for the stencil of the corresponding halfedge-vertex. For a vertex-vertex we use the corresponding old vertex and the members of its 1-ring neighbourhood, see Fig. 4.

Next we tune the coefficients in the stencils so that the resulting subdivision surfaces are as smooth as possible.

3.1 Background

Traditionally, there are two major tools for analysing the smoothness properties of a subdivision surface. The generating functions [9], and the spectral analysis of the subdivision matrix [8, 3, 20]. Here we use a version of spectral analysis with a more geometric flavor in the form of eigenpolygons [14, 16]. Although our analysis is elementary it is worth going into some detail, especially because the study of schemes with complex subdominant eigenvalues is scattered in the literature of subdivision.

Recall from [7] that the eigenvalues of the n -dimensional circulant matrix

$$C = \text{circ}(c_0, c_1, \dots, c_{n-1}) = \begin{pmatrix} c_0 & c_1 & c_2 & \dots & c_{n-2} & c_{n-1} \\ c_{n-1} & c_0 & c_1 & \dots & c_{n-3} & c_{n-2} \\ & & & \dots & & \\ c_2 & c_3 & c_4 & \dots & c_0 & c_1 \\ c_1 & c_2 & c_3 & \dots & c_{n-1} & c_0 \end{pmatrix} \quad (1)$$

are the values of the generating polynomial

$$\lambda_t = p(\omega^t), \quad p(z) = c_0 + c_1 z + c_2 z^2 + \dots + c_{n-1} z^{n-1}, \quad \omega = e^{\frac{2\pi i}{n}} \quad (2)$$

for $t = 0, 1, \dots, n - 1$. The corresponding eigenvectors are given by the rows of the matrix

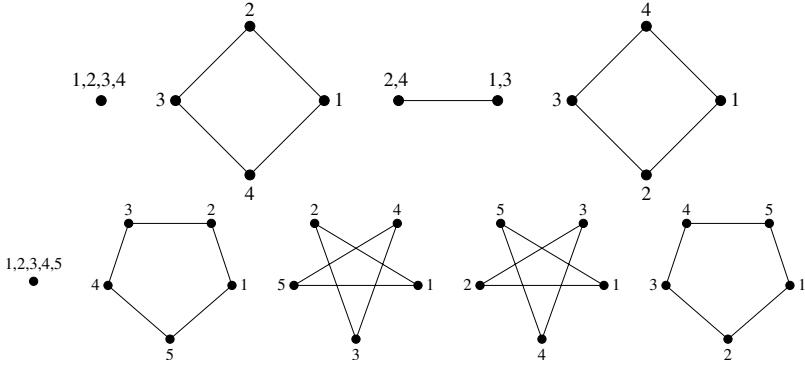


Fig. 5. The four eigenquadrilaterals and the five eigenpentagons.

$$F = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \dots & \omega^{(n-1)(n-1)} \end{pmatrix} = (\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{n-1})^T. \quad (3)$$

These eigenvectors can also be seen as vertices of planar regular polygons, allowing multiple vertices and self-intersections. Fig. 5 shows these eigenpolygons for the cases $n = 4, 5$, while a detailed study of them can be found in [2].

Every planar polygon, thought of here as an n -tuple of coplanar points, or equivalently, an n -dimensional complex vector, can be uniquely written as a linear combination of the n eigenpolygons. A non-planar polygon can be written as a linear combination of the eigenpolygons, with the additional property that any two eigenpolygons corresponding to conjugate eigenvalues, i.e.

$$\mathbf{w}_k, \mathbf{w}_{n-k}, \quad k = 1, 2, \dots, \left\lfloor \frac{n}{2} \right\rfloor \quad (4)$$

lie on the same plane. The planes where the pairs of eigenpolygons lie can be computed by solving a linear system, see [5] for the details.

3.2 Analysis Around a Centreface

A very distinct property of the $\sqrt{5}$ scheme is that it is both primal and dual [12]. That is, one iteration of the subdivision process maps faces to faces and vertices to vertices. This also means that we have to study the behaviour of the scheme both around centrefaces and around vertices.

In the study of the behaviour of the scheme around a centreface we assume that all the faces are quadrilateral and thus, at each step, they are

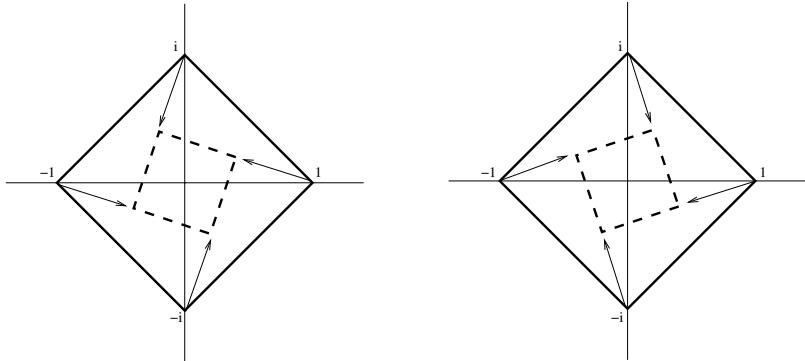


Fig. 6. The two convex eigenquads after one iteration of the $QP(2,1)$ subdivision. The result can be described as multiplication by $\frac{2}{5} + i\frac{1}{5}$ and $\frac{2}{5} - i\frac{1}{5}$, respectively.

transformed by a circulant matrix of dimension $n = 4$. By evaluating the generating polynomial at $1, i, -1, -i$ we get the system

$$\begin{aligned} c_0 + c_1 + c_2 + c_3 &= 1 \\ c_0 + ic_1 - c_2 - ic_3 &= \lambda_1 \\ c_0 - c_1 + c_2 - c_3 &= \lambda_2 \\ c_0 - ic_1 - c_2 + ic_3 &= \lambda_3 \end{aligned} \quad (5)$$

To find the exact values of λ_1, λ_3 we notice that the transformation of the eigenpolygon \mathbf{w}_1 by one step of the subdivision scheme corresponds to a multiplication by

$$z = re^{i\theta} = \frac{2}{5} + i\frac{1}{5} \quad (6)$$

where r is the scaling and θ the rotation induced by the scheme. Similarly, the transformation on \mathbf{w}_3 , which is a copy of \mathbf{w}_1 with opposite orientation, corresponds to a multiplication by

$$\bar{z} = re^{-i\theta} = \frac{2}{5} - i\frac{1}{5}. \quad (7)$$

But as $\mathbf{w}_1, \mathbf{w}_3$ are eigenpolygons their transformation is also equal to a multiplication by the corresponding eigenvalue. Thus, we have $\lambda_1 = \frac{2}{5} + i\frac{1}{5}$ and $\lambda_3 = \frac{2}{5} - i\frac{1}{5}$ (see Fig. 6).

In subdivision, the standard requirement for the fourth, fifth, and sixth eigenvalues is to depend quadratically on the two subdominant eigenvalues [21]. Indeed, as the two subdominant eigenvalues represent the transformation within the tangent plane, for nice curvature behaviour we would expect the dominant eigencomponents outside the tangent plane to shrink with a speed depending quadratically on the shrinkage of the tangent plane.

Thus, in our case, a natural choice for the fourth eigenvalue, which by the third equation of the system (5) is real, would be $\lambda_2 = |\lambda_1|^2 = |\lambda_3|^2 = \frac{1}{5}$. That leads to the coefficients

$$c_0 = \frac{5}{10} \quad c_1 = \frac{3}{10} \quad c_2 = \frac{1}{10} \quad c_3 = \frac{1}{10} \quad (8)$$

Notice that, with the above coefficients, points at different distances from the new point have the same influence on it.

Although in the rest of the paper we use the above coefficients, we have also examined other possibilities. The coefficients $\frac{12}{25}, \frac{8}{25}, \frac{2}{25}, \frac{3}{25}$, were obtained as Wachspress coordinates of the point $(\frac{2}{5}, \frac{1}{5})$ with respect to the unit square, see for example [11]. Another alternative is to use the coefficients $\frac{9}{20}, \frac{7}{20}, \frac{1}{20}, \frac{3}{20}$, which give $\lambda_2 = 0$ meaning that any new face corresponding to an old face is planar, because the only eigencomponent which possibly lies outside the tangent plane becomes 0. Finally, the coefficients $\frac{2}{5}, \frac{2}{5}, 0, \frac{1}{5}$, give $\lambda_2 = -\frac{1}{5}$.

Experimentally, we found that for $\lambda_2 = \frac{3}{25}$ we get visual results similar to those for $\lambda_2 = \frac{1}{5}$. For $\lambda_2 = 0$ the visual quality deteriorates slightly, while for $\lambda_2 = -\frac{1}{5}$ it is significantly worse, see Fig. 7. The latter shows that the behaviour of the scheme depends on the actual eigenvalues and not on their absolute values.

3.3 Analysis Around a Vertex

To study the smoothness properties of a surface around a vertex of valence n we consider the subdivision matrix

$$M = \begin{pmatrix} 1 - \beta - \gamma & \frac{\beta}{n} & \frac{\gamma}{n} & \frac{\beta}{n} & \frac{\gamma}{n} & \frac{\beta}{n} & \frac{\gamma}{n} & \dots & \frac{\beta}{n} & \frac{\gamma}{n} \\ c_0 & c_1 & c_2 & c_3 & 0 & 0 & 0 & \dots & 0 & 0 \\ c_1 & c_2 & c_3 & c_0 & 0 & 0 & 0 & \dots & 0 & 0 \\ c_0 & 0 & 0 & c_1 & c_2 & c_3 & 0 & \dots & 0 & 0 \\ c_1 & 0 & 0 & c_2 & c_3 & c_0 & 0 & \dots & 0 & 0 \\ & & & & & & & & & \\ c_0 & & c_3 & 0 & 0 & 0 & 0 & \dots & c_1 & c_2 \\ c_1 & & c_0 & 0 & 0 & 0 & 0 & \dots & c_2 & c_3 \end{pmatrix} \quad (9)$$

acting on the vector

$$V = (O, P_0, Q_0, P_1, Q_1, \dots, P_{n-1}, Q_{n-1})^T \quad (10)$$

with O, P_i, Q_i as shown in Fig. 4 (right). As the c_i 's are given by (8) we have to optimise for β and γ .

To streamline the computations we calculate the two eigenvalues corresponding to each frequency separately, see [4]. Apart from the obvious eigenvalue 1, we will call *elliptic* the two eigenvalues corresponding to frequency 0

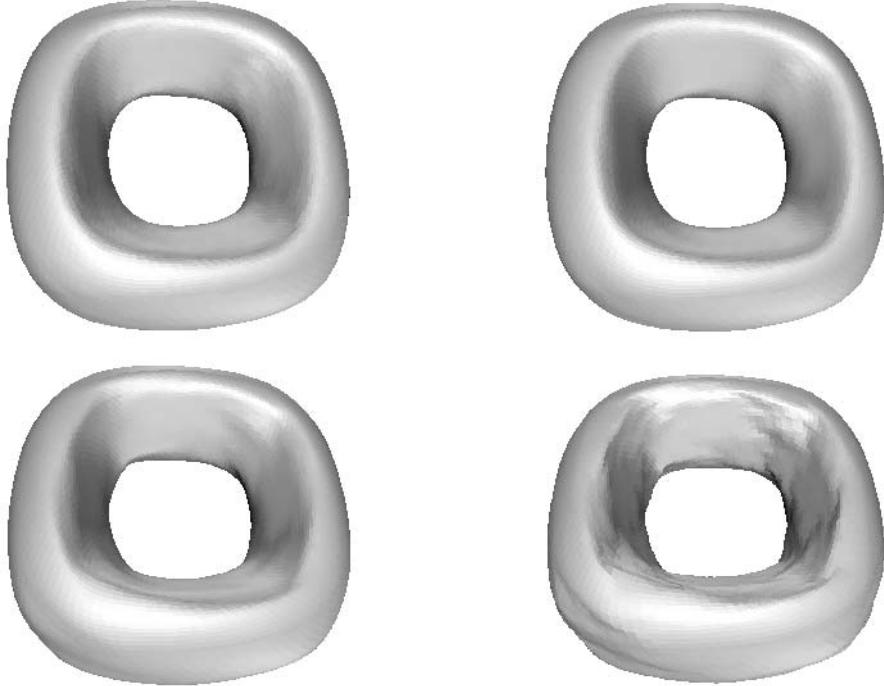


Fig. 7. Top left: $\lambda_2 = \frac{1}{5}$, top right: $\lambda_2 = \frac{3}{25}$, bottom left: $\lambda_2 = 0$, bottom right: $\lambda_2 = -\frac{1}{5}$.

because they are responsible for the elliptic properties of the scheme, and the remaining $2n - 2$ eigenvalues *non-elliptic*.

We notice that β, γ seen here as variables, affect only the two elliptic eigenvalues, see for example [16]. Using this observation we prove in two stages that our scheme gives C^1 surfaces for generic input meshes. First we show that the largest eigenvalues λ_1, λ_{n-1} correspond to frequencies 1 and $n - 1$, and all the other non-elliptic eigenvalues have smaller norm. Secondly we compute β, γ such that the elliptic eigenvalues are equal to $|\lambda_1|^2$ and 0.

The Non-elliptic Eigenvalues

The two non-elliptic eigenvalues corresponding to frequency j are given by the eigenvalues of the matrix

$$\begin{pmatrix} \lambda_j^p & c_2 \\ \lambda_j^q & c_3 \end{pmatrix} \quad (11)$$

where λ_j^p, λ_j^q are the j th eigenvalues of

$$C^p = \text{circ}(c_1, c_3, 0, \dots, 0) = \text{circ} \left(\frac{3}{10}, \frac{1}{10}, 0, \dots, 0 \right) \quad (12)$$

and

$$C^q = \text{circ}(c_2, c_0, 0, \dots, 0) = \text{circ} \left(\frac{1}{10}, \frac{5}{10}, 0, \dots, 0 \right) \quad (13)$$

respectively. For a geometric interpretation of the above we use a special decomposition of the two n -gons

$$\mathbf{P} = (P_0, P_1, \dots, P_{n-1}), \quad \mathbf{Q} = (Q_0, Q_1, \dots, Q_{n-1}) \quad (14)$$

as linear combinations of eigenpolygons, coming in pairs $\mathbf{P}_j, \mathbf{Q}_j$ of parallel polygons with the same frequency j . Then, excluding the influence of O on them (which is a similarity), the subdivision process transforms them by

$$\begin{pmatrix} \mathbf{P}'_j \\ \mathbf{Q}'_j \end{pmatrix} = \begin{pmatrix} \lambda_j^p & c_2 \\ \lambda_j^q & c_3 \end{pmatrix} \begin{pmatrix} \mathbf{P}_j \\ \mathbf{Q}_j \end{pmatrix} \quad (15)$$

For the details on constructing such a decomposition see [16].

Substituting c_2, c_3 from (8) and using (2) to calculate λ_j^p, λ_j^q we find the characteristic polynomial of (11)

$$\left| \begin{array}{cc} \frac{3}{10} + \frac{1}{10}\omega^j - x & \frac{1}{10} \\ \frac{1}{10} + \frac{5}{10}\omega^j & \frac{1}{10} - x \end{array} \right| = \frac{2}{100} - \frac{4}{100}\omega^j - \left(\frac{4}{10} + \frac{1}{10}\omega^j \right)x + x^2 \quad (16)$$

with roots

$$\lambda_j = \frac{\frac{4}{10} + \frac{1}{10}\omega^j \pm \sqrt{\frac{8}{100} + \frac{24}{100}\omega^j + \frac{1}{100}\omega^{2j}}}{2} \quad (17)$$

From this we can verify that for every n , the eigenvalues with the largest norms correspond to $j = 1, n-1$.

For the regular case $n = 4$ in particular, we find that the two subdominant eigenvalues are $\frac{2}{5} \pm i\frac{1}{5}$ as expected. The fourth and fifth eigenvalues are $\frac{3}{20} \pm i\frac{\sqrt{15}}{20}$ which means that, even in the regular case, the scheme is not C^2 . Nevertheless, as their norm is near to the square of the norm of the subdominant eigenvalues, the quadratic properties of the scheme are acceptable in practice.

Tuning the Elliptic Eigenvalues

The elliptic eigenvalues of the scheme can be found from the 3×3 matrix

$$\begin{pmatrix} 1 - \beta - \gamma & \beta & \gamma \\ c_0 & c_1 + c_3 & c_2 \\ c_1 & c_0 + c_3 & c_2 \end{pmatrix} = \begin{pmatrix} 1 - \beta - \gamma & \beta & \gamma \\ \frac{5}{10} & \frac{4}{10} & \frac{1}{10} \\ \frac{3}{10} & \frac{6}{10} & \frac{1}{10} \end{pmatrix} \quad (18)$$

see [4]. After adding the second and third columns to the first, and then subtracting the first row from the second and third, we find the characteristic polynomial to be

$$(1-x) \begin{vmatrix} \frac{4}{10} - \beta - x & \frac{1}{10} - \gamma \\ \frac{6}{10} - \beta & \frac{1}{10} - \gamma - x \end{vmatrix} \quad (19)$$

and the two elliptic eigenvalues are

$$\frac{\beta + \gamma - \frac{1}{2} \pm \sqrt{(\beta + \gamma - \frac{1}{2})^2 + \frac{8}{100} - \frac{8}{10}\gamma}}{2} \quad (20)$$

We notice that even after putting the largest elliptic eigenvalue equal to $|\lambda_1|^2$ we still have a degree of freedom left. We use this extra freedom to make the smallest elliptic eigenvalue equal to zero. This is a good strategy because the tuning of the eigenvalues is done for the limit after infinitely many subdivision steps, and by making zero the eigenvalues with no rôle in the limit, we avoid unwanted artifacts in the intermediate steps. Solving the system we find

$$\beta = \frac{2}{5} - |\lambda_1|^2 \quad \gamma = \frac{1}{10} \quad (21)$$

For example, for $n = 4$ we have $|\lambda_1|^2 = \frac{1}{5}$ giving $\beta = \frac{1}{5}$. Notice that $\beta = \frac{1}{5}$ is not optimal for every n , because $|\lambda_1|$ is not equal to $\frac{1}{5}$ for every n . Fig. 8 shows the results of smoothing a valence 3 vertex with $\beta = \frac{1}{5}$ instead of the correct $\beta \approx 0.2518$. The difference is small but noticeable.

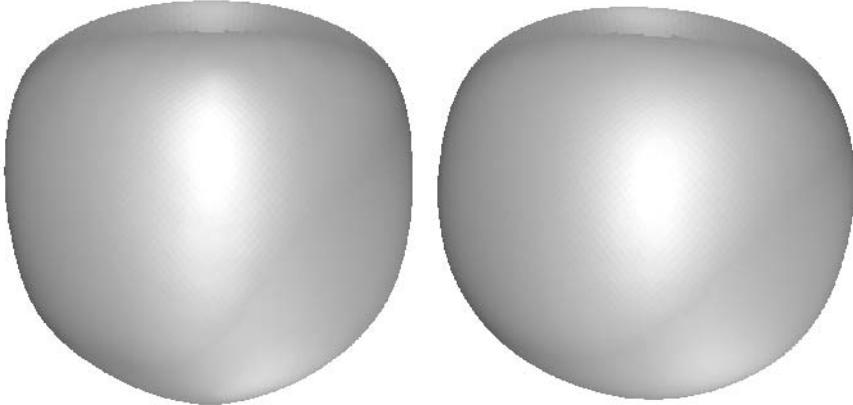


Fig. 8. Left: $\beta = 0.2$ for the valence 3 vertices. Right: $\beta = 0.2518$ for the valence 3 vertices.

3.4 The Support

After defining the stencils, the support of any $\sqrt{5}$ subdivision scheme depends only on the direction of the rotation of the grid at each step, that is, on the combination of steps of $QP(2, 1)$ and $QP(1, 2)$ we use. In the case we always keep the same direction for the rotation then the grid never aligns with the original grid and the support is fractal.

On the other hand, if we alternate the direction of the rotation at each step, then the grid aligns with the original every even number of iterations. In this case the *arity* of the double step is 5. Fig. 9 (left) shows the *footprint* of a double step, that is, the non-zero points of the basis function after a double step, and following [13] we can see that the support is again fractal. The same Figure also shows a polygonal subset of the support calculated with the method in [13]. Fig. 9 (right) shows an approximation of the support based on four iterations of the scheme.

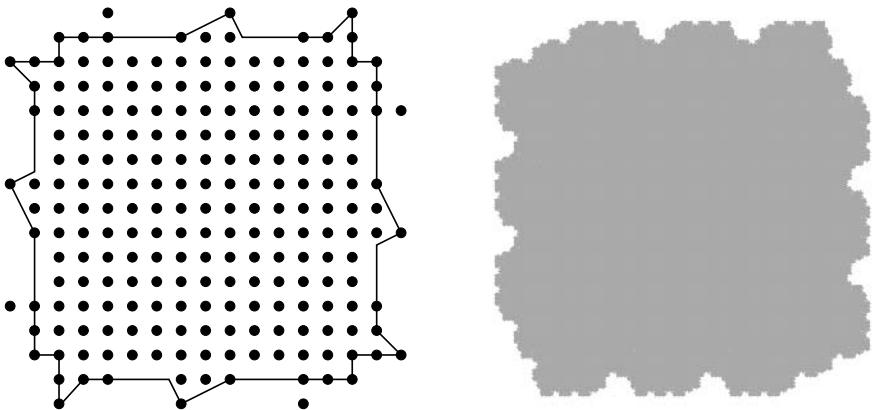


Fig. 9. Left: The footprint of the $\sqrt{5}$ scheme and a polygonal inner bound for the support, calculated using the method in [13]. Right: An approximation to the support based on four iterations of the scheme.

4 Results

We implemented the $\sqrt{5}$ scheme for quadrilateral meshes without boundaries. Due to the direct interpretation of the subdivision process in terms of half-edges, the code was less than three hundred lines, including the implementation of a half-edge structure. Fig. 10 shows some results.

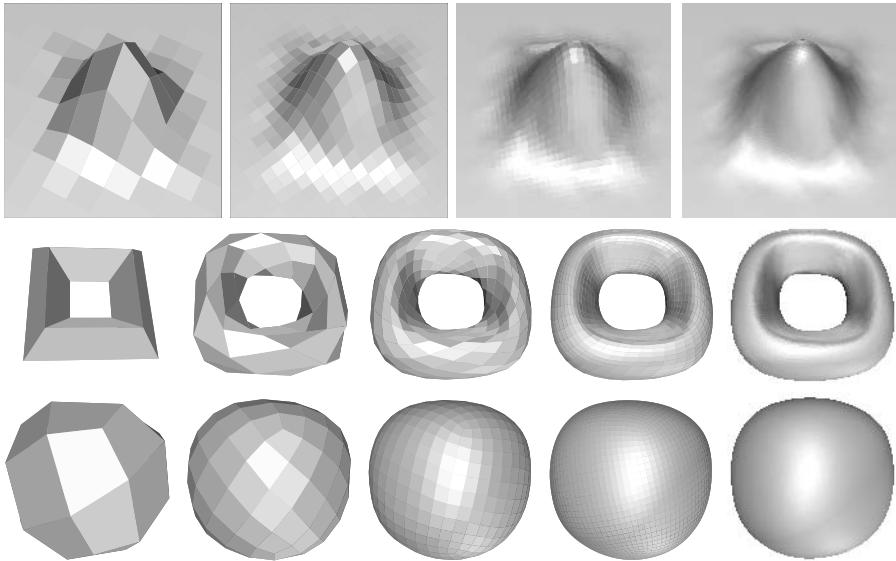


Fig. 10. First row: The basis function. Second row: A torus. Third row: A cube.

5 Conclusion – Future Work

We have presented a new $\sqrt{5}$ subdivision scheme for quad meshes. The most interesting aspects of this scheme are:

- The new points have unique x and y coordinates with respect to the old faces, giving room for fine-tuning with relatively small stencils.
- There is a natural correspondence between the new vertices and the old half-edges. That makes the implementation almost trivial, and leads to a general approach to subdivision as an averaging process with the half-edges as the main primitive.

The rotation of the grid at each subdivision step raises many questions on subdivision matrices with complex subdominant eigenvalues and, as we saw, the reduction of the problem to the case of real eigenvalues through norms, does not give satisfactory answers. In the future we plan a more systematic study of the fourth, fifth and sixth eigenvalues of a subdivision matrix when the second and the third eigenvalues are complex.

Acknowledgement

This work was supported in part by the European Union research project “Multiresolution in Geometric Modelling (MINGLE)” under grant HPRN-CT-1999-00117.

Appendix

The way we calculate a halfedge-vertex in the $\sqrt{5}$ -scheme suggests a general averaging operator which can be used for a general description of subdivision. Let

$$(P_0, P_1, \dots, P_{n-1})^T \quad (22)$$

be a face of the mesh consisting of n vertices, written here as a $n \times 1$ vector, and let

$$c_0 P_0 + c_1 P_1 + \dots + c_{n-1} P_{n-1} \quad (23)$$

be a new point corresponding to the half-edge $\mathbf{P}_0\mathbf{P}_1$. Then, by the rotational symmetry of the connectivity of the face, all the new points corresponding to its half-edges are given by

$$C \times (P_0, P_1, \dots, P_{n-1})^T \quad (24)$$

where C is a circulant matrix.

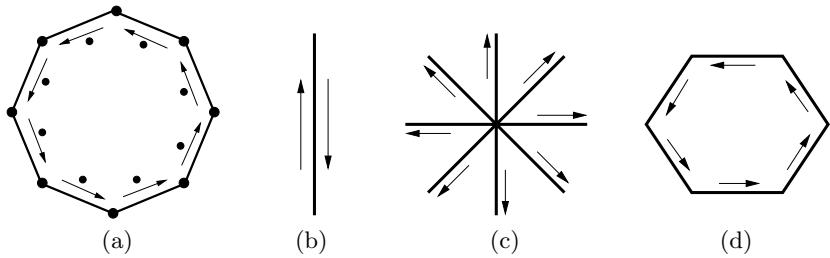


Fig. 11. (a) The circulant averaging operator \mathcal{C} . The small dots, each one corresponding to a half-edge of the face, are the new vertices (halfedge vertices). (b),(c),(d) The mean average of the halfedge-vertices corresponding to an old edge, vertex, or face, is used as a second operator.

Thus, the calculation of new points corresponding to old half-edges can be seen as the action on the mesh of a *circulant averaging operator* \mathcal{C} . Interestingly, many major approximating subdivision schemes can be described as the combination of \mathcal{C} with the mean averaging of the halfedge-vertices corresponding to the same old edge, vertex, or face, see Fig. 11.

Below we give examples of well-known schemes described in this way:

- **Doo-Sabin:** Traditionally, the Doo-Sabin scheme is not described with the use of half-edges. However it has a trivial description in terms of the operator $\mathcal{C} = \text{circ}(\frac{9}{16}, \frac{3}{16}, \frac{1}{16}, \frac{3}{16})$.
- **Loop:** To describe the Loop [18] scheme for triangle meshes we use $\mathcal{C} = \text{circ}(\frac{3}{8}, \frac{1}{4}, \frac{3}{8})$. Then we calculate a vertex-vertex as a linear combination of the corresponding old vertex and the mean average of the points

corresponding to the half-edges starting from it, see Fig. 11(c). To find a mid-edge vertex we average the two points corresponding to the two half-edges of that edge, see Fig. 11(b).

- **Catmull-Clark:** Similarly, for the Catmull-Clark scheme we use $\mathcal{C} = \text{circ}(\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{5}{8})$. We notice that for any vertex, averaging around all the faces adjacent to it gives six times as much weight to the vertices directly connected to it than to the vertex opposite to it, thus leading to the original coefficients proposed in [6].

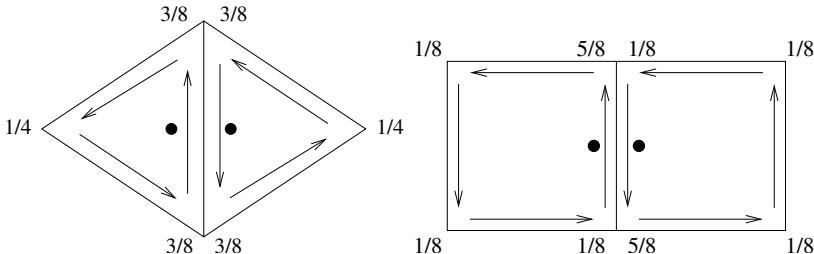


Fig. 12. Left: In the Loop scheme \mathcal{C} acts with coefficients $(\frac{3}{8}, \frac{1}{4}, \frac{3}{8})$. Right: In the Catmull-Clark scheme \mathcal{C} acts with coefficients $(\frac{1}{8}, \frac{5}{8}, \frac{1}{8}, \frac{1}{8})$.

The mean averaging operator has been extensively studied in the context of subdivision [10, 19, 15], and shown to have nice smoothness properties. But it also has serious limitations in describing the existing schemes with their usual stencils. Several extensions of the mean averaging operator have been proposed in [19], but, to the best of our knowledge the circulant averaging operator \mathcal{C} has not been studied in full generality in the context of subdivision.

References

1. M. Alexa. Split operators for triangular refinement. *Computer Aided Geometric Design*, 19(3):169–172, 2002.
2. F. Bachmann and E. Schmidt. *n-Ecke*. B.I.-Hochschultaschenbücher, 1970.
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, C. Gérot, M. A. Sabin, and L. Kobbelt. Simple computation of the eigencomponents of a subdivision matrix in the Fourier domain. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 245–257 (this book). Springer, 2004.
5. E. R. Berlekamp, E. N. Gilbert, and F. W. Sinden. A polygon problem. *Am. Math. Mon.*, 72:233–241, 1965.
6. E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10:350–355, 1978.

7. P. J. Davis. *Circulant matrices*. Chichester: Wiley, New York, 1979.
8. D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10:356–360, 1978.
9. N. Dyn. Subdivision schemes in computer-aided geometric design. In W. Light, editor, *Advances in Numerical Analysis*, volume 2, pages 36–104. Clarendon Press, 1992.
10. N. Dyn, D. Levin, and J. Simoens. Face value subdivision schemes on triangulations by repeated averaging. In A. Cohen, J.-L. Merrien, and L. L. Schumaker, editors, *Curve and Surface Fitting: Saint-Malo 2002*, pages 129–138. Nashboro Press, 2003.
11. M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–28, 2003.
12. I. P. Ivrissimtzis, N. A. Dodgson, and M. A. Sabin. A generative classification of mesh refinement rules with lattice transformations. *Computer Aided Geometric Design*, 21(1):99–109, 2004.
13. I. P. Ivrissimtzis, M. A. Sabin, and N. A. Dodgson. On the support of recursive subdivision. *ACM Transactions on Graphics*. To appear.
14. I. P. Ivrissimtzis and H.-P. Seidel. Evolutions of polygons in the study of subdivision surfaces. *Computing*. To appear.
15. I. P. Ivrissimtzis, K. Shrivastava, and H.-P. Seidel. Subdivision rules for general meshes. In A. Cohen, J.-L. Merrien, and L. L. Schumaker, editors, *Curve and Surface Fitting: Saint-Malo 2002*, pages 229–238. Nashboro Press, 2003.
16. I. P. Ivrissimtzis, R. Zayer, and H.-P. Seidel. Polygonal decomposition of the 1-ring neighborhood of the Catmull-Clark scheme. In *Proc. Shape Modeling International*, 2004. To appear.
17. L. Kobbelt. $\sqrt{3}$ subdivision. In *Proc. ACM SIGGRAPH 2000*, pages 103–112, 2000.
18. C. T. Loop. Smooth subdivision surfaces based on triangles. Master’s thesis, University of Utah, Department of Mathematics, 1987.
19. P. Oswald and P. Schröder. Composite primal/dual $\sqrt{3}$ -subdivision schemes. *Computer Aided Geometric Design*, 20(3):135–164, 2003.
20. U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design*, 12(2):153–174, 1995.
21. M. A. Sabin. Eigenanalysis and artifacts of subdivision curves and surfaces. In A. Iske, E. Quak, and M. S. Floater, editors, *Tutorials on Multiresolution in Geometric Modelling*, pages 69–92. Springer, 2002.
22. I. H. Sloan and S. Joe. *Lattice methods for multiple integration*. Oxford Science Publications. Oxford: Clarendon Press., 1994.
23. M. Stamminger and G. Drettakis. Interactive sampling and rendering for complex and procedural geometry. In *Proc. Eurographics Workshop on Rendering*, pages 151–162, 2001.

Geometrically Controlled 4-Point Interpolatory Schemes

Martin Marinov¹, Nira Dyn², and David Levin²

¹ Computer Graphics Group, RWTH Aachen, Germany

marinov@cs.rwth-aachen.de

² School of Mathematical Sciences, Tel-Aviv University, Israel

{niradyn|levin}@math.tau.ac.il

Summary. We present several non-linear 4-point interpolatory schemes, derived from the “classical” linear 4-point scheme. These new schemes have variable tension parameter instead of the fixed tension parameter in the linear 4-point scheme. The tension parameter is adapted locally according to the geometry of the control polygon within the 4-point stencil. This allows the schemes to remain local and in the same time to achieve two important shape-preserving properties - artifact elimination and convexity-preservation. The proposed schemes are robust and have special features such as “double-knot” edges corresponding to continuity without geometrical smoothness and inflection edges support for convexity-preservation. A convergence proof is given and experimental smoothness analysis is done in detail, which indicates that the limit curves are C^1 .

1 Introduction

1.1 An Overview of the 4-point Scheme

We explore the 4-point subdivision scheme [2] defined by the following mask:

$$p_{2i}^{k+1} = p_i^k, \quad (1)$$

$$p_{2i+1}^{k+1} = (p_i^k + p_{i+1}^k)(w + 1/2) - w(p_{i-1}^k + p_{i+2}^k), \quad (2)$$

where $\{p_i^k\}_i$ is a given control point sequence and w is a *tension parameter*. It is known that the 4-point scheme is C^1 for $0 < w < \frac{1}{8}$. The tension parameter plays a crucial rôle in the geometrically controlled schemes, so now we take a closer look at it. For the trivial value of $w = 0$ the 4-point scheme generates the piecewise linear interpolant to the initial control polygon. For the value $w = \frac{1}{16}$ the limit curve of the 4-point scheme has the best possible Hölder regularity $R_H = 2 - \varepsilon$, i.e., it is almost C^2 . Also, only for this value of w , the scheme reproduces polynomials of degree up to 3 [2]. For $0 < w < 1/16$,

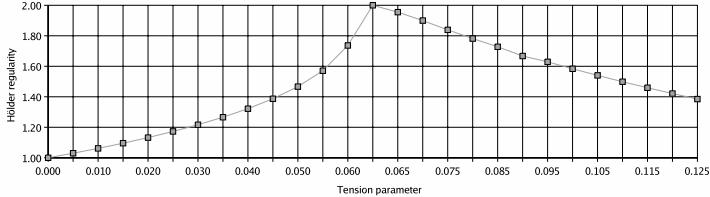


Fig. 1. Hölder regularity of the 4-point scheme for $w \in [0, \frac{1}{8}]$.

R_H is monotone in w , $1 \leq R_H < 2 - \varepsilon$ (Fig. 1). So for the value $w = \frac{1}{16}$ the 4-point scheme has the best regularity and approximation properties.

Fig. 2(upper left) demonstrates the visual effect when changing the tension parameter from 0 to $\frac{1}{16}$: for smaller w the generated curves are closer to the initial control polygon. With $w = \frac{1}{16}$, the smoothest curve is produced. However, $w = \frac{1}{16}$ may cause unpleasant problems as shown in Fig. 2(upper right). Lowering the tension parameter to $w = 0.01$ (Fig. 2(lower left)) solves these problems, but now the curve, which is still C^1 , is hardly recognisable visually from the initial control polygon. So we conclude that an appropriate choice of one tension parameter for an entire control polygon is not always possible.

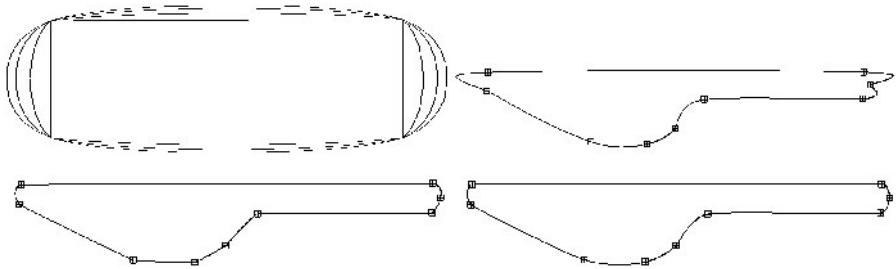


Fig. 2. (Upper left) the effect of w on the shape of the limit curve, (upper right) problems with the visual quality caused by $w = \frac{1}{16}$, (lower left) visually non-smooth curve caused by $w = 0.01$, (lower right) specific w_i^k for each edge of the control polygon alleviates the artifacts and keeps the visual appearance of the curve smooth enough.

1.2 Adaptive Tension Parameter

The idea of using different tension parameters for the edges of a given control polygon at each refinement step was initially investigated in [6]. While it was proved that the non-uniform 4-point scheme in the form

$$p_{2i}^{k+1} = p_i^k, \quad (3)$$

$$p_{e_i^k} = p_{2i+1}^{k+1} = (p_i^k + p_{i+1}^k)(w_i^k + 1/2) - w_i^k(p_{i-1}^k + p_{i+2}^k), \quad (4)$$

produces C^1 limit functions if w_i^k is always chosen in $[\varepsilon \dots \frac{1}{8} - \varepsilon]$ with $\varepsilon > 0$, one still needs to find a procedure for prescribing an appropriate w_i^k for every subdivision step. In this paper, we present several such procedures, where the choice of the tension parameter depends on the geometry of the control polygon. We call them *geometrically controlled* 4-point schemes or shortly *geometrically controlled* schemes.

For a given sequence of control points at refinement level k , $P^k = \{p_i^k\}_i$, where $p_i^k \in \mathbb{R}^2$ or $p_i^k \in \mathbb{R}^3$, we define $e_i^k = p_{i+1}^k - p_i^k$. The subdivision rule (2) is then expressed as

$$p_{e_i^k} = (p_i^k + p_{i+1}^k)(w + \frac{1}{2}) - w(p_{i-1}^k + p_{i+2}^k) = p_i^k + \frac{e_i^k}{2} + w(e_{i-1}^k - e_{i+1}^k). \quad (5)$$

This gives an interesting geometrical interpretation of the insertion rule of the 4-point scheme: the position of $p_{e_i^k}$ is actually a *displacement vector* d_i^k from the middle point of the edge e_i^k , which depends on the difference between the neighbouring edges, scaled by w . We extend its definition to the case of variable tension parameter:

$$d_i^k = w_i^k(e_{i-1}^k - e_{i+1}^k). \quad (6)$$

2 Displacement-safe Geometrically Controlled Schemes

2.1 Definition and Basic Properties

The first class of geometrically controlled schemes which we present here is aimed at solving problems in the shape of the limit curves such as those in the example in Fig. 2(upper right). There are two main questions which have to be answered in order to create such an *artifact-free* scheme:

1. How to detect possible locations in a given control polygon which create artifacts in the limit curve?
2. What subdivision rules to apply in such locations in order to avoid artifacts and at the same time to preserve the limit curve continuity and, if possible, its smoothness?

Our approach for selecting the tension parameter $w_i^k = \omega(i, k)$ tries to answer these two questions, by the composition of two functions $w_i^k = f(g(i, k))$:

1. a *characterising* function $g : I^k \rightarrow [0 \dots c]$. Here I^k is the set of all edge indices in a given refinement level k . $g(i, k)$ characterises the regularity of e_i^k in P^k .

2. a *selecting* function $f : [0 \dots c] \rightarrow [0 \dots W]$. $f(x)$ maps the range of $g(i, k)$ into the range of the available tension parameters.

We restrict our choice to functions which satisfy the following conditions:

1. $g(i, k)$ depends only on $p_{i-1}^k, p_i^k, p_{i+1}^k, p_{i+2}^k$, which ensures that our non-linear schemes have the same support as the original 4-point scheme.
2. $g(i, k)$ is invariant under similitudes.
3. $g(i, k) = 0 \Leftrightarrow |e_i^k| = 0$ and $f(x) = 0 \Leftrightarrow x = 0$, i.e., $w_i^k = 0 \Leftrightarrow |e_i^k| = 0$, i.e., we use the average operation for zero-length edges, thus creating corners. In the vicinity of such edges the limit curve is not G^1 anymore. We choose this behaviour for the following reasons:
 - a) To prevent the loops (see Fig. 8, row 3, column 1).
 - b) To mimic the behaviour of NURBS curves - while we keep the uniform parametrization, we allow corners to be defined by repeated control points in the geometry.
4. $g(i, k) > 0 \Leftrightarrow |e_i^k| > 0$ and $0 < f(x) < W \Leftrightarrow x > 0$, where W is some predefined value for the tension parameter, corresponding to the regular edges. We usually take $W = \frac{1}{16}$.
5. $f(1) = W$. $g(i, k) = 1$ characterises the *regular* edges of P^k , i.e., areas where we can safely use subdivision rules which generate curves with maximum smoothness without creating artifacts.

We found out experimentally that the linear 4-point scheme produces visually-pleasing curves when the edges included in the insertion rule (5) are equidistant. Based on that observation we propose the following two possible combinations of characterising/selecting functions:

$$g(i, k) = \frac{3|e_i^k|}{|e_{i-1}^k| + |e_i^k| + |e_{i+1}^k|}, \quad f(x) = \begin{cases} Wx & , 0 \leq x \leq 1, \\ \frac{W(3-x)}{2} & , 1 < x \leq 3, \end{cases} \quad (7)$$

$$g(i, k) = \frac{3|e_i^k|}{|e_{i-1}^k| + |e_i^k| + |e_{i+1}^k|}, \quad f(x) = \begin{cases} Wx & , 0 \leq x \leq 1, \\ W & , 1 < x \leq 3. \end{cases} \quad (8)$$

We define $g(i, k) = 0$ if $|e_{i-1}^k| + |e_i^k| + |e_{i+1}^k| = 0$.

Furthermore, the geometrically controlled schemes should not introduce new corners during the subdivision process by multiplying consequent control points, i.e., creating zero-length edges. Therefore we introduce the notions:

Definition 1. A geometrically controlled 4-point scheme is safe if for every (i, k) , such that $|e_i^k| > 0$, the newly inserted control point $p_{e_i^k} = p_{2i+1}^{k+1}$ satisfies $p_{e_i^k} \neq p_{2i}^{k+1}$ and $p_{e_i^k} \neq p_{2i+2}^{k+1}$.

Definition 2. A geometrically controlled 4-point scheme is displacement-safe, if there exist $0 < C < \frac{1}{2}$, such that for every (i, k) such that $|e_i^k| \neq 0$, $|d_i^k| \leq C|e_i^k|$.

Lemma 1. *If a given geometrically controlled 4-point scheme is displacement-safe, then it is safe.*

Lemma 2. *If $0 < W < \frac{1}{8}$, then the geometrically controlled schemes defined by (7) or (8) are displacement-safe.*

Proof. Suppose that $g(i, k) \leq 1$. Then we have

$$|d_i^k| = \frac{3W |e_i^k|}{|e_{i-1}^k| + |e_i^k| + |e_{i+1}^k|} |e_{i-1}^k - e_{i+1}^k| < \frac{\frac{3}{8} |e_i^k| (|e_{i+1}^k| + |e_{i-1}^k|)}{|e_{i-1}^k| + |e_i^k| + |e_{i+1}^k|} \leq \frac{3}{8} |e_i^k| .$$

If $g(i, k) > 1$, then

$$\frac{3 |e_i^k|}{|e_{i-1}^k| + |e_i^k| + |e_{i+1}^k|} \geq 1 \Leftrightarrow 2 |e_i^k| \geq |e_{i-1}^k| + |e_{i+1}^k| ,$$

Hence, if $g(i, k) > 1$, $|e_{i+1}^k - e_{i-1}^k| \leq |e_{i+1}^k| + |e_{i-1}^k| < 2 |e_i^k|$. Since $W > \frac{W(3-x)}{2}$ when $x > 1$ then

$$|d_i^k| \leq W |e_{i+1}^k - e_{i-1}^k| < 2W |e_i^k| < \frac{1}{4} |e_i^k| .$$

Corollary 1. *The two proposed geometrically controlled schemes are safe.*

An important consequence of the displacement-safe condition is that p_{2i+1}^{k+1} will be close to the subdivided edge e_i^k even if the neighbouring points p_{i-1}^k and p_{i+2}^k are very far away. A similar property is not true for the original 4-point scheme and our experiments show that it is the major reason for the appearance of artifacts in the generated limit curves. It also leads us to the idea of proposing a geometrical scheme based on the displacement-safe definition:

$$g(i, k) = \frac{C}{W} \frac{|e_i^k|}{|e_{i+1}^k - e_{i-1}^k|} , \quad f(x) = \min(Wx, W) \quad (9)$$

where $0 < C < \frac{1}{2}$, $0 < W < \frac{1}{8}$ and $g(i, k) = 0 \Leftrightarrow |e_{i+1}^k - e_{i-1}^k| = 0$. (9) performs on par with the two other proposed schemes in terms of visual quality, while having an additional global parameter C affecting its behaviour.

2.2 Convergence and Smoothness

We cite relevant results from [6].

Theorem 1. *If for every (i, k) , $w_i^k \in [0 : \frac{1}{8} - \varepsilon]$ for some $\varepsilon > 0$ then the non-uniform 4-point scheme (3), (4) produces C^0 limit functions. If for every (i, k) , $w_i^k \in [\varepsilon : \frac{1}{8} - \varepsilon]$ for some $\varepsilon > 0$ then the scheme (3), (4) generates C^1 limit functions.*

Corollary 2. *The proposed geometrically controlled displacement-safe schemes produce C^0 limit functions.*

Proof. By construction $w_i^k \in [0 : W]$, where $W \in [\varepsilon : \frac{1}{8} - \varepsilon]$ and we apply component-wise Theorem 1.

Remark 1. C^1 continuity of the limit curves of the displacement-safe schemes is not guaranteed since the tension parameters are not bounded away from zero. It can be achieved by the following modification. First we perform several unmodified subdivision steps; then, in every step, we select the new tension parameters so that they will not be smaller than those used in the previous step. Here are two possible strategies to achieve this:

$$\begin{aligned} w_{2i}^{k+1} &= \max(\tilde{w}_{2i}^{k+1}, w_i^k), & w_{2i+1}^{k+1} &= \max(\tilde{w}_{2i+1}^{k+1}, w_i^k) \quad \text{or} \\ w_{2i}^{k+1} &= \max(\tilde{w}_{2i}^{k+1}, \min(w_i^k, w_{i-1}^k)), & w_{2i+1}^{k+1} &= \max(\tilde{w}_{2i+1}^{k+1}, \min(w_i^k, w_{i+1}^k)), \end{aligned}$$

where $\tilde{w}_{2i}^{k+1}, \tilde{w}_{2i+1}^{k+1}$ are computed as in the unmodified scheme. Since $w_i^k > 0$ away from the corner control points (which correspond to repeated vertices in the initial control polygon), and no new corner points are introduced during the subdivision, we can define $\varepsilon = \min_i \{w_i^K \mid w_i^K > 0\}$ for the last refinement step K used with the unmodified rules. Then in view of Theorem 1 this suffices for proving C^1 away from the predefined corners.

Although by applying these modified methods the displacement-safe condition is broken, the visual appearance of the limit curves is similar to the unmodified schemes limit curves, mainly because most of the artifacts are introduced in the first few steps. As in the unmodified schemes, the modified schemes do not increase the support of the 4-point scheme. The cost of using the modification is the additional memory required for keeping the values of the tension parameters used at the previous step.

3 Convexity-preserving Geometrically Controlled Scheme

An important shape-preserving property is convexity-preservation. A considerable amount of research related to subdivision schemes and convexity-preservation was conducted in [3, 4, 5]. The scheme proposed in [4] is the only one which works in the geometrical case, while the schemes investigated in [5] preserve convexity only in the functional case. The method presented in [3] chooses a global w depending on the initial convex functional data, such that the limit function of the 4-point scheme is convex.

Here we define a *geometrically controlled* scheme which preserves convexity in the geometrical case. The scheme handles correctly not only closed convex polygons (Fig. 3(left)), but also open convex polygons (Fig. 3(middle)) and other interesting cases (Fig. 3(right)).

We examine the 4-point scheme stencil for a given e_i^k and distinguish three configurations (Fig. 4):

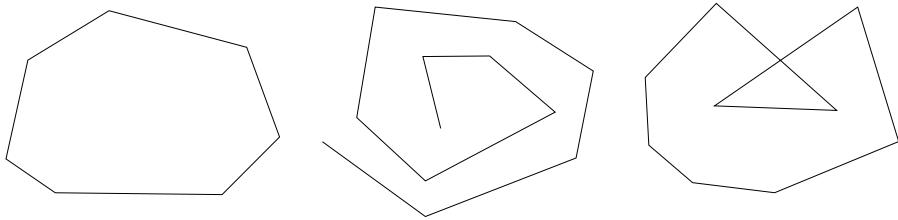


Fig. 3. Convex polygons – (left) closed, (middle) open, (right) self-intersecting.

Definition 3. $p_{i-1}^k, p_i^k, p_{i+1}^k, p_{i+2}^k$ form a convex stencil if p_{i-1}^k, p_{i+2}^k lie in a common half-plane with respect to the line defined by p_i^k, p_{i+1}^k . If at least one of p_{i-1}^k, p_{i+2}^k lies on the line through p_i^k, p_{i+1}^k , then the points form a straight stencil. Otherwise they form an inflection stencil.

Definition 4. e_i^k is a convex edge if $p_{i-1}^k, p_i^k, p_{i+1}^k, p_{i+2}^k$ form a convex stencil. It is a straight edge if the control points form a straight stencil. Otherwise it is an inflection edge.

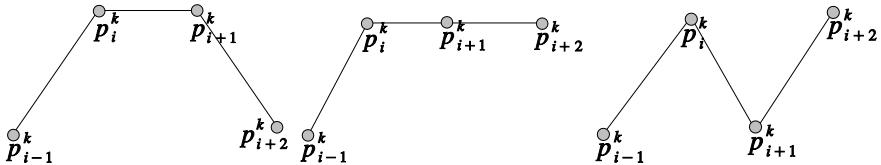


Fig. 4. Stencils – (left) convex, (middle) straight, (right) inflection.

Remark 2. Our definition for a convex stencil/edge is independent of the convexity of the edge with respect to the entire control polygon. So under *convex* we term both convex and concave stencils/edges in a given control polygon. However, in Sect. 4 we employ a special treatment to *inflection* edges and obtain co-convexity-preservation.

Definition 5. The polygon P^k is strictly convex if every e_i^k in P^k is convex.

Definition 6. The polygon P^k is convex if every e_i^k in P^k is either convex or straight.

Definition 7. A line t_i^k passing through p_i^k is a convex tangent if the points p_{i-1}^k, p_{i+1}^k lie in a common half-plane with respect to t_i^k . If p_{i-1}^k or p_{i+1}^k lies on t_i^k , it is a straight tangent. Otherwise t_i^k is an inflection tangent.

Suppose we have defined a convex tangent $\forall p_i^k \in P^k$. $\forall e_i^k$ we define the points $M_i^k = \frac{p_i^k + p_{i+1}^k}{2}$, $L_i^k = t_i^k \cap (M_i^k + \lambda_i^k \tilde{d}_i^k)$, $N_i^k = t_{i+1}^k \cap (M_i^k + \nu_i^k \tilde{d}_i^k)$, $\tilde{d}_i^k = e_{i-1}^k - e_{i+1}^k$, $\lambda_i^k, \nu_i^k \in \mathbb{R}$ (Fig. 5). We use the values λ_i^k, ν_i^k as bounds on w_i^k . In case $\lambda_i^k, \nu_i^k > 0$ we take $0 < w_i^k < \min(\lambda_i^k, \nu_i^k)$ in order to guarantee that if P^k is strictly convex, so is P^{k+1} . To see that this is necessary, assume that $w_i^k = v_i^k$ and $w_{i+1}^k = \lambda_{i+1}^k$. Then $p_{2i+1}^{k+1} = N_i^k$ and $p_{2i+3}^{k+1} = L_{i+1}^k$, and since $N_i^k \in t_{i+1}^k$ and $L_{i+1}^k \in t_{i+1}^k$, the points $p_{2i+1}^{k+1}, p_{2i+2}^{k+1} = p_{i+1}^k, p_{2i+3}^{k+1}$ lie on t_{i+1}^k and P^{k+1} fails to be strictly convex. So we define $\mu_i^k = C \cdot \min(\lambda_i^k, \nu_i^k)$, where $0 < C < 1$ is a user defined constant. We bound w_i^k in the range of values which guarantees convergence: $w_i^k = \min(W, \mu_i^k)$, where $0 < W < \frac{1}{8}$.

Remark 3. It is possible that $\lambda_i^k < 0$ or that $\nu_i^k < 0$. Negative bounds on w_i^k are ignored. We simply substitute $\lambda_i^k = \frac{W}{C}$ if $\lambda_i^k < 0$ and $\nu_i^k = \frac{W}{C}$ if $\nu_i^k < 0$.

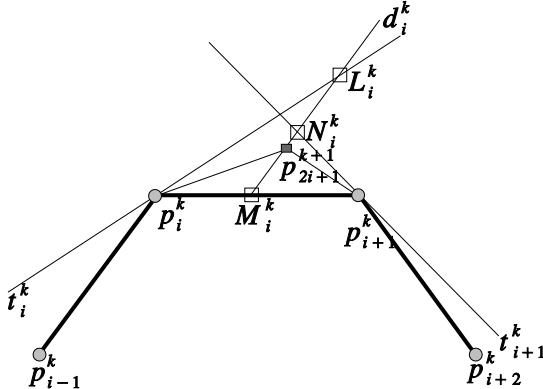


Fig. 5. Convexity-preserving scheme – inserting rule.

To prove that the resulting scheme is convexity-preserving and convergent we use the following notation, illustrated in Fig. 6:

- Every convex tangent t_i^k divides the plane in two half-planes: η_i^k and ξ_i^k such that $p_{i-1}^k, p_{i+1}^k \in \eta_i^k$.
- Every inflection tangent \widehat{t}_i^k divides the plane into two half-planes: $\widehat{\eta}_i^k$ and $\widehat{\xi}_i^k$ such that $p_{i-1}^k \in \widehat{\eta}_i^k$ and $p_{i+1}^k \in \widehat{\xi}_i^k$.
- Every edge e_i^k defines a line l_i^k , which divides the plane in two half-planes: τ_i^k and σ_i^k .
- If e_i^k is convex, then $p_{i-1}^k, p_{i+2}^k \in \tau_i^k$.
- If e_i^k is inflection, then $p_{i-1}^k \in \tau_i^k$ and $p_{i+2}^k \in \sigma_i^k$.
- If e_i^k is straight, then either $p_{i+2}^k \in l_i^k$ and $p_{i-1}^k \in \tau_i^k$, or $p_{i-1}^k \in l_i^k$ and $p_{i+2}^k \in \sigma_i^k$.

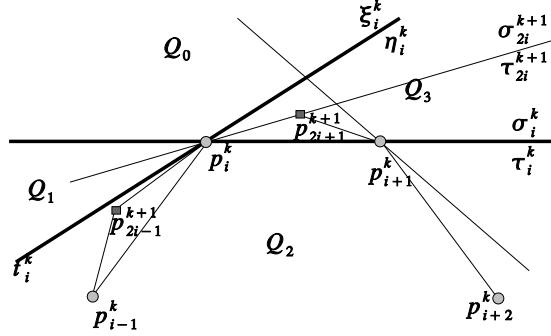


Fig. 6. Convexity-preserving scheme – illustration of the proof of Lemma 3.

Lemma 3. If e_i^k and e_{i-1}^k belonging to an arbitrary P^k are convex, then e_{2i}^{k+1} and e_{2i-1}^{k+1} constructed by the convexity-preserving scheme are convex.

Proof. Let us examine e_{2i}^{k+1} . The lines t_i^k and τ_i^k , which cross at p_i^k , divide the plane into four quadrants $Q_0 = \xi_i^k \cap \sigma_i^k$, $Q_1 = \xi_i^k \cap \tau_i^k$, $Q_2 = \eta_i^k \cap \tau_i^k$ and $Q_3 = \eta_i^k \cap \sigma_i^k$ (Fig. 6). Since e_{2i}^{k+1} and e_{i-1}^k are convex edges and t_i^k is a convex tangent then by construction $p_{2i+1}^{k+1} \in Q_3$ and $p_{2i-1}^{k+1} \in Q_2$. By definition, $p_{2i+2}^{k+1} = p_{i+1}^k \in Q_3 \cap Q_2$. Thus, the line l_{2i}^{k+1} containing e_{2i}^{k+1} is in the interior of the cone $Q_3 \cup Q_1$, and the points p_{2i-1}^{k+1} , p_{2i+2}^{k+1} are in the same half-plane relative to e_{2i}^{k+1} . This proves that e_{2i}^{k+1} is convex. The proof of the convexity of e_{2i-1}^{k+1} is similar.

Remark 4. The requirement that e_{i-1}^k is convex is necessary for the convexity of e_{2i}^{k+1} , since in case e_{i-1}^k is an inflection edge and e_i^k is convex, it is possible to construct P^k such that e_{2i}^{k+1} is an inflection edge.

Remark 5. In the case that P^k is open, we treat the boundary edges $e_0^k, e_{N-1}^k \in P^k$ as convex, by adding “boundary points” p_{-1}^k, p_{N+1}^k such that e_1^{k+1} and e_{2N-2}^{k+1} are also convex. We do this is by taking $p_{-1}^k = p_0^k$ and $p_{N+1}^k = p_N^k$ and $t_0 \perp e_0$ and $t_N \perp e_{N-1}$. Any other choice of p_{-1}^k and p_{N+1}^k such that e_0^k and e_{N-1}^k are convex, generates convex boundary edges $e_1^{k+1}, e_{2N-2}^{k+1}$.

Corollary 3. Given a strictly convex P^k , the refined control polygon P^{k+1} produced by the convexity-preserving scheme is also strictly convex.

Proof. We apply Lemma 3 for every inner edge $e_i^{k+1} \in P^{k+1}$, and Remark 5 for the boundary edges.

The proposed scheme can handle correctly convex control polygons as well. If $p_{i-1}^k, p_i^k, p_{i+1}^k$ lie on the same line then a convex tangent t_i^k does not exist, and we take t_i^k to be a straight tangent. Application of the scheme with a straight

t_i^k obviously leads to $\mu_{i-1}^k = \mu_i^k = 0$, i.e., $w_{i-1}^k = w_i^k = 0$. Thus e_{2i-2}^{k+1} , e_{2i-1}^{k+1} , e_{2i}^{k+1} , e_{2i+1}^{k+1} are straight. Furthermore if e_i^k is convex, but e_{i-1}^k is straight, then e_{2i}^{k+1} is also convex, since $p_{2i-1}^{k+1} \in e_{i-1}^k$ and the proof of Lemma 3 still applies. This proves the following claim:

Corollary 4. *Given a convex P^k , the refined control polygon P^{k+1} produced by the proposed scheme is also convex, so that for every convex $e_i^k \in P^k$ the edges $e_{2i}^{k+1}, e_{2i+1}^{k+1} \in P^{k+1}$ are convex and for every straight $e_i^k \in P^k$ the edges $e_{2i}^{k+1}, e_{2i+1}^{k+1} \in P^{k+1}$ are straight.*

Lemma 4. *Given an initial convex control polygon P^0 , the convexity-preserving scheme produces convex C^0 limit curve, which is a linear segment between the boundary points of each straight edge of P^0 and is a strictly convex curve between the boundary points of each strictly convex sub-polygon of P^0 .*

Proof. By construction $w_i^k \in [0 : W]$, where $W \in [\varepsilon : \frac{1}{8} - \varepsilon]$. Applying component-wise Theorem 1, we conclude that the scheme is convergent and that the limit curve is continuous.

Since all the edges between the boundary points of a straight edge in P^0 lie on that straight edge in every P^k , the limit curve is a line segment between these two boundary points. To see that the limit curve between the boundary points of a strictly convex sub-polygon of P^0 is strictly convex, assume that part of such a segment of the limit curve is a line segment s . Then for k large enough there are three points $p_{i-1}^k, p_i^k, p_{i+1}^k$ on s . This contradicts Corollary 4, since all edges in P^k between the boundary points of a convex edge in P^0 are convex and the strictly convex sub-polygon consists only of convex edges.

We propose the following ways to approximate the tangents t_i^k :

1. $t_i^k = p_{i+1}^k - p_{i-1}^k$
2. $t_i^k \perp b_i^k$, where b_i^k is the bisector of the angle defined by $p_{i-1}^k, p_i^k, p_{i+1}^k$
3. t_i^k is the tangent at p_i^k of the circle passing through $p_{i-1}^k, p_i^k, p_{i+1}^k$

The first method is the simplest and most natural. The tangent coincides with the tangent of the quadratic function passing through $p_{i-1}^k, p_i^k, p_{i+1}^k$, it is easy and fast to compute and it gives the best results in visual quality. Certainly other choices are also possible.

4 Co-convex Geometrically Controlled Scheme

While convexity-preservation is a very useful property it is still not enough to handle a lot of “real world” situations. Artists and engineers which employ CAD systems often desire to have the freedom to define curves which consist of convex and concave parts, joined smoothly. Here we present an extension of the scheme defined in Sect. 3 that produces curves which preserve convexity

and concavity according to the convexity/concavity of the edges of the initial control polygon. We call it the *co-convex* geometrically controlled 4-point scheme.

As we mentioned in Remark 4, in case e_{i-1}^k is an inflection edge, it is not enough that e_i^k is convex for e_{2i}^{k+1} to be convex. To guarantee that e_{2i}^{k+1} is convex when e_i^k is convex, we have to ensure that $p_{2i-1}^{k+1} \in \tau_{2i}^{k+1}$. We define additional inflection tangents $\widehat{t}_i^k, \widehat{t}_{i+1}^k$ for the boundary points p_i^k, p_{i+1}^k of an inflection edge e_i^k and compute $\widehat{L}_i^k = \widehat{t}_i^k \cap (M_i^k + \lambda_i^k \widehat{d}_i^k), \widehat{N}_i^k = \widehat{t}_{i+1}^k \cap (M_i^k + \widehat{\nu}_i^k \widehat{d}_i^k), \widehat{\lambda}_i^k, \widehat{\nu}_i^k \in \mathbb{R}$ (Fig. 7(left)). We replace the negative values among $\lambda_i^k, \nu_i^k, \widehat{\lambda}_i^k, \widehat{\nu}_i^k$ with $\frac{W}{C}$ and select w_i^k for e_i^k as:

$$\mu_i^k = C \cdot \min(\lambda_i^k, \nu_i^k, \widehat{\lambda}_i^k, \widehat{\nu}_i^k), \quad w_i^k = \min(W, \mu_i^k), \quad 0 < W < \frac{1}{8}$$

We define the inflection tangents \widehat{t}_i^k as the bisector of the angle defined by $p_{i-1}^k, p_i^k, p_{i+1}^k$. Any other line dividing this angle can also be used. For $k > 0$ we do not employ convexity-preserving rules when subdividing edges which are descendants of the inflection edges in the refinement hierarchy of P^0 - we use the original 4-point scheme for them (Fig. 7(right)). To see why, examine the following 4-point stencil: $p_{i-1}^0 = (0, -1), p_i^0 = (0, 0), p_{i+1}^0 = (1, 0), p_{i+2}^0 = (1, 1)$. Regardless of the used tangents the inserted point is $p_{2i+1}^1 = (0, 0.5)$. Therefore the configuration $p_{2i}^1, p_{2i+1}^1, p_{2i+2}^1$ is straight and if we apply the convexity-preserving rule, it will lead to a line segment on the limit curve between p_i^0 and p_{i+1}^0 . Still this is probably not the intention of the designer, who can add an additional point on the edge e_i^0 in order to create a straight segment in the curve.

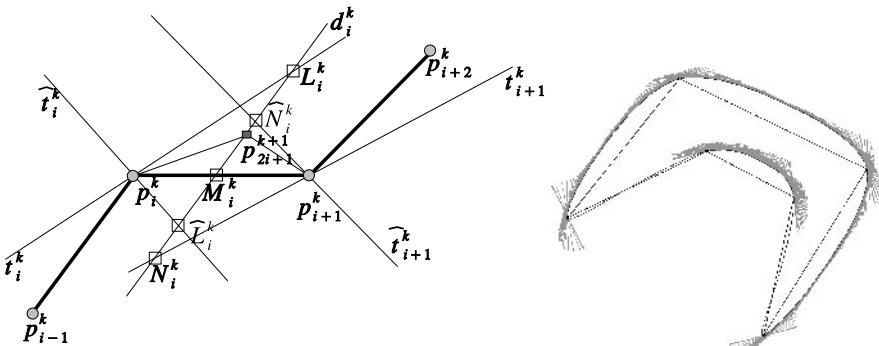


Fig. 7. Co-convex scheme – (left) insertion rule for an inflection edge, (right) co-convex limit curve with the computed convex tangents. Note the segments corresponding to the inflection edges where the linear 4-point scheme is used.

Lemma 5. *If the edge $e_i^k \in P^k$ is convex, then the edges e_{2i}^{k+1} and e_{2i+1}^{k+1} constructed by the proposed co-convex scheme are convex.*

Proof. Let us examine e_{2i}^{k+1} . In case e_{i-1}^k is convex, we apply Lemma 3. If e_{i-1}^k is an inflection edge, then $p_i^k \in \widehat{\xi}_i^k$ with respect to the tangent t_i^k and by construction $p_{2i-1}^{k+1} \in (\eta_i^k \cap \widehat{\eta}_i^k) \subset Q_2$, which implies the convexity of e_{2i}^{k+1} . The proof for e_{2i+1}^{k+1} is analogous.

Remark 6. If the subdivided edge e_i^k defines a corner, i.e., $|e_i^k| = 0$, then similar to the schemes defined in Sect. 2 we prevent loops by using $w_i^k = 0$. If $|e_i^k| = 0$, but $|e_{i-1}^k| > 0$ we treat e_{i-1}^k as convex edge and define the tangent $t_i^k \perp e_{i-1}^k$. By symmetry if $|e_{i+1}^k| > 0$, we define the tangent $t_{i+1}^k \perp e_{i+1}^k$.

Finally we are able to define a robust co-convex scheme with the following properties:

Lemma 6. *Given an arbitrary control polygon P^0 the proposed co-convex scheme produces a C^0 limit curve with the following properties: (a) For every convex sub-polygon of P^0 the corresponding curve segment is strictly convex. (b) For every straight edge e_i^0 the corresponding curve segment is a line segment. (c) All inflection points on the curve are contained in segments of the curve corresponding to inflection edges in P^0 .*

Remark 7. A corner point or a control point p_i^0 in P^0 joining two consecutive straight edges e_{i-1}^0 and e_i^0 can also be inflection points in the limit curve.

Remark 8. The convexity-preserving and co-convex schemes which we propose can be easily applied to 3D control points. Given a tangent line, we define a tangent plane $T_i^k = \{P : P = \lambda_o t_i^k + \lambda_1 (e_{i-1}^k \times e_i^k) + p_i^k\}$. Then all intersections are performed using the tangent plane. If $e_{i-1}^k \times e_i^k = \emptyset$, then either $e_i^k = \emptyset$ or $e_{i-1}^k = \emptyset$, or e_i^k and e_{i-1}^k are on the same line. The first two cases are corner cases and we can define T_i^k to be perpendicular to the non-zero edge. The third one is a straight configuration and every tangent plane passing through the line defined by e_i^k can be used.

Remark 9. One could also combine the co-convex scheme with a displacement-safe scheme in order to get rid of artifacts. This is done easily by taking the minimum of the tension parameters determined by the two schemes. The resulting tension parameter satisfies both conditions and thus the limit curve is artifact-free and co-convex.

5 Experimental Analysis of Smoothness and Examples

For non-linear schemes the conventional tools for checking smoothness are not applicable. An experimental method for analysing these schemes is employed

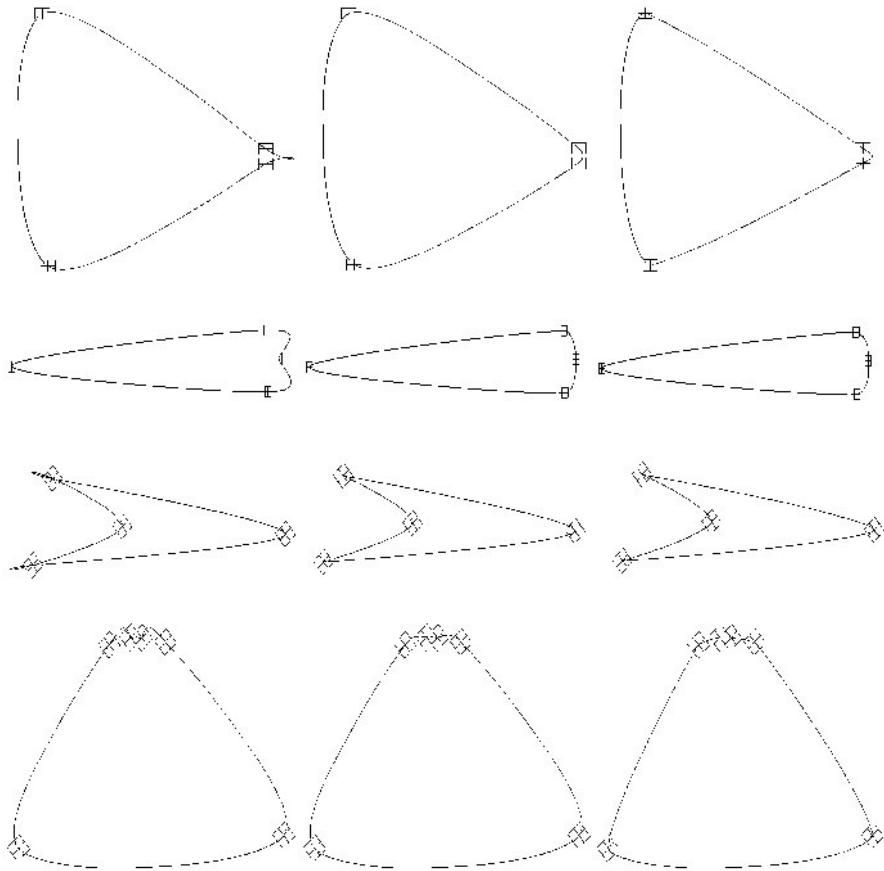


Fig. 8. Examples: left column – the original 4-point scheme with $w = 1/16$, middle column – the displacement-safe scheme (9) with $C = 0.2$, right column – the convexity-preserving scheme with $C = 0.9$.

in [1, 5] by checking numerically the Hölder regularity of several computed limit curves. Since the parametrization we use is uniform, we can employ the algorithm proposed in [5], which is related to the following definitions:

Definition 8. An l times differentiable function $f : \Omega \subset R \rightarrow R$ is said to have Hölder regularity $R_H = l + \alpha$, if there exist $C < \infty$ such that:

$$\left| \frac{\partial^l f(x_1)}{\partial x_1^l} - \frac{\partial^l f(x_2)}{\partial x_2^l} \right| \leq C |x_1 - x_2|^\alpha, \quad \forall x_1, x_2 \in \Omega.$$

Definition 9. An interpolatory subdivision scheme is said to have Hölder regularity $R_H = l + \alpha_l$, if there exist $C < \infty$ and $h > 0$ such that:

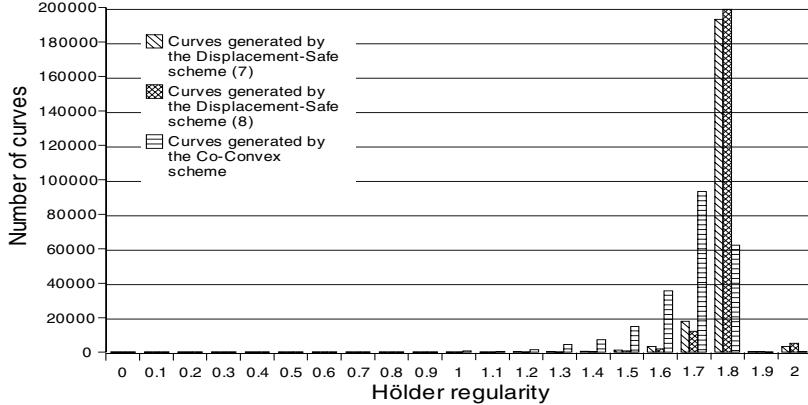


Fig. 9. Experimental Hölder regularity of the limit curves generated by the displacement-safe schemes (7), (8) and the co-convex scheme with $C = 0.9$ applied to 219436 randomly generated control polygons, $W = \frac{1}{16}$. The scheme (9) was not included in the experiment, since its parameter C is a trade off between analytical smoothness and visual quality and therefore can not be determined automatically.

$$\lim_{k \rightarrow \infty} l! 2^{kl} |(\Delta^l f^k)_{i+1} - (\Delta^l f^k)_i| \leq C(2^{-k} h)^{\alpha_l}$$

where $(\Delta f^k)_i = f_{i+1}^k - f_i^k$ and Δ^l is defined recursively as $\Delta \Delta^l f = \Delta^{l+1} f$.

We define $\rho_l^k = l! 2^{kl} \max_i |(\Delta^l f^k)_{i+1} - (\Delta^l f^k)_i|$ and under the assumption that $\rho_l^k \approx C(2^{-k} h)^{\alpha_l}$, one can define the contraction factor λ_l and compute an estimate of α_l :

$$\lambda_l := \frac{\rho_l^{k+1}}{\rho_l^k} \approx \frac{C(2^{-(k+1)} h)^{\alpha_l}}{C(2^{-k} h)^{\alpha_l}} = 2^{-\alpha_l}, \quad \alpha_l := -\log_2 \left(\frac{\rho_l^{k+1}}{\rho_l^k} \right).$$

Computing α_l makes sense only if $\alpha_j \approx 1$ for $j = 0, 1, \dots, l-1$. The proposed method is verified by applying it to schemes for which R_H has been determined analytically - for example the 4-point scheme with $w = \frac{1}{16}$ is proved to have $R_H = 2 - \varepsilon$, and the above method obtains numerically the values $\alpha_1 \approx 1$ and $\alpha_2 \approx 0$. Our implementation checks α_l of the geometrically controlled schemes component-wise, i.e., for $f_i = x_i$ we compute α_l^x , for $f_i = y_i$ we compute α_l^y , and for $f_i = z_i$ we compute α_l^z . α_l then is defined as $\alpha_l = \min(\alpha_l^x, \alpha_l^y, \alpha_l^z)$.

Since R_H of the original 4-point scheme depends on the tension parameter, not surprisingly the geometrically controlled schemes have different R_H depending on the initial geometry. We tested the proposed schemes on the examples given in the paper and we also applied the schemes (7), (8) and the co-convex scheme on about 220K randomly generated control polygons (Fig. 9). All of the curves generated by the schemes (7), (8) after 18 subdivision

steps were C^1 , i.e., $\alpha_0 \approx 1$ and $\alpha_1 > 0$, while only 0.019% of the curves generated by the co-convex scheme were not C^1 . This leads us to the conclusion that the proposed displacement-safe schemes are most probably C^1 , and the co-convex scheme is also C^1 away from corners and straight edges. It is clear also, that the co-convex scheme sacrifices regularity in order to achieve convexity-preservation in some cases (Fig. 9).

Acknowledgement

This work was supported by the European Union research project “Multiresolution in Geometric Modelling (MINGLE)” under grant HPRN-CT-1999-00117.

References

1. Aspert, N., Ebrahimi, T., Vandergheynst, P.: Non-linear subdivision using local spherical coordinates. *Computer Aided Geometric Design*, 20, 165–187 (2003).
2. Dyn, N., Gregory, J., Levin, D.: A 4-point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design*, 4, 257–268 (1987).
3. Dyn, N., Kuijt, F., Levin, D., van Damme, R. M. J.: Convexity preservation of the four-point interpolatory subdivision scheme. *Computer Aided Geometric Design*, 16, 789–792 (1999).
4. Dyn, N., Levin, D., Liu, D.: Interpolatory convexity-preserving subdivision schemes for curves and surfaces. *Computer Aided Geometric Design*, 24, 211–216 (1992).
5. Kuijt, F.: Convexity Preserving Interpolation - Stationary Nonlinear Subdivision and Splines. PhD thesis, University of Twente, Faculty of Mathematical Sciences, (1998).
6. Levin, D.: Using Laurent polynomial representation for the analysis of the non-uniform binary subdivision schemes. *Advances in Computational Mathematics*, 11, 41–54 (1999).

Part VI

— Thinning

Adaptive Thinning for Terrain Modelling and Image Compression

Laurent Demaret¹, Nira Dyn², Michael S. Floater³, and Armin Iske¹

¹ Zentrum Mathematik, Technische Universität München, Munich, Germany
`{demaret|iske}@ma.tum.de`

² School of Mathematical Sciences, Tel-Aviv University, Israel
`niradyn@post.tau.ac.il`

³ Computer Science Department, Oslo University, Norway
`michaelf@ifi.uio.no`

Summary. Adaptive thinning algorithms are greedy point removal schemes for bivariate scattered data sets with corresponding function values, where the points are recursively removed according to some data-dependent criterion. Each subset of points, together with its function values, defines a linear spline over its Delaunay triangulation. The basic criterion for the removal of the next point is to minimise the error between the resulting linear spline at the bivariate data points and the original function values. This leads to a hierarchy of linear splines of coarser and coarser resolutions.

This paper surveys the various removal strategies developed in our earlier papers, and the application of adaptive thinning to terrain modelling and to image compression. In our image test examples, we found that our thinning scheme, adapted to diminish the least squares error, combined with a post-processing least squares optimisation and a customised coding scheme, often gives better or comparable results to the wavelet-based scheme SPIHT.

1 Introduction

This paper concerns the construction of multiresolution approximations to bivariate functions from irregular point samples. These approximations are linear splines over decremental Delaunay triangulations, generated by adaptive thinning algorithms.

A thinning algorithm is a scheme which recursively removes points from a set of scattered data, according to some specific criterion. By recursively removing points, one at a time, a thinning algorithm yields an ordering of the points, which in turn yields a data hierarchy of the input point set.

In *adaptive* thinning algorithms, the criterion for the removal of points depends on both the locations of the given points and the sampled function values at the points. This is in contrast to *non-adaptive* thinning, where the

criterion for removal depends only on the geometry of the given planar point set [10].

Linear splines over Delaunay triangulations are used in order to compute multiresolution approximations to the sampled function. Each subset of points defines a unique Delaunay triangulation (up to co-circularity) and a corresponding linear spline function. The resulting sequence of linear splines constitutes the multiresolution approximations of the sampled function.

For every point in the current subset, we maintain an *anticipated error* which provides a local estimate for the true error incurred by its removal. The error is the deviation of the spline function at the 2D data points from the given function values, measured in some specific norm. The point with minimal anticipated error is considered to be the least significant in the current situation, and is removed. In order to obtain good multiresolution approximations to the sampled function, the choice of removal criterion requires care. We customise our removal strategies according to the application.

The idea of thinning scattered data is closely related to *mesh simplification* methods. Indeed, thinning combined with linear splines over Delaunay triangulations is only one of several mesh simplification methods. One of the earliest methods for mesh simplification is the incremental algorithm of Fowler and Little [11]. An early survey paper is that of Lee [16]. De Floriani, Puppo, and co-workers have proposed several algorithms for mesh simplification and developed good data structures for hierarchical triangulations, see [4] and the references therein. Heckbert and Garland [13] give an extensive survey of simplification methods both for terrain models (triangulated scattered data in the plane) and free form models (manifold surfaces represented by 3D triangle meshes). Specific mesh simplification algorithms include techniques like edge-collapse, half-edge collapse, and vertex collapse. For a more recent survey paper on these methods, see the tutorial [12].

Adaptive thinning algorithms are useful for both model simplification and data compression, and have been applied to hierarchical terrain modelling and to image compression. This paper starts with a generic introduction to adaptive thinning algorithms, before going on to various application-specific measures of anticipated errors, and ends with numerical examples of the algorithms applied to terrain modelling and image compression.

In the image compression application, adaptive thinning constructs a hierarchy of most significant pixel positions in a digital image. We use a thinning scheme, whose anticipated error is measured in the ℓ_2 -norm, combined with a post-processing step which optimises the luminance at the most significant pixels. The positions of the pixels in the set of most significant pixels, along with their optimised luminance values, are then converted into a bit-stream, using a customised coding scheme for scattered data, developed in our previous paper [5]. At the decoder, the transmitted data is used for the image reconstruction, by evaluating the linear spline over the Delaunay triangulation of the transmitted pixels, interpolating the optimised luminance values at the vertices.

The result is a novel image compression scheme, **AT₂^{*}**, which, in our numerical examples, often gives better or comparable compression rates to the well-established wavelet-based compression method SPIHT (Set Partitioning Into Hierarchical Trees).

2 Generic Formulation of Thinning

This section provides a generic introduction to the basic features and concepts of adaptive thinning algorithms. Let $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^2$ denote a finite scattered point set in \mathbb{R}^2 , and let $f_X = (f(x_1), \dots, f(x_N))^T \in \mathbb{R}^N$ denote a corresponding data vector containing point samples taken from an unknown function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ at the points of X .

Thinning is a recursive point removal scheme for bivariate scattered data, whose generic formulation is given by the following algorithm, where n is the number of removals.

Algorithm 1 (Thinning).

- (1) Let $X_N = X$;
- (2) **FOR** $k = 1, \dots, n$
 - (2a) Locate a **removable** point $x \in X_{N-k+1}$;
 - (2b) Let $X_{N-k} = X_{N-k+1} \setminus x$;

Note that thinning constructs, for given data (X, f_X) , a nested sequence

$$X_{N-n} \subset \cdots \subset X_{N-1} \subset X_N = X \quad (1)$$

of subsets of X , where the size $|X_k|$ of any subset X_k in (1) is k , $N-n \leq k \leq N$. Two consecutive subsets in (1) differ only by one point.

In order to select a specific thinning strategy, it remains to give a definition for a removable point in step (2a) above. Before we propose several different preferred removal strategies, let us first discuss our motivation for the construction of the data hierarchy in (1). Our intention is to use the data hierarchy (1) in order to create a multiresolution approximation of the sampled function f from the given data f_X .

The multiresolution approximation of f combines the data hierarchy (1) with *linear splines*. Recall that a linear spline is a continuous function, which is piecewise linear over a partitioning of its domain $\Omega \subset \mathbb{R}^2$. In the setting of this paper, we let the domain Ω coincide with the convex hull $[X]$ of the input point set X . This makes sense, if the convex hull $[Y]$ of any subset $Y \subset X$, constructed by thinning, coincides with the convex hull of X . We ensure this by not removing *extremal* points from X . A convenient choice for the partitioning of Ω is the *Delaunay triangulation* $\mathcal{D}(Y)$ of Y . Although we assume that the reader is familiar with Delaunay triangulation methods, let us recall some of their basic properties, which are relevant to the construction of adaptive thinning algorithms. For a comprehensive discussion of triangulation

methods, in particular Delaunay triangulations, we refer to the textbook [17] and the paper [19].

Firstly, we remark that a Delaunay triangulation $\mathcal{D}(Y)$ of a finite planar point set Y is one, such that for any triangle in $\mathcal{D}(Y)$ its corresponding circumcircle does not contain any point from Y in its interior. This property is termed the *Delaunay property*. Moreover, there is a unique triangulation of Y with the Delaunay property, provided that no four points in Y are cocircular [17]. We assume this condition on the given set X in order to avoid lengthy but immaterial discussions concerning the non-uniqueness of $\mathcal{D}(Y)$, for $Y \subset X$.

Secondly, note that the removal of one point y from Y requires an update of $\mathcal{D}(Y)$ in order to obtain the Delaunay triangulation $\mathcal{D}(Y \setminus y)$. Due to the Delaunay property, this update of $\mathcal{D}(Y)$ is *local*. Indeed, the required retriangulation, incurred by the removal of the vertex y in $\mathcal{D}(Y)$, can be performed by the retriangulation of its cell $C(y)$. Recall that the cell $C(y)$ of y is the union of all triangles in $\mathcal{D}(Y)$ which contain y as a vertex. Fig. 1 shows, for a vertex y in a Delaunay triangulation, the retriangulation of its cell $C(y)$.

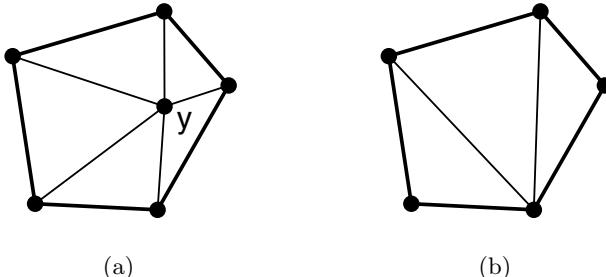


Fig. 1. Removal of the vertex y , and retriangulation of its cell. The five triangles of the cell in (a) are replaced by the three triangles in (b).

For any $Y \subset X$ let

$$\mathcal{S}_Y = \{s : s \in C([Y]) \text{ and } s|_T \text{ linear for all } T \in \mathcal{D}(Y)\},$$

be the spline space containing all continuous functions over $[Y]$ whose restriction to any triangle in $\mathcal{D}(Y)$ is linear. Any element in \mathcal{S}_Y is referred to as a *linear spline* over $\mathcal{D}(Y)$. For given function values f_Y , there is a unique linear spline $L(Y; f) \in \mathcal{S}_Y$ which interpolates f at the points of Y ,

$$L(Y; f)(y) = f(y), \quad \text{for all } y \in Y.$$

3 Adaptive Anticipated Error Measures

The aim of adaptive thinning is to construct a data hierarchy of the form (1) from the given data (X, f_X) , such that for any subset $Y = X_k \subset X$ in (1) the interpolatory linear spline $L(Y; f) \in \mathcal{S}_Y$, is *close* at X to the given function values $f_X \in \mathbb{R}^N$.

In order to establish this, we require that, for some norm $\|\cdot\|$ on \mathbb{R}^N , the *approximation error*

$$\eta(Y; X) = \|L(Y; f)|_X - f_X\| \quad (2)$$

is small. Note that $\eta(Y; X)$ in (2) depends also on the input values f_X , but for notational simplicity we omit this.

For the discrete ℓ_∞ -norm, the approximation error $\eta(Y; X)$ becomes

$$\eta_\infty(Y; X) = \max_{x \in X} |L(Y; f)(x) - f(x)|, \quad (3)$$

whereas for the discrete ℓ_2 -norm, we obtain

$$\eta_2(Y; X) = \sqrt{\sum_{x \in X} |L(Y; f)(x) - f(x)|^2}. \quad (4)$$

Ideally, for any k , $N - n < k < N$, we would like to locate a subset $Y \subset X$ which minimises the error in (2) among all subsets of X of equal size $|Y| = k$. We remark, however, that the problem of finding an algorithm which outputs for any possible input (X, f_X, k) , $N - n < k < |X|$, such an optimal subset Y^* of size $|Y^*| = k$ satisfying

$$\eta(Y^*; X) = \min_{\substack{Y \subset X \\ |Y|=k}} \eta(Y; X) \quad (5)$$

is closely related to the NP-hard *k-centre problem*. For a comprehensive discussion on the *k*-centre problem we refer to the textbook [14, Section 9.4.1] and the survey [20]. To overcome this difficulty, thinning algorithms are based on a *greedy* removal strategy. The application of greedy algorithms to the *k*-centre problem is developed in [15].

Greedy algorithms are in general known as efficient and effective methods of dynamic programming for solving optimisation problems *approximately*. Greedy algorithms typically go through a sequence of steps, where for each step a choice is made that looks best at the moment. For a general introduction to greedy algorithms we recommend the textbook [2, Chapter 16].

For the thinning Algorithm 1, the most natural removal criterion in step (2a) for approximately solving (5) by a greedy algorithm is

Definition 1. (Removal Criterion AT)

For $Y \subset X$, a point $y^* \in Y$ is said to be **removable** from Y , if and only if it satisfies

$$\eta(Y \setminus y^*; X) = \min_{y \in Y} \eta(Y \setminus y; X).$$

We refer to the adaptive thinning algorithm, resulting from this removal criterion in Algorithm 1, as **AT**.

Let us make a few remarks on the idea of this particular definition of a removable point. When using the above removal criterion **AT**, we assign to each current point $y \in Y$ an *anticipated error*

$$e(y) = \eta(Y \setminus y; X),$$

which is incurred by the removal of y . Moreover, we interpret the value $e(y)$ as the significance of the point y in the current subset Y . In this sense, a point y^* whose removal gives the least anticipated error $e(y^*)$ is considered as *least significant* in the current situation, and so it is removed from Y .

For the implementation of the thinning algorithm, we use a priority queue of the points, according to their significances. This priority queue has a least significant point at its head, and is updated after each removal of its head. For more details concerning the efficient maintenance of the priority queue of the scattered points, we refer to our paper [8].

In the remainder of this section, we propose different removal criteria, which are adapted to terrain modelling and to image compression. Let us briefly explain the differences between these two applications. In terrain modelling, it is of primary importance to keep the *maximal* deviation $\eta_\infty(Y; X)$ between the linear spline interpolant $L(f; Y)$ and the given point samples f_X as small as possible. This is in contrast to applications in image compression, where the quality measure relies on the *mean square error*

$$\bar{\eta}_2^2(Y; X) = \eta_2^2(Y; X)/N. \quad (6)$$

We develop two classes of customised adaptive thinning criteria. Those for terrain modelling work with the error measure η_∞ in (3), whereas the anticipated error measures for image compression rely on the discrete ℓ_2 -error η_2^2 in (4). Accordingly, we denote by **AT** $_\infty$ the adaptive thinning algorithm **AT** which works with the ℓ_∞ -norm, whereas **AT** $_2$ is the algorithm **AT** for the choice of the ℓ_2 -norm. The removal criterion for **AT** $_\infty$ cannot be computed locally, but alternative local removal criteria are suggested in the next subsection.

3.1 Anticipated Errors for Terrain Modelling

In this subsection, we propose three locally computable, alternative removal criteria, **AT1**, **AT2** and **AT3**, which reduce the computational costs of the resulting thinning algorithm, in comparison with **AT** $_\infty$. The removal criteria **AT1** and **AT2** require the retriangulation of $Y \setminus y$ for the computation of the anticipated error of any y in Y . Due to the Delaunay property, only the retriangulation of the cell $C(y)$ is required. The removal criterion **AT3** does not require the retriangulation of the cell $C(y)$ for the computation of the anticipated error of y in Y .

The first alternative, **AT1**, measures the anticipated error of a point y only in its cell $C(y)$,

$$e_1(y) = \eta_\infty(Y \setminus y; X \cap C(y)).$$

Definition 2. (Removal Criterion AT1)

For $Y \subset X$, a point $y^* \in Y$ is said to be removable from Y , if and only if it satisfies

$$e_1(y^*) = \min_{y \in Y} e_1(y).$$

We remark that the adaptive thinning algorithm **AT** $_\infty$ is not equivalent to **AT1**. This is confirmed by the following counter-example.

*Example 1. (**AT** $_\infty \neq \textbf{AT1}$)*

Consider the eight data points, $X = \{x_1, \dots, x_8\}$ and the values f_X given as follows.

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|----------|--------|--------|--------|--------|--------|--------|--------|--------|
| x_i | (1, 0) | (2, 0) | (3, 0) | (4, 0) | (5, 0) | (6, 0) | (7, 0) | (1, 1) |
| $f(x_i)$ | 5 | -1 | 0 | -3 | 0 | -1.1 | 2.5 | 0 |

In this case, the extremal points of X are x_1, x_7 and x_8 , so that only the five points x_i , $i = 2, \dots, 6$, can be removed. It is easy to see that both **AT** $_\infty$ and **AT1** remove the point $x_3 = (3, 0)$ first, and then they remove $x_5 = (5, 0)$. In the third step, however, the algorithm **AT1** removes $x_6 = (6, 0)$, whereas the algorithm **AT** $_\infty$ removes $x_4 = (4, 0)$.

Now let us turn to the adaptive thinning algorithm **AT2**, being a simplification of the previous **AT1**. In order to further reduce the required computational costs, the removal criterion **AT2** depends only on the sample values f_Y of the points in the current subset $Y \subset X$. This is in contrast to both **AT** $_\infty$ and **AT1**, which depend on points in X which were removed in previous steps.

The anticipated error of **AT2** is, for any $y \in Y$, given by

$$e_2(y) = \eta_\infty(Y \setminus y; Y).$$

Note that this expression can be rewritten as

$$e_2(y) = |L(Y \setminus y; f)(y) - f(y)|.$$

Definition 3. (Removal Criterion AT2)

For $Y \subset X$, a point $y^* \in Y$ is said to be removable from Y , if and only if it satisfies

$$e_2(y^*) = \min_{y \in Y} e_2(y).$$

We have also explored an adaptive thinning algorithm, **AT3**, which is faster than **AT2**. The algorithm **AT3** does not only ignore the points already removed but also computes an *anticipated error* for each point without needing to temporarily retriangulate its cell.

The basic idea behind **AT3** is to define this anticipated error $e_3(y)$ as the maximum of the *directional anticipated errors* at y in a certain sample of directions. For each neighbouring vertex z of y in $\mathcal{D}(Y)$ we consider the unique point p lying at the intersection of the boundary of $C(y)$ and the straight line passing through z and y (other than z itself). Such a point exists, since $C(y)$ is a *star-shaped polygon*.

The point p is either a vertex of the cell's boundary $\partial C(y)$ or a point on one of its sides. In either case, p lies on at least one edge of $\partial C(y)$. Let us denote such an edge by $[z_2, z_3]$; see Fig. 2. Then the triangle $T_z = [z, z_2, z_3]$, with vertices in $Y \setminus y$, contains y . We call T_z a *directional triangle* of y . We then let

$$e_3^z(y) = |L(T_z; f)(y) - f(y)|$$

be the (unique) directional anticipated error of y in the direction $z - y$, where $L(T_z; f)$ is the linear function which interpolates f at the vertices of T_z . Now, we let

$$e_3(y) = \max_{z \in V_y} e_3^z(y)$$

for the anticipated error of the adaptive thinning algorithm **AT3**, where V_y is the set of all neighbouring vertices of y in $\mathcal{D}(Y)$.

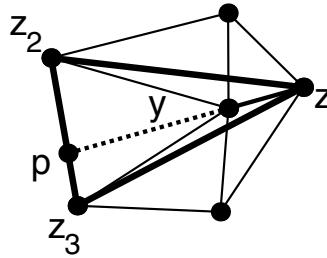


Fig. 2. Directional triangle of y .

Definition 4. (Removal Criterion AT3)

For $Y \subset X$, a point $y^* \in Y$ is said to be **removable** from Y , if and only if it satisfies

$$e_3(y^*) = \min_{y \in Y} e_3(y).$$

A detailed analysis of the complexity of the adaptive thinning algorithms **AT**_∞, **AT1**, **AT2**, and **AT3** can be found in [8]. It is shown in [8] that the asymptotic computational costs of any of these three algorithms is $O(N \log(N))$, but with different constants. For further illustration, we refer to the numerical examples in Sect. 4, where these algorithms are applied to one selected test case from terrain modelling.

3.2 Anticipated Errors for Image Compression

In this subsection, we propose one customised removal criterion for image compression. In this application, the quality of image compression schemes is usually measured in *Peak Signal to Noise Ratio* (PSNR) (7), see the discussion in Sect. 5. It is sufficient for the moment to say that PSNR is an equivalent measure to the reciprocal of the *mean square error* $\bar{\eta}_2^2(Y; X)$ in (6).

The aim of adaptive thinning, when applied to image compression, is to keep the mean square error as small as possible. This is accomplished by using adaptive thinning algorithms, which generate subsets $Y \subset X$ in (1), whose square error $\eta_2^2(Y; X)$ is, among subsets of equal size, small. Therefore, we work with the discrete ℓ_2 -norm η_2 in (4).

Let us now discuss the adaptive thinning algorithm **AT2**, whose anticipated error is given by

$$e(y) = \eta_2^2(Y \setminus y; X), \quad \text{for } y \in Y.$$

By the additivity of η_2^2 and by the observations $X = (X \setminus C(y)) \cup (X \cap C(y))$, and $\eta_2^2(Y \setminus y; X \setminus C(y)) = \eta_2^2(Y; X \setminus C(y))$, for any $y \in Y$, we get

$$\begin{aligned} \eta_2^2(Y \setminus y; X) &= \eta_2^2(Y \setminus y; X \setminus C(y)) + \eta_2^2(Y \setminus y; X \cap C(y)) \\ &= \eta_2^2(Y; X \setminus C(y)) + \eta_2^2(Y \setminus y; X \cap C(y)) \\ &= \eta_2^2(Y; X) + \eta_2^2(Y \setminus y; X \cap C(y)) - \eta_2^2(Y; X \cap C(y)). \end{aligned}$$

Hence, for any $Y \subset X$, the minimisation of $\eta_2^2(Y \setminus y; X)$ is equivalent to minimising the difference

$$e_\delta(y) = \eta_2^2(Y \setminus y; X \cap C(y)) - \eta_2^2(Y; X \cap C(y)), \quad \text{for } y \in Y,$$

where $C(y)$ is the cell of y in $\mathcal{D}(Y)$.

Definition 5. (Removal Criterion **AT2**)

For $Y \subset X$, a point $y^* \in Y$ is said to be **removable** from Y , if and only if it satisfies

$$e_\delta(y^*) = \min_{y \in Y} e_\delta(y).$$

We remark that we can establish the complexity $O(N \log(N))$ for the adaptive thinning algorithm **AT2**, by following along the lines of the analysis in [8]. This, however, is beyond the scope of this survey.

4 Adaptive Thinning in Terrain Modelling

We have implemented the thinning algorithms **AT1**, **AT2**, and **AT3**, together with one non-adaptive thinning algorithm, called **NAT** [8]. The algorithm **NAT**, proposed in [10], ignores the given samples f_X , and favours evenly distributed subsets of points $Y \subset X$. We refrained from implementing the algorithm **AT** $_{\infty}$, since it requires significantly more computations [8], and due to our experience in the univariate setting [7], we do not expect **AT** $_{\infty}$ to be significantly better than **AT1**.

In this section, we compare the performance of the four algorithms **AT1**, **AT2**, **AT3**, and **NAT** in terms of both approximation quality and computational cost on one specific example from terrain modelling. The corresponding data set, *Hurrungane*, contains 23092 data points. Each data point is of the form $(x, f(x))$, where $f(x)$ denotes the terrain's height value sampled at the location $x \in \mathbb{R}^2$. This data set is displayed in Fig. 3 (a) (2D view) and in Fig. 3 (b) (3D view).

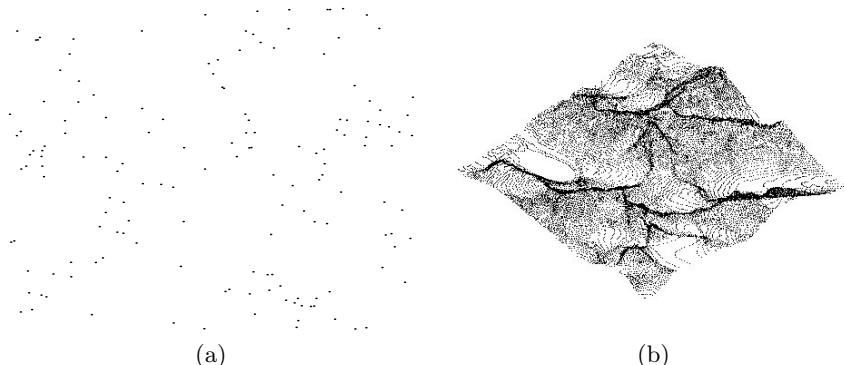


Fig. 3. Hurrungane: (a) 2D view and (b) 3D view.

For all four thinning algorithms, we have recorded both the required seconds of CPU time (without considering the computational costs required for building the initial data structures, such as the Delaunay triangulation) and the sequence of approximation errors $\eta_{\infty}(Y; X)$ after the removal of $n = 1000, 2000, \dots, 22000$ points from X .

Not surprisingly, we found that **NAT** is the fastest method but also the worst one in terms of its approximation error. For example, for $n = 22000$ the algorithm **AT1** takes 247.53 seconds of CPU time, whereas **NAT** takes only 11.37 seconds. On the other hand, we obtain in this particular example $\eta_{\infty}(Y; X) = 278.61$ for **NAT**, but only $\eta_{\infty}(Y; X) = 30.09$ when using **AT1**. The two corresponding triangulations $\mathcal{D}(Y)$ output by **NAT** and **AT1** are displayed in Fig. 4 (a) and (b) (2D view), and in Fig. 5 (a) and (b) (3D view).

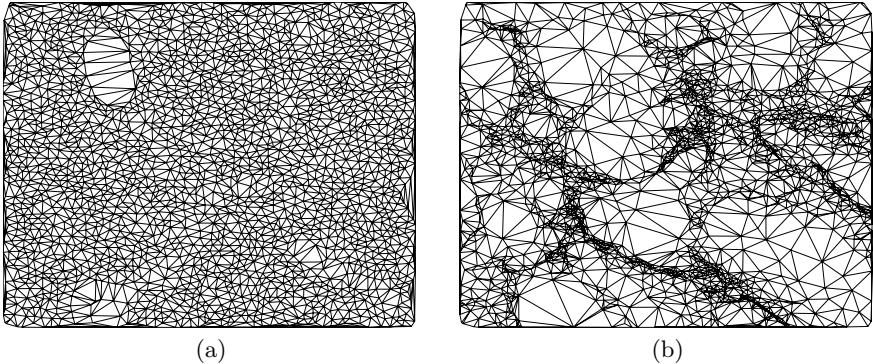


Fig. 4. Thinned Hurrungane with 1092 points, 2D view. (a) **NAT** and (b) **AT1**.

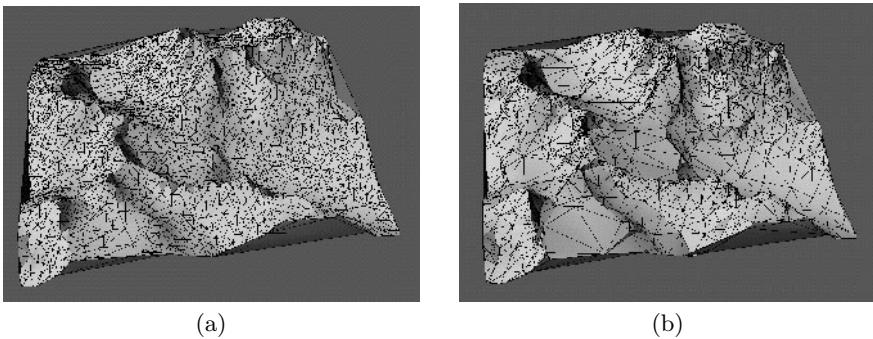


Fig. 5. Thinned Hurrungane with 1092 points, 3D view. (a) **NAT** and (b) **AT1**.

In Fig. 6 (a) and in Fig. 7 (a) the approximation error $\eta_\infty(Y; X)$ as a function of the number of removed points is plotted for the four different thinning algorithms. In Fig. 6 (b) and in Fig. 7 (b) the corresponding seconds of CPU time are displayed.

The graphs show that, with respect to approximation error, the three adaptive thinning algorithms **AT1**, **AT2**, and **AT3** are much better than **NAT**. Among the three adaptive thinning algorithms, **AT1** is the best, followed by **AT3**, and **AT2** is the worst. Note that by definition **AT3** can only be inferior to **AT2** after one removal. In the numerical example, **AT3** has continued to be inferior for about 50 removal steps, after which its approximation error is smaller than that of **AT2**.

As to the computational costs for the adaptive thinning algorithms, **AT3** is the fastest, and **AT1** the slowest, see Fig. 6 (b) and Fig. 7 (b). Our conclusion is that **AT1** is our recommended thinning algorithm. But if computational time is a critical issue, **AT3** is a good alternative.

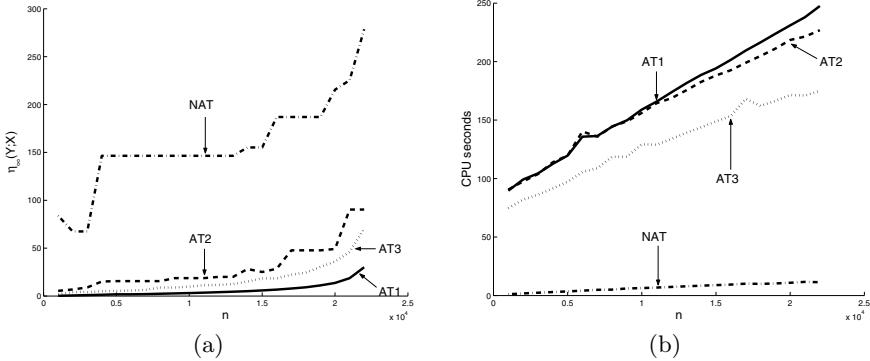


Fig. 6. Hurrungane: comparison between **NAT** (dash-dot line), **AT1** (solid), **AT2** (dashed), and **AT3** (dotted), (a) approximation error and (b) seconds of CPU time.

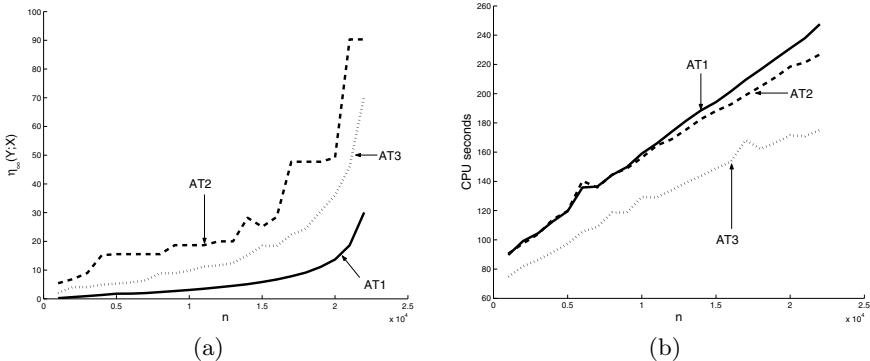


Fig. 7. Hurrungane: comparison between **AT1** (solid line), **AT2** (dashed), and **AT3** (dotted), (a) approximation error and (b) seconds of CPU time.

5 Adaptive Thinning in Image Compression

This section reviews our recent research on the application of adaptive thinning to image compression. The information reduction and efficient coding of digital images are essential for fast transmission across an information channel, such as the Internet. For a comprehensive introduction to image compression and coding, we recommend the textbook [22].

Many of the well-established techniques in image compression, including JPEG2000 [22], are based on wavelets and related techniques, see [3] for a recent survey on wavelet-based image coding. When working with wavelets, digital images are represented by using *rectangular grids* of wavelet coefficients. This is in contrast to adaptive thinning, which works with *scattered data*.

Adaptive thinning constructs a hierarchy of sets of most significant pixels, where for each set the image is approximated by the linear spline over the Delaunay triangulation of the pixels in the set. The idea to approximate an image by first identifying significant pixels is not new (see e.g. [9]). In this section we go further and so obtain a competitive compression scheme, based on adaptive thinning.

Any compression scheme is mainly concerned with the following sequence of tasks.

- (1) data reduction;
- (2) encoding of the reduced data at the sender;
- (3) transmission of the encoded data from the sender to the receiver;
- (4) decoding of the transmitted data at the receiver;
- (5) data reconstruction.

Adaptive thinning is mainly used in the above step (1), the data reduction. In the following discussion, we first explain how adaptive thinning works in image data reduction, before we show how the reconstruction step (5) is accomplished. For a discussion of scattered data coding, required in steps (2) and (4), we refer to our paper [5].

Before we explain any details on our compression scheme, we wish to make a few preliminary remarks concerning our image model. Note that many images contain discontinuities. It might therefore seem like a good idea to approximate such images by *discontinuous* functions over triangulations, instead of keeping to the *continuous* piecewise linear functions of Algorithm 1.

One possibility would be to use piecewise *constant* functions over triangulations. This, however, reduces the approximation quality of the model, because piecewise constants have only $O(h)$ approximation order, rather than $O(h^2)$ for piecewise linear functions. Another possibility would be to use piecewise linear functions that are not necessarily globally continuous. This, however, requires assigning several height values to each vertex, and so in this approach there is significantly more data attached to each triangulation, which leads to higher coding costs.

Unless one works with a very sophisticated coding scheme, which makes fundamental use of the statistical correlation of the data around the image's discontinuities, either of these two discontinuous models leads, in our experience, to inferior compression ratios. Therefore, we prefer to work with a continuous image model, such as the one using continuous piecewise linear functions in Algorithm 1.

5.1 Adaptive Thinning and Image Reduction and Reconstruction

A digital image is a rectangular grid of *pixels*. Each pixel bears a colour value or greyscale *luminance*. For the sake of simplicity, we restrict the following discussion to greyscale images. The image can be viewed as a matrix $F = (f(i, j))_{i,j}$, whose entries $f(i, j)$ are the luminance values at the pixels. The

pixel positions $(i, j) \in X$ are pairs of non-negative integers i and j , whose range is often of the form $[0 \dots 2^p - 1] \times [0 \dots 2^q - 1]$, for some positive integers p, q , where we let $[0 \dots n] = [0, n] \cap \mathbb{Z}$ for any non-negative integer $n \in \mathbb{Z}$. In this case, the size of the pixel set X is $2^p \times 2^q$. Likewise, the entries $f(i, j)$ in F are non-negative integers whose range is typically $[0 \dots 2^r - 1]$, for some positive integer r . In the examples of the test images below, we work with 256 greyscale luminances in $[0 \dots 255]$, so that in this case $r = 8$.

A well-known quality measure for the evaluation of image compression schemes is the *Peak Signal to Noise Ratio* (PSNR),

$$\text{PSNR} = 10 * \log_{10} \left(\frac{2^r \times 2^r}{\bar{\eta}_2^2(Y; X)} \right), \quad (7)$$

which is an equivalent measure to the reciprocal of the mean square error $\bar{\eta}_2^2(Y; X)$ in (6). The PSNR is expressed in dB (decibels). Good image compressions typically have PSNR values of 30 dB or more [22] for the reconstructed image. The popularity of PSNR as a measure of *image distortion* derives partly from the ease with which it may be calculated, and partly from the tractability of linear optimisation problems involving squared error metrics. More appropriate measures of *visual distortion* are discussed in [22].

Adaptive thinning, when applied to digital images, recursively deletes pixels using the thinning Algorithm 1, in combination with the adaptive removal criterion **AT₂** of Sect. 3. In other words, the pixel positions form the initial point set X on which the adaptive thinning algorithm is applied. At any step of the algorithm, a *removable pixel* (point) is removed from the image. The output of adaptive thinning is a set $Y \subset X$ of pixels combined with their corresponding luminances F_Y .

However, due to the regular distribution of pixel positions, the Delaunay triangulations of X , and of its subsets $Y \subset X$, might be non-unique. To avoid this ambiguity, we apply a small perturbation to the pixels X and apply the thinning algorithm to the perturbed pixels.

As a post-process to the thinning, we further minimise the mean square error by *least squares approximation* [1]. More precisely, we compute from the output set Y and the values F the unique *best ℓ_2 -approximation* $L^*(Y; F) \in \mathcal{S}_Y$ satisfying

$$\sum_{(i,j) \in X} |L^*(Y; F)(i, j) - f(i, j)|^2 = \min_{s \in \mathcal{S}_Y} \sum_{(i,j) \in X} |s(i, j) - f(i, j)|^2. \quad (8)$$

Such a unique solution exists since $Y \subset X$. The compressed information to be transferred consists of the output set Y and the corresponding optimised luminances $\{f^*(i, j) = L^*(Y; F)(i, j) : (i, j) \in Y\}$.

Following along the lines of our papers [5, 6], we apply a uniform quantisation to these *optimised* luminances. This yields the quantised symbols $\{Q(f^*(i, j)) : (i, j) \in Y\}$, corresponding to the quantised luminance values $\{\tilde{f}(i, j) : (i, j) \in Y\}$, where $\tilde{f}(i, j) \approx f^*(i, j)$ for $(i, j) \in Y$. The elements

of the set $\{(i, j, Q(f^*(i, j))) : (i, j) \in Y\}$ are coded by using the customised scattered data coding scheme of [5].

At the receiver, the reconstruction of the image F (step (5)) is then accomplished as follows. The *unique* Delaunay triangulation $\mathcal{D}(Y)$ of the pixel positions Y is computed at the decoder, using the same perturbation rules applied previously at the encoder. This defines, in combination with the decoded luminance values $\tilde{F}_Y = \{\tilde{f}(i, j) : (i, j) \in Y\}$, the unique linear spline $\tilde{L}(Y; \tilde{F}_Y) \in \mathcal{S}_Y$ satisfying $\tilde{L}(Y; \tilde{F}_Y)(i, j) = \tilde{f}(i, j)$ for every $(i, j) \in Y$. Finally, the reconstruction of the image is given by the image matrix $\tilde{F} = (\tilde{L}(Y; \tilde{F}_Y)(i, j))_{(i, j) \in X}$.

We denote the novel image compression scheme, presented in this subsection, by **AT₂***.

5.2 Comparison between **AT₂*** and SPIHT

In this subsection we compare the performance of our compression scheme **AT₂*** with that of the wavelet-based compression scheme *Set Partitioning Into Hierarchical Trees* (SPIHT) [18] on two test images. We work with greyscale values of the luminances $f(i, j)$ in $[0 \dots 255]$, i.e., $r = 8$. In the test examples below, we use the range $[0 \dots 31]$ for the quantised symbols $Q(f^*(i, j))$, with $(i, j) \in Y$. In each test case, the compression rate, measured in *bits per pixel* (bpp), is fixed. The quality of the resulting reconstructions is then evaluated by the comparison of the differences in PSNR, and in visual quality.

We remark that the good compression rate of SPIHT is, at low bit rates, comparable with that of the powerful method *EBCOT* [21], which is the basis algorithm of the standard *JPEG2000* [22].

A Geometric Test Image

We first consider one artificial test image, **Reflex**, of small size 128×128 ($p = q = 7$). This geometric test image is displayed in Fig. 11 (a). The purpose of this test case is to demonstrate the good performance of our compression scheme **AT₂*** on texture-free images with sharp edges.

In this test case, we fix the compression rate to 0.251 bpp. The resulting reconstructions corresponding to **AT₂*** and to SPIHT are displayed in Fig. 11 (b),(d). Our compression scheme **AT₂*** yields the PSNR value 41.73 dB, whereas SPIHT provides the inferior PSNR value 30.42 dB. Hence, with respect to this quality measure, our compression method **AT₂*** is much better. Moreover, the reconstruction by **AT₂*** provides also a superior *visual quality* to that of the reconstructed image by SPIHT, see Figs. 11 (b),(d). Indeed, our compression scheme **AT₂*** manages to localise the sharp edges of the test image **Reflex**. Moreover, it avoids undesired oscillations around the edges, unlike SPIHT. This is due to the well-adapted distribution of the 384 most significant pixels, output by the adaptive thinning algorithm **AT₂**, whose Delaunay triangulation is displayed in Fig. 11 (c).



Fig. 8. Fruits. Original image of size 512×512 .

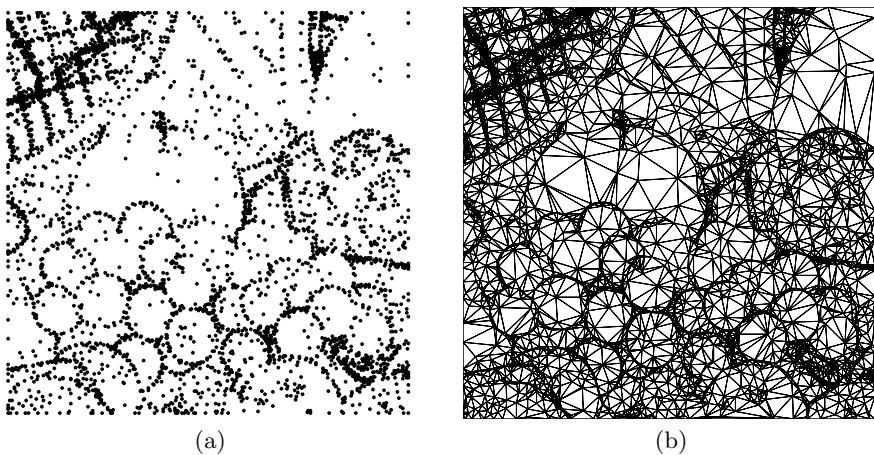


Fig. 9. Fruits. (a) 4044 most significant pixels output by AT_2 and (b) their Delaunay triangulation.



(a)



(b)

Fig. 10. Fruits. Compression at 0.185 bpp and reconstruction by (a) SPIHT with PSNR 32.33 dB and (b) AT_2^* with PSNR 31.85 dB.

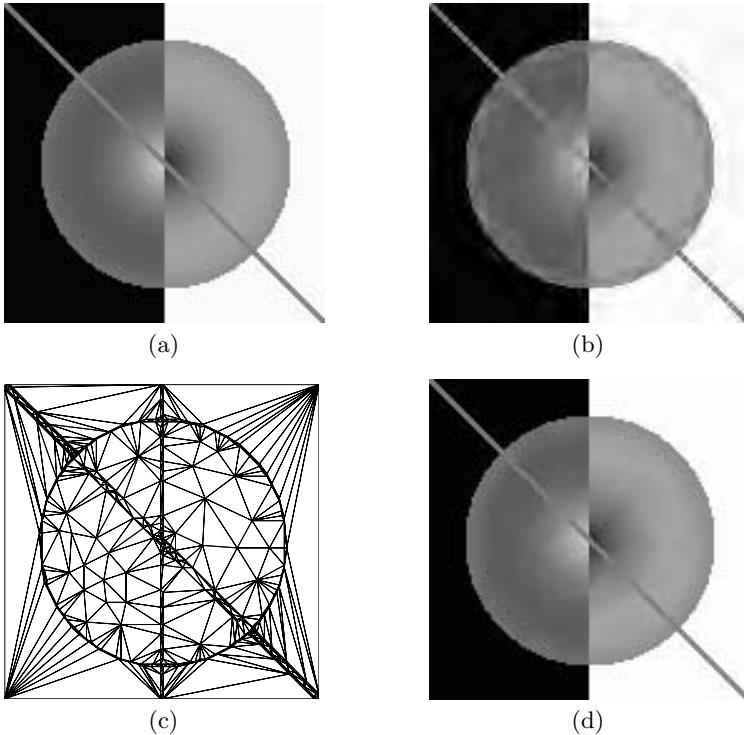


Fig. 11. Reflex. (a) Original image of size 128×128 . Compression at 0.251 bpp and reconstruction by (b) SPIHT with PSNR 30.42 db, (d) \mathbf{AT}_2^* with PSNR 41.73 db. (c) The Delaunay triangulation of the 384 most significant pixels output by \mathbf{AT}_2 .

We have recorded the results of this example, along with those of the following test case, in Table 1.

A Popular Test Case of a Real Image

We considered also applying our compression scheme \mathbf{AT}_2^* to one popular test case of a real image, called **Fruits**, which is also used as a standard test case in the textbook [22]. The original image **Fruits**, of size 512×512 , is displayed in Fig. 8.

It is remarkable that our compression scheme \mathbf{AT}_2^* is, at low bit-rates, quite competitive with SPIHT. This is confirmed by the following comparison between SPIHT and \mathbf{AT}_2^* at the bit-rate 0.185 bpp. The different PSNR values are shown in the second row of Table 1. Note that the PSNR obtained by \mathbf{AT}_2^* is only slightly smaller than that obtained by SPIHT.

Now let us turn to the visual quality of the reconstructions. The reconstruction by SPIHT is shown in Fig. 10 (a), whereas Fig. 10 (b) shows the reconstruction by \mathbf{AT}_2^* .

The set Y of most significant pixel positions obtained by \mathbf{AT}_2 , along with its Delaunay triangulation $\mathcal{D}(Y)$, are displayed in Fig. 9. Note that by the distribution of the most significant pixels, the main features of the image, such as sharp edges and silhouettes, are captured very well. Moreover, our compression scheme \mathbf{AT}_2^* manages to denoise the test image **Fruits** quite successfully, in contrast to SPIHT.

On balance, in terms of the *visual* quality of the two reconstructions of **Fruits**, we feel that our compression scheme \mathbf{AT}_2^* is at least as good as SPIHT.

Table 1. Comparison between the compression schemes SPIHT and \mathbf{AT}_2^* .

| Test Case | bpp | Peak Signal to Noise Ratio (PSNR) | |
|-----------|-------|-----------------------------------|-------------------|
| | | SPIHT | \mathbf{AT}_2^* |
| Reflex | 0.251 | 30.42 | 41.73 |
| Fruits | 0.185 | 32.33 | 31.85 |

Acknowledgements

The assistance of Konstantinos Panagiotou, Bertolt Meier, Eugene Rudy, Georgi Todorov and Rumen Traykov with the implementation of the compression schemes and the preparation of the numerical examples is gratefully appreciated. The authors were partly supported by the European Union within the project MINGLE (Multiresolution in Geometric Modelling), HPRN-CT-1999-00117.

References

1. Å. Björck. *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.
2. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*, 2nd edition. MIT Press, Cambridge, Massachusetts, 2001.
3. G. M. Davis and A. Nosratinia. Wavelet-based image coding: an overview, *Appl. Comp. Control, Signal & Circuits*, B. N. Datta (ed), Birkhauser, 205–269, 1999.
4. L. De Floriani, P. Magillo, and E. Puppo. Building and traversing a surface at variable resolution. *Proceedings of IEEE Visualization* **97**:103–110, 1997.
5. L. Demaret and A. Iske. Scattered data coding in digital image compression. *Curve and Surface Fitting: Saint-Malo 2002*, A. Cohen, J.-L. Merrien, and L. L. Schumaker (eds.), Nashboro Press, Brentwood, 107–117, 2003.

6. L. Demaret and A. Iske. Advances in digital image compression by adaptive thinning. To appear in the *MCFA Annals*, Volume III, 2004.
7. N. Dyn, M. S. Floater, and A. Iske. Univariate adaptive thinning. *Mathematical Methods for Curves and Surfaces: Oslo 2000*, T. Lyche and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 123–134, 2001.
8. N. Dyn, M. S. Floater, and A. Iske. Adaptive thinning for bivariate scattered data. *J. Comput. Appl. Math.* **145**(2):505–517, 2002.
9. Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing* **6**(9):1305–1315, Sep 1997.
10. M. S. Floater and A. Iske. Thinning algorithms for scattered data interpolation. *BIT* **38**:705–720, 1998.
11. R. J. Fowler and J. J. Little. Automatic extraction of irregular network digital terrain models. *Computer Graphics* **13**:199–207, 1979.
12. C. Gotsman, S. Gumhold, and L. Kobbelt. Simplification and Compression of 3D Meshes. *Tutorials on Multiresolution in Geometric Modelling*, A. Iske, E. Quak, and M. S. Floater (eds.), Springer-Verlag, Heidelberg, 319–361, 2002.
13. P. S. Heckbert and M. Garland. Survey of surface simplification algorithms. Technical Report, Computer Science Dept., Carnegie Mellon University, 1997.
14. D. S. Hochbaum (ed.). *Approximation algorithms for NP-hard problems*. PWS Publishing Company, Boston, 1997.
15. A. Iske. Progressive scattered data filtering. *J. Comput. Appl. Math.* **158**(2):297–316, 2003.
16. J. Lee. Comparison of existing methods for building triangular irregular network models of terrain from grid digital elevation models. *Int. J. of Geographical Information Systems* **5**(3):267–285, 1991.
17. F. P. Preparata and M. I. Shamos. *Computational Geometry*. Springer, New York, 1988.
18. A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees, *IEEE Trans. Circuits and Systems for Video Technology* **6**:243–250, 1996.
19. L. L. Schumaker. Triangulation methods. *Topics in Multivariate Approximation*, C. K. Chui, L. L. Schumaker, and F. Utreras (eds.), Academic Press, New York, 219–232, 1987.
20. D. B. Shmoys. Computing near-optimal solutions to combinatorial optimization problems. *DIMACS, Ser. Discrete Math. Theor. Comput. Sci.* **20**:355–397, 1995.
21. D. Taubman. High performance scalable image compression with EBCOT, *IEEE Trans. on Image Processing*, July 2000, 1158–1170, 2000.
22. D. Taubman and M. W. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Kluwer, Boston, 2002.

Simplification of Topologically Complex Assemblies

Carlos Andújar, Marta Fairén, Pere Brunet, and Víctor Cebollada

Department of Computer Science, Universitat Politècnica de Catalunya,
Barcelona, Spain
andujar@lsi.upc.es

Summary. In this paper we present a new simplification approach intended for scenes containing a huge number of simple objects forming a topologically complex assembly. Our method combines appearance preservation and topology reduction by converting a 3D model to and from an intermediate octree representation. During the conversion of the input mesh into an octree, appearance attributes such as colour are stored in the octree nodes. Unlike related approaches, the inside/outside values at octree vertices are computed according to neighbourhood configuration rather than by direct sampling. This allows the reconstructed surface to span only a reduced subset of the terminal nodes of the octree (those which are classified as border nodes), thus avoiding small cracks and removing internal structures not visible from the outside. The reconstruction step of our method succeeds in preserving the appearance of most of the scene objects while drastically simplifying the geometry and topology.

1 Introduction

Over the last few years, many different approaches for surface simplification have been proposed. Most simplification methods seem to be optimised for simplifying a single densely-tessellated surface. Scenes containing a huge number of objects forming a complex assembly pose a serious problem in most surface simplification methods, which are unable to modify the topology of the model. Some methods attempt topological simplification (reviewed in Sect. 2) but they are based on error metrics defined on the boundary surface of the objects thus limiting the amount of topology reduction. Volume-based models [8, 2] provide a convenient framework for controlled topology reduction. However, these methods do not address the problem of preserving surface-appearance attributes such as colour and orientation.

In this paper we present a new simplification method specifically designed for scenes containing millions of simple objects forming a topologically complex assembly (see Fig. 6). Our method combines appearance preservation

and topology reduction by converting a 3D model to and from an intermediate octree representation. The algorithm extends the framework proposed in [2] with two significant contributions: improved surface fitting and colour preservation.

Our simplification algorithm proceeds through three major steps. First, during the discretisation step, the input model is converted into an octree. Then, a polygonal surface is extracted from the octree using a corrected Discretized Marching Cubes [14] algorithm which only depends on the octree nodes. Finally, the topology-reduced surface is further simplified by the iterative application of edge collapses driven by a quadric error metric [6]. The discretisation and reconstruction steps have been designed to preserve colour information and the orientation of important faces.

A key aspect of our method is that, unlike related approaches, the inside/outside values at octree vertices are computed according to neighbourhood configuration rather than by direct sampling. This allows the reconstructed surface to extend across a reduced subset of the terminal nodes of the octree (those which are classified as border nodes), thus aggregating disconnected components and removing internal structures which are not visible from the outside.

Our method is intended to be used in combination with classic surface simplification methods: the leaf nodes of the scene graph containing single objects are simplified using existing surface-based simplification methods, whereas intermediate nodes containing complex assemblies are simplified using our method.

The remainder of this paper is organised as follows: Sect. 2 surveys some previous work on the subject focusing on topology-reducing methods, appearance-preservation and feature-sensitive surface extraction. Sect. 3 outlines volume-based simplification and presents an overview of the algorithm. Octree reconstruction and appearance preservation are explained respectively in Sects. 4 and 5. In Sect. 6 some results are discussed and compared with alternative methods. Finally, Sect. 7 provides concluding remarks and plans for future work.

2 Previous Work

Many automatic simplification algorithms have been published in recent years, and detailed discussion of them is beyond the scope of this paper. The interested reader is referred to [13] for a survey.

Surface simplification literature has focused mainly on simplification algorithms tuned for highly-tessellated, topologically-simple surfaces, which are frequently modelled as triangular meshes. Most methods follow a top-down strategy, performing face reduction directly on the mesh by the iterative application of reduction operators. In the rest of this section, we concentrate

on topology simplification algorithms and feature-preserving reconstruction using volume data.

Vertex clustering methods [17, 12] group nearby vertices into *clusters*, and then replace all vertices inside a cluster by a single vertex. *Pair contraction* methods proceed by the iterative elimination of geometric entities through a local transformation consisting in joining a pair of vertices not necessarily connected by an edge. These methods differ basically in the decimation criteria adopted [6, 16, 4, 18, 11]. All these pair-contraction methods allow topological changes in regions with nearby vertices, but they do not identify and remove triangles that become internal after genus reductions. El-Sana and Varshney [3] proposed a controlled topology simplification scheme based on the notion of alpha-hulls. Alpha prisms are constructed in regions with holes and cavities, and their union is performed by computing their pairwise intersections. This method reduces the genus, removes protuberances and fills cracks, although it does not aggregate unconnected components.

Volume-based methods achieve topology simplification by converting a 3D model to and from a volumetric representation. Andujar *et al.* [1] proposed a simplification algorithm that worked by using an octree as the intermediate model. A two-manifold surface was computed from the octree by a discretized, orthogonal version of Marching Cubes, and the final surface was obtained by a further decimation of the reconstructed surface. He *et al.* [8] proposed a similar topology reduction strategy, based on the discretisation into volume rasters followed by isosurface extraction. Andujar *et al.* [2] extended their initial method to general models by a further decimation of the reconstructed surface, and they proposed a general framework for volume-based simplification.

Another group of algorithms can perform feature-preserving reconstruction and simplification. They use intermediate volume representations (or require volume input data), and they have evolved from the Marching Cubes algorithm, trying to obtain more compact final representations. Montani *et al.* [14] proposed the Discretized Marching Cubes (DMC) algorithm to compact the resulting Marching Cubes isosurface. Using a very small set of planes to approximate the isosurface, a very compact surface is obtained. In [10] and [15] two different approaches are proposed to obtain high quality isosurface approximations while maintaining sharp features. The algorithm in [15] is based on a mesh evolution process where three different filters are combined to control the smoothness of the extracted surface. The approach by Kobbelt *et al.* [10] maintains the original simple structure of the Marching Cubes algorithm (using a vector distance field evaluated at the mesh points). It identifies the cells that contain sharp edges of the original surfaces and additional sample points lying on the features are computed and inserted into the mesh. A similar dual approach is presented in [9].

3 Algorithm Overview

We aim to produce level-of-detail simplifications suitable for accelerating navigation through complex scenes containing hundreds of objects. These scenes are common in many application areas such as ship building and chemical plant design. Given such a scene, we want to precompute level-of-detail representations both for single objects and for groups of objects forming an assembly. Approximations at intermediate levels of the scene graph make sense because such approximations can be cheaper owing to topology changes (e.g. aggregation, genus reduction) and removal of internal structures (e.g. double walls, cavities and even full parts enclosed by other objects and therefore not visible from the outside). Current approaches do not address these problems.

Our method starts by building a new hierarchy of objects. CAD models often exhibit hierarchies based on semantics rather than geometric proximity criteria. Since this grouping is not suitable for LOD generation, we build a new hierarchy using a simplified version of [7], keeping the original graph only for selection and interrogation purposes. Note that in industrial applications, such as ship design, having a good spatial hierarchy and hierarchical levels of detail is mandatory. Object grouping is discussed in more detail in Sect. 5.

The next step is to compute LOD representations at several levels of the new visualisation-friendly scene-graph. LOD representations at the leaves are computed using existing surface simplification methods. At this level volume-based techniques are not suitable because not many topology changes are likely to happen and, for a small error threshold, surface-based approaches are efficient and provide high-quality approximations. Our current implementation uses a sequence of edge collapses driven by a quadric error metric [5] for simplifying single objects.

On the other hand, LOD representations at intermediate levels (i.e. for groups of objects or assemblies) are produced using our method. Each assembly has to be simplified so that its face count is less than the sum of faces of its sons. This way LOD switch management is straightforward because switching to a higher level never increases the face count.

Assembly simplification proceeds through three major steps: discretisation, reconstruction and face reduction. First, during the discretisation step, the input model is converted into an octree having its terminal nodes labelled as *black*, *white* or *terminal grey*. Black and white nodes correspond to cubic regions completely inside (resp. outside) the input model, and terminal grey nodes correspond to cubic regions traversed by the boundary of the input model. From now on terminal grey nodes will be referred to as TG nodes. All TG nodes occur at the same user-provided maximum subdivision level.

The next step is the conversion of the octree back to a polyhedral surface. We compute a triangle mesh by first computing a 3D grid and then applying Marching Cubes over it. The reconstruction starts with a traversal of the octree looking for *border terminal grey* (BTG) nodes. A TG node is said to be border if it has at least one 26-neighbour labelled as white (see Fig. 1). The

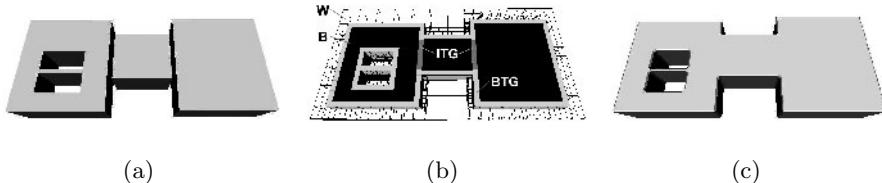


Fig. 1. Terminal grey node classification: (a) input model (three objects); (b) interior of the octree showing node classification; (c) reconstruction (one component).

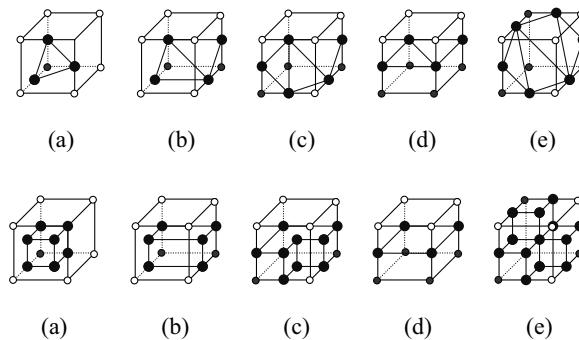


Fig. 2. Patterns for non-ambiguous configurations from (top) Discrete Marching Cubes and (bottom) Orthogonal Reconstruction.

grid definition, above, implies that only a reduced subset of the TG nodes are reconstructed and this is the key to achieving aggregation of disconnected point sets [2] (see Fig. 1).

Regarding the reconstruction from the octree, our two main contributions are related to appearance preservation. The first one deals with colour preservation and it is discussed in Sect. 5. The second one is related to geometry fitting. The main idea is to use the original geometry for reconstructing BTG nodes whenever possible, using an approach similar to Extended Marching Cubes [10]. Details are given in Sect. 4. At this point the two-sided Hausdorff distance between the interior points of the original solid and the interior points of the extracted solids is guaranteed to be less than the length of the main diagonal of terminal octree nodes.

At this point the original surface has been converted into a new, error-bounded, two-manifold mesh. The last step is a simplification process through a sequence of edge collapses again driven by quadric error metrics. Note that since the input of this simplification stage is guaranteed to be a two-manifold without boundary, there is no need to use boundary preservation planes for weighting the quadrics. An important difference with respect to volume-based

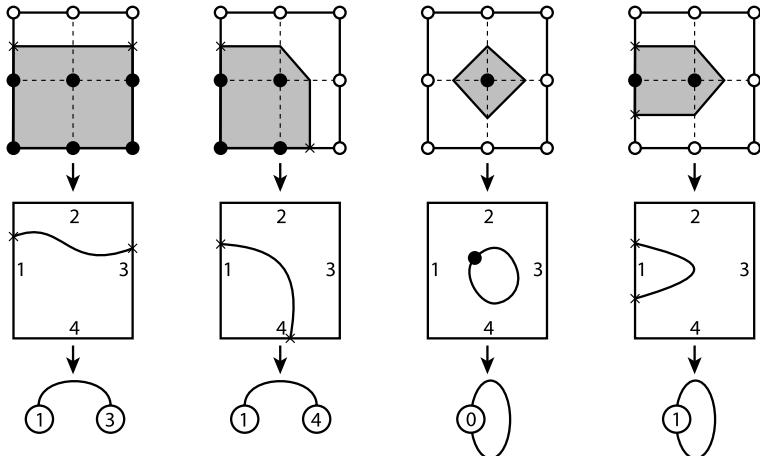


Fig. 3. Connectivity on a BTG face. A BTG face subdivided into eight octants and the surface traversing it (*top*); schematic view of the edges being intersected by the surface (*middle*); connectivity graph (*bottom*). Note that the surface might intersect the BTG face while not intersecting any BTG edge; this is represented in the graph as a face node (*right*).

methods [2] is that during the simplification process, the surface is not restricted to lie inside the set of BTG nodes. That means that the error bound provided by the octree depth no longer applies. However, our experiments show that omitting this restriction at this stage allows for better approximations and shorter running times.

Before simplifying an assembly we filter its open surfaces. These will be simplified separately as single objects because our method would produce an approximation for both sides of the open surface. After filtering, each assembly is converted into an octree. Colour preservation at this stage is discussed in Sect. 5. Note that, as a by-product of the octree construction, we have available the list of input faces passing through each BTG node.

4 Feature-preserving Reconstruction

The input to the reconstruction process is an octree corresponding to a group of objects from the scene being simplified. We aim to compute a new polyhedral surface from the octree which tries to simplify the topology while preserving, as far as possible, the main features of the original model.

A key concept is the distinction between *feasible* and *infeasible* BTG nodes. This distinction is based on comparing the original surface S traversing a BTG node (which is available as a by-product of the octree construction) and the new surface S' that will be produced by Marching Cubes over its octants. A

```

// identify feasible nodes
for each BTG n
    if connectivityMC(n) == connectivityOrig(n) then
        computeVertexPositionsAndNormals() // exact intersection
        // at this point loops are completely determined
        connectLoopsIntoSheets() // uses MC lookup table
        if featureNode(n) then addVertices() endif
        triangulateLoops() // triangle fan for each sheet
    endif
end for

// identify infeasible nodes
for each BTG n
    if connectivityMC(n) != connectivityOrig(n) then
        completeLoops() // midpoint selection
        connectLoopsIntoSheets() // uses MC lookup table
        if featureNode(n) then addVertices() endif
        triangulateLoops() // triangle fan for each sheet
    endif
end for

```

Fig. 4. Reconstruction algorithm.

BTG node is said to be feasible if S and S' intersect the BTG faces with the same connectivity. This connectivity can be encoded by the following graph, $G = (V, E)$ (see Fig. 3): each node of G is assigned to an edge of the BTG node, and there is a link between two nodes in V if and only if the two corresponding edges are intersected by a connected component of S (resp. S').

The reconstruction algorithm is shown in Fig. 4. The algorithm performs two similar traversals of BTG nodes (the only octree nodes that will generate geometry), the former only processing feasible BTG nodes and the later processing infeasible BTG nodes. Triangles inside each BTG node are created as the algorithm processes them.

The first traversal proceeds as follows. First, feasible BTG nodes are detected by comparing the connectivity of the surface that will be produced by a MC reconstruction with the connectivity of the original surface. This comparison can be done quickly by comparing the connectivity graph for the six faces of the BTG node (Fig. 3). In our current implementation we extend each entry of the MC look-up table with the connectivity graph of the six faces to speed-up the comparison process. If the node is identified as feasible, then we compute the exact vertex position and vertex normals by intersecting the original surface inside the node with the edges of the BTG node. Recall that the original surface inside the node is available as a by-product of the octree construction. At this point the loops (the intersection of the output surface with BTG faces) have been completely determined: the topology is given by the MC look-up table, and the vertex positions and normals have been computed

by exact intersection rather than by linear interpolation. Before connecting the loops to form triangles, we check whether the node includes a feature. We use a detection method similar to [10] using the exact vertex normals computed at the vertices. The triangles are finally created by triangulating the loops including the sampled vertices.

The second step performs an almost identical traversal, but this time only infeasible BTG nodes are processed and exact intersections are replaced by midpoint selection.

Maximum deviation and topology reduction are controlled through the maximum subdivision levels. Two originally disconnected point sets A and B will be aggregated in those regions where nodes in BTG(A) are 6-adjacent to nodes in BTG(B). A tunnel is closed in those regions where its TG nodes form a single 6-connected component (i.e. when terminal nodes are large enough to keep white nodes from appearing inside the hole).

There is a drawback of trying to preserve the original geometry on the reconstruction step: unlike [2], the output of our method is no longer independent of the way the input surface is tessellated. However, this is a minor drawback which in practice simply keeps exact edges from appearing when using small subdivision levels.

5 Colour Preservation

This section presents our approach for preserving colour during the different steps of our simplification process. We also provide two alternative methods which are compared in Sect. 6.

Preserving colour in our approach implies keeping track of colour information during the discretisation, reconstruction and face reduction processes. During discretisation we assign to each BTG node a colour which depends on the input faces traversing it. This colour will be referred to as the *BTG colour*. Our current implementation simply selects the colour of the largest face traversing the node. Our experiments show that this choice provides better colour uniformity than averaging colours. During reconstruction, a colour is assigned to each new vertex created inside a BTG node. If the vertex belongs to a feature node, it simply inherits the colour of the corresponding input face; otherwise, it inherits the BTG colour. Note that neighbour BTG nodes can assign the same vertex with a different colour because vertices are shared by up to four BTG nodes. In this case we consider these vertices separately, although referring to the same point. Before entering the face reduction stage, we group triangles into groups of similar colour before doing edge collapses. At this point colour differences can be computed in any appropriate colour space (RGB, CIE, ...). We simply measure RGB difference because our target CAD models use a small colour palette. Therefore our current implementation simply partitions the extracted mesh into face sets with homogeneous colour. This is possible due to the largest-face colour choice discussed above. Finally,

each group of faces with similar colour is simplified using the quadric error metric presented in [5].

We now present two alternative approaches for managing colour preservation. The first alternative consists in omitting the colour segmentation prior to the face reduction step. In this case, colour preservation is left completely to the metric which drives the edge collapses [5]. Our experiments show that our approach provides better approximations in models containing a reduced subset of colours such as the CAD models targeted by our method. The second alternative consists in performing the colour segmentation right at the beginning of our approach, i.e. before discretisation. More precisely, we group objects in the input scene first by colour and then by geometric proximity. As we show in Sect. 6, this alternative requires longer computation times and does not result in better approximations.

6 Results and Discussion

Our simplification algorithm has been implemented and tested on three CAD models. The *engine* model includes 1,976 objects comprising engine parts, piping and diverse equipment. The *oil tanker* model includes 4,096 objects comprising also the hull, bulkheads, pillars and covers. The *taxis* model includes 21 disconnected shells.

Fig. 5 shows our results on the engine model. The model features 92,876 triangles, although from the selected viewpoint only 45,659 of them are visible. Fig. 5(b) shows the output of our simplification method. More precisely, the image corresponds to the eight LOD representations created at the fourth level of the scene graph. The resulting number of triangles is 6,947 (about an 84% reduction on the original); note that the overall appearance has been preserved. Fig. 5(c) shows the output produced by the alternative method consisting in taking into account colour during hierarchy creation. As we can see, overall appearance has also been preserved quite well but the running times for generating these simplifications are much more expensive (around 3 times greater in this example – see Table 1). Fig. 5(d) has been generated by omitting the colour separation process. Notice that some polygons exhibit a noticeable colour degradation. Fig. 6 shows our results on the oil tanker. The oil tanker model has 235,088 triangles, although from the selected viewpoint only 115,103 of them are visible.

Fig. 7 shows our results on the taxi model and compares them with QSlim [6]. The original model has 37,380 triangles. The simplifications on the top have 2,332 (QSlim) and 2,130 triangles (our approach). Our method guarantees a two-manifold result even when the original model is not. Note how the frontal part of the car, which had many shells, has been aggregated with our method into a single connected component.

Running times for the test models on a Pentium III at 800MHz are summarised in Table 1. These times include all the steps, from hierarchy creation

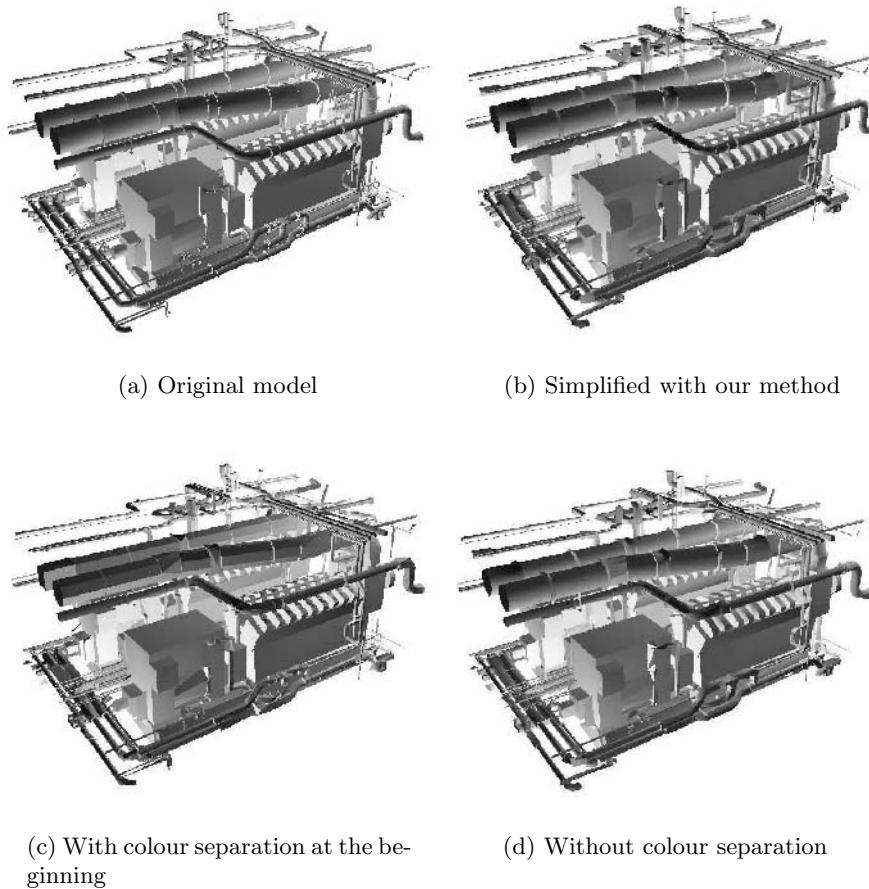
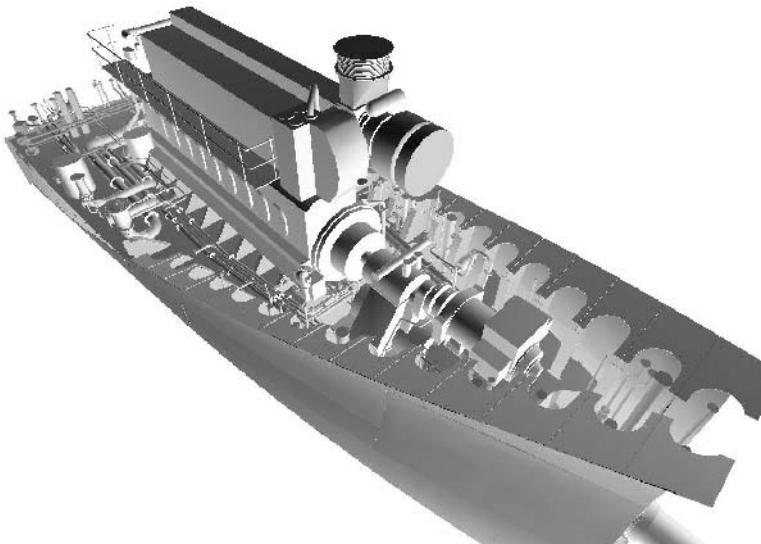


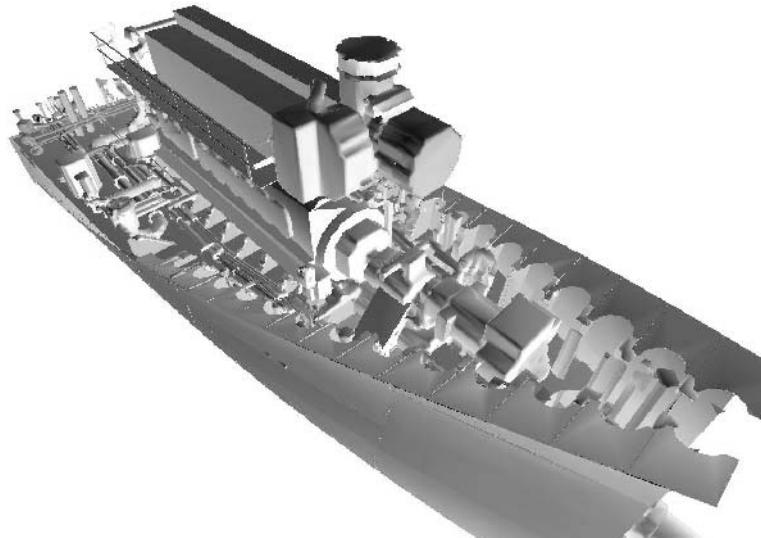
Fig. 5. [Reproduced in colour in Plates 17 and 18.] Results on the engine model.

to face reduction. We would like to point out that our code has not been particularly optimised.

Note that our method might slightly enlarge some objects, but no object disappears unless aggregated with a nearby object. This behaviour contrasts with most surface-based methods which tend to collapse and remove small disconnected objects. We think that in some applications, such as design reviews of CAD models, it is more suitable to avoid this latter behaviour.



(a) Original image



(b) Simplified with our method

Fig. 6. [Reproduced in colour in Plate 19.] Results on the oil tanker model.

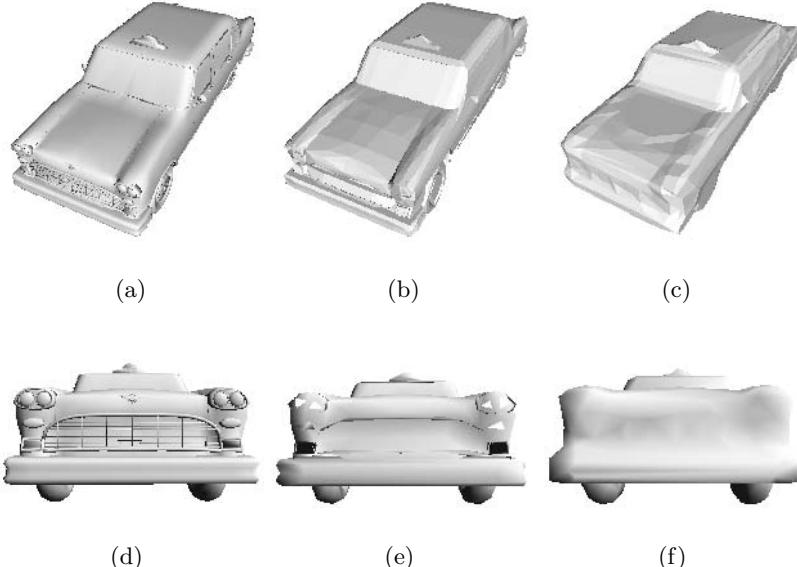


Fig. 7. Results on the taxi model compared to QSlim [6]. Original model (left); Simplification generated by QSlim (middle) and simplification generated by our algorithm (right).

Table 1. Running times for the test models in Figs. 5 and 6.

| | Faces | reduction % | CPU time |
|-----------|--------------|-------------|-----------------|
| Fig. 5(a) | 45,659 | | |
| Fig. 5(b) | 6,947 | 84.8% | 4 min 9 s |
| Fig. 5(c) | 6,945 | 84.8% | 15 min 18 s |
| Fig. 5(d) | 6,979 | 84.7% | 4 min 14 s |
| Fig. 6(a) | 115,103 | | |
| Fig. 6(b) | 57,984 | 49.7% | 9 min 4 s |

7 Conclusions and Future Work

We have presented a new simplification approach intended for generating LOD-representations of scenes containing hundreds of simple objects. Our method is intended to be used in combination with other surface simplification methods: the leaf nodes of the scene graph containing single objects are simplified using existing surface-based simplification methods, whereas intermediate nodes containing complex assemblies are simplified using our method.

Our method combines appearance preservation and topology reduction. Unlike other volume-based approaches, the inside/outside values at octree vertices are computed according to neighbourhood configuration rather than by direct sampling. This allows the reconstructed surface to span only a reduced subset of the terminal nodes of the octree, thus enabling topology changes such as removal of internal structures not visible from the outside.

Due to the adopted colour definition scheme, the approximations produced by our method extend across the set of BTG nodes and completely enclose interior TG nodes. Surfaces inside interior TG nodes are not reconstructed because during visualisation they would be occluded by the new boundary. Since internal structures, which are not visible from the outside, are automatically removed, our method can be seen as a combination of surface simplification and view-independent occlusion culling.

We plan to experiment with other strategies for assigning octree nodes with vertex colours; particularly, we want to explore the use of normal vectors for limiting the number of faces that are considered when selecting the largest face inside a node.

Through simple octree operations, our approach provides a convenient framework for non-penetrating simplification, i.e. approximations completely bounding or completely enclosed inside the input solid. We plan to study the use of bounding approximations along with sampled texture maps using RGBA colour where alpha values are used to clip the approximation to the original silhouette.

Acknowledgements

This work has been partially funded by the projects TIC2000–1009 and TIC2001–2226–C02–01 from the Spanish government. We would like to thank *Sener, Ingeniería y Sistemas* for providing the ship models and Amazing 3D Graphics for the taxi model.

References

1. C. Andújar, D. Ayala, P. Brunet, R. Joan-Arinyo, and J. Solé. Automatic generation of multiresolution boundary representations. *Computer Graphics Forum*, 15(3), 1996.
2. C. Andújar, P. Brunet, and D. Ayala. Topology-reducing surface simplification using a discrete solid representation. *ACM Transactions on Graphics*, 21(2):88–105, 2002.
3. J. El-Sana and A. Varshney. Topology simplification for polygonal virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 4(2), April–June 1998. ISSN 1077-2626.
4. J. El-Sana and A. Varshney. Generalized view-dependent simplification. 18(3):83–94, 1999.

5. M. Garland and P. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization '98*, pages 263–269. IEEE, 1998.
6. M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proc. ACM SIGGRAPH '97*, pages 209–216. Addison Wesley, August 1997.
7. G. Müller and S. Schäfer and D. Fellner. A rapid clustering algorithm for efficient rendering. In *Eurographics Conference, short paper*, 1999.
8. T. He, L. Hong, A. Varshney, and S. W. Wang. Controlled topology simplification. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):171–184, 1996.
9. T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *Proc. ACM SIGGRAPH 2002*, pages 339–346, 2002.
10. L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature sensitive surface extraction from volume data. In *Proc. ACM SIGGRAPH 2001*, pages 57–66, 2001.
11. P. Lindstrom and C. Silva. A memory insensitive technique for large model simplification. In *Proc. IEEE Visualization 2001*, pages 121–126, 2001.
12. K.-L. Low and T.-S. Tan. Model simplification using vertex-clustering. In *Proc. Symposium on Interactive 3D Graphics*, New York, 1997. ACM Press.
13. D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Morgan Kaufmann publishers, 2003.
14. C. Montani, R. Scateni, and R. Scopigno. Discretized Marching Cubes. In *Visualization'94*, pages 281–287. IEEE Computer Society Press, 1994.
15. Y. Ohtake and A. Belyaev. Mesh optimization for polygonized isosurfaces. *Computer Graphics Forum*, 20(3), 2001.
16. J. Popovic and H. Hoppe. Progressive simplicial complexes. In *Proc. ACM SIGGRAPH '97*, pages 217–224, August 1997.
17. J. Rossignac and P. Borrel. Multiresolution 3D approximations for rendering complex scenes. In *Modeling in Computer Graphics*. Springer-Verlag, 1993.
18. W. J. Schroeder. A topology modifying progressive decimation algorithm. In *IEEE Visualization '97*, pages 205–212. IEEE, November 1997.

Topology Preserving Thinning of Vector Fields on Triangular Meshes

Holger Theisel, Christian Rössl, and Hans-Peter Seidel

Max-Planck-Institut für Informatik, Saarbrücken, Germany

{theisel|roessl|hpseidel}@mpi-sb.mpg.de

Summary. We consider the topology of piecewise linear vector fields whose domain is a piecewise linear 2-manifold, i.e. a triangular mesh. Such vector fields can describe simulated 2-dimensional flows, or they may reflect geometric properties of the underlying mesh. We introduce a thinning technique which preserves the complete topology of the vector field, i.e. the critical points and separatrices. As the theoretical foundation, we have shown in an earlier paper 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. This result is applied in a number of compression algorithms which are based on a repeated local modification of the vector field – namely a repeated edge-collapse of the underlying piecewise linear domain.

1 Introduction

Topological methods have become a standard tool for visualising 2D vector fields because they give the opportunity to represent even complex flow structures by only a small number of graphical primitives. Since the introduction of topological methods as a visualisation tool in [11], a number of extensions and modifications of topological concepts have been introduced. The original work [11] considered only first order critical points, i.e. critical points with a non-vanishing Jacobian matrix. Based on an eigenvector/eigenvalue analysis, these critical points were classified into sources, sinks and saddles. Then separatrices starting from the saddle points in the direction of the eigenvectors of the Jacobian matrix were integrated. In addition, separatrices from detachment and attachment points at no-slip boundaries were considered. [14] treats higher order critical points while [19] considers critical points at infinity. In [3], separatrices starting from boundary switch points are considered to separate regions of different inflow/outflow behaviour across the boundary of the flow. [21] considers closed separatrices in the flow. Attachment and separation lines are treated in [12] as additional topological features. In [2] and [5], the topology of scalar fields is treated for visualisation purposes. Initial approaches for visualising 3D topological skeletons are presented in [9].

Flow data sets to be visually analysed are increasingly large and increasingly complex. To deal with this problem, two general approaches have been developed which make use of topological concepts: topological simplification and topology preserving compression of vector fields.

Topology simplification methods are motivated by the assumption that not all topological features of a vector field have the same importance. This happens when some of the critical points and separatrices result from noise in the vector field. The simplest way to solve this problem is to apply a smoothing of the vector field before extracting the topology [4]. 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 to the index of critical points and the index theorem.) [3] uses an area metric to denote unimportant critical points. These points are repeatedly collapsed to more important critical points in the neighbourhood. [4] collapses pairs of first order critical points of opposite index (i.e. a saddle is collapsed with a source, sink, or centre). [18] uses a similar approach but provides a way of consistently updating the underlying vector field. [17] merges clusters of critical points to a higher order critical point. [20] analyses the curvature normal of certain time surfaces to obtain a topology-preserving smoothing of a vector field. The simplification of the topology of scalar fields (which can be considered as a special case of vector field topology) is treated in [6] and [1].

Topology preserving compression techniques can be considered as a contrasting approach to topology preserving simplification techniques. Here, the complete topological skeleton is considered to be important, and compression techniques for the vector field are sought which preserve this topological skeleton completely. [13] is the first approach at an algorithm to compress a vector field under the consideration of preserving the characteristics of critical points. In [15] a method is introduced which preserves not only the critical points but also the behaviour of the separatrices. Unfortunately, this approach gives reasonable compression ratios only for vector fields with a rather poor topology. An approach which gives good compression ratios even for complex topologies (under consideration of both critical points and separatrices) was recently presented in [16]. This approach is based on a theorem which shows that – although the topology of a vector field is a global feature – it can be decided entirely by a local analysis whether a local modification of the vector field is going to change the topology. Based on this, repeated local modifications of the vector field are applied which compress the data set but preserve its topology.

The vector fields we consider in this paper are piecewise linear: in the 2D domain there is a finite number of sample points in which a velocity vector is measured or simulated. To get a vector field, the sample points are trian-

gulated, and a linear interpolation is applied inside each triangle. This way, a piecewise linear vector field can be considered as a 2D triangular mesh with velocity information in each vertex. Furthermore, compression approaches for such vector fields are highly related to thinning approaches for triangular meshes. Thinning approaches reduce the number of triangles in a mesh by applying local collapsing operations. This process is steered by minimising certain error functions between the original and the thinned mesh.

The main approach of this paper is repeatedly to apply half-edge collapses to piecewise linear vector fields (i.e. a triangular mesh) in such a way that the topology of the vector field is preserved. This approach is based on the observation that the topology reflects important properties of the vector field.

The rest of the paper is organised as follows: Sect. 2 gives a short introduction to the topology of 2D vector fields. Sect. 3 introduces a number of topology based equivalence concepts for vector fields. Based on this, Sect. 4 describes three topology preserving thinning algorithms. One of them was already presented in [16], the other two are new approaches. Sect. 5 shows the results while Sect. 6 draws some conclusions.

2 The Topology of 2D Vector Fields

The application of topological methods to a 2D vector field \mathbf{v} aims to separate regions of different flow behaviour in the domain of \mathbf{v} . To do so, the topological skeleton of \mathbf{v} has to be extracted. Here we consider the following features for constructing this skeleton:

- *Critical points* [11] are isolated points with a vanishing velocity. Based on an eigenvector/eigenvalue analysis of the Jacobian of \mathbf{v} , we distinguish between sources, sinks, and saddles¹.
- *Boundary switch points* [3] separate outflow regions and inflow regions across the boundary of the domain of \mathbf{v} . (See Fig. 1a for an example.)
- *Separatrices* [11] are particular stream lines starting either from the saddle points in the direction of the eigenvectors or from the boundary switch points in both forward and backward directions.

Fig. 1b shows an example of a topological skeleton.

This system of points and lines separates the domain of \mathbf{v} into regions of similar flow behaviour: considering two points \mathbf{x}_2 and \mathbf{x}_3 in the same sector of the topological skeleton of \mathbf{v} , the stream lines passing through \mathbf{x}_2 and \mathbf{x}_3 originate in the same source or inflow region, and terminate in the same sink or outflow region. Fig. 1c and 1d give an illustration. In this paper we restrict ourselves to the topological features mentioned above. In particular

¹ There are also centre critical points, i.e. focus points of a circular flow. They are not considered in this paper because they are structurally unstable: adding some noise to a vector field, a centre becomes a source or a sink.

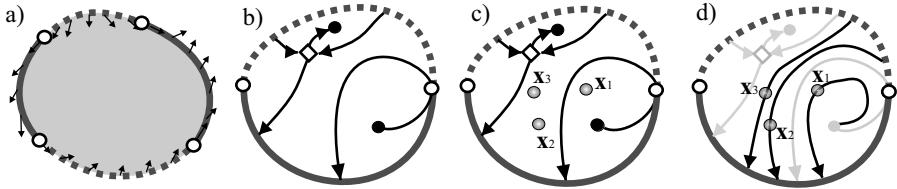


Fig. 1. (a) Vector field with 2 inflow areas (dotted lines) and 2 outflow areas (solid lines) across the boundary of the domain; these areas are separated by boundary switch points (white points); (b) topological skeleton of a vector field consisting of 2 boundary switch points (white points), one boundary outflow region (solid grey line), one boundary inflow region (dotted grey line), one saddle point (white diamond), one source (black point), one sink (black point), and the separatrices (black lines); (c) x_2 and x_3 are in the same sector of the topological skeleton while x_1 is located in a different one; (d) the stream lines through x_2 and x_3 start in the same inflow region and end in the same outflow region while the stream line through x_1 shows a different behaviour: it originates at a source within the domain.

we assume that no higher order critical points [14], closed stream lines [21] or no-slip boundaries [11] appear in the flow.

3 Topologically Equivalent Vector Fields

In order to evaluate a thinning algorithm on piecewise linear vector fields, we have to compare the topological skeleton of the original and the compressed vector field. To do so, a number of topology based equivalence concepts are possible:

1. Two topological skeletons are equivalent if both their critical points and separatrices are identical.
2. Two topological skeletons are equivalent if they have the same critical points (both location and Jacobian matrices), and the corresponding separatrices end in the same critical points or inflow/outflow regions.
3. The topological skeletons of \mathbf{v}_1 and \mathbf{v}_2 are equivalent if there is a one-to-one map between the critical points of \mathbf{v}_1 and \mathbf{v}_2 , such that saddles are mapped to saddles, sources to sources, and sinks to sinks, and corresponding separatrices of \mathbf{v}_1 and \mathbf{v}_2 end in corresponding critical points or inflow/outflow regions.

Note that the equivalence concept 1 is a rather strong one: \mathbf{v}_1 and \mathbf{v}_2 are supposed to have the same critical points (including the Jacobian matrices) and the same separatrices. Concept 2 relaxes this by allowing that corresponding separatrices have different paths (as long as they end in the same critical point or inflow/outflow region). In concept 3 we further relax this by allowing the critical points to move, so long as they do not merge or change their classification.

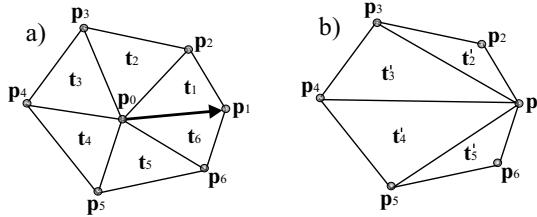


Fig. 2. Configuration for half-edge collapse $\mathbf{p}_0 \rightarrow \mathbf{p}_1$; (a) triangles t_1, \dots, t_6 ; (b) new triangles t'_2, \dots, t'_5 .

4 Thinning the Mesh

In this section we discuss three thinning methods each preserving one of the equivalence concepts mentioned above. All these methods are based on a controlled half-edge collapse of the underlying triangular mesh. This means that a half-edge collapse is only carried out if it is guaranteed to keep the topology unchanged (using one of the equivalence concepts from Sect. 3). The whole process is a greedy optimisation driven by a priority queue. In a 3D setup the priority of a collapse would be some kind of quality measure, such as distance to the original surface. In the 2D case we have an additional degree of freedom. A natural choice would be to locally apply some difference measure for flow fields [8, 10]. In our current implementation we merely assign priorities proportional to edge lengths, preferring short edges for collapse.

The core of this thinning algorithm is therefore an algorithm which decides if a particular half-edge collapse changes the topology of the whole vector field.

Consider a one-ring around a vertex \mathbf{p}_0 consisting of the triangles t_1, \dots, t_n and the vertices $\mathbf{p}_1, \dots, \mathbf{p}_n$ (see Fig. 2a). We describe three algorithms which decide if a half-edge collapse $\mathbf{p}_0 \rightarrow \mathbf{p}_1$ (see Figs. 2a and 2b) changes the topology of \mathbf{v} in the sense of one of the equivalence concepts introduced in Sect. 3.

A thinning algorithm which preserves equivalence concept 1 can easily be formulated as

Algorithm 1 (*check whether a half-edge collapse $\mathbf{p}_0 \rightarrow \mathbf{p}_1$ changes the topology in the sense of concept 1*):

1. If one of the triangles t_1, \dots, t_n contains a critical point or a part of a separatrix, stop and prohibit the half-edge collapse.
2. Simulate the half-edge collapse $\mathbf{p}_0 \rightarrow \mathbf{p}_1$.
3. If one of the new triangles t'_2, \dots, t'_{n-1} contains a critical point, stop and prohibit the half-edge collapse.
4. Allow the half-edge collapse and stop.

This algorithm is justified by the fact that any local modification is going to change the location of a critical point or a separatrix. Hence, a half-edge collapse can only be allowed in regions without any topological features.

A thinning algorithm which preserves equivalence concept 2 was introduced in [16]. There it was shown that it can be decided entirely by a local analysis of the area to be modified, whether or not a local modification (i.e. a half-edge collapse) preserves the topology. This property is remarkable because the topology of a vector field is a global feature: a local modification of a vector field might change the topology at a completely different location. In [16] it was also shown that to check whether a local modification changes the topology, a number of points on the boundary of the modified area has to be collected, and their cyclic order before and after the collapse has to be compared. This gives the following algorithm:

Algorithm 2 (*check whether a half-edge collapse $\mathbf{p}_0 \rightarrow \mathbf{p}_1$ changes the topology in the sense of concept 2*):

1. *Check if there are critical points inside $\mathbf{D}' = (\mathbf{t}_1, \dots, \mathbf{t}_n)$. If so, prohibit the half-edge collapse and stop.*
2. *Collect all separatrices which pass through \mathbf{D}' . For each separatrix, store the entry point and exit point of \mathbf{D}' in a cyclic list L_1 which is ordered in the same cyclic order as the points on the closed polygon $((\mathbf{p}_1, \mathbf{p}_2), \dots, (\mathbf{p}_{n-1}, \mathbf{p}_n), (\mathbf{p}_n, \mathbf{p}_1))$.*
3. *If a separatrix enters \mathbf{D}' more than once, prohibit the half-edge collapse² and stop.*
4. *Compute the boundary switch points of the vector field on the polygon $((\mathbf{p}_1, \mathbf{p}_2), \dots, (\mathbf{p}_{n-1}, \mathbf{p}_n), (\mathbf{p}_n, \mathbf{p}_1))$, insert these points into L_1 .*
5. *Simulate the half-edge collapse $\mathbf{p}_0 \rightarrow \mathbf{p}_1$ while storing the original configuration (to allow an undo of the half-edge collapse).*
6. *Apply linear interpolation of the vector field inside the new triangles $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3), (\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4), \dots, (\mathbf{p}_1, \mathbf{p}_{n-1}, \mathbf{p}_n)$). Check whether there are critical points inside one of the new triangles. If so, prohibit half-edge collapse and stop.*
7. *Construct a new cyclic ordered list L_2 of points on the polygon $((\mathbf{p}_1, \mathbf{p}_2), \dots, (\mathbf{p}_{n-1}, \mathbf{p}_n), (\mathbf{p}_n, \mathbf{p}_1))$ consisting of the following points:*
 - a) *all boundary switch points from step 4 of the algorithm*
 - b) *the entry points to \mathbf{D}' of all separatrices*
 - c) *the exit points, from \mathbf{D}' . These are compute by integrating the stream lines starting from all points of step 7b of this algorithm inside \mathbf{D}' until they reach the boundary again.*
8. *Compare the cyclic order of the points in L_1 and L_2 . If the corresponding points do not have the same cyclic order in L_1 and L_2 , prohibit the half-edge collapse and stop.*
9. *Allow the half-edge collapse and stop.*

² This is necessary to fulfil the theorem in [16] on which this algorithm is based upon. In fact, the algorithm is based on the assumption that a local modification of the vector field does not change the entry points of the separatrices into the area to be modified. This does not hold any more for re-entering stream lines.

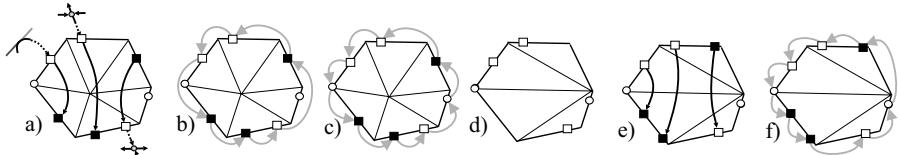


Fig. 3. Example of algorithm 2; (a) three separatrices passing through \mathbf{D}' , and 2 boundary switch points (white circles) are present; the empty boxes are the entry points of the separatrices into \mathbf{D}' (in integration direction), the solid boxes are the exit points; (b) cyclic list L_1 (grey arrows) after step 2; (c) L_1 after step 4; (d) collecting points of new list L_2 after half-edge collapse: after step 7a and 7b; (e) integrate new stream lines (step 7c); (f) cyclic list L_2 after step 7c; half-edge collapse is allowed, since the corresponding points in L_1 and L_2 (shown in (c) and (f)) are in the same order.

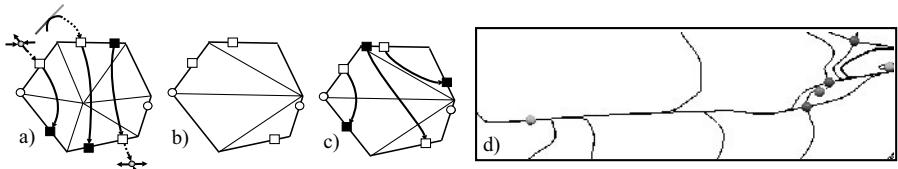


Fig. 4. (a) Another example of algorithm 2; (a) three separatrices passing through \mathbf{D}' , and 2 boundary switch points (white circles) are present: L_1 consists of the marked points on the boundary; (b) points of L_2 after step 7b; (c) points of L_2 after step 7c; edge collapse is not allowed, since the corresponding points in L_1 and L_2 (shown in (a) and (c)) are in a different order; (d) example of separatrices which tend to be very close to each other.

Fig. 3 illustrates an example of this algorithm where an edge collapse is allowed. Figs. 4a – 4c show an example where the algorithm prohibits a half-edge collapse.

Now we want to modify algorithm 22 to handle equivalence concept 3. To do so, we have to compare the critical points in \mathbf{D}' before and after the half-edge collapse if some of the critical points collapsed. We get the following

Algorithm 3 (*check whether a half-edge collapse $\mathbf{p}_0 \rightarrow \mathbf{p}_1$ changes the topology in the sense of concept 3*):

1. Extract and store the critical points inside $\mathbf{D}' = (\mathbf{t}_1, \dots, \mathbf{t}_n)$. If there is more than one saddle, or if there is more than one source/sink, then prohibit the half-edge collapse and stop.
2. as in algorithm 2.
3. as in algorithm 2.
4. as in algorithm 2.
5. as in algorithm 2.
6. Apply linear interpolation of the vector field inside the new triangles $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3), (\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4), \dots, (\mathbf{p}_1, \mathbf{p}_{n-1}, \mathbf{p}_n)$). Check the new triangles for

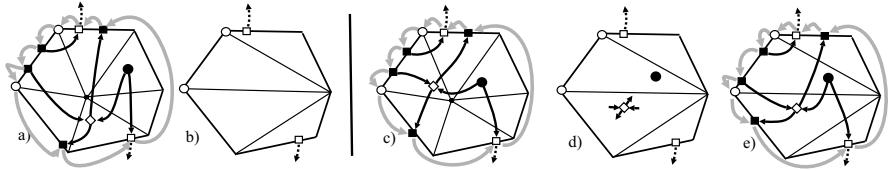


Fig. 5. (a) 1-ring containing one saddle (white diamond) and one source (black circle); 3 of the 4 separatrices created by the saddle leave the 1-ring while one ends in the source; in addition, two separatrices enter the region from outside (hollow boxes): one ends in the source, the other leaves the region; (b) simulated half-edge collapse removes the critical points: half-edge collapse is not allowed; (c) another example of a 1-ring containing one saddle (white diamond) and one source (black circle); (d) simulated half-edge collapse gives two new critical points: one saddle and one sink; (e) cyclic list L_2 after step 7 of algorithm 3; the half-edge collapse is allowed.

critical points. If the number of saddles or the number of sources/sinks does not coincide with the numbers found in step 1, prohibit the half-edge collapse and stop. If there is one saddle, integrate its 4 separatrices until they leave \mathbf{D}' . Store the 4 exit points into a new cyclic list L_2 of points on the polygon $((\mathbf{p}_1, \mathbf{p}_2), \dots, (\mathbf{p}_{n-1}, \mathbf{p}_n), (\mathbf{p}_n, \mathbf{p}_1))$.

7. Insert the following points into L_2 :

- a) as in algorithm 2.
- b) as in algorithm 2.
- c) as in algorithm 2.

8. as in algorithm 2.

9. as in algorithm 2.

Figs. 5a and 5b illustrate an example of algorithm 3 where the half-edge collapse is not allowed. Figs. 5c – 5e show an example with an allowed half-edge collapse.

An analysis of the algorithm in [16], especially on the skin friction data set (described in the next section), had shown that this data set tends to have many separatrices very close to each other (see Fig. 4d for an example). This can be explained with the presence of attachment and separation lines [12]. For these cases the algorithm in [16] may forbid a half-edge collapse due to numerical instabilities. To solve this, we collected the entry points of separatrices at the 1-ring of a vertex to clusters: entry points of separatrices which are very close to each other³ are set to the same entry point and thus have the same exit point as well.

³ We used 1/1000 of the length of the boundary edge.

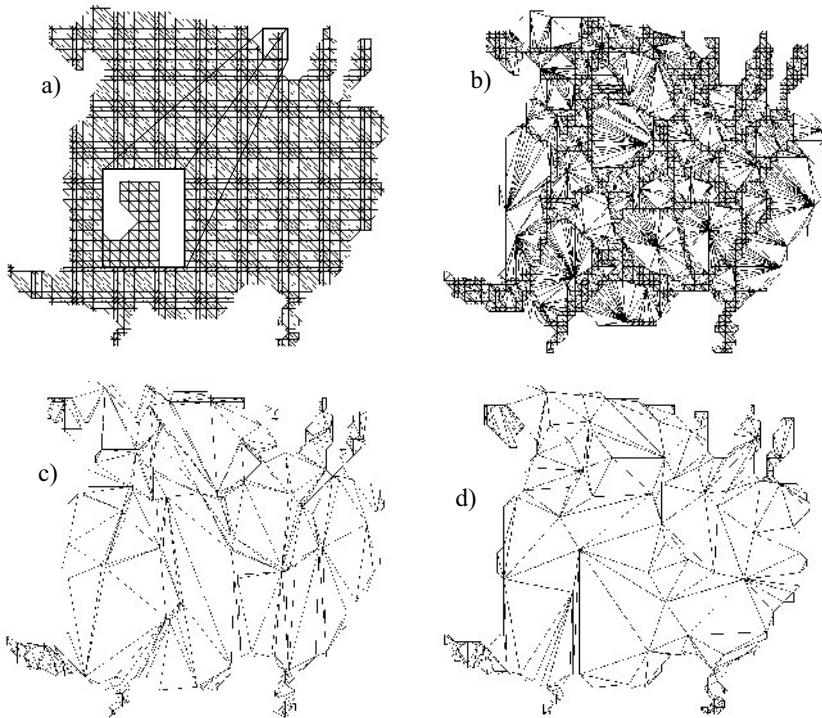


Fig. 6. Test data set 1 (flow in a bay area near Greifswald); (a) piecewise triangular domain of the original data set; (b) domain of the compressed data set (algorithm 1); (c) domain of the compressed data set (algorithm 2); (d) domain of the compressed data set (algorithm 3).

5 Results

We applied our thinning algorithms to two test data sets. The first data set describes (the perpendicular of) the flow of a bay area of the Baltic Sea near Greifswald in Germany. The data set was created by the Department of Mathematics, University of Rostock. The data is given as an incomplete flow data set on a regular 115×103 grid. Triangulating the defined cells, we have a piecewise linear vector field consisting of 14,086 triangles (see Fig. 6a).

Fig. 7a shows the topological skeleton of the vector field. This flow data set consists of 71 critical points, 44 boundary switch points, and 168 separatrices. Fig. 6b shows the resulting triangular grid after applying algorithm 1. This grid consists of 4,944 triangles. We can clearly see that areas containing separatrices or critical points are left untouched by the algorithm. Fig. 7b shows the topological skeleton after algorithm 1 (which by definition has to be identical to Fig. 7a).

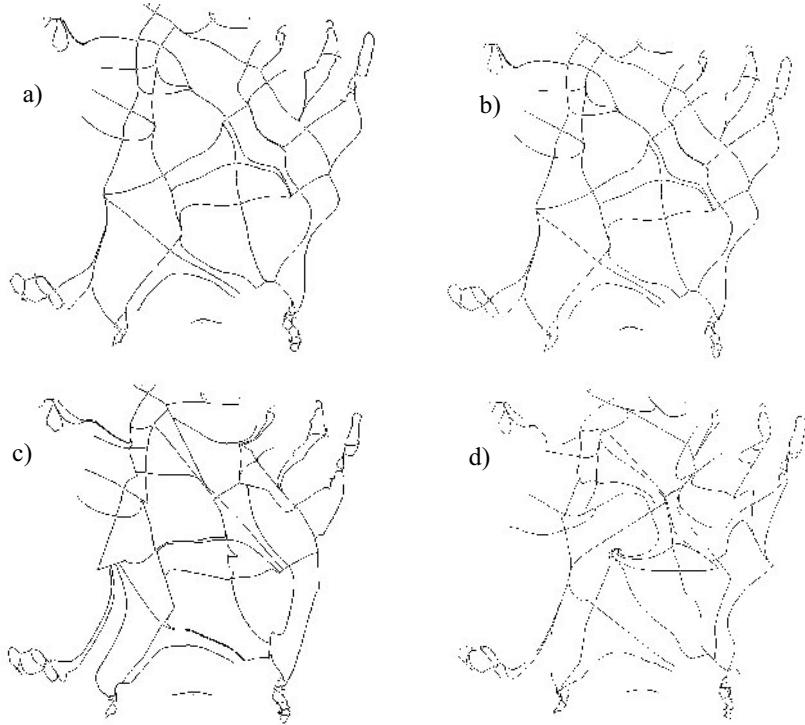


Fig. 7. Test data set 1; (a) topological skeleton of original data set; (b) topological skeleton of compressed data set (algorithm 1); (c) topological skeleton of compressed data set (algorithm 2); (d) topological skeleton of compressed data set (algorithm 3).

Applying algorithm 2, we obtained a new piecewise linear vector field which consists of 660 triangles. Fig. 6c shows the piecewise triangular domain of the compressed vector field. Fig. 7c shows the topological skeleton of the compressed vector field. The compression ratio is 95.3%. The complete compression algorithm took 280 seconds on an Intel Xeon 1.7 GHz processor. Figs. 6d and 7d show the same results for algorithm 3. Here, the number of triangles was reduced to 374. Fig. 6d shows the resulting triangular grid while Fig. 7d shows the new topological skeleton.

The second test data set describes the skin friction on a face of a cylinder which was obtained by a numerical simulation of a flow around a square cylinder. The data set was generated by Verstappen and Veldman of the University of Groningen. This data set is also analysed in [3],[13] and [16]. The data is given on a rectangular 102×64 grid with varying grid size. To get a piecewise linear vector field, we divided each grid cell into two triangles which gives a piecewise triangular domain consisting of 12,726 triangles. Fig. 8a shows the

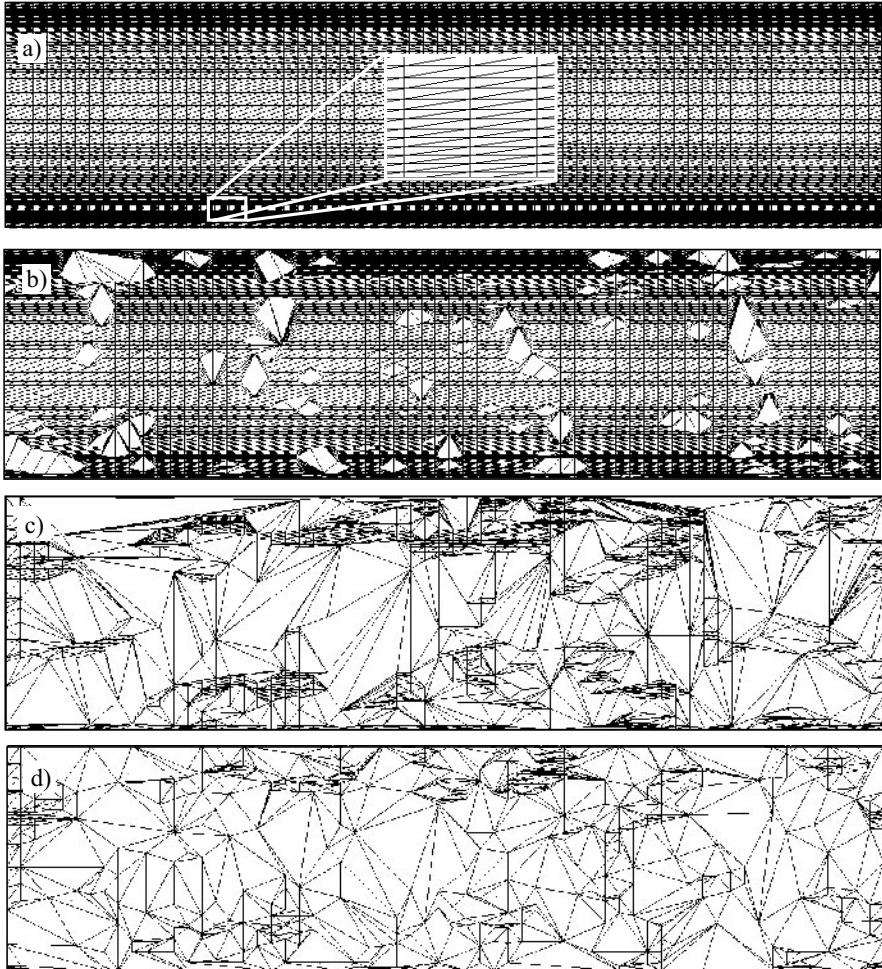


Fig. 8. Test data set 2 (skin friction); (a) piecewise triangular domain of the original data set; (b) domain of the compressed data set (algorithm 1); (c) domain of the compressed data set (algorithm 2); (d) domain of the compressed data set (algorithm 3).

piecewise triangular domain of the vector field. As we can see in this picture, all triangles there tend to be long and thin. Fig. 9a shows the topological skeleton of the vector field. This vector field consists of 338 critical points, 34 boundary switch points, and 714 separatrices. Therefore, it can be considered as a vector field of complex topology. Fig. 9b shows the resulting grid after applying algorithm 1. This grid consists of 10,680 triangles. As we can see, almost no thinning took place because the separatrices of this vector field are rather dense. Fig. 9b shows the topological skeleton after algorithm 1 (identical to

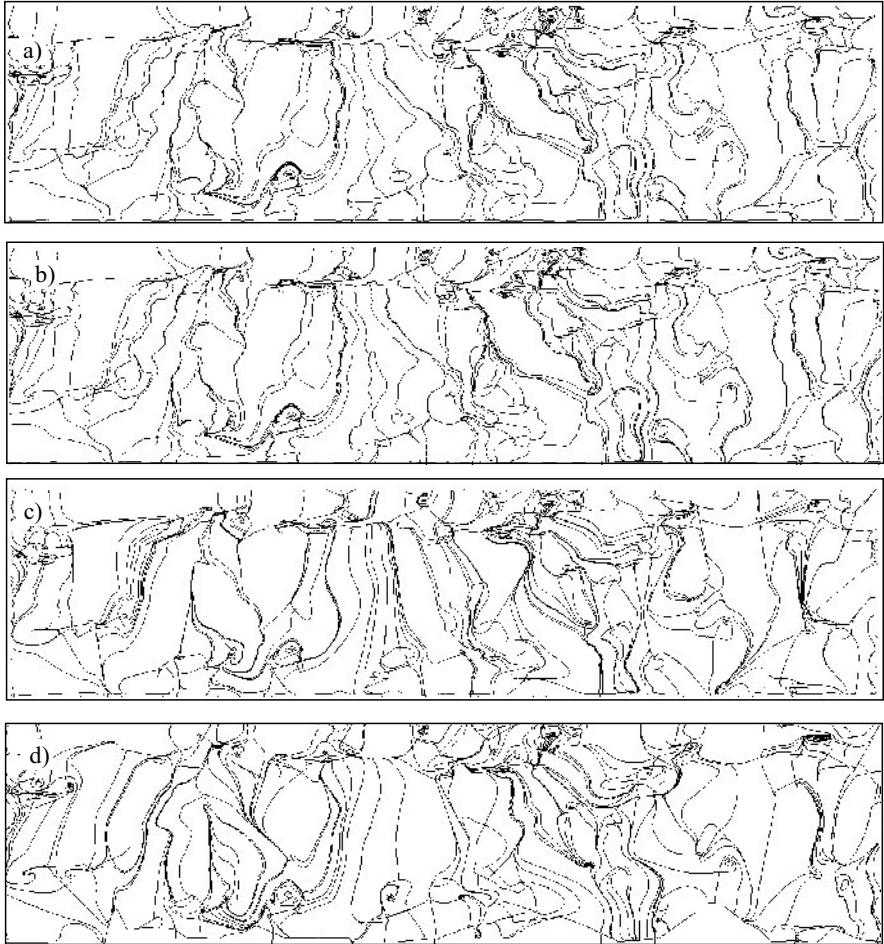


Fig. 9. Test data set 2 (skin friction); (a) topological skeleton of original data set; (b) topological skeleton of compressed data set (algorithm 1); (c) topological skeleton of compressed data set (algorithm 2); (d) topological skeleton of compressed data set (algorithm 3).

Fig. 9a). By applying our compression algorithm 2, we obtained a vector field with the piecewise triangular domain shown in Fig. 8c. This domain consists of 2,153 triangles which gives a compression ratio of 83.1%. Fig. 9c shows the topological skeleton. The complete compression algorithm took 299 seconds on an Intel Xeon 1.7 GHz processor. Fig. 8d shows the underlying grid resulting from algorithm 3 consisting of 1071 triangles. The topological skeleton of this vector field is shown in Fig. 9d.

6 Conclusions

We have introduced and compared a number of new topology preserving thinning algorithms for piecewise linear vector fields which are based on one of the topology based equivalence concepts of Sect. 3. We applied the algorithms to two test data sets of moderate and complex topology respectively.

Equivalence concept 1 (and its thinning algorithm) gives very poor compression ratios for topologically complex data. This is due to the fact that the appearance of a critical point or separatrix in a triangle prevents it from being collapsed.

Equivalence concept 2 gives significant compression ratios even for topologically complex data sets. Applying this algorithm guarantees that the topological skeleton of the original and the thinned vector field coincide in the critical points and the connectivity of the separatrices.

In comparison to equivalence concept 2, concept 3 gives a further reduction of the number of triangles in the thinned mesh by a factor of approximately 50%. This was achieved for both test data sets. The visualisation of the topological skeleton shows that the critical points change their locations, but the change tends to be limited to the neighbourhood of the original critical point.

Acknowledgements

The authors thank Wim de Leeuw for providing the second test data set. This work was supported in part by the European Union research project “Multiresolution in Geometric Modelling (MINGLE)” under grant HPRN-CT-1999-00117.

References

1. C. Bajaj and D. Schikore. Topology-preserving data simplification with error bounds. *Comput. & Graphics*, 22(1):3–12, 1998.
2. C. L. Bajaj, V. Pascucci, and D. R. Schikore. Visualization of scalar topology for structural enhancement. In *Proc. IEEE Visualization '98*, pages 51–58, 1998.
3. W. de Leeuw and R. van Liere. Collapsing flow topology using area metrics. In *Proc. IEEE Visualization '99*, 1999.
4. 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.
5. T. K. Dey, H. Edelsbrunner, S. Guha, and D. V. Nekhayev. Topology preserving edge contraction. *Publ. Inst. Math. (Beograd)*, 66(1999):23–45, 1999.
6. H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse complexes for piecewise linear 2-manifolds. In *Proc. 17th Sympos. Comput. Geom. 2001*, 2001.
7. P. A. Firby and C. F. Gardiner. *Surface Topology*, chapter 7, pages 115–135. Ellis Horwood Ltd., 1982. Vector Fields on Surfaces.

8. H. Garcke, T. Preusser, M. Rumpf, A. Telea, U. Weikardt, and J. van Wijk. A continuous clustering method for vector fields. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proc. IEEE Visualization 2000*, pages 351–358, 2000.
9. A. Globus and C. Levit. A tool for visualizing of three-dimensional vector fields. In *Proc. IEEE Visualization '91*, pages 33–40. IEEE Computer Society Press, 1991.
10. 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.
11. J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *IEEE Computer*, 22(8):27–36, August 1989.
12. D. N. Kenwright, C. Henze, and C. Levit. Feature extraction of separation and attachment lines. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):135–144, 1999.
13. 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.
14. 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.
15. H. Theisel. Designing 2D vector fields of arbitrary topology. *Computer Graphics Forum (Eurographics 2002)*, 21(3):595–604, 2002.
16. H. Theisel, C. Rössl, and H.-P. Seidel. Compression of 2D vector fields under guaranteed topology preservation. *Computer Graphics Forum (Eurographics 2003)*, 22(3):333–342, 2003.
17. X. Tricoche, G. Scheuermann, and H. Hagen. A topology simplification method for 2D vector fields. In *Proc. IEEE Visualization 2000*, pages 359–366, 2000.
18. X. Tricoche, G. Scheuermann, and H. Hagen. Continuous topology simplification of planar vector fields. In *Proc. Visualization 01*, pages 159 – 166, 2001.
19. I. Trotts, D. Kenwright, and R. Haimes. Critical points at infinity: a missing link in vector field topology. In *Proc. NSF/DoE Lake Tahoe Workshop on Hierarchical Approximation and Geometrical Methods for Scientific Visualization*, 2000.
20. R. Westermann, C. Johnson, and T. Ertl. Topology-preserving smoothing of vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):222–229, 2001.
21. 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.

Part VII

— Wavelets

Periodic and Spline Multiresolution Analysis and the Lifting Scheme

Jürgen Prestin¹ and Ewald Quak²

¹ Institute of Mathematics, University of Lübeck, Germany

prestin@math.uni-luebeck.de

² Department of Applied Mathematics, SINTEF ICT, Oslo, Norway

Ewald.Quak@sintef.no

Summary. The lifting scheme is a well-known general framework for the construction of wavelets, especially in finite-dimensional settings. After a short introduction about the basics of lifting, we discuss how wavelet constructions, in two specific finite settings, can be related to the lifting approach. These examples concern, on the one hand, polynomial splines and, on the other, the Fourier approach for translation-invariant spaces of periodic functions.

1 Introduction

The goal of this contribution is to address how two types of multiresolution analysis that have been subject to research activities within the MINGLE project are related to a well-known general framework for the construction of wavelets, namely the lifting scheme, as introduced by Sweldens [19], see also Sweldens & Schröder [18], Daubechies & Sweldens [7] and the references therein. In the original paper [19] the approach is formulated for the whole real axis, but it can also be used to generate various sorts of “second-generation wavelets” in finite settings or to describe decomposition and reconstruction algorithms without ever explicitly working with basis functions. In our paper, we describe a general finite setting with basis functions and related matrices that nicely fits the specific types of multiresolution we are most interested in, namely polynomial splines and periodic functions, yet we do not aim to address the topic in the full generality given by the concept of stable completions as described by Carnicer, Dahmen & Pena [1]. Thus we also do not address in any detail the construction of biorthogonal wavelets on the interval as addressed for example in [10], [6], [16] and [4]. The lifting framework we want to relate to is the topic of Sect. 2, spline examples are presented in Sect. 3, and the Fourier approach for periodic functions is covered in Sect. 4.

2 Basics of the Lifting Scheme

2.1 Decomposition of a Finite-dimensional Space

We will be dealing with real-valued square-integrable univariate functions defined over a compact interval and we use for simplicity always the standard inner product $\langle \cdot, \cdot \rangle$. Our setting is strictly finite-dimensional, meaning that a linear space of functions U of dimension $n + m$ is represented as the direct sum of two spaces, namely V^0 of dimension n and W^0 of dimension m , i.e.

$$U = V^0 + W^0. \quad (1)$$

Let a basis of U be given by functions $\vartheta_0, \dots, \vartheta_{n+m-1}$, a basis of V^0 by functions $\varphi_0^0, \dots, \varphi_{n-1}^0$ and a basis of W^0 by functions $\psi_0^0, \dots, \psi_{m-1}^0$. For ease of notation we use column vectors of functions as

$$\boldsymbol{\vartheta} = (\vartheta_0, \dots, \vartheta_{n+m-1})^T, \boldsymbol{\varphi}^0 = (\varphi_0^0, \dots, \varphi_{n-1}^0)^T \text{ and } \boldsymbol{\psi}^0 = (\psi_0^0, \dots, \psi_{m-1}^0)^T.$$

Due to (1), there exist matrices \mathbf{P}^0 of dimension $(n + m) \times n$ and \mathbf{Q}^0 of dimension $(n + m) \times m$, such that

$$\begin{pmatrix} \boldsymbol{\varphi}^0 \\ \boldsymbol{\psi}^0 \end{pmatrix}^T = \boldsymbol{\vartheta}^T (\mathbf{P}^0, \mathbf{Q}^0) = \boldsymbol{\vartheta}^T \mathbf{M}^0. \quad (2)$$

As the functions are chosen as bases of their respective spaces, the *two-scale* matrix \mathbf{M}^0 is nonsingular and we write $(\mathbf{M}^0)^{-1} = (\tilde{\mathbf{P}}^0, \tilde{\mathbf{Q}}^0)^T$, again with matrices $\tilde{\mathbf{P}}^0$ of dimension $(n + m) \times n$ and $\tilde{\mathbf{Q}}^0$ of dimension $(n + m) \times m$, yielding

$$\boldsymbol{\vartheta}^T = \begin{pmatrix} \boldsymbol{\varphi}^0 \\ \boldsymbol{\psi}^0 \end{pmatrix}^T (\tilde{\mathbf{P}}^0, \tilde{\mathbf{Q}}^0)^T.$$

Proposition 1. *The fact that the matrices \mathbf{M}^0 and $(\mathbf{M}^0)^{-1}$ are inverses is reflected in the matrix relations*

$$\begin{pmatrix} \tilde{\mathbf{P}}^{0^T} \mathbf{P}^0 & \tilde{\mathbf{P}}^{0^T} \mathbf{Q}^0 \\ \tilde{\mathbf{Q}}^{0^T} \mathbf{P}^0 & \tilde{\mathbf{Q}}^{0^T} \mathbf{Q}^0 \end{pmatrix} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0}_{n \times m} \\ \mathbf{0}_{m \times n} & \mathbf{I}_m \end{pmatrix} \quad (3)$$

and

$$\mathbf{P}^0 \tilde{\mathbf{P}}^{0^T} + \mathbf{Q}^0 \tilde{\mathbf{Q}}^{0^T} = \mathbf{I}_{n+m}. \quad (4)$$

An element $h \in U$ can be written as $h = f + g$, where $f \in V^0$ and $g \in W^0$, and in the respective basis representations as

$$\boldsymbol{\vartheta}^T \mathbf{h} = \boldsymbol{\varphi}^{0^T} \mathbf{f}^0 + \boldsymbol{\psi}^{0^T} \mathbf{g}^0$$

with coefficient vectors \mathbf{f}^0 , \mathbf{g}^0 and \mathbf{h} of length n , m and $n + m$, respectively. We can derive from (3) and (4)

Proposition 2. *Decomposition of the coefficient vector \mathbf{h} into the coefficient vectors of its components is achieved by*

$$\mathbf{f}^0 = \tilde{\mathbf{P}}^{0^T} \mathbf{h} \text{ and } \mathbf{g}^0 = \tilde{\mathbf{Q}}^{0^T} \mathbf{h},$$

while reconstruction of the coefficient vector \mathbf{h} from the coefficient vectors of its components is carried out by

$$\mathbf{h} = \mathbf{P}^0 \mathbf{f}^0 + \mathbf{Q}^0 \mathbf{g}^0.$$

In the *lazy wavelet* approach, the spaces V^0 and W^0 are generated by just dividing the basis functions into two disjoint groups, typically by separating the even-index ones from the odd-index ones. For the coefficients this means that decomposition is *even* and *odd subsampling*, and the two-scale matrices \mathbf{M}^0 and $(\mathbf{M}^0)^{-1}$ are simply permutation matrices.

A different strategy is a *hierarchical basis* approach. If U and V^0 are given together with basis functions that are somehow linked to a coarse knot sequence τ for V^0 and a refinement \mathbf{t} of τ as the knot sequence for U , then it is natural to define a complement W^0 as the span of those basis functions in the large space U that can be associated with the newly added knots in $\mathbf{t} \setminus \tau$. With the matrix P^0 fixed by the choice of basis functions in U and V^0 , the columns of the two-scale matrix \mathbf{Q}^0 are then some unit vectors in \mathbb{R}^{n+m} , as determined by the positioning of the new knots with respect to the old ones.

Given once more U and V^0 together with their basis functions, the *semi-orthogonal* approach uses the uniquely determined relative orthogonal complement W^\perp of V^0 with respect to U . The term *semi* indicates that there is orthogonality between spaces, but each basis need not be an orthogonal basis of the corresponding space.

Taking the point of view of *filtering* discrete data, one is more interested in the properties of the matrices used for decomposition and reconstruction according to Propositions 1 and 2 than in the actual underlying spaces of functions. A general investigation of the matrix relations (3) and (4) concerning banded matrices can be found in [5].

2.2 Change of Basis

We now want to replace the direct sum in (1) by another one, where we still keep the dimensions of the subspaces, such that

$$U = V + W \tag{5}$$

with new bases

$$\boldsymbol{\varphi} = (\varphi_0, \dots, \varphi_{n-1})^T \text{ and } \boldsymbol{\psi} = (\psi_0, \dots, \psi_{m-1})^T$$

for V and W , respectively. Note that we leave the basis of U unchanged. Let a nonsingular change-of-basis or *transfer* matrix be given by

$$\begin{pmatrix} \varphi \\ \psi \end{pmatrix}^T = \begin{pmatrix} \varphi^0 \\ \psi^0 \end{pmatrix}^T \begin{pmatrix} \mathbf{A}_n & \mathbf{B}_{n \times m} \\ \mathbf{C}_{m \times n} & \mathbf{D}_m \end{pmatrix}. \quad (6)$$

Lemma 1. *The two-scale matrix in*

$$\begin{pmatrix} \varphi \\ \psi \end{pmatrix}^T = \boldsymbol{\vartheta}^T (\mathbf{P}, \mathbf{Q}) = \boldsymbol{\vartheta}^T \mathbf{M}$$

and the one in (2) are connected by the same change-of-basis matrix as for the basis functions in (6), in the sense that

$$(\mathbf{P}, \mathbf{Q}) = (\mathbf{P}^0, \mathbf{Q}^0) \begin{pmatrix} \mathbf{A}_n & \mathbf{B}_{n \times m} \\ \mathbf{C}_{m \times n} & \mathbf{D}_m \end{pmatrix}.$$

Condition (3) implies

$$(\tilde{\mathbf{P}}, \tilde{\mathbf{Q}})^T = \begin{pmatrix} \mathbf{A}_n & \mathbf{B}_{n \times m} \\ \mathbf{C}_{m \times n} & \mathbf{D}_m \end{pmatrix}^{-1} (\tilde{\mathbf{P}}^0, \tilde{\mathbf{Q}}^0)^T.$$

Special cases for transfer matrices are of course

$$\begin{pmatrix} \mathbf{A}_n & \mathbf{B}_{n \times m} \\ \mathbf{0}_{m \times n} & \mathbf{D}_m \end{pmatrix} \text{ and } \begin{pmatrix} \mathbf{A}_n & \mathbf{0}_{n \times m} \\ \mathbf{C}_{m \times n} & \mathbf{D}_m \end{pmatrix}. \quad (7)$$

The first one leaves the space V^0 intact, and just changes its basis, while the basis ψ^0 is changed so that the complement space W^0 changes to (in general a different) W . Analogously, the second one leaves the complement space W^0 intact with a change of basis, and replaces φ^0 and thus the space V^0 by V . Choosing for the first matrix in addition \mathbf{A}_n as the identity \mathbf{I}_n , leaves also the basis of the space V^0 unchanged, and analogously the choice of $\mathbf{D}_m = \mathbf{I}_m$ preserves the basis of the complement space W^0 in the second case.

For what is called *lifting*, the transfer matrix is even further specialised, as both blocks on the diagonals in (7) are chosen as identities.

Definition 1. *A basis ψ for a new complement space W in the decomposition (5) is generated from the old basis ψ^0 of the old complement space W^0 in (1) by a lifting step if the change-of-basis matrix has the form*

$$\begin{pmatrix} \mathbf{I}_n & \mathbf{S}_{n \times m} \\ \mathbf{0}_{m \times n} & \mathbf{I}_m \end{pmatrix}.$$

Analogously a basis φ for the new space V in the decomposition (5) is generated from the old basis φ^0 of the old space V^0 in (1) by a dual lifting step if the change-of-basis matrix has the form

$$\begin{pmatrix} \mathbf{I}_n & \mathbf{0}_{n \times m} \\ \mathbf{S}_{m \times n} & \mathbf{I}_m \end{pmatrix}.$$

The intention of lifting is to change one of the subspaces in (1) by replacing its given and typically “simpler” basis by a new, in some sense “better” basis in a straightforward manner, namely by a suitable choice of the free parameters which are represented by the coefficients in the matrix \mathbf{S} . A significant advantage of a basis created by a lifting step is that the inverse of its transfer matrix is very straightforward, namely

$$\begin{pmatrix} \mathbf{I}_n & \mathbf{S}_{n \times m} \\ \mathbf{0}_{m \times n} & \mathbf{I}_m \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{I}_n & -\mathbf{S}_{n \times m} \\ \mathbf{0}_{m \times n} & \mathbf{I}_m \end{pmatrix}.$$

Using Lemma 1 this implies

Lemma 2. *A lifting step changes the decomposition (1) to $U = V^0 + W$ by performing the change of basis*

$$\begin{pmatrix} \varphi \\ \psi \end{pmatrix} = \begin{pmatrix} \varphi^0 \\ \psi^0 + \mathbf{S}_{n \times m}^T \varphi^0 \end{pmatrix},$$

and the change in two-scale matrices given by

$$\begin{aligned} (\mathbf{P}, \mathbf{Q}) &= (\mathbf{P}^0, \mathbf{Q}^0 + \mathbf{P}^0 \mathbf{S}_{n \times m}), \\ (\tilde{\mathbf{P}}, \tilde{\mathbf{Q}})^T &= (\tilde{\mathbf{P}}^0 - \tilde{\mathbf{Q}}^0 \mathbf{S}_{n \times m}^T, \tilde{\mathbf{Q}}^0)^T. \end{aligned}$$

A dual lifting step changes the decomposition (1) to $U = V + W^0$ through the change of basis

$$\begin{pmatrix} \varphi \\ \psi \end{pmatrix} = \begin{pmatrix} \varphi^0 + \mathbf{S}_{m \times n}^T \psi^0 \\ \psi^0 \end{pmatrix},$$

and the change of matrices

$$\begin{aligned} (\mathbf{P}, \mathbf{Q}) &= (\mathbf{P}^0 + \mathbf{Q}^0 \mathbf{S}_{m \times n}, \mathbf{Q}^0), \\ (\tilde{\mathbf{P}}, \tilde{\mathbf{Q}})^T &= (\tilde{\mathbf{P}}^0, \tilde{\mathbf{Q}}^0 - \tilde{\mathbf{P}}^0 \mathbf{S}_{m \times n}^T)^T. \end{aligned}$$

In terms of Proposition 2 this means that if the original decomposition matrices $\tilde{\mathbf{P}}^0$ and $\tilde{\mathbf{Q}}^0$ are sparse and the matrix $\mathbf{S}_{n \times m}$ used to satisfy additional properties is sparse as well, the lifting scheme produces a better basis that still allows efficient computations.

2.3 The Biorthogonal Setting

So far we have only considered one decomposition (1). Since there are two pairs of matrices (\mathbf{P}, \mathbf{Q}) and $(\tilde{\mathbf{P}}, \tilde{\mathbf{Q}})$ associated with this decomposition, it is natural to consider a second related decomposition as well. So we will now be looking at two sets of decompositions that are *dual* to each other. Let

$$\begin{aligned} U &= V^0 + W^0, \\ \tilde{U} &= \tilde{V}^0 + \tilde{W}^0, \end{aligned}$$

with bases for U, V^0, W^0 ,

$$\boldsymbol{\vartheta} = (\vartheta_0, \dots, \vartheta_{n+m-1})^T, \boldsymbol{\varphi}^0 = (\varphi_0^0, \dots, \varphi_{n-1}^0)^T \text{ and } \boldsymbol{\psi}^0 = (\psi_0^0, \dots, \psi_{m-1}^0)^T,$$

and for $\tilde{U}, \tilde{V}^0, \tilde{W}^0$,

$$\tilde{\boldsymbol{\vartheta}} = (\tilde{\vartheta}_0, \dots, \tilde{\vartheta}_{n+m-1})^T, \tilde{\boldsymbol{\varphi}}^0 = (\tilde{\varphi}_0^0, \dots, \tilde{\varphi}_{n-1}^0)^T \text{ and } \tilde{\boldsymbol{\psi}}^0 = (\tilde{\psi}_0^0, \dots, \tilde{\psi}_{m-1}^0)^T,$$

with the corresponding matrices $\mathbf{P}^0, \mathbf{Q}^0, \tilde{\mathbf{P}}^0$ and $\tilde{\mathbf{Q}}^0$ given by

$$\begin{pmatrix} \boldsymbol{\varphi}^0 \\ \boldsymbol{\psi}^0 \end{pmatrix}^T = \boldsymbol{\vartheta}^T (\mathbf{P}^0, \mathbf{Q}^0) \text{ and } \begin{pmatrix} \tilde{\boldsymbol{\varphi}}^0 \\ \tilde{\boldsymbol{\psi}}^0 \end{pmatrix}^T = \tilde{\boldsymbol{\vartheta}}^T (\tilde{\mathbf{P}}^0, \tilde{\mathbf{Q}}^0). \quad (8)$$

Note that the decomposition $\tilde{U} = \tilde{V}^0 + \tilde{W}^0$ typically involves completely different types of functions from those of the decomposition $U = V^0 + W^0$. Only the dimensions of the corresponding spaces are supposed to be the same. If in this setting the bases in the large spaces U and \tilde{U} are related by biorthogonality, and the two-scale matrices are inverse transposes, then the bases of the component spaces are dual, as formulated in the following result.

Lemma 3. *Let the bases of the large spaces U and \tilde{U} be biorthogonal, i.e.*

$$\langle \boldsymbol{\vartheta}, \tilde{\boldsymbol{\vartheta}} \rangle = \left(\langle \vartheta_i, \tilde{\vartheta}_{\tilde{i}} \rangle \right)_{i,\tilde{i}=1}^{n+m} = \mathbf{I}_{n+m}, \quad (9)$$

and let the two-scale matrices be related by

$$(\mathbf{P}^0 \mathbf{Q}^0) (\tilde{\mathbf{P}}^0 \tilde{\mathbf{Q}}^0)^T = \mathbf{I}_{n+m}. \quad (10)$$

Then the bases for the subspaces are biorthogonal or dual in the sense that

$$\left\langle \begin{pmatrix} \boldsymbol{\varphi}^0 \\ \boldsymbol{\psi}^0 \end{pmatrix}, \begin{pmatrix} \tilde{\boldsymbol{\varphi}}^0 \\ \tilde{\boldsymbol{\psi}}^0 \end{pmatrix} \right\rangle = \begin{pmatrix} \langle \boldsymbol{\varphi}^0, \tilde{\boldsymbol{\varphi}}^0 \rangle & \langle \boldsymbol{\varphi}^0, \tilde{\boldsymbol{\psi}}^0 \rangle \\ \langle \boldsymbol{\psi}^0, \tilde{\boldsymbol{\varphi}}^0 \rangle & \langle \boldsymbol{\psi}^0, \tilde{\boldsymbol{\psi}}^0 \rangle \end{pmatrix} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0}_{n \times m} \\ \mathbf{0}_{m \times n} & \mathbf{I}_m \end{pmatrix}. \quad (11)$$

In this biorthogonal setting the investigation of a change of bases in the first decomposition must address how the change of bases in the dual spaces must be performed to keep the duality of (11) intact.

Lemma 4. *For a nonsingular change-of-basis matrix with*

$$\begin{pmatrix} \boldsymbol{\varphi} \\ \boldsymbol{\psi} \end{pmatrix}^T = \begin{pmatrix} \boldsymbol{\varphi}^0 \\ \boldsymbol{\psi}^0 \end{pmatrix}^T \begin{pmatrix} \mathbf{A}_n & \mathbf{B}_{n \times m} \\ \mathbf{C}_{m \times n} & \mathbf{D}_m \end{pmatrix},$$

the change-of-basis matrix for the dual decomposition must be the inverse transpose to keep the duality intact, i.e. the equation

$$\begin{pmatrix} \tilde{\varphi} \\ \tilde{\psi} \end{pmatrix}^T = \begin{pmatrix} \tilde{\varphi}^0 \\ \tilde{\psi}^0 \end{pmatrix}^T \begin{pmatrix} \mathbf{A}_n & \mathbf{B}_{n \times m} \\ \mathbf{C}_{m \times n} & \mathbf{D}_m \end{pmatrix}^{-T}$$

results in new, but still dual, decompositions

$$\begin{aligned} U &= V + W, \\ \tilde{U} &= \tilde{V} + \tilde{W}, \end{aligned}$$

where the new bases also satisfy

$$\begin{pmatrix} \langle \varphi, \tilde{\varphi} \rangle & \langle \varphi, \tilde{\psi} \rangle \\ \langle \psi, \tilde{\varphi} \rangle & \langle \psi, \tilde{\psi} \rangle \end{pmatrix} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0}_{n \times m} \\ \mathbf{0}_{m \times n} & \mathbf{I}_m \end{pmatrix}.$$

In the dual setting the simplicity of a lifting step as a change of basis is especially significant. Given that for a lifting step the change-of-basis matrix is

$$\begin{pmatrix} \mathbf{I}_n & \mathbf{S}_{n \times m} \\ \mathbf{0}_{m \times n} & \mathbf{I}_m \end{pmatrix},$$

its inverse transpose is

$$\begin{pmatrix} \mathbf{I}_n & \mathbf{0}_{n \times m} \\ -(S_{n \times m})^T & \mathbf{I}_m \end{pmatrix},$$

meaning that a dual lifting step is performed on the dual decomposition, explaining why the term dual is used in Definition 1.

Corollary 1. A lifting step in $U = V^0 + W^0$ with $\mathbf{S}_{n \times m}$ and a dual lifting step with $-(\mathbf{S}_{n \times m})^T$ in $\tilde{U} = \tilde{V}^0 + \tilde{W}^0$ result in the dual decompositions

$$\begin{aligned} U &= V^0 + W, \\ \tilde{U} &= \tilde{V} + \tilde{W}^0, \end{aligned}$$

with a new pair of dual bases

$$\begin{pmatrix} \varphi \\ \psi \end{pmatrix} = \begin{pmatrix} \varphi^0 \\ \psi^0 + \mathbf{S}_{n \times m}^T \varphi^0 \end{pmatrix} \text{ and } \begin{pmatrix} \tilde{\varphi} \\ \tilde{\psi} \end{pmatrix} = \begin{pmatrix} \tilde{\varphi}^0 - \mathbf{S}_{n \times m} \tilde{\psi}^0 \\ \tilde{\psi}^0 \end{pmatrix},$$

and matrices

$$\begin{aligned} (\mathbf{P}, \mathbf{Q}) &= (\mathbf{P}^0, \mathbf{Q}^0 + \mathbf{P}^0 \mathbf{S}_{n \times m}), \\ (\tilde{\mathbf{P}}, \tilde{\mathbf{Q}})^T &= (\tilde{\mathbf{P}}^0 - \tilde{\mathbf{Q}}^0 \mathbf{S}_{n \times m}^T, \tilde{\mathbf{Q}}^0)^T. \end{aligned}$$

Corollary 2. A dual lifting step in $U = V^0 + W^0$ with $\mathbf{S}_{m \times n}$ and a lifting step with $-(\mathbf{S}_{m \times n})^T$ in $\tilde{U} = \tilde{V}^0 + \tilde{W}^0$ generate

$$\begin{aligned} U &= V + W^0, \\ \tilde{U} &= \tilde{V}^0 + \tilde{W}, \end{aligned}$$

with a pair of new dual bases

$$\begin{pmatrix} \varphi \\ \psi \end{pmatrix} = \begin{pmatrix} \varphi^0 + \mathbf{S}_{m \times n}^T \psi^0 \\ \psi^0 \end{pmatrix} \text{ and } \begin{pmatrix} \tilde{\varphi} \\ \tilde{\psi} \end{pmatrix} = \begin{pmatrix} \tilde{\varphi}^0 \\ \tilde{\psi}^0 - \mathbf{S}_{m \times n} \tilde{\varphi}^0 \end{pmatrix}, \quad (12)$$

and matrices

$$\begin{aligned} (\mathbf{P}, \mathbf{Q}) &= (\mathbf{P}^0 + \mathbf{Q}^0 \mathbf{S}_{m \times n}, \mathbf{Q}^0), \\ (\tilde{\mathbf{P}}, \tilde{\mathbf{Q}})^T &= (\tilde{\mathbf{P}}^0, \tilde{\mathbf{Q}}^0 - \tilde{\mathbf{P}}^0 \mathbf{S}_{m \times n}^T)^T. \end{aligned}$$

2.4 Beyond the Original Spaces

So far we have not left the spaces U and \tilde{U} , only changed subspaces and their bases. A major point in the real axis setting is that given a biorthogonal setting with finite filters, all other possible pairs with finite filters that are biorthogonal to the original pair can be characterised completely (see Lemma 5 of [19]). Such a characterisation is of course impossible in a finite setting. According to the lemma below we can, however, say the following. Let U^* be another, possibly completely different, large space of functions U^* of the correct dimension $n+m$, such that it possesses a basis which is biorthogonal to the basis of U . Now we can actually generate a decomposition $V^* + W^*$ of U^* by using the two-scale matrices of \tilde{U} to define the basis functions for the subspaces V^* and W^* . These new basis functions are then still dual to the ones given by the decomposition of U .

Lemma 5. Given decompositions $U = V + W$ and $\tilde{U} = \tilde{V} + \tilde{W}$, with biorthogonal bases $\boldsymbol{\vartheta}$ and $\tilde{\boldsymbol{\vartheta}}$ of U and \tilde{U} , respectively, let the two-scale matrices be related by

$$(\mathbf{P} \ \mathbf{Q}) (\tilde{\mathbf{P}} \ \tilde{\mathbf{Q}})^T = \mathbf{I}_{n+m}, \quad (13)$$

so that by Lemma 3,

$$\begin{pmatrix} \langle \varphi, \tilde{\varphi} \rangle & \langle \varphi, \tilde{\psi} \rangle \\ \langle \psi, \tilde{\varphi} \rangle & \langle \psi, \tilde{\psi} \rangle \end{pmatrix} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0}_{n \times m} \\ \mathbf{0}_{m \times n} & \mathbf{I}_m \end{pmatrix}.$$

Assume that a space U^* is spanned by $\boldsymbol{\vartheta}^* = (\vartheta_0^*, \dots, \vartheta_{n+m-1}^*)^T$, where the basis $\boldsymbol{\vartheta}^*$ is still biorthogonal to $\boldsymbol{\vartheta}$, i.e. $\langle \boldsymbol{\vartheta}, \boldsymbol{\vartheta}^* \rangle = \mathbf{I}_{n+m}$, and introduce a decomposition $U^* = V^* + W^*$ by defining

$$\begin{pmatrix} \varphi^* \\ \psi^* \end{pmatrix}^T := \vartheta^{*^T} (\tilde{\mathbf{P}}, \tilde{\mathbf{Q}}),$$

with V^* and W^* spanned by φ^* and ψ^* , respectively. Then the systems φ^* and ψ^* are also biorthogonal to φ and ψ in the sense that

$$\begin{pmatrix} \langle \varphi, \varphi^* \rangle & \langle \varphi, \psi^* \rangle \\ \langle \psi, \varphi^* \rangle & \langle \psi, \psi^* \rangle \end{pmatrix} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0}_{n \times m} \\ \mathbf{0}_{m \times n} & \mathbf{I}_m \end{pmatrix}.$$

The two-scale matrices for $U^* = V^* + W^*$ remain the same as in (13).

Note that in the classical real axis setting, the basis functions are not just linked within each space by translation, but also the ones in U and V are strongly linked, namely by dilation. In this whole section we have never used such relationships. In the following section, however, we give a short example explaining how one can go beyond the initial spaces in the real axis setting by using the fact that the functions in U and V , and \tilde{U} and \tilde{V} , respectively, are linked by dilation.

3 Polynomial Splines

3.1 Piecewise Constants

We will start by looking at piecewise constant functions, not so much because they are typically used as an example, but because they allow us in fact to investigate a finite setting without too many boundary effects, so that we can go back to an infinite setting of functions on the whole real axis – as we would like to do at the end of this subsection – without too many technical difficulties.

Example 1. Let the interval $[0, 1]$ be partitioned by the knots $\tau_j = j/4, j = 0, \dots, 4$. This partition is then uniformly refined to $t_i = i/8, i = 0, \dots, 8$. The space U is the space of piecewise constant functions on the intervals $[t_i, t_{i+1})$ and the space V^0 is the space of piecewise constant functions on the intervals $[\tau_j, \tau_{j+1})$. Denoting by $\chi_{[a,b]}$ the characteristic function of the interval $[a, b]$, the elementary basis functions of U are $\vartheta_i = \chi_{[t_i, t_{i+1})}$, $i = 0, \dots, 7$, and for $V = V^0$ they are $\varphi_j = \chi_{[\tau_j, \tau_{j+1})}$, $j = 0, \dots, 3$. This fixes the matrix \mathbf{P}^0 as given in (14).

A hierarchical complement space W^0 is described by choosing its basis functions to complement the φ_j 's with those elements from U that correspond exactly to the newly added knots, namely $\psi_k^0 = \vartheta_{2k+1}$, $k = 0, \dots, 3$, so that

$$(\mathbf{P}^0)^T = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } (\mathbf{Q}^0)^T = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad (14)$$

inducing

$$\left(\tilde{\mathbf{P}}^0\right)^T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \text{ and } \left(\tilde{\mathbf{Q}}^0\right)^T = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}. \quad (15)$$

According to Definition 1, a general lifting step matrix in this setting is

$$\begin{pmatrix} \mathbf{I}_4 & \mathbf{S}_{4 \times 4} \\ \mathbf{0}_{4 \times 4} & \mathbf{I}_4 \end{pmatrix},$$

where the entries $s_{k,j}$ of $\mathbf{S}_{4 \times 4}$ can still be chosen freely. The old “wavelets” ψ_k^0 are characteristic functions and thus possess no vanishing moments. We now demand that the new basis functions ψ_k at least have one vanishing moment (their integrals be zero). This is reflected by the four conditions that $\langle \psi_k, 1 \rangle = 0$ for all k . With a total of 16 coefficients in $\mathbf{S}_{4 \times 4}$, that leaves us with a good deal of free parameters. As we want $\mathbf{S}_{4 \times 4}$ to be as sparse as possible, it turns out that a diagonal matrix is sufficient, namely

$$\mathbf{S}_{4 \times 4} = \begin{pmatrix} -\frac{1}{2} & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & -\frac{1}{2} \end{pmatrix}.$$

Lemma 2 gives us the new matrices as

$$\mathbf{Q}^T = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \text{ and } \tilde{\mathbf{P}}^T = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \quad (16)$$

while \mathbf{P}^0 remains as in (14) and $\tilde{\mathbf{Q}}^0$ as in (15).

Consequently in this very special case a lifting step has not just created a new basis (and corresponding complement space) with basis functions that now all possess one vanishing moment, the new functions are in fact the Haar wavelets and the new complement space W is in fact the orthogonal complement W^\perp , as $\mathbf{S}_{4 \times 4}$ is actually the unique solution to the 16 equations $\langle \psi_k, \varphi_j^0 \rangle = 0$ for all k, j . That of course is really a special case only valid for piecewise constants, as we will see in the next subsection.

Example 2. In the setting of Example 1, the functions ϑ_i form an orthogonal, but not orthonormal system. It is straightforward to choose a biorthogonal system as $\tilde{\vartheta}_i = 8\vartheta_i$ for all i , so that $\tilde{U} = U$. As matrices we choose \mathbf{P}^0 from (14), $\tilde{\mathbf{Q}}^0$ from (15) and as $\tilde{\mathbf{P}}^0$ and \mathbf{Q}^0 the matrices from (16), as they satisfy the condition (10). Then (8) allows it to compute φ_j^0 as used in Example 1, and their duals are $\tilde{\varphi}_j^0 = 4\varphi_j^0$, with $\psi_k^0 = \frac{1}{2}\vartheta_{2k+1} - \frac{1}{2}\vartheta_{2k}$ and $\tilde{\psi}_k^0 = \tilde{\vartheta}_{2k+1} - \tilde{\vartheta}_{2k}$.

In this setting we want to perform a dual lifting step with a matrix $\mathbf{S}_{4 \times 4}$. Let us say that we want to create new dual wavelets $\tilde{\psi}_k$ that have three vanishing moments, as the $\tilde{\psi}_k^0$'s have only one according to their construction above. For $k = 1, 2$ there are three free parameters, which we can fix by the conditions for three vanishing moments, i.e. for $e_i(x) = x^i, i = 0, 1, 2$, we demand that $\langle \tilde{\psi}_k, e_i \rangle = 0$ for $k = 1, 2$. A direct computation establishes

$$\mathbf{S}_{4 \times 4} = \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & 0 \\ -\frac{1}{32} & 0 & \frac{1}{32} & 0 \\ 0 & -\frac{1}{32} & 0 & \frac{1}{32} \\ 0 & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix}. \quad (17)$$

Note here the boundary effect. If we want the boundary wavelets $\tilde{\psi}_k, k = 0, 3$, to have three vanishing moments as well, we have to use all three parameters, which results in a matrix that is not tridiagonal in the corresponding two corners. If we want a tridiagonal matrix, the boundary functions can only have two vanishing moments at most.

As indicated at the end of the previous section, let us now have a quick look, indicating how the lifting scheme on the real axis actually works to define new spaces and new types of functions. To this end choose the matrix in (17) such that we get two diagonals with constant entries, i.e.

$$\mathbf{S}_{4 \times 4}^b = \begin{pmatrix} -\frac{1}{32} & \frac{1}{32} & 0 & 0 \\ -\frac{1}{32} & 0 & \frac{1}{32} & 0 \\ 0 & -\frac{1}{32} & 0 & \frac{1}{32} \\ 0 & 0 & -\frac{1}{32} & \frac{1}{32} \end{pmatrix}.$$

That way the boundary dual wavelets only have one vanishing moment, but that is not so important. The main point, however, is the expression we can derive from (12) for the new inner scaling functions, which is in this case completely free of boundary effects. We obtain for $i = 1$ (for $i = 2$ this works just as well),

$$\varphi_1 = \varphi_1^0 - \frac{1}{32}\psi_0^0 + \frac{1}{32}\psi_2^0.$$

In terms of the fine basis of U that means

$$\varphi_1 = -\frac{1}{32} \left(\frac{1}{2}\vartheta_1 - \frac{1}{2}\vartheta_2 \right) + \vartheta_3 + \vartheta_4 + \frac{1}{32} \left(\frac{1}{2}\vartheta_5 - \frac{1}{2}\vartheta_6 \right). \quad (18)$$

Now recall that in the real axis setting the fine functions are linked to the coarse ones through dilation. Thus (18) can be re-interpreted as a refinement equation in the sense that

$$\begin{aligned} \varphi(x) = & -\frac{1}{32} \left(\frac{1}{2}\varphi(2x+2) - \frac{1}{2}\varphi(2x+1) \right) + \varphi(2x) + \varphi(2x-1) \\ & + \frac{1}{32} \left(\frac{1}{2}\varphi(2x-2) - \frac{1}{2}\varphi(2x-3) \right), \end{aligned}$$

which defines a completely new scaling function which is, of course, no longer piecewise constant. The corresponding new wavelet is given by

$$\psi(x) = -\frac{1}{2}\varphi(2x) + \frac{1}{2}\varphi(2x-1).$$

According to the construction, the functions are biorthogonal to the basis pair corresponding on the real axis to $\tilde{\varphi} = \tilde{\varphi}^0$ and $\tilde{\psi} = \tilde{\psi}^0 - S_{4 \times 4} \tilde{\varphi}^0$, which are still piecewise constant. As $\tilde{\psi}$ was constructed to have three vanishing moments, the new function φ reproduces polynomials of order three.

The functions φ and ψ are of course well-known. The lifting scheme produced here the biorthogonal pairs originally constructed by Cohen, Daubechies and Feauveau [3]. In fact a general result of Daubechies and Sweldens [7] shows how on the real axis any biorthogonal pairs with finite filters (finitely many two-scale coefficients) can be factored into lifting steps.

3.2 Piecewise Linear Splines

Given the space constraints of this paper, we outline quickly, just using piecewise linear splines, how higher order splines behave differently to the simple piecewise constants of the previous subsection.

Example 3. We investigate the same knot sequences as in Example 2. This time, however, we consider all piecewise linear functions with respect to the knots t_i as forming the space U , and all piecewise linear functions with respect to the knots τ_j as forming the subspace V . As basis functions of these spaces we consider the “hat” functions (piecewise linear B-splines), that are 1 in one knot and 0 in all others. We denote by $\Lambda_{t_i}, i = 0, \dots, 8$, the fine hats for the knots t_i , and by $\lambda_{\tau_j}, j = 0, \dots, 4$, the coarse hats for the knots τ_j . Thus $\vartheta_i = \Lambda_{t_i}$ and $\varphi_j = \lambda_{\tau_j}$. Again a hierarchical choice for a basis of a complement space W^0 is fairly obvious: the fine hat functions corresponding to the newly inserted knots. So we set $\psi_k^0 = \vartheta_{2k+1} = \Lambda_{t_{2k+1}}, k = 0, \dots, 3$. This is also called the Faber decomposition, see the tutorial [15]. The matrices \mathbf{P}^0 and \mathbf{Q}^0 are also given explicitly there.

In this case the lifting matrix assumes the form

$$\begin{pmatrix} \mathbf{I}_5 & \mathbf{S}_{5 \times 4} \\ \mathbf{0}_{4 \times 5} & \mathbf{I}_4 \end{pmatrix},$$

where the entries $s_{k,j}$ of $\mathbf{S}_{5 \times 4}$ can still be chosen freely.

The basis elements of W^0 have no vanishing moments at all. So again we can use the free parameters that are provided through $S_{5 \times 4}$ to generate a new basis ψ_k^1 for a new complement W^1 , so that the new functions have vanishing moments. For piecewise linear functions we thus try $\langle \psi_k^1, e_0 \rangle = \langle \psi_k^1, e_1 \rangle = 0$ for all k . On the other hand we would like to have only a few non-zero coefficients. It turns out that it is possible to satisfy the conditions

by only involving the two coarse hats neighbouring the fine hat ψ_k^0 . In fact the resulting inner functions ψ_k^1 are symmetric, namely

$$\mathbf{S}_{5 \times 4}^T = \begin{pmatrix} -\frac{3}{4} & -\frac{1}{8} & 0 & 0 & 0 \\ 0 & -\frac{1}{4} & -\frac{1}{4} & 0 & 0 \\ 0 & 0 & -\frac{1}{4} & -\frac{1}{4} & 0 \\ 0 & 0 & 0 & -\frac{1}{8} & -\frac{3}{4} \end{pmatrix}. \quad (19)$$

Also for linear (and higher order) splines it is then possible to pursue lifting in a biorthogonal setting, producing Cohen-Daubechies-Feauveau pairs on the real axis, but we will not pursue that any further here.

Instead we would like to see how a new basis ψ_k^\perp spanning the orthogonal complement W^\perp of V relative to U can be found by performing a lifting of the Faber basis. Recall that in Example 2, this was already achieved by simply satisfying the condition of one vanishing moment. The analogous matrix (19), however, is not sufficient. The condition of orthogonality of V and W^\perp leads to 20 conditions $\langle \psi_k^\perp, \varphi_j^0 \rangle = 0$, which determine uniquely that

$$(\mathbf{S}_{5 \times 4}^\perp)^T = \frac{1}{224} \begin{pmatrix} -142 & -52 & 14 & -4 & 2 \\ 38 & -76 & -70 & 20 & -10 \\ -10 & 20 & -70 & -76 & 38 \\ 2 & -4 & 14 & -52 & -142 \end{pmatrix}.$$

Consequently, the matrix $\mathbf{Q} = \mathbf{Q}^0 + \mathbf{P}^0 \mathbf{S}_{5 \times 4}^\perp$ is dense (see again [15]). We conclude that the basis ψ^\perp of the orthogonal complement space generated by lifting the Faber basis is the one obtained by solving the normal equations for the fine hats $A_{t_{2k+1}}$ (similar results hold for higher order splines).

Thus the following question arises: what is then the change-of-basis matrix that produces the minimally supported B-wavelet basis ([15], Lyche, Mørken, Quak [9]) for W^\perp from the hierarchical basis of the Faber decomposition? It is

$$\begin{pmatrix} \mathbf{I}_5 & \mathbf{B}_{5 \times 4} \\ \mathbf{0}_{4 \times 5} & \mathbf{D}_4 \end{pmatrix},$$

where

$$\mathbf{B}_{5 \times 4}^T = \begin{pmatrix} -12 & -6 & 0 & 0 & 0 \\ 0 & -6 & -6 & 0 & 0 \\ 0 & 0 & -6 & -6 & 0 \\ 0 & 0 & 0 & -6 & -12 \end{pmatrix} \text{ and } \mathbf{D}_4 = \begin{pmatrix} 20 & 4 & 0 & 0 \\ 4 & 16 & 4 & 0 \\ 0 & 4 & 16 & 4 \\ 0 & 0 & 4 & 20 \end{pmatrix},$$

which is not the matrix of a lifting step, since D_4 is not the identity. Again similar results using banded matrices, but involving a larger number of bands, hold also for higher order spline B-wavelets. Note finally that the B-wavelets resulting from this change of basis have the same support as the functions obtained through lifting with (19).

As a concluding remark let us emphasise that in this short treatment of some spline examples, we have obviously downplayed the effect of boundaries

and nonuniform knot sequences. A look at [9] clearly shows that the construction of semi-orthogonal B-wavelets for nonuniform knots involves significant technical effort. The boundary effects when restricting to an interval the biorthogonal pairs of Cohen, Daubechies and Feauveau, where one component consists of splines over uniform dyadic knots, require also a really substantial treatment, as given by Dahmen, Kunoth and Urban in [4], based on the stable completions of [1].

4 Periodic Functions

As a further example we study translation-invariant spaces of 2π -periodic functions. We are again interested in the change of basis algorithms as described in Sect. 2. Contrary to Sect. 3, where the transformation matrices in space domain were discussed, it is advantageous in this case to go to the Fourier domain.

Basic results for this approach are studied by Chui & Mhaskar [2], Koh, Lee & Tan [8], Plonka & Tasche [12],[13], Narcowich & Ward [11] and Selig [17]. Concerning notations and basic results we follow mainly [17]. A matrix formulation of algorithms can also be found in [14].

To begin with let the Fourier matrix \mathbf{F} of dimension $N \times N$ be defined as

$$\mathbf{F} = \mathbf{F}_N = \frac{1}{\sqrt{N}} \left(e^{-\frac{2\pi i k l}{N}} \right)_{k,l=0}^{N-1}.$$

The discrete Fourier transform of length N maps a vector $\mathbf{a} = (a_k)_{k=0}^{N-1} \in \mathbb{C}^N$ to a vector $\hat{\mathbf{a}} = (\hat{a}_l)_{l=0}^{N-1}$ by

$$\hat{a}_l = \sum_{k=0}^{N-1} a_k e^{-\frac{2\pi i k l}{N}}, \quad l = 0, \dots, N-1,$$

which can be rewritten as $\hat{\mathbf{a}} = \sqrt{N} \mathbf{F} \mathbf{a}$. From the unitarity and symmetry of the Fourier matrix $\mathbf{F}^{-1} = \bar{\mathbf{F}}^T = \bar{\mathbf{F}}$ one concludes

$$\mathbf{a} = \frac{1}{\sqrt{N}} \bar{\mathbf{F}} \hat{\mathbf{a}}.$$

Strongly related to the Fourier matrix \mathbf{F} is the decomposition of an arbitrary circulant matrix

$$\text{circ } \mathbf{a} = \left(a_{(s-r) \bmod N} \right)_{r,s=0}^{N-1}$$

given by a vector $\mathbf{a} \in \mathbb{C}^N$, and we have

$$\text{circ } \mathbf{a} = \mathbf{F} \text{ diag } \hat{\mathbf{a}} \bar{\mathbf{F}},$$

where $\hat{a}_l, l = 0, \dots, N-1$, turn out to be the eigenvalues of the circulant matrix $\text{circ } \mathbf{a}$. To introduce the particular function space we have in mind we

recall the definition of $L^2_{2\pi}$, the Hilbert space of 2π -periodic functions with inner product

$$\langle \varphi, \psi \rangle = \frac{1}{2\pi} \int_0^{2\pi} \varphi(t) \overline{\psi(t)} dt$$

and finite norm $\|\varphi\|^2 = \langle \varphi, \varphi \rangle$. Every function $\varphi \in L^2_{2\pi}$ can be written as its Fourier series

$$\varphi(t) = \sum_{k \in \mathbb{Z}} c_k(\varphi) e^{ikt}$$

with Fourier coefficients

$$c_k(\varphi) = \langle \varphi, e^{ik \cdot} \rangle = \frac{1}{2\pi} \int_0^{2\pi} \varphi(t) e^{-ikt} dt.$$

Here we will exploit extensively the relationship to the Hilbert space $\ell^2(\mathbb{Z})$ of all sequences $\mathbf{a} = (a_k)_{k \in \mathbb{Z}}$ with

$$\|\mathbf{a}\|_{\ell^2} = \left(\sum_{k \in \mathbb{Z}} |a_k|^2 \right)^{1/2}$$

and

$$\langle \mathbf{a}, \mathbf{b} \rangle_{\ell^2} = \sum_{k \in \mathbb{Z}} a_k \overline{b_k}.$$

The isomorphism between $L^2_{2\pi}$ and $\ell^2(\mathbb{Z})$ is represented by Parseval's equation

$$\|\varphi\|^2 = \sum_{k \in \mathbb{Z}} |c_k(\varphi)|^2 \quad \text{for all } \varphi \in L^2_{2\pi}$$

and more generally

$$\langle \varphi, \psi \rangle = \sum_{k \in \mathbb{Z}} c_k(\varphi) \overline{c_k(\psi)} \quad \text{for all } \varphi, \psi \in L^2_{2\pi}.$$

Since we are interested in finite-dimensional translation-invariant subspaces we define for arbitrary sets of functions $\{\varphi_k : k = 0, \dots, N-1\}$, $\varphi_k \in L^2_{2\pi}$, the Gram matrix

$$\mathbf{G} = \mathbf{G}(\varphi) = (\langle \varphi_k, \varphi_l \rangle)_{k,l=0}^{N-1}.$$

Here and in the following we use the notation $\boldsymbol{\varphi}$ for the vector of functions

$$\boldsymbol{\varphi} = (\varphi_k)_{k=0}^{N-1}.$$

We recall for $\mathbf{a} = (a_k)_{k=0}^{N-1} \in \mathbb{C}^N$ the simple fact

$$\bar{\mathbf{a}}^T \mathbf{G} \mathbf{a} = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \bar{a}_k \langle \varphi_k, \varphi_l \rangle a_l = \left\| \sum_{k=0}^{N-1} \bar{a}_k \varphi_k \right\|^2 \geq 0.$$

Hence, it follows that \mathbf{G} is regular if and only if the functions φ_k are linearly independent. Moreover, if $\mathbf{G} = \mathbf{I} = \mathbf{I}_N$ is the identity matrix, then the functions φ_k are orthonormal. If \mathbf{G} is regular, then

$$V = \text{span } \{\varphi_k : k = 0, \dots, N-1\} \subset L_{2\pi}^2$$

is a space of dimension N with basis $\{\varphi_k : k = 0, \dots, N-1\}$. It is now a simple observation that for the basis $\{\varphi_k\}$ there exists a unique biorthogonal basis $\{\tilde{\varphi}_k : k = 0, \dots, N-1\}$ for the space V , i.e.,

$$(\langle \varphi_k, \tilde{\varphi}_l \rangle)_{k,l=0}^{N-1} = \mathbf{I} \quad (20)$$

is the identity matrix, and furthermore,

$$\tilde{\varphi} = \mathbf{G}^{-1} \varphi.$$

More generally, every vector of functions $\tilde{\varphi} = (\tilde{\varphi}_l)_{l=0}^{N-1}$ satisfying (20), but not necessarily in V will be called a biorthogonal vector for $\varphi = (\varphi_k)_{k=0}^{N-1}$.

From now on we restrict ourselves to finite-dimensional translation-invariant spaces

$$V = \text{span} \left\{ \varphi \left(\cdot - \frac{2\pi k}{N} \right) : k = 0, \dots, N-1 \right\} \subset L_{2\pi}^2$$

generated by a function $\varphi \in L_{2\pi}^2$.

To understand the structure of bases of these spaces we take into account so-called orthogonal splines (cf. [8])

$$\sum_{p \in \mathbb{Z}} c_{k+Np}(\varphi) e^{i(k+Np)\cdot}, \quad k = 0, \dots, N-1.$$

These orthogonal splines generated from $\varphi \in L_{2\pi}^2$ are related to the translation-invariant space V by the following result.

Lemma 6. *For $\varphi \in L_{2\pi}^2$,*

$$\begin{aligned} & \text{span} \left\{ \varphi \left(\cdot - \frac{2\pi k}{N} \right) : k = 0, \dots, N-1 \right\} \\ &= \text{span} \left\{ \sum_{p \in \mathbb{Z}} c_{k+Np}(\varphi) e^{i(k+Np)\cdot} : k = 0, \dots, N-1 \right\}. \end{aligned}$$

Proof. To show this equality we simply remark that

$$\varphi = \sqrt{N} \mathbf{F} \left(\sum_{p \in \mathbb{Z}} c_{k+Np}(\varphi) e^{i(k+Np)\cdot} \right)_{k=0}^{N-1}, \quad (21)$$

which can be verified by direct calculation. \square

As an immediate consequence of this equality we can compute the dimension of the space V from the sequence of Fourier coefficients $(c_k(\varphi))$ of its generating function φ .

Corollary 3. *Let $\varphi \in L^2_{2\pi}$. The dimension d of*

$$V = \text{span} \left\{ \varphi \left(\cdot - \frac{2\pi k}{N} \right) : k = 0, \dots, N-1 \right\}$$

is given by the number of integers k , $0 \leq k \leq N-1$, such that there exists some $p \in \mathbb{Z}$ with $c_{k+Np}(\varphi) \neq 0$.

In other words

$$d = \sharp \left\{ k : \sum_{p \in \mathbb{Z}} |c_{k+Np}(\varphi)|^2 > 0, \quad k = 0, \dots, N-1 \right\}.$$

In particular, the space V has maximal dimension N if and only if for all k the orthogonal splines are not the zero function.

Proof. By Parseval's equation the non-zero orthogonal splines are linearly independent. Then the assertion follows immediately from (21). \square

In the next lemma we discuss under which conditions a function $\psi \in L^2_{2\pi}$ is an element of the subspace V .

Lemma 7. *The function $\psi \in L^2_{2\pi}$ is an element of $V = \text{span} \left\{ \varphi \left(\cdot - \frac{2k\pi}{N} \right) : k = 0, \dots, N-1 \right\}$ if and only if there exists a vector $\mathbf{a} = (a_s)_{s=0}^{N-1}$ such that*

$$c_{k+Np}(\psi) = \hat{a}_k c_{k+Np}(\varphi) \quad \text{for all } k = 0, \dots, N-1 \quad \text{and } p \in \mathbb{Z}. \quad (22)$$

Then

$$\psi = \sum_{s=0}^{N-1} a_s \varphi \left(\cdot - \frac{2s\pi}{N} \right).$$

Proof. If (22) is satisfied, then $\psi \in L^2_{2\pi}$ can be written as

$$\begin{aligned} \psi(t) &= \sum_{k=0}^{N-1} \hat{a}_k \sum_{p \in \mathbb{Z}} c_{k+Np}(\varphi) e^{i(k+Np)t} \\ &= \sum_{k=0}^{N-1} \sum_{s=0}^{N-1} a_s e^{-\frac{2\pi i k s}{N}} \sum_{p \in \mathbb{Z}} c_{k+Np}(\varphi) e^{i(k+Np)t} \\ &= \sum_{s=0}^{N-1} a_s \varphi \left(t - \frac{2\pi s}{N} \right). \end{aligned}$$

The reverse direction follows in the same way. \square

From now on we consider different translation-invariant spaces. The first result describes the direct sum property.

Lemma 8. *Let $\varphi, \psi \in L^2_{2\pi}$ be given and let*

$$V = \text{span} \left\{ \varphi \left(\cdot - \frac{2k\pi}{N} \right), \quad k = 0, \dots, N-1 \right\}, \quad (23)$$

$$W = \text{span} \left\{ \psi \left(\cdot - \frac{2k\pi}{N} \right), \quad k = 0, \dots, N-1 \right\}. \quad (24)$$

Then the following two propositions are equivalent:

I. *The sum $V + W$ is a direct one, i.e.*

$$\dim V + \dim W = \dim(V + W).$$

II. *If there exist $(\hat{a}_k)_{k=0}^{N-1}, (\hat{b}_k)_{k=0}^{N-1} \in \mathbb{C}^N$ such that*

$$\hat{a}_k c_{k+Np}(\varphi) = \hat{b}_k c_{k+Np}(\psi) \quad \text{for all } k = 0, \dots, N-1, p \in \mathbb{Z}, \quad (25)$$

then $(\hat{a}_k) = (\hat{b}_k) = \mathbf{0}$.

Proof. We follow the same lines as in the proof of Lemma 7. Writing

$$\sum_{s=0}^{N-1} a_s \varphi \left(\cdot - \frac{2s\pi}{N} \right) - b_s \psi \left(\cdot - \frac{2s\pi}{N} \right) = 0$$

in terms of Fourier coefficients gives equivalently

$$\sum_{s=0}^{N-1} a_s e^{-\frac{2\pi i \ell s}{N}} c_\ell(\varphi) - b_s e^{-\frac{2\pi i \ell s}{N}} c_\ell(\psi) = 0 \quad \text{for all } \ell \in \mathbb{Z},$$

which can be easily reformulated as (25). \square

The next step is to discuss orthogonality and biorthogonality.

Lemma 9. *Let $\varphi, \psi \in L^2_{2\pi}$ be given and let V and W be as in Lemma 8. Then a) $V \perp W$ if and only if*

$$\sum_{p \in \mathbb{Z}} c_{q+Np}(\varphi) \overline{c_{q+Np}(\psi)} = 0 \quad \text{for all } q = 0, \dots, N-1.$$

b) *The vector of translates $\psi = (\psi(\cdot - \frac{2k\pi}{N}))_{k=0}^{N-1}$ is a biorthogonal vector for $\varphi = (\varphi(\cdot - \frac{2k\pi}{N}))_{k=0}^{N-1}$ if and only if*

$$\sum_{p \in \mathbb{Z}} c_{q+Np}(\varphi) \overline{c_{q+Np}(\psi)} = \frac{1}{N} \quad \text{for all } q = 0, \dots, N-1.$$

Proof. Computing for all $0 \leq k, l \leq N - 1$ the inner product between the translates of φ and ψ we obtain

$$\begin{aligned} \left\langle \frac{0}{\delta_{k,l}} \right\rangle &= \left\langle \varphi \left(\cdot - \frac{2k\pi}{N} \right), \psi \left(\cdot - \frac{2l\pi}{N} \right) \right\rangle \\ &= \sum_{s \in \mathbb{Z}} c_s(\varphi) \overline{c_s(\psi)} e^{\frac{2\pi i s(k-l)}{N}} \\ &= \sum_{q=0}^{N-1} \sum_{p \in \mathbb{Z}} c_{q+Np}(\varphi) \overline{c_{q+Np}(\psi)} e^{\frac{2\pi i q(k-l)}{N}}. \end{aligned}$$

Applying the inverse discrete Fourier transform of length N gives the desired results. \square

Now we are interested in the reverse direction. Let a function $\vartheta \in L^2_{2\pi}$ and a space U be given as

$$U = \text{span} \left\{ \vartheta \left(\cdot - \frac{k\pi}{N} \right) : k = 0, \dots, 2N - 1 \right\}.$$

Furthermore, we assume $\dim U = 2N$ which means that none of the $2N$ orthogonal splines generated by ϑ is the zero function. Now we want to decompose U into the direct sum of translation-invariant spaces V and W of the form (23–24), i.e.

$$U = V + W \quad \text{with} \quad \dim V = \dim W = N.$$

Hence,

$$\varphi = \sum_{s=0}^{2N-1} \alpha_s \vartheta \left(\cdot - \frac{s\pi}{N} \right), \quad \psi = \sum_{s=0}^{2N-1} \beta_s \vartheta \left(\cdot - \frac{s\pi}{N} \right). \quad (26)$$

Under these assumptions one verifies that

$$\hat{\alpha}_k \hat{\beta}_{k+N} \neq \hat{\alpha}_{k+N} \hat{\beta}_k \quad \text{for all } k = 0, \dots, N - 1,$$

where $(\hat{\alpha}_k)_{k=0}^{2N-1}, (\hat{\beta}_k)_{k=0}^{2N-1}$ are the discrete Fourier transforms of length $2N$ of the vectors $(\alpha_s)_{s=0}^{2N-1}$ and $(\beta_s)_{s=0}^{2N-1}$, respectively. Note that one can prove the existence of such functions φ, ψ starting from an arbitrary $\vartheta \in L^2_{2\pi}$ with $\dim U = 2N$. Note on the other hand that the converse of this statement is not true in general (cf. [17], where an example of two orthogonal translation-invariant spaces V, W is given for which $U = V + W$ is not a translation-invariant space). Furthermore, in [17] it is studied how one can describe the basis transformation algorithms in case of $V \perp W$. Here we generalise to the case of a direct sum $V + W$.

Theorem 1. *Let the spaces V and W be defined as in Lemma 8. Moreover, let $\dim V + W = 2N$. Then, the basis transformation between*

$$\vartheta = \left(\vartheta \left(\cdot - \frac{s\pi}{N} \right) \right)_{s=0}^{2N-1} \quad \text{and} \quad \begin{pmatrix} \varphi \\ \psi \end{pmatrix} = \begin{pmatrix} (\varphi \left(\cdot - \frac{2s\pi}{N} \right))_{s=0}^{N-1} \\ (\psi \left(\cdot - \frac{2s\pi}{N} \right))_{s=0}^{N-1} \end{pmatrix}$$

can be described easily in the Fourier domain as

$$\begin{pmatrix} \bar{\mathbf{F}}_N \varphi \\ \bar{\mathbf{F}}_N \psi \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \text{diag}(\hat{\alpha}_k)_{k=0}^{N-1} & \text{diag}(\hat{\alpha}_{k+N})_{k=0}^{N-1} \\ \text{diag}(\hat{\beta}_k)_{k=0}^{N-1} & \text{diag}(\hat{\beta}_{k+N})_{k=0}^{N-1} \end{pmatrix} \bar{\mathbf{F}}_{2N} \vartheta.$$

The inverse formula reads as

$$\begin{aligned} \bar{\mathbf{F}}_{2N} \vartheta &= \sqrt{2} \begin{pmatrix} \text{diag} \left(\frac{\hat{\beta}_{k+N}}{\hat{\alpha}_k \hat{\beta}_{k+N} - \hat{\alpha}_{k+N} \hat{\beta}_k} \right)_{k=0}^{N-1} & \text{diag} \left(\frac{-\hat{\alpha}_{k+N}}{\hat{\alpha}_k \hat{\beta}_{k+N} - \hat{\alpha}_{k+N} \hat{\beta}_k} \right)_{k=0}^{N-1} \\ \text{diag} \left(\frac{-\hat{\beta}_k}{\hat{\alpha}_k \hat{\beta}_{k+N} - \hat{\alpha}_{k+N} \hat{\beta}_k} \right)_{k=0}^{N-1} & \text{diag} \left(\frac{\hat{\alpha}_k}{\hat{\alpha}_k \hat{\beta}_{k+N} - \hat{\alpha}_{k+N} \hat{\beta}_k} \right)_{k=0}^{N-1} \end{pmatrix} \\ &\times \begin{pmatrix} \bar{\mathbf{F}}_N \varphi \\ \bar{\mathbf{F}}_N \psi \end{pmatrix}. \end{aligned}$$

Proof. With (26) we conclude that

$$\begin{pmatrix} \varphi \\ \psi \end{pmatrix} = \begin{pmatrix} (\alpha_{s-2k})_{k=0, s=0}^{N-1, 2N-1} \\ (\beta_{s-2k})_{k=0, s=0}^{N-1, 2N-1} \end{pmatrix} \vartheta, \quad (27)$$

where we extended the vectors (α_k) and (β_k) to be $2N$ -periodic. Multiplying from the left by the block matrix

$$\begin{pmatrix} \bar{\mathbf{F}}_N & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{F}}_N \end{pmatrix}$$

and using $\bar{\mathbf{F}}_N \mathbf{F}_N = \mathbf{I}$, we rewrite the right hand side of (27) as

$$\begin{pmatrix} \bar{\mathbf{F}}_N & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{F}}_N \end{pmatrix} \begin{pmatrix} (\alpha_{s-2k})_{k=0, s=0}^{N-1, 2N-1} \\ (\beta_{s-2k})_{k=0, s=0}^{N-1, 2N-1} \end{pmatrix} \mathbf{F}_{2N} \bar{\mathbf{F}}_{2N} \vartheta$$

which is simply

$$\begin{aligned} &\frac{1}{\sqrt{2N}} \begin{pmatrix} \bar{\mathbf{F}}_N & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{F}}_N \end{pmatrix} \begin{pmatrix} (\hat{\alpha}_\ell e^{-2\pi i \ell k/N})_{k=0, s=0}^{N-1, 2N-1} \\ (\hat{\beta}_\ell e^{-2\pi i \ell k/N})_{k=0, \ell=0}^{N-1, 2N-1} \end{pmatrix} \bar{\mathbf{F}}_{2N} \vartheta \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} \text{diag}(\hat{\alpha}_k)_{k=0}^{N-1} & \text{diag}(\hat{\alpha}_{k+N})_{k=0}^{N-1} \\ \text{diag}(\hat{\beta}_k)_{k=0}^{N-1} & \text{diag}(\hat{\beta}_{k+N})_{k=0}^{N-1} \end{pmatrix} \bar{\mathbf{F}}_{2N} \vartheta. \end{aligned} \quad (28)$$

□

Defining the circulant matrices

$$\mathbf{A} = \mathbf{F}_N \text{diag}(\hat{\alpha}_k)_{k=0}^{N-1} \bar{\mathbf{F}}_N, \quad \mathbf{B} = \mathbf{F}_N \text{diag}(\hat{\alpha}_k)_{k=N}^{2N-1} \bar{\mathbf{F}}_N,$$

$$\mathbf{C} = \mathbf{F}_N \text{diag}(\hat{\beta}_k)_{k=0}^{N-1} \bar{\mathbf{F}}_N \quad \text{and} \quad \mathbf{D} = \mathbf{F}_N \text{diag}(\hat{\beta}_k)_{k=N}^{2N-1} \bar{\mathbf{F}}_N,$$

and using (28) we can rewrite the basis transformation from Theorem 1 as

$$\begin{pmatrix} \varphi \\ \psi \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{F}_N & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_N \end{pmatrix} \bar{\mathbf{F}}_{2N} \boldsymbol{\vartheta}.$$

Having expressed the transformations as in Theorem 1 it is now straightforward to apply a lifting step or a dual lifting step as described in Definition 1. So one can change directly from arbitrary bases to orthogonal or interpolatory bases. In case of orthogonal bases for W one would change ψ into ψ^\perp given by

$$c_k(\psi^\perp) = \frac{c_k(\psi)}{\sqrt{N \sum_{p \in \mathbb{Z}} |c_{k+Np}(\psi)|^2}} \quad \text{for all } k \in \mathbb{Z},$$

and for interpolatory bases with the additional condition

$$\sum_{p \in \mathbb{Z}} c_{k+Np}(\psi) \neq 0 \quad \text{for all } k = 0, \dots, N-1,$$

one could change to

$$c_k(\psi^L) = \frac{c_k(\psi)}{N \sum_{p \in \mathbb{Z}} c_{k+Np}(\psi)} \quad \text{for all } k \in \mathbb{Z},$$

obtaining bases with $\psi^L(\frac{2k\pi}{N}) = \delta_{k,0}$ for $k = 0, \dots, N-1$. To achieve, for example, orthogonality of the translates of ψ we can choose $\mathbf{A} = \mathbf{I}$, $\mathbf{B} = \mathbf{0}$ and we have to replace $\text{diag}(\hat{\beta}_k)_{k=0}^{N-1}$ and $\text{diag}(\hat{\beta}_k)_{k=N}^{2N-1}$ by

$$\text{diag} \left(\frac{\hat{\beta}_k}{\sum_{p \in \mathbb{Z}} \hat{\beta}_{k+Np} \hat{\vartheta}_{k+Np}} \right)_{k=0}^{N-1} \quad \text{and} \quad \text{diag} \left(\frac{\hat{\beta}_{k+N}}{\sum_{p \in \mathbb{Z}} \hat{\beta}_{k+Np} \hat{\vartheta}_{k+Np}} \right)_{k=0}^{N-1},$$

respectively.

Acknowledgement

This work was supported in part by the European Union research project “Multiresolution in Geometric Modelling (MINGLE)” under grant HPRN-CT-1999-00117.

References

1. Carnicer, J.M., Dahmen, W., Pena, J.M.: Local decomposition of refinable spaces and wavelets. *Appl. Comput. Harmon. Anal.*, **3**, 127–153 (1996).
2. Chui, C.K., Mhaskar, H.N.: On trigonometric wavelets. *Constr. Approx.*, **9**, 167–190 (1993).
3. Cohen, A., Daubechies, I., Feauveau, J.-C.: Biorthogonal bases of compactly supported wavelets. *Commun. Pure Appl. Math.*, **45**, 485–560 (1992).
4. Dahmen, W., Kunoth, A., Urban, K.: Biorthogonal spline wavelets on the interval – stability and moment conditions. *Appl. Comput. Harmon. Anal.*, **6**, 132–196 (1999).
5. Dahmen, W., Micchelli, C.A.: Banded matrices with banded inverses. II: Locally finite decomposition of spline spaces. *Constr. Approx.*, **9**, 263–281 (1993).
6. Dahmen, W., Schneider, R.: Wavelets on manifolds. I: Construction and domain decomposition. *SIAM J. Math. Anal.*, **31**, 184–230 (1999).
7. Daubechies, I., Sweldens, W.: Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.*, **4**, 247–269 (1998).
8. Koh, Y.W., Lee, S.L., Tan, H.H. Periodic orthogonal splines and wavelets. *Appl. Comput. Harmonic Anal.*, **2**, 201–218 (1995).
9. Lyche, T., Mørken, K., Quak, E.: Theory and algorithms for nonuniform spline wavelets. In: Dyn, N., Leviatan, D., Levin, D., and Pinkus, A., (eds.), *Multivariate approximation and applications*, Cambridge University Press, 152–187 (2001).
10. Masson, R.: Biorthogonal spline wavelets on the interval for the resolution of boundary problems. *Math. Models Methods Appl. Sci.*, **6**, 749–791 (1996).
11. Narcowich, F.J., Ward, J.D.: Wavelets associated with periodic basis functions. *Appl. Comput. Harmon. Anal.*, **3**, 40–56 (1996).
12. Plonka, G., Tasche, M.: A unified approach to periodic wavelets. In: Chui, C.K., Montefusco, L., and Puccio, L. (eds.), *Wavelets: Theory, Algorithms, and Applications*, Academic Press, New York, 137–151 (1994).
13. Plonka, G., Tasche, M.: On the computation of periodic spline wavelets. *Appl. Comput. Harmon. Anal.*, **2**, 1–14 (1995).
14. Prestin, J., Quak, E.: Trigonometric interpolation and wavelet decompositions. *Num. Alg.*, **9**, 293–317 (1995).
15. Quak, E.: Nonuniform B -splines and B -wavelets. In: Iske, A., Quak, E., Floater M. S., (eds.), *Tutorials on Multiresolution in Geometric Modelling*, Springer, 101–146 (2002).
16. Schneider, R.: Multiskalen- und Wavelet-Matrixkompression. Analysisbasierte Methoden zur effizienten Lösung großer vollbesetzter Gleichungssysteme. *Advances in Numerical Mathematics*. B. G. Teubner Stuttgart (1998).
17. Selig, K.: Periodische Wavelet-Packets und eine gradoptimale Schauderbasis. PhD thesis, Universität Rostock, Germany, Shaker Verlag, Aachen (1998).
18. Sweldens, W., Schröder, P.: Building your own wavelets at home. *Wavelets in Computer Graphics, ACM SIGGRAPH Course Notes* (1996).
19. Sweldens, W.: The lifting scheme: A custom-design construction of biorthogonal wavelets. *Appl. Comput. Harmon. Anal.*, **3**, 186–200 (1996).

Nonstationary Sibling Wavelet Frames on Bounded Intervals: the Duality Relation

Laura Beutel

Institute of Applied Mathematics, University of Dortmund, Germany

`laura.beutel@math.uni-dortmund.de`

and

SINTEF Applied Mathematics, Oslo, Norway

`laura.beutel@sintef.no`

Summary. This note presents the setting of sibling frames on a compact interval together with a discussion on the duality relation.

1 Introduction

Hilbert frames are *overcomplete* and *stable* families of functions in a Hilbert space which provide (not necessarily unique) series representations for each element of the space. Frames play an important rôle in signal processing and other areas of applied mathematics and they can be considered to be a natural generalisation of Riesz bases. The overcompleteness of the system incorporates redundant information in the frame coefficients, such that the loss of some of them does not necessarily imply loss of information.

A sequence of functions $\{f_i\}_{i \in \mathbb{N}}$ from a separable Hilbert space \mathcal{H} (usually chosen as L_2) is called a *Bessel sequence* if there exists a constant B ($0 < B < \infty$) such that for every $f \in \mathcal{H}$

$$\sum_{i=1}^{\infty} |\langle f, f_i \rangle|^2 \leq B \cdot \|f\|_{\mathcal{H}}^2. \quad (1)$$

A Bessel sequence with Bessel bound B is called a *frame* if, in addition, there exists a constant A ($0 < A \leq B < \infty$) such that for every $f \in \mathcal{H}$ one has

$$A \cdot \|f\|_{\mathcal{H}}^2 \leq \sum_{i=1}^{\infty} |\langle f, f_i \rangle|^2. \quad (2)$$

A and B are the lower and upper frame bound, respectively, and the inequalities (1) and (2) together are called the *frame condition* and they express the (numerical) *stability* of the function family.

If one can choose equal frame bounds then the frame is called *tight*. A tight frame can always be *normalised*, such that $A = B = 1$. The tight frames generalise the orthonormal wavelets. For basic theory on Hilbert frames see for example [10, 5].

Two frames $\{f_i\}_{i \in \mathbb{N}}$ and $\{g_i\}_{i \in \mathbb{N}}$ from the same separable Hilbert space \mathcal{H} are dual to each other if for all $f, g \in \mathcal{H}$ we have the identity

$$\langle f, g \rangle = \sum_{i=1}^{\infty} \langle f, f_i \rangle \langle g_i, g \rangle. \quad (3)$$

Every frame has a dual, which is usually not unique. The tight frames form a very important subclass of frames, because the canonical dual of a tight frame is very easy to determine, for example the dual of a normalised tight frame is the frame itself. In the MRA setting a frame together with a dual, where both f_i and g_i are elements of the *same* spline-like space, are called *sibling frames*.

We are interested in a general construction scheme for locally supported frames (“*wavelet frames*”) and corresponding sibling duals, which are defined from a nonstationary MRA, where both irregular “shifts” on the same refinement level and nonuniform refinement between levels are allowed. In this paper we discuss one part of this construction scheme. We focus on the duality relation. Our construction yields analytical formulations of the frame elements and enables us to choose frame elements with local support and vanishing moments, features which are important in applications.

The generic example of this construction is that of tight spline frames on a bounded interval I (see [3, 4]), where the MRA of $L_2(I)$ is generated from nested knot sequences, whose maximal knot distances converge to zero. Only methods from spline theory (such as knot insertion) and linear algebra are needed in order to develop the theory; there is no need for any Fourier techniques. This allows *arbitrary* nested knot sequences with multiple knots instead of uniformly refined ones. In the sibling case, the spline frame will have a *spline* dual, a property which cannot be achieved in the orthonormal wavelet case, but which is desirable for applications.

2 Preliminaries

2.1 Equivalent Matrices and Invariant Properties

We recall first the classical definition of the equivalence between two matrices and afterwards we propose an extension of this concept. We use this generalised equivalence of matrices when we list some propositions and remarks which, to some extent, have been used implicitly in [3] in the process of characterising wavelet tight frames. This subsection – as well as the next one – is meant to supplement Sect. 2 in [3].

In A_m and $A_{m \times n}$ the subindices denote the dimensions of the (square, respectively rectangular) matrix A .

Definition 1. Two rectangular (complex) matrices $A_{m \times n}$ and $B_{m \times n}$ are called equivalent, if there exist two nonsingular matrices S_m and T_n such that A and B are related by the transformation $B_{m \times n} = S_m A_{m \times n} T_n$. In particular, two equivalent matrices A_m and B_m are called

- a) similar, if $T_m = S_m^{-1}$,
- b) congruent, if $T_m = S_m^T$, and
- c) conjunctive, if $T_m = S_m^*$.

Here, and also further in this paper, S^* denotes the Hermitian conjugate of S (i.e. $S^* := \overline{S^T}$).

For a square matrix A and a non-singular matrix S the following properties are well known. SAS^T is symmetric if and only if A is symmetric. SAS^* is Hermitian if and only if A is Hermitian. Similar matrices have identical spectra. For further purposes we wish to have a more relaxed definition of the equivalence in the sense that it is applicable to rectangular matrices of different dimensions. To this end we extend Definition 1 in a natural way as follows: two rectangular (complex) matrices $A_{m \times n}$ and $B_{p \times q}$ are called equivalent, if there exist two matrices $S_{p \times m}$ and $T_{n \times q}$ of maximal rank such that A and B are related by the transformation $B_{p \times q} = S_{p \times m} A_{m \times n} T_{n \times q}$.

For this extended concept we are still able to formulate two of the three aforementioned cases, namely: two equivalent square matrices A_m and B_p are called

- d) congruent, if $T_{m \times p} = S_{p \times m}^T$, and
- e) conjunctive, if $T_{m \times p} = S_{p \times m}^*$.

We are interested in finding properties which remain invariant with respect to the presented transformations. The following two results are proved by standard techniques from linear algebra. Note that Proposition 1 is a special case of Proposition 2.

Proposition 1. Symmetry, skew-symmetry, positive semi-definiteness and negative semi-definiteness are properties which remain invariant under the congruence transformations described in d).

Proposition 2. Hermiticity, skew-Hermiticity, positive semi-definiteness and negative semi-definiteness are properties which remain invariant under the conjunctivity transformations given in e).

We want to emphasise at this point that, in the setting of Proposition 2, from the positive definiteness of A we could only infer that B is positive semi-definite. Thus, in general, the (positive or negative) definiteness is not invariant under the conjunctivity transformation.

Next we consider a special recurrence relation, (4), between some matrices which arises naturally and is fundamental in the study of the duality condition of sibling frames. Particular instances of it can be encountered in the tight

frame case (when $Q_j = \tilde{Q}_j$, see [3]) as well as in the orthonormal wavelet case (when, along with some stability conditions, $Q_j = \tilde{Q}_j$ and S_{j+1}, S_j are identity matrices, see [8]).

Proposition 3. *Let S_0 be a square matrix of dimension M_0 and let $\{Q_j\}_{j \geq 0}$, $\{\tilde{Q}_j\}_{j \geq 0}$ and $\{P_j\}_{j \geq 0}$ be families of nonzero real rectangular matrices with $\dim Q_j = \dim \tilde{Q}_j = M_{j+1} \times N_j$ and $\dim P_j = M_{j+1} \times M_j$. Furthermore let the square matrices S_j of dimension M_j , $j > 0$, be defined by the recurrence*

$$S_{j+1} := P_j S_j P_j^T + Q_j \tilde{Q}_j^T, \quad j \geq 0. \quad (4)$$

Then the following statements hold.

- a) *If S_0 is symmetric positive semi-definite and for every $j \geq 0$ the matrix $Q_j \tilde{Q}_j^T$ is symmetric positive semi-definite (respectively, symmetric positive definite), then all the matrices S_j are symmetric positive semi-definite (respectively, symmetric positive definite).*
- b) *If in a) the term “positive” is exchanged with “negative”, the new statement holds as well.*

The proof is based on Proposition 1. In condensed form Proposition 3 states that a sequence of matrices $\{S_j\}_{j \geq 0}$ with the recurrence structure described by (4) inherits the (semi)definiteness property of the sequence $\{Q_j \tilde{Q}_j^T\}_{j \geq 0}$. The first matrix S_0 cannot be arbitrary. It is important for the first step that it also has some definiteness property, but this is not enough for the recurrence in (4) to produce a whole sequence of matrices $\{S_j\}_{j \geq 0}$ with the same property.

In the particular case, where the sequences $\{Q_j\}_{j \geq 0}$ and $\{\tilde{Q}_j\}_{j \geq 0}$ are identical, the request that $Q_j \tilde{Q}_j^T$ should be symmetric positive semi-definite for all $j \geq 0$ is automatically satisfied and we get for arbitrary $\{Q_j\}_{j \geq 0}$ and arbitrary $\{P_j\}_{j \geq 0}$ a sequence $\{S_j\}_{j \geq 0}$ of symmetric positive semi-definite matrices. This is exactly the tight frame case presented in [3].

In the setting of Proposition 3 if the sequences $\{S_j\}_{j \geq 0}$ and $\{P_j\}_{j \geq 0}$ are given, then in order to ensure that all the matrices $Q_j \tilde{Q}_j^T$ be symmetric positive semi-definite, one has (obviously) to demand (or to test) that the differences $S_{j+1} - P_j S_j P_j^T$ are symmetric positive semi-definite for every $j \geq 0$.

In the following we use row vector notation. To condense the notation, we introduce $\mathbb{M}_j := \{1, 2, \dots, M_j\}$, $\mathbb{N}_j := \{1, 2, \dots, N_j\}$ and for the function family $\Phi_j = \{\phi_{j,k}\}_{k \in \mathbb{M}_j}$ we will set $\langle f, \Phi_j \rangle := [\langle f, \phi_{j,k} \rangle]_{k \in \mathbb{M}_j}$.

2.2 Bilinear Forms and Kernels

In this subsection, for a finite function family and a real matrix, we introduce two entities: a bilinear form and a kernel. They will be our tools in characterising sibling frames. In [3] the authors use, for the characterisation of tight

frames, the same kernel as we do and the quadratic form associated to the bilinear form from below.

Let $I = [a, b]$ be a compact interval on the real axis. For a finite family $\Phi_j = [\phi_{j,k}]_{k \in M_j}$ from $L_2(I)$ with cardinality M_j ($j \geq 0$) and a real matrix $S_j = [s_{k,l}^{(j)}]_{k,l \in M_j}$ we consider:

a) the bilinear form

$$\begin{aligned} T_j(f, g) &:= [\langle f, \phi_{j,k} \rangle]_{k \in M_j} \cdot S_j \cdot [\langle g, \phi_{j,k} \rangle]_{k \in M_j}^T \\ &=: \langle f, \Phi_j \rangle \cdot S_j \cdot \langle g, \Phi_j \rangle^T, \quad f, g \in L_2(I), \end{aligned} \quad (5)$$

b) the kernel

$$\begin{aligned} K_{S_j}(x, y) &:= \Phi_j(x) \cdot S_j \cdot \Phi_j^T(y) \\ &= \sum_{k,l \in M_j} \phi_{j,k}(x) \cdot s_{k,l}^{(j)} \cdot \phi_{j,l}(y), \quad x, y \in I. \end{aligned} \quad (6)$$

$T_j(f, g)$ and $K_{S_j}(x, y)$ inherit the symmetry and definiteness properties of the matrix S_j . Furthermore, they are related by

$$T_j(f, g) = \int_I f(x) \int_I g(y) \cdot K_{S_j}(x, y) dy dx, \quad f, g \in L_2(I).$$

The following theorem is the first main result we present in this paper. It describes inheritance properties of T_j and K_{S_j} .

Theorem 1. Let Φ_j , $j \geq 0$, be finite families from $L_2(I)$, Φ_j with cardinality M_j , such that a refinement relation of the form $\Phi_j = \Phi_{j+1} \cdot P_j$ exists for all $j \geq 0$, where the P_j 's are real matrices of dimensions $M_{j+1} \times M_j$.

Furthermore let the families $\Psi = \{\Psi_j\}_{j \geq 0}$ and $\tilde{\Psi} = \{\tilde{\Psi}_j\}_{j \geq 0}$ have the structure

$$\begin{aligned} \Psi_j &:= \Phi_{j+1} \cdot Q_j =: [\psi_{j,k}]_{k \in N_j}, \quad j \geq 0, \\ \tilde{\Psi}_j &:= \Phi_{j+1} \cdot \tilde{Q}_j =: [\tilde{\psi}_{j,k}]_{k \in N_j}, \quad j \geq 0, \end{aligned}$$

where Q_j and \tilde{Q}_j are (typically sparse, or even banded) real matrices of dimensions $M_{j+1} \times N_j$.

If there exists a sequence of real matrices $(S_j)_{j \geq 0}$ which are related by the recurrence

$$S_{j+1} = P_j S_j P_j^T + Q_j \tilde{Q}_j^T, \quad j \geq 0,$$

then the following statements hold.

a) The associated bilinear forms T_J on $L_2(I)^2$ w.r.t. $\{\Phi_J\}_{J \geq 0}$ inherit this structure. They satisfy the recurrence relation

$$T_{J+1}(f, g) = T_J(f, g) + \sum_{l \in \mathbb{N}_J} \langle f, \psi_{J,l} \rangle \langle g, \tilde{\psi}_{J,l} \rangle$$

and the representation formula

$$T_{J+1}(f, g) = T_0(f, g) + \sum_{j=0}^J \sum_{l \in \mathbb{N}_j} \langle f, \psi_{j,l} \rangle \langle g, \tilde{\psi}_{j,l} \rangle \quad (7)$$

for all $f, g \in L_2(I)$ and all $J \geq 0$.

- b) The associated kernels K_{S_j} w.r.t. $\{\Phi_j\}_{j \geq 0}$ inherit this structure, i.e. there holds the recurrence

$$K_{S_{J+1}}(x, y) = K_{S_J}(x, y) + \sum_{k \in \mathbb{N}_J} \psi_{J,k}(x) \cdot \tilde{\psi}_{J,k}(y),$$

and further we get the representation formula

$$K_{S_{J+1}}(x, y) = K_{S_0}(x, y) + \sum_{j=0}^J \sum_{k \in \mathbb{N}_j} \psi_{j,k}(x) \cdot \tilde{\psi}_{j,k}(y),$$

both being valid for all $x, y \in I$ and any $J \geq 0$.

Proof. We have

$$\begin{aligned} T_{J+1}(f, g) &= \langle f, \Phi_{J+1} \rangle \cdot S_{J+1} \cdot \langle g, \Phi_{J+1} \rangle^T \\ &= (\langle f, \Phi_{J+1} \rangle \cdot P_J) \cdot S_J \cdot (\langle g, \Phi_{J+1} \rangle \cdot P_J)^T \\ &\quad + (\langle f, \Phi_{J+1} \rangle \cdot Q_J) \cdot \left(\langle g, \Phi_{J+1} \rangle \cdot \tilde{Q}_J \right)^T \\ &= \langle f, \Phi_{J+1} \cdot P_J \rangle \cdot S_J \cdot \langle g, \Phi_{J+1} \cdot P_J \rangle^T \\ &\quad + \langle f, \Phi_{J+1} \cdot Q_J \rangle \cdot \langle g, \Phi_{J+1} \cdot \tilde{Q}_J \rangle^T \\ &= \langle f, \Phi_J \rangle \cdot S_J \cdot \langle g, \Phi_J \rangle^T + \langle f, \Psi_J \rangle \cdot \langle g, \tilde{\Psi}_J \rangle^T \\ &= T_J(f, g) + \sum_{l \in \mathbb{N}_J} \langle f, \psi_{J,l} \rangle \langle g, \tilde{\psi}_{J,l} \rangle, \quad \forall J \geq 0, \\ &= T_0(f, g) + \sum_{j=0}^J \sum_{l \in \mathbb{N}_j} \langle f, \psi_{j,l} \rangle \langle g, \tilde{\psi}_{j,l} \rangle, \quad \forall J \geq 0, \end{aligned}$$

$$\begin{aligned}
\text{and } K_{S_{J+1}}(x, y) &= \Phi_{J+1}(x) \cdot S_{J+1} \cdot \Phi_{J+1}^T(y) \\
&= (\Phi_{J+1}(x) \cdot P_J) \cdot S_J \cdot (\Phi_{J+1}(y) \cdot P_J)^T \\
&\quad + (\Phi_{J+1}(x) \cdot Q_J) \left(\Phi_{J+1}(y) \cdot \tilde{Q}_J \right)^T \\
&= \Phi_J(x) \cdot S_J \cdot \Phi_J^T(y) + \Psi_J(x) \cdot \tilde{\Psi}_J^T(y) \\
&= K_{S_J}(x, y) + \sum_{k \in \mathbb{N}_J} \psi_{J,k}(x) \cdot \tilde{\psi}_{J,k}(y), \quad \forall J \geq 0, \\
&= K_{S_0}(x, y) + \sum_{j=0}^J \sum_{k \in \mathbb{N}_j} \psi_{j,k}(x) \cdot \tilde{\psi}_{j,k}(y), \quad \forall J \geq 0. \quad \square
\end{aligned}$$

We want to emphasise here, that – in order to get the above recurrence relations for T_j and K_{S_j} – we do not have to assume any special properties of the matrices in use; the relation between the matrices is the crucial point.

3 MRA Framework and Definition of Sibling Frames

This section presents the nonstationary MRA setting under which we will work and gives a definition of (nonstationary MRA) sibling frames of $L_2(I)$ associated with the locally supported and finite refinable function vectors Φ_j and with the ground level characterised by a matrix S_0 .

The notion of affine sibling frames of $L_2(\mathbb{R})$ was introduced for the first time in [2, Definition 1] in order to achieve more flexibility and thus additional properties for the frame elements such as symmetry (or anti-symmetry), small support, high order of vanishing moments, approximate shift-invariance and inter-orthogonality. A parallel and independent development of some similar and overlapping results is presented in [6]. Chui et al. [3] present a general construction scheme as well as practical procedures for (non-affine, nonstationary) tight wavelet frames with maximal number of vanishing moments and minimal support on a compact interval of the real line.

We present here a more general (i.e. non-affine, nonstationary, non-tight) approach for sibling frames of $L_2(I)$, where I is a compact interval of \mathbb{R} . To our knowledge this approach has not been studied so far.

Let $I = [a, b]$ be a bounded interval on the real axis \mathbb{R} and let $V_0 \subset V_1 \subset \dots \subset V_n \subset \dots$ be a sequence of nested finite dimensional subspaces of $L_2(I)$ which are dense in the space, i.e. $\text{clos}_{L_2}(\cup_{j \geq 0} V_j) = L_2(I)$. Giving up the affine structure we will assume more generally that each V_j is spanned by a finite system $\Phi_j = [\phi_{j,k}; 1 \leq k \leq M_j]$, where $M_j \geq \dim V_j$. The refinement relation of $V_j \subset V_{j+1}$ will be described by a real matrix $P_j = [p_{k,l}^{(j)}]$ of dimensions $M_{j+1} \times M_j$ which reads as follows: $\Phi_j = \Phi_{j+1} \cdot P_j$.

Note that we do not require any conditions of uniform refinement (neither shift invariance nor dilation invariance). Nonstationarity refers here to both irregular “shifts” on the same level and nonuniform refinement from one level

to the next one which are allowed in the described MRA. This general framework is also used in [3] and the special case of the spline MRA generated by a sequence of nested knot sequences which are dense in I can be encountered, for example in [3, 8]. For the frame elements to be useful in applications, we will require in the sequel the same localisation property of the refinable function vectors Φ_j as in [3].

Definition 2. A function family $\Phi = \{\Phi_j\}_{j \geq 0} = \{[\phi_{j,k}; 1 \leq k \leq M_j]\}_{j \geq 0}$ is said to be locally supported, if the sequence of the maximal support length on each level

$$h(\Phi_j) := \max_{k \in M_j} \text{length}(\text{supp } \phi_{j,k})$$

converges to zero.

Definition 3. Assume that $\Phi = \{\Phi_j\}_{j \geq 0}$ is a locally supported family, where Φ_j has cardinality M_j , and adjacent families are related by the refinement relation $\Phi_j = \Phi_{j+1} \cdot P_j$. Let S_0 be the matrix that defines the bilinear form T_0 (i.e. the ground level component).

Then the families Ψ and $\tilde{\Psi}$ determined by the sequences of matrices $\{Q_j\}_{j \geq 0}$ and $\{\tilde{Q}_j\}_{j \geq 0}$ ($\dim Q_j = \dim \tilde{Q}_j = M_{j+1} \times N_j$) through the relations

$$\Psi = \{\Psi_j\}_{j \geq 0} = \{\Phi_{j+1} \cdot Q_j\}_{j \geq 0} = \{[\psi_{j,k}]_{k \in N_j}\}_{j \geq 0} \quad (8)$$

and

$$\tilde{\Psi} = \{\tilde{\Psi}_j\}_{j \geq 0} = \{\Phi_{j+1} \cdot \tilde{Q}_j\}_{j \geq 0} = \{[\tilde{\psi}_{j,k}]_{k \in N_j}\}_{j \geq 0} \quad (9)$$

constitute sibling frames of $L_2(I)$ w.r.t. T_0 , if the following conditions are satisfied.

- a) They are Bessel families, i.e. there exist constants B and \tilde{B} with $0 < B, \tilde{B} < \infty$ such that
 i)

$$T_0(f, f) + \sum_{j \geq 0} \sum_{k \in N_j} |\langle f, \psi_{j,k} \rangle|^2 \leq B \cdot \|f\|_2^2 \quad (10)$$

and

ii)

$$T_0(f, f) + \sum_{j \geq 0} \sum_{k \in N_j} |\langle f, \tilde{\psi}_{j,k} \rangle|^2 \leq \tilde{B} \cdot \|f\|_2^2 \quad (11)$$

for all $f \in L_2(I)$.

- b) They are dual, i.e. for all $f, g \in L_2(I)$ we have the identity

$$T_0(f, g) + \sum_{j \geq 0} \sum_{k \in N_j} \langle f, \psi_{j,k} \rangle \langle \tilde{\psi}_{j,k}, g \rangle = \langle f, g \rangle. \quad (12)$$

Note that in this case both families Ψ and $\tilde{\Psi}$ are indeed frames of $L_2(I)$. Using the duality relation one can prove that the lower frame bound of Ψ is \tilde{B}^{-1} and that of $\tilde{\Psi}$ is B^{-1} . The numbers N_j of frame elements on the corresponding levels j ($j \geq 0$) are free parameters in the construction of sibling frames and they govern the redundancy degree of the frame system.

The assumption that Ψ and $\tilde{\Psi}$ are Bessel families is not needed for tight frames. For this special case the boundedness is contained in the duality relation (i.e. all three conditions in the above definition collapse to one identity). Unlike this, in the (non-tight) sibling frame case one has to find suitable (necessary and) sufficient conditions for the boundedness. We have successfully extended Meyer's "vaguelettes" collection (see [9, p. 270]) to the non-stationary setting presented in the above. Furthermore, we have proven that if such a family possesses a suitable "overlapping constant", then it is a Bessel family and we have obtained a concrete Bessel bound. These results are very technical, their detailed presentation goes beyond the scope of this note and will be presented elsewhere.

For the remainder of this section let $\Phi = \{\Phi_j\}_{j \geq 0}$ be a locally supported family with $\#\Phi_j = M_j$ and $\Phi_j = \Phi_{j+1} \cdot P_j$. The matrix S_0 defines the bilinear form T_0 . Further let Ψ and $\tilde{\Psi}$ be function families as defined by (8) and (9) and satisfying the Bessel conditions (10) and (11). Using identity (7) and applying the Cauchy-Schwarz inequality we get

Proposition 4. *If the matrix S_0 is symmetric positive definite, then the bilinear forms T_J ($J \geq 1$) are bounded from above as follows:*

$$|T_J(f, g)| \leq |T_0(f, g)| + \sqrt{B\tilde{B}} \cdot \|f\| \cdot \|g\|, \quad f, g \in L_2(I). \quad (13)$$

If, in addition, Φ_0 is a Bessel family with Bessel bound B_0 , then the T_J 's are uniformly bounded, i.e. the following relation holds.

$$|T_J(f, g)| \leq \left(B_0 \cdot \|S_0\|_2 + \sqrt{B\tilde{B}} \right) \cdot \|f\| \cdot \|g\|, \quad f, g \in L_2(I), \quad (14)$$

where by $\|S_0\|_2$ we denote the spectral norm of S_0 .

The monotonicity of the sequence of bilinear forms $(T_J)_J$ is not ensured in the general case, but we can state

Proposition 5. *If all the matrices $Q_J \tilde{Q}_J^T$ are symmetric positive (negative) semi-definite, then the sequence of quadratic forms $(T_J)_J$ is monotonically increasing (decreasing, respectively). Obviously, if the matrices are definite then we get strict monotonicity.*

4 Characterisation of the Duality Relation

In this section we present and discuss necessary and sufficient conditions for the existence of the duality relation (12). Our next main result generalises

Theorem 1 in [3] and gives a precise characterisation of the duality of two Bessel families.

Theorem 2. Let $\Phi = \{\Phi_j\}_{j \geq 0}$ be a locally supported family with $\#\Phi_j = M_j$ and $\Phi_j = \Phi_{j+1} \cdot P_j$. The matrix S_0 defines the ground level component T_0 . Furthermore, let Ψ and $\tilde{\Psi}$ be function families as defined by (8) and (9) and satisfying the Bessel conditions (10) and (11).

Ψ and $\tilde{\Psi}$ are dual (and thus sibling frames w.r.t. T_0), if and only if there exists a sequence of matrices $(S_j)_{j \geq 1}$, $\dim S_j = M_j$, such that

- a) the bilinear forms T_j satisfy

$$\lim_{j \rightarrow \infty} T_j(f, g) = \langle f, g \rangle, \quad f, g \in L_2(I), \quad (15)$$

and

- b) for every $j \geq 0$ we have

$$S_{j+1} - P_j S_j P_j^T = Q_j \tilde{Q}_j^T. \quad (16)$$

Proof. Let f and g be two arbitrarily fixed functions from $L_2(I)$.

Sufficiency. According to Theorem 1, property b) implies identity (7) which, combined with a), gives the desired duality relation.

Necessity. S_0 is given and for $j \geq 1$ we define the matrices recursively by $S_{j+1} := P_j S_j P_j^T + Q_j \tilde{Q}_j^T$. Thus condition (16) is satisfied. Equation (7) follows by an application of Theorem 1. Thus the duality relation implies the convergence of the sequence $(T_J(f, g))_J$ and, therefore, the desired limit (15). \square

Note that identity (16) describes the relation of all the matrices involved in the definition of sibling frames and thus it points out their interplay in the construction process.

Next we want to find some special cases where the limit (15) exists. For this purpose we consider the kernels introduced in (6) and we follow a classical and well studied approach from Approximation Theory (see [1, 7]) which seems to be very useful in our Hilbert frame setting. First of all we recall the definition of an approximate identity.

Definition 4. A sequence of kernels $K_n : I^2 \rightarrow \mathbb{R}$, $n \in \mathbb{N}$, is called an approximate identity if the functions K_n are continuous and satisfy

- a) Normalisation: $\int_I K_n(x, t) dt \rightarrow 1$ uniformly in $x \in I$ when $n \rightarrow \infty$;
- b) Uniform boundedness w.r.t. n : for every $x \in I$ there exists $M(x) > 0$ such that $\int_I |K_n(x, t)| dt \leq M(x)$ for all $n \in \mathbb{N}$;
- c) Localisation: for every $\delta \in (0, |I|]$ we have $\int_{|x-t| \geq \delta} |K_n(x, t)| dt \rightarrow 0$ uniformly in x for $n \rightarrow \infty$.

If the boundedness constant $M(x)$ does not depend on the variable x , then we call $\{K_n\}_n$ a uniformly bounded approximate identity.

A fundamental result from the approximation theory literature (see e.g. [7, Theorem 2.1, p. 5]) states that for an approximate identity $(K_n)_{n \in \mathbb{N}}$, for every continuous function $f : I \rightarrow \mathbb{R}$ and every $x \in I$ we have the following convergence:

$$\int_I K_n(x, t) \cdot f(t) dt \rightarrow f(x) \quad \text{for } n \rightarrow \infty. \quad (17)$$

Furthermore, this convergence is uniform in x if the approximate identity is uniformly bounded. This powerful result will be the main ingredient in the proof of Theorem 3. Now we present some sufficient conditions on the kernels in order to satisfy (15). These demands can be verified in practical situations more easily than (15).

Theorem 3. *If the kernels $(K_{S_j})_{j \geq 0}$ form a uniformly bounded approximate identity, then the forms $(T_j)_{j \geq 0}$ form a bilinear approximation method of the scalar product operator on $L_2(I^2)$, i.e. identity (15) holds.*

Proof. Let f and g be real continuous functions on I . Without loss of generality we assume that $f \not\equiv 0$. By using the aforementioned fundamental result for uniformly bounded approximate identities, for each $\varepsilon > 0$, there exists $N_\varepsilon \in \mathbb{N}$ such that for all $j \geq N_\varepsilon$ there holds

$$\begin{aligned} |T_j(f, g) - \langle f, g \rangle| &= \left| \int_I f(x) \left[\int_I g(y) \cdot K_{S_j}(x, y) dy - g(x) \right] dx \right| \\ &\leq \int_I |f(x)| \cdot \left| \int_I g(y) \cdot K_{S_j}(x, y) dy - g(x) \right| dx \\ &< \int_I |f(x)| \cdot \frac{\varepsilon}{\|f\|_1} dx = \varepsilon. \end{aligned}$$

Applying a density argument we get the desired limit for all functions in the space $L_2(I)$. \square

Proposition 4 and Theorem 3 generalise Theorem 9 in [3]. Note further that the kernels K_{S_j} are continuous if we choose families Φ_j of continuous functions. Condition c) in Definition 4 is satisfied if all the matrices S_j have a fixed maximal bandwidth and the function family $\Phi = \{\Phi_j\}_{j \geq 0}$ is locally supported in the sense of Definition 2. Namely, in this case the integral appearing in condition c) of Definition 4 is equal to zero for indices j which are large enough.

In the spline setting mentioned in Sect. 3, one usually chooses Φ_j to be the (suitably normalised) B-spline basis of V_j . Furthermore, if the matrices S_j are chosen as in [3] (i.e. constructed directly and only from the knot sequences t_j), then it follows immediately that $\{K_{S_j}\}_j$ are continuous and local. The uniform boundedness of the kernels $K_{S_j}(x, y)$ with respect to both j and x , as well as their normalisation, is given by Theorem 8 in [3]. Thus, in this case,

all the hypotheses of Theorem 3 are fulfilled. In order to obtain sibling frames we have to factorise the matrices $S_{j+1} - P_j S_j P_j^T$ appropriately into $Q_j \cdot \tilde{Q}_j^T$, i.e. such that the Bessel conditions are satisfied. We are currently studying possible factorisations. In order to give the reader an idea of our approach, we present an example of a Bessel family, which is our frame candidate. We consider a MRA generated by the dense sequence of finite knot vectors $\mathbf{t}_0 \subset \dots \subset \mathbf{t}_j \subset \mathbf{t}_{j+1} \subset \dots \subset [a, b]$. Each \mathbf{t}_j has $\text{int}(j)$ inner knots of multiplicity at most m and stacked boundary knots of maximal multiplicity m . The L^2 -normalised B-splines over the knot sequence \mathbf{t}_j are denoted by $\{N_{\mathbf{t}_j; m, k}^B : k = -m + 1, \dots, \text{int}(j)\}$ and weighted knot differences are defined by

$$d_{\mathbf{t}_j; m, k} := \frac{t_{m+k}^{(j)} - t_k^{(j)}}{m}.$$

The family

$$\cup_{j \geq 0} \{N_{\mathbf{t}_j; m, k}^B : k = -m + 1, \dots, \text{int}(j)\}$$

is locally supported, due to the density of the nested knot sequences in $[a, b]$ and the refinement matrices $P_{\mathbf{t}_j, \mathbf{t}_{j+1}; m}$ are given by the Oslo algorithm. The sequence of maximal support length $\{h(\Phi_j)\}_{j \geq 0}$ is in this case monotonically decreasing; this is a positive aspect for applications.

At present, frame candidates with one vanishing moment are given by

$$\psi_{j,k}(x) := \text{norm}_{j,k} \cdot \left(N_{\mathbf{t}_j; m+1, k}^B \right)'(x), \quad k \in \{-m + 1, \dots, \text{int}(j) - 1\}, \quad j \geq 0,$$

with normalisation

$$\text{norm}_{j,k} := \min \{d_{\mathbf{t}_j; m, k}, d_{\mathbf{t}_j; m, k+1}\} \cdot \frac{\min \{d_{\mathbf{t}_j; m-1, k}, d_{\mathbf{t}_j; m-1, k+1}, d_{\mathbf{t}_j; m-1, k+2}\}}{d_{\mathbf{t}_j; m+1, k}}.$$

We have proven that, under additional geometrical assumptions on the refinement of the knot vectors, this is a Bessel family by using our extension of Meyer's results.

5 Conclusions

We have presented results concerning the duality relation which describe necessary conditions for the bilinear forms T_j , the kernels K_{S_j} and the matrices S_j to verify a matrix factorisation technique for the construction of sibling frames.

In the future, our goal is to give a general construction scheme for sibling frames on a bounded interval and to construct such frames in the special spline MRA setting generated by a sequence of nested knot sequences which are dense in I .

Acknowledgement

This work was supported in part by the European Union research project “Multiresolution in Geometric Modelling (MINGLE)” under grant HPRN-CT-1999-00117.

References

1. P. L. Butzer and R. J. Nessel. *Fourier Analysis and Approximation*. Birkhäuser, Basel und Stuttgart, 1971.
2. C. K. Chui, W. He, and J. Stöckler. Compactly supported tight and sibling frames with maximum vanishing moments. *Applied Comput. Harmonic Anal.* 13:224–262, 2002.
3. C. K. Chui, W. He, and J. Stöckler. Nonstationary tight wavelet frames on bounded intervals, submitted. Preprint available as: Universität Dortmund, Ergebnisberichte Angewandte Mathematik Nr. 230, April 2003.
4. C. K. Chui and J. Stöckler. Recent development of spline wavelet frames with compact support. To appear in *Beyond wavelets*, G. V. Welland (ed.). Academic Press, San Diego et al., 2003.
5. I. Daubechies. *Ten Lectures on Wavelets*. SIAM, Philadelphia, Pa., 1992.
6. I. Daubechies, B. Han, A. Ron, and Z. Shen. Framelets: MRA-based constructions of wavelet frames. *Applied Comput. Harmonic Anal.* 14:1–46, 2003.
7. R. A. DeVore and G. G. Lorentz. *Constructive Approximation*. Springer, Berlin, 1993.
8. T. Lyche, K. Mørken, and E. Quak. *Theory and algorithms for nonuniform spline wavelets*. In “Multivariate Approximation and Applications”, N. Dyn, D. Leviatan, D. Levin and A. Pinkus (eds.), Cambridge University Press, pages 152–187, 2001.
9. Y. Meyer. *Ondelettes et Opérateurs: II. Opérateurs de Calderón Zygmund*. Hermann et Cie, Paris, 1990.
10. R. M. Young. *An Introduction to Nonharmonic Fourier Series*. Academic Press, San Diego, 2001.

Haar Wavelets on Spherical Triangulations

Daniela Roșca

Institute of Mathematics, University of Lübeck, Germany
and

Department of Mathematics, Technical University of Cluj-Napoca, Romania
`Daniela.Catinas@math.utcluj.ro`

Summary. We construct piecewise constant wavelets on spherical triangulations, which are orthogonal with respect to a scalar product on $L^2(\mathbb{S}^2)$. Our classes of wavelets include certain wavelets obtained by Bonneau and by Nielson et al. We also prove the Riesz stability and show some numerical experiments.

1 Introduction

In [1] and [2] some “nearly orthogonal” piecewise constant wavelets defined on arbitrary triangulations of the sphere \mathbb{S}^2 of \mathbb{R}^3 are presented. In [2] a spherical wavelet basis is said to be *nearly orthogonal* if it becomes orthogonal when the subdivision depth increases (i.e. when the spherical triangles are “near” planar). Actually, the orthogonality occurs if, at each level of the multiresolution, the areas of the spherical triangles are approximated by the areas of the corresponding planar triangles. Some numerical examples show that this idea works well in practice, but no mathematical arguments were given to assure that it works in practice all the time.

In this paper we use a scalar product $\langle \cdot, \cdot \rangle_*$ on $L^2(\mathbb{S}^2)$, defined in [3], which induces a norm $\|\cdot\|_*$ equivalent to the usual 2-norm of $L^2(\mathbb{S}^2)$. Then we construct piecewise constant wavelets which are orthogonal with respect to this scalar product. The equivalence of the norms $\|\cdot\|_*$ and the usual 2-norm of $L^2(\mathbb{S}^2)$ will help us to prove the Riesz stability in $L^2(\mathbb{S}^2)$ of our wavelets.

2 Preliminaries

Consider the unit sphere \mathbb{S}^2 of \mathbb{R}^3 with centre O and Π a convex polyhedron having triangular faces¹ and the vertices situated on the sphere. Also we have

¹ The polyhedron could also have faces which are not triangles. In that case we triangulate each of these faces and consider it as having triangular faces, with some of the faces coplanar.

to suppose that no face contains the origin O and O is situated inside the polyhedron. We denote by $\mathcal{T}^0 = \{T_1, T_2, \dots, T_n\}$ the set of the faces of Π and by Ω the surface (the “cover”) of Π . Then we consider the radial projection onto \mathbb{S}^2 , $p : \Omega \rightarrow \mathbb{S}^2$,

$$p(x, y, z) = \frac{1}{\sqrt{x^2 + y^2 + z^2}} (x, y, z), \quad (x, y, z) \in \Omega, \quad (1)$$

and its inverse $p^{-1} : \mathbb{S}^2 \rightarrow \Omega$,

$$p^{-1}(\eta_1, \eta_2, \eta_3) = \frac{-d}{a\eta_1 + b\eta_2 + c\eta_3} (\eta_1, \eta_2, \eta_3), \quad (\eta_1, \eta_2, \eta_3) \in \mathbb{S}^2,$$

where $ax + by + cz + d = 0$ is the equation of the face of Π onto which the point $(\eta_1, \eta_2, \eta_3) \in \mathbb{S}^2$ projects. In case this projection is situated on an edge, then one of the two faces containing that edge is taken.

Being given Ω , we can say that $\mathcal{T} = \mathcal{T}^0$ is a triangulation of Ω . Next we wish to consider its uniform refinement \mathcal{T}^1 . For a given triangle $[M_1 M_2 M_3]$ in \mathcal{T}^0 , let A_1, A_2, A_3 denote the midpoints of the edges $M_2 M_3, M_3 M_1$ and $M_1 M_2$, respectively. Then we consider the set

$$\mathcal{T}^1 = \bigcup_{[M_1 M_2 M_3] \in \mathcal{T}^0} \{[M_1 A_2 A_3], [A_1 M_2 A_3], [A_1 A_2 M_3], [A_1 A_2 A_3]\},$$

which is also a triangulation of Ω . Continuing the refinement process in the same manner, we can obtain a triangulation \mathcal{T}^j of Ω , for $j \in \mathbb{N}$. The projection of \mathcal{T}^j onto the sphere will be $\mathcal{U}^j = \{p(T^j), T^j \in \mathcal{T}^j\}$, which is a triangulation of \mathbb{S}^2 . The number of triangles in \mathcal{U}^j will be $|\mathcal{U}^j| = n \cdot 4^j$.

Let $\langle \cdot, \cdot \rangle_\Omega$ be the following inner product, based on the initial coarsest triangulation \mathcal{T}^0 :

$$\langle f, g \rangle_\Omega = \sum_{T \in \mathcal{T}^0} \frac{1}{a(T)} \int_T f(\mathbf{x}) g(\mathbf{x}) d\mathbf{x}, \quad \text{for } f, g \in C(T) \quad \forall T \in \mathcal{T}^0.$$

Here $a(T)$ denotes the area of the triangle T . Also, we consider the induced norm

$$\|f\|_\Omega = \langle f, f \rangle_\Omega^{1/2}.$$

For L^2 -integrable functions F and G defined on \mathbb{S}^2 , the following scalar product associated to the given polyhedron Π was defined in [3]:

$$\langle F, G \rangle_* = \langle F \circ p, G \circ p \rangle_\Omega. \quad (2)$$

There it was proved that, in the space $L^2(\mathbb{S}^2)$, the norm $\|\cdot\|_*$ induced by this scalar product is equivalent to the usual norm $\|\cdot\|_{L^2(\mathbb{S}^2)}$ of $L^2(\mathbb{S}^2)$ and

$$m \|F\|_{L^2(\mathbb{S}^2)}^2 \leq \|F\|_*^2 \leq M \|F\|_{L^2(\mathbb{S}^2)}^2, \quad (3)$$

with

$$m = \frac{1}{4} \min_{T \in \mathcal{T}^0} \frac{d_T^2}{a(T)^3}, \quad M = 2 \max_{T \in \mathcal{T}^0} \frac{1}{|d_T|}, \quad d_T = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix},$$

for each triangle T having the vertices $B_i(x_i, y_i, z_i)$, $i = 1, 2, 3$. If we use the relation $|d_T| = 2a(T) \operatorname{dist}(O, T)$, with $\operatorname{dist}(O, T)$ representing the distance from the origin to the plane of the triangle T , then the values m and M become

$$m = \min_{T \in \mathcal{T}^0} \frac{\operatorname{dist}^2(O, T)}{a(T)},$$

$$M = \max_{T \in \mathcal{T}^0} \frac{1}{a(T) \operatorname{dist}(O, T)}.$$

In the following we construct a multiresolution on \mathbb{S}^2 consisting of piecewise constant functions on the triangles of $\mathcal{U}^j = \{U_1^j, U_2^j, \dots, U_{n \cdot 4^j}^j\}$, $j \in \mathbb{N}$.

By definition, a multiresolution of $L^2(\mathbb{S}^2)$ is a sequence of subspaces $\{V^j : j \geq 0\}$ of $L^2(\mathbb{S}^2)$ which satisfies the following properties:

1. $V^j \subseteq V^{j+1}$ for all $j \in \mathbb{N}$.

2. $\operatorname{clos}_{L^2(\mathbb{S}^2)} \bigcup_{j=0}^{\infty} V^j = L^2(\mathbb{S}^2)$.

3. There are index sets $\mathcal{K}_j \subseteq \mathcal{K}_{j+1}$ such that for every level j there exists a Riesz basis $\{\varphi_t^j, t \in \mathcal{K}_j\}$ of the space V^j . This means that there exist constants $0 < c < C < \infty$, independent of the level j , such that

$$c2^{-j} \left\| \left\{ c_t^j \right\}_{t \in \mathcal{K}_j} \right\|_{l_2(\mathcal{K}_j)} \leq \left\| \sum_{t \in \mathcal{K}_j} c_t^j \varphi_t^j \right\|_{L^2(\mathbb{S}^2)} \leq C2^{-j} \left\| \left\{ c_t^j \right\}_{t \in \mathcal{K}_j} \right\|_{l_2(\mathcal{K}_j)}.$$

3 The Spaces V^j and W^j

For a fixed $j \in \mathbb{N}$, to each triangle $U_k^j \in \mathcal{U}^j$, $k = 1, 2, \dots, n \cdot 4^j$, we associate the function $\varphi_{U_k^j} : \mathbb{S}^2 \rightarrow \mathbb{R}$,

$$\varphi_{U_k^j}(\eta) = \begin{cases} 1, & \text{inside the triangle } U_k^j, \\ 1/2, & \text{on the edges of } U_k^j, \\ 0, & \text{elsewhere.} \end{cases}$$

Then we define the spaces of functions $V^j = \operatorname{span} \{\varphi_{U_k^j}, k = 1, 2, \dots, n \cdot 4^j\}$, consisting of piecewise constant functions on the triangles of \mathcal{U}^j .

It follows immediately that the set $\{\varphi_{U_k^j}, k = 1, 2, \dots, n \cdot 4^j\}$ is a basis for V^j , so $|V^j| = n \cdot 4^j$. We must establish the relation between the spaces

V^j and V^{j+1} . Let $U^j \in \mathcal{U}^j$ and $U_k^{j+1} = p(T_k^{j+1})$, $k = 1, 2, 3, 4$, the refined triangles obtained from U^j , as in Fig. 1. We have

$$\varphi_{U^j} = \varphi_{U_1^{j+1}} + \varphi_{U_2^{j+1}} + \varphi_{U_3^{j+1}} + \varphi_{U_4^{j+1}},$$

which holds in $L^2(\mathbb{S}^2)$. Thus, $V^j \subseteq V^{j+1}$ for all $j \in \mathbb{N}$. With respect to the scalar product $\langle \cdot, \cdot \rangle_*$, the spaces V^j and V^{j+1} become Hilbert spaces, with the corresponding norm $\|\cdot\|_* = \langle \cdot, \cdot \rangle_*^{1/2}$.

Next we define the space W^j as the orthogonal complement, with respect to the scalar product $\langle \cdot, \cdot \rangle_*$, of the coarse space V^j in the fine space V^{j+1} :

$$V^{j+1} = V^j \bigoplus W^j.$$

The spaces W^j are called *the wavelet spaces*. The dimension of W^j is $|W^j| = |V^{j+1}| - |V^j| = 3n \cdot 4^j$. In the following we will construct a basis of W^j . Let us take the triangle U^j and its refinements $U_1^{j+1}, U_2^{j+1}, U_3^{j+1}, U_4^{j+1}$ and denote $F_{U^j}^1, F_{U^j}^2, F_{U^j}^3$ the projections onto \mathbb{S}^2 of the mid-points of the edges of the plane triangle $p^{-1}(U^j)$, as in Fig. 1. Note that, except for the case $j = 0$,

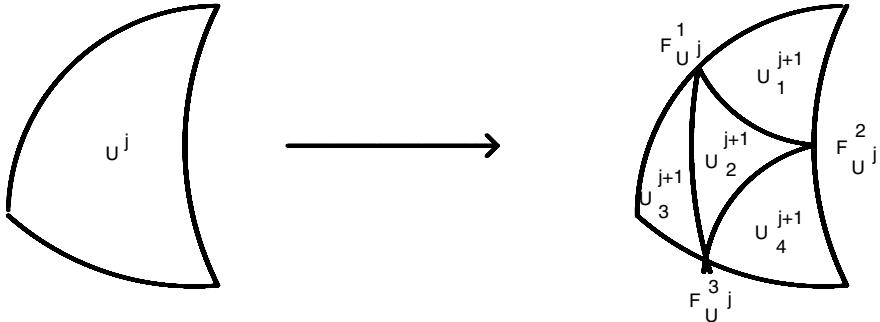


Fig. 1. The triangle U^j and its refined triangles U_k^{j+1} , $k = 1, 2, 3, 4$.

the points $F_{U^j}^l$, $l = 1, 2, 3$, are not in general mid-points of the edges of the spherical triangle U^j . To each of these points $F_{U^j}^l$ a wavelet will be associated in the following way

$$\begin{aligned} \Psi_{F_{j+1}^1, U^j} &= \alpha_1 \varphi_{U_1^{j+1}} + \alpha_2 \varphi_{U_3^{j+1}} + \beta \varphi_{U_2^{j+1}} + \gamma \varphi_{U_4^{j+1}}, \\ \Psi_{F_{j+1}^2, U^j} &= \alpha_1 \varphi_{U_4^{j+1}} + \alpha_2 \varphi_{U_1^{j+1}} + \beta \varphi_{U_2^{j+1}} + \gamma \varphi_{U_3^{j+1}}, \\ \Psi_{F_{j+1}^3, U^j} &= \alpha_1 \varphi_{U_3^{j+1}} + \alpha_2 \varphi_{U_4^{j+1}} + \beta \varphi_{U_2^{j+1}} + \gamma \varphi_{U_1^{j+1}}, \end{aligned} \quad (4)$$

with $\alpha_1, \alpha_2, \beta, \gamma \in \mathbb{R}$. Let us mention that $\text{supp } \Psi_{F_{j+1}^k, U^j} = U^j$ for $k = 1, 2, 3$.

Next we will find conditions on the coefficients $\alpha_1, \alpha_2, \beta, \gamma$, which assure that the set

$$\left\{ \Psi_{F_{j+1}^k, U^j}, k = 1, 2, 3, U^j \in \mathcal{U}^j \right\}$$

is an orthonormal basis of W^j with respect to the scalar product defined in (2). First we must have

$$\left\langle \Psi_{F_{j+1}^k, U^j}, \varphi_{S^j} \right\rangle_* = 0, \quad (5)$$

for $k = 1, 2, 3$ and $U^j, S^j \in \mathcal{U}^j$. If $U^j \neq S^j$, then the equality is immediate since $\text{supp } \Psi_{F_{j+1}^k, U^j} = \text{supp } \varphi_{U^j}$ and $\text{supp } \varphi_{S^j} \cap \text{supp } \varphi_{U^j}$ is either the empty set or an edge, whose measure is zero. For $U^j = S^j$, evaluating the scalar product (5) we obtain

$$\left\langle \Psi_{F_{j+1}^1, U^j}, \varphi_{S^j} \right\rangle_* = \frac{\alpha_1 \mathcal{A}_1^{j+1} + \alpha_2 \mathcal{A}_3^{j+1} + \beta \mathcal{A}_2^{j+1} + \gamma \mathcal{A}_4^{j+1}}{a(p^{-1}(U))},$$

U being the triangle of the initial triangulation \mathcal{U}^0 which includes the triangle U^j and $\mathcal{A}_k^{j+1} = a(p^{-1}(U_k^{j+1}))$. Since

$$\frac{\mathcal{A}_k^{j+1}}{a(p^{-1}(U))} = 4^{-(j+1)} \text{ for } k = 1, 2, 3, 4,$$

the orthogonality conditions (5) reduce to

$$\alpha_1 + \alpha_2 + \beta + \gamma = 0. \quad (6)$$

Now we have to find conditions on the parameters $\alpha_1, \alpha_2, \beta, \gamma$ such that the functions $\left\{ \Psi_{F_{j+1}^k, U^j}, k = 1, 2, 3, U^j \in \mathcal{U}^j \right\}$ are linearly independent. Let $\lambda_{F^1, U^j}, \lambda_{F^2, U^j}, \lambda_{F^3, U^j} \in \mathbb{R}$ for $U^j \in \mathcal{U}^j$. Taking the linear combination

$$\sum_{k=1}^3 \sum_{U^j \in \mathcal{U}^j} \lambda_{F^k, U^j} \Psi_{F_{j+1}^k, U^j} = 0,$$

it follows that for each $U^j \in \mathcal{U}^j$ we must have

$$\sum_{k=1}^3 \lambda_{F^k, U^j} \Psi_{F_{j+1}^k, U^j} = 0. \quad (7)$$

In order to simplify the writing we let $\lambda_k = \lambda_{F^k, U^j}$. Linear independence occurs if each relation (7) implies $\lambda_1 = \lambda_2 = \lambda_3 = 0$. Using the definitions (4) we obtain

$$\begin{aligned} \lambda_1 \alpha_1 + \lambda_2 \alpha_2 + \lambda_3 \gamma &= 0, \\ \lambda_1 \beta + \lambda_2 \beta + \lambda_3 \beta &= 0, \\ \lambda_1 \alpha_2 + \lambda_2 \gamma + \lambda_3 \alpha_1 &= 0, \\ \lambda_1 \gamma + \lambda_2 \alpha_1 + \lambda_3 \alpha_2 &= 0. \end{aligned}$$

Taking into account the condition (6), we can deduce that this system of four equations in three unknowns has only the zero solution if and only if

$$\alpha_1^3 + \alpha_2^3 + \gamma^3 - 3\alpha_1\alpha_2\gamma \neq 0. \quad (8)$$

So, if this condition is satisfied, then a basis in W^j is constructed.

Now we want to look for an orthogonal basis. Each of the orthogonality conditions

$$\left\langle \Psi_{F_{j+1}^k, U^j}, \Psi_{F_{j+1}^l, U^j} \right\rangle_* = 0$$

for $l, k \in \{1, 2, 3\}$, $l \neq k$ and $U^j \in \mathcal{U}^j$ is equivalent to

$$\alpha_1\alpha_2 + (\alpha_1 + \alpha_2)\gamma + \beta^2 = 0. \quad (9)$$

Solving the system consisting of the equations (6) and (9) we get

$$\beta^2 - (\alpha_1 + \alpha_2)\beta - (\alpha_1^2 + \alpha_1\alpha_2 + \alpha_2^2) = 0. \quad (10)$$

We wish to have orthonormal bases, so we impose the condition

$$\left\| 2^j \cdot \Psi_{F_{j+1}^l, U^j} \right\|_* = 1 \text{ for } l = 1, 2, 3.$$

Using the relations (6) and (10) we obtain, for $l = 1, 2, 3$,

$$\left\| 2^j \cdot \Psi_{F_{j+1}^l, U^j} \right\|_* = \alpha_1^2 + \alpha_2^2 + \beta^2 + \gamma^2 = 4\beta^2.$$

Hence, $\beta = \pm \frac{1}{2}$. For $\beta = \frac{1}{2}$ condition (10) reduces to

$$4(\alpha_1^2 + \alpha_1\alpha_2 + \alpha_2^2) + 2(\alpha_1 + \alpha_2) - 1 = 0,$$

and condition (8) reduces to

$$2(\alpha_1^2 + \alpha_1\alpha_2 + \alpha_2^2) + (\alpha_1 + \alpha_2) \neq 0.$$

The small ellipse, having the equation $2(\alpha_1^2 + \alpha_1\alpha_2 + \alpha_2^2) + (\alpha_1 + \alpha_2) = 0$, contains the points (α_1, α_2) for which the wavelets become linearly dependent. In conclusion, there exist orthogonal wavelets for all (α_1, α_2) situated on the big ellipse plotted in Fig. 2. These wavelets have the expression

$$\begin{aligned} {}^1\Psi_{F_{j+1}^1, U^j} &= \alpha_1\varphi_{U_1^{j+1}} + \alpha_2\varphi_{U_3^{j+1}} + \frac{1}{2}\varphi_{U_2^{j+1}} - \left(\frac{1}{2} + \alpha_1 + \alpha_2 \right) \varphi_{U_4^{j+1}}, \\ {}^1\Psi_{F_{j+1}^2, U^j} &= \alpha_1\varphi_{U_4^{j+1}} + \alpha_2\varphi_{U_1^{j+1}} + \frac{1}{2}\varphi_{U_2^{j+1}} - \left(\frac{1}{2} + \alpha_1 + \alpha_2 \right) \varphi_{U_3^{j+1}}, \\ {}^1\Psi_{F_{j+1}^3, U^j} &= \alpha_1\varphi_{U_3^{j+1}} + \alpha_2\varphi_{U_4^{j+1}} + \frac{1}{2}\varphi_{U_2^{j+1}} - \left(\frac{1}{2} + \alpha_1 + \alpha_2 \right) \varphi_{U_1^{j+1}}. \end{aligned}$$

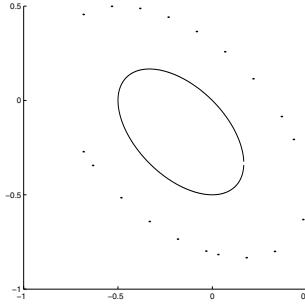


Fig. 2. The graph of the curve $4(\alpha_1^2 + \alpha_1\alpha_2 + \alpha_2^2) + 2(\alpha_1 + \alpha_2) - 1 = 0$ (the big ellipse), resp. $2(\alpha_1^2 + \alpha_1\alpha_2 + \alpha_2^2) + (\alpha_1 + \alpha_2) = 0$ (the small ellipse).

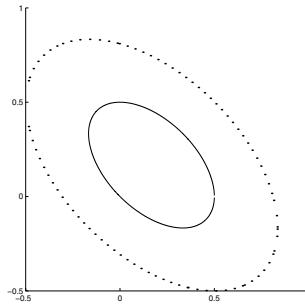


Fig. 3. The graph of the curve $4(\alpha_1^2 + \alpha_1\alpha_2 + \alpha_2^2) - 2(\alpha_1 + \alpha_2) - 1 = 0$ (the big ellipse), resp. $2(\alpha_1^2 + \alpha_1\alpha_2 + \alpha_2^2) - (\alpha_1 + \alpha_2) = 0$ (the small ellipse).

For $\beta = -\frac{1}{2}$ condition (10) reduces to

$$4(\alpha_1^2 + \alpha_1\alpha_2 + \alpha_2^2) - 2(\alpha_1 + \alpha_2) - 1 = 0,$$

while condition (8) reduces to

$$2(\alpha_1^2 + \alpha_1\alpha_2 + \alpha_2^2) - (\alpha_1 + \alpha_2) \neq 0.$$

Again, there exist orthogonal wavelets for all (α_1, α_2) situated on the big ellipse plotted in Fig. 3. These wavelets have the expression

$$\begin{aligned} {}^2\Psi_{F_{j+1}^1, U^j} &= \alpha_1 \varphi_{U_1^{j+1}} + \alpha_2 \varphi_{U_3^{j+1}} - \frac{1}{2} \varphi_{U_2^{j+1}} + \left(\frac{1}{2} - \alpha_1 - \alpha_2 \right) \varphi_{U_4^{j+1}}, \\ {}^2\Psi_{F_{j+1}^2, U^j} &= \alpha_1 \varphi_{U_4^{j+1}} + \alpha_2 \varphi_{U_1^{j+1}} - \frac{1}{2} \varphi_{U_2^{j+1}} + \left(\frac{1}{2} - \alpha_1 - \alpha_2 \right) \varphi_{U_3^{j+1}}, \\ {}^2\Psi_{F_{j+1}^3, U^j} &= \alpha_1 \varphi_{U_3^{j+1}} + \alpha_2 \varphi_{U_4^{j+1}} - \frac{1}{2} \varphi_{U_2^{j+1}} + \left(\frac{1}{2} - \alpha_1 - \alpha_2 \right) \varphi_{U_1^{j+1}}. \end{aligned}$$

We note that if we choose $\alpha_1 = \alpha_2 = \alpha$, then we obtain the families of wavelets $\{{}^1\Psi_{F_{j+1}^l, U^j}^1\}$, $\{{}^1\Psi_{F_{j+1}^l, U^j}^2\}$, $\{{}^2\Psi_{F_{j+1}^l, U^j}^1\}$ and $\{{}^2\Psi_{F_{j+1}^l, U^j}^2\}$, given by

$$\begin{aligned} {}^1\Psi_{F_{j+1}^l, U^j}^1 &= -\frac{1}{2} \left(\varphi_{U_1^{j+1}} + \varphi_{U_3^{j+1}} - \varphi_{U_2^{j+1}} - \varphi_{U_4^{j+1}} \right), \\ {}^1\Psi_{F_{j+1}^l, U^j}^2 &= \frac{1}{6} \left(\varphi_{U_1^{j+1}} + \varphi_{U_3^{j+1}} + 3\varphi_{U_2^{j+1}} - 5\varphi_{U_4^{j+1}} \right), \\ {}^2\Psi_{F_{j+1}^l, U^j}^1 &= \frac{1}{2} \left(\varphi_{U_1^{j+1}} + \varphi_{U_3^{j+1}} - \varphi_{U_2^{j+1}} - \varphi_{U_4^{j+1}} \right), \\ {}^2\Psi_{F_{j+1}^l, U^j}^2 &= -\frac{1}{6} \left(\varphi_{U_1^{j+1}} + \varphi_{U_3^{j+1}} + 3\varphi_{U_2^{j+1}} - 5\varphi_{U_4^{j+1}} \right), \end{aligned}$$

for $l = 1$ and similarly for $l = 2, 3$. These wavelets are exactly the wavelets obtained in [2], in the case when the spherical areas are approximated with the plane areas.

4 The Stability of the Bases

To be useful in practice, the wavelets must satisfy the Riesz stability conditions. Next we prove the Riesz stability of the bases that we have constructed in V^j and W^j , for arbitrary $j \in \mathbb{N}$.

First we check Condition 3 of the definition of multiresolution. The basis $\{2^j \varphi_{U_k^j}, k = 1, 2, \dots, n \cdot 4^j\}$ of V^j is orthonormal since

$$\left\| 2^j \varphi_{U_k^j} \right\|_*^2 = 4^j \left\langle \varphi_{U_k^j}, \varphi_{U_k^j} \right\rangle_* = 4^j \cdot \frac{a(p^{-1}(U_k^j))}{a(p^{-1}(U))} = 1$$

and $\left\langle 2^j \varphi_{U_k^j}, 2^j \varphi_{U_l^j} \right\rangle_* = 0$ for $k \neq l$ because the intersection of their supports is either empty or an edge, which has measure zero.

Being an orthonormal basis with respect to the inner product $\langle \cdot, \cdot \rangle_*$, the following equality holds

$$\left\| \sum_{U^j \in \mathcal{U}^j} c_U^j 2^j \varphi_{U^j} \right\|_* = \left\| \left\{ c_U^j \right\}_{U \in \mathcal{U}^j} \right\|_{l^2}.$$

Using now the equality (3), which expresses the equivalence of the norms $\|\cdot\|_*$ and $\|\cdot\|_{L^2(\mathbb{S}^2)}$, we get

$$\frac{1}{M} \left\| \left\{ c_U^j \right\}_{U \in \mathcal{U}^j} \right\|_{l^2}^2 \leq \left\| \sum_{U^j \in \mathcal{U}^j} c_U^j 2^j \varphi_{U^j} \right\|_{L^2(\mathbb{S}^2)}^2 \leq \frac{1}{m} \left\| \left\{ c_U^j \right\}_{U \in \mathcal{U}^j} \right\|_{l^2}^2,$$

which is exactly Condition 3 of the definition of a multiresolution. Using the same arguments for the wavelets bases

$$\left\{ 2^j {}^i \Psi_{F_{j+1}^l, U^j}^k \right\}_{l=1,2,3, U^j \in \mathcal{U}^j}, \quad i = 1, 2, \quad k = 1, 2,$$

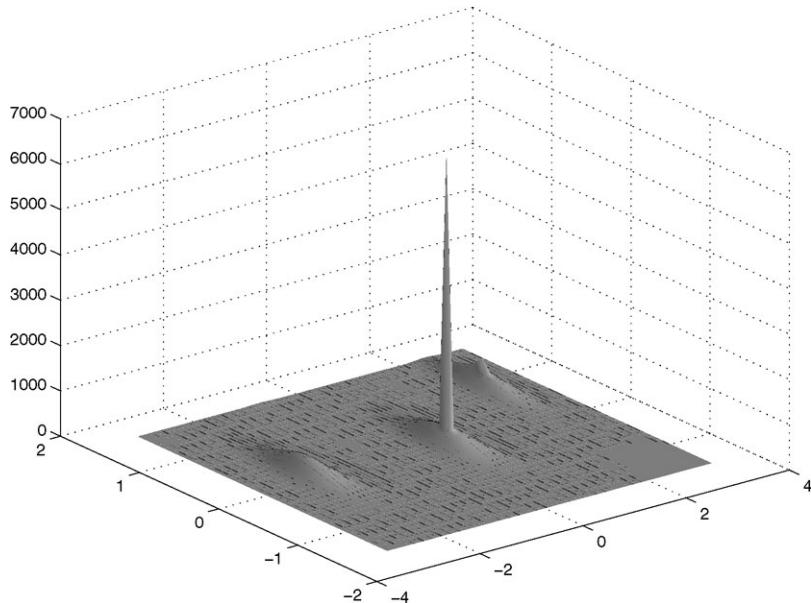


Fig. 4. The initial data set *pol5*.

we can prove that

$$\frac{1}{M} \left(\sum_{l=1}^3 \sum_{U \in \mathcal{U}^j} d_{l,U^j} \right)^2 \leq \left\| \sum_{U^j \in \mathcal{U}^j} d_{l,U^j} 2^{j \frac{l}{m}} \Psi_{F_{j+1}^l, U^j}^k \right\|_{L^2(\mathbb{S}^2)}^2 \leq \frac{1}{m} \left(\sum_{l=1}^3 \sum_{U \in \mathcal{U}^j} d_{l,U^j} \right)^2.$$

Some evaluations of the number $\kappa = \sqrt{M/m}$ for some particular polyhedra show that κ is $3^{3/2} = 5.19615\dots$ for the regular tetrahedron, $3^{3/4} = 2.27951\dots$ for the cube and regular octahedron and $(15/(5 + 2\sqrt{5}))^{3/4} = 1.41167\dots$ for the regular dodecahedron and regular icosahedron. However, the number κ is not significant for the performance of the wavelets, since the matrices involved in the decomposition and reconstruction algorithms are orthogonal.

5 Numerical Tests

In order to illustrate our prewavelets, we took as the initial polyhedron Π an octahedron with six vertices and we performed five levels of decomposition. At level five, the total number of triangles is 8196. Then we considered a particular data set *pol5* from texture analysis of crystals (cf. [4]) and we represented it as shown in Fig. 4. It consists of 36×72 measurements on the sphere at the points

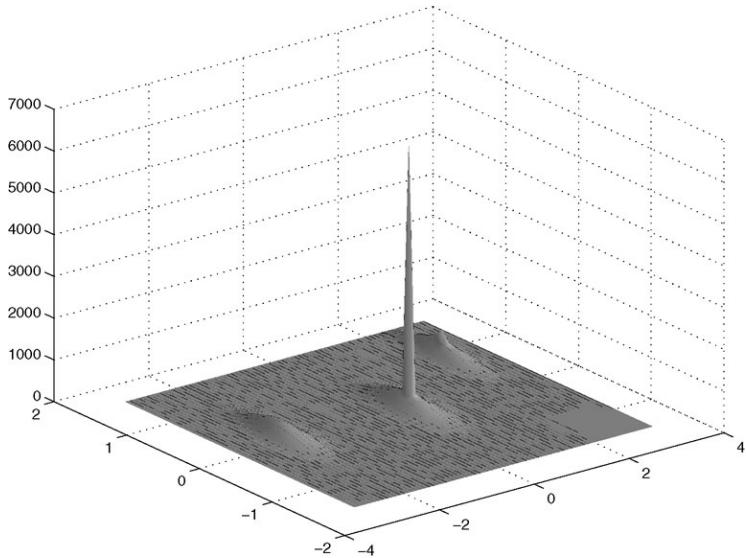


Fig. 5. The function $f^5 \in V^5$, approximation of $pol5$ at level 5.

$$\{P_{ij} (\cos \theta_j \sin \rho_i, \sin \theta_j \sin \rho_i, \cos \rho_i)\},$$

with $\theta_j = \frac{\pi j}{36} - \frac{\pi}{72}$, $j = 1, \dots, 72$, $\rho_i = \frac{\pi i}{36} - \frac{\pi}{72}$, $i = 1, \dots, 36$. Its main characteristic is that the values over the whole sphere are constant, except for some peaks. First we have approximated this data with the function $f^5 \in V^5$ (see Fig. 5), considering $pol5$ as a piecewise constant function on the set

$$\{p(\mathcal{Q}_{ij}), i = 1, \dots, 36, j = 1, \dots, 72\},$$

where p is the projection defined in (1) and \mathcal{Q}_{ij} are quadrants with centres at P_{ij} and edge $\pi/72$. The approximation error

$$e = \frac{1}{36 \cdot 72} \sum_{i=1}^{36} \sum_{j=1}^{72} |f^5(i, j) - pol5(i, j)|$$

was 1.0984. Since the set $\{\varphi_t^j\}_{t \in \mathcal{U}^j}$ is a basis for V^j , for $j = 0, 1, 2, \dots$, we can write

$$f^5(\eta) = \sum_{t \in \mathcal{U}^5} f_t^5 \varphi_t^5(\eta), \quad \eta \in \mathbb{S}^2. \quad (11)$$

The vector $\mathbf{f}^5 = (f_t^5)_{t \in \mathcal{U}^5}$ associated with the function f^5 was then decomposed into \mathbf{f}^0 and $\mathbf{g}^0, \mathbf{g}^1, \mathbf{g}^2, \mathbf{g}^3, \mathbf{g}^4$, using the wavelet with coefficients $(\alpha_1, \alpha_2, \beta, \gamma) = (\frac{1}{6}, \frac{1}{6}, \frac{3}{6}, -\frac{5}{6})$. The detail coefficients \mathbf{g}^j , $j = 0, \dots, 4$, were

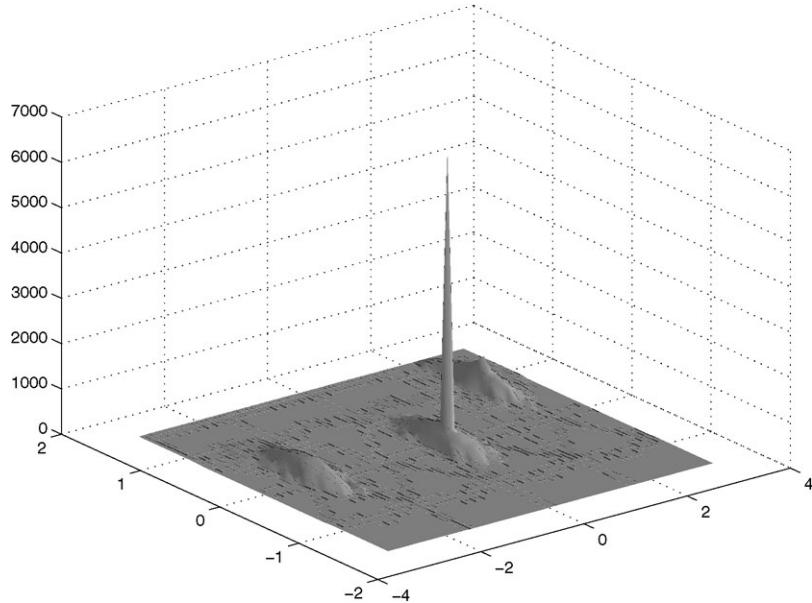


Fig. 6. The reconstructed function \hat{f}^5 for the compression rate 0.05.

thresholded to obtain a specific compression rate. More precisely, their components $(g_k^j)_{k=1,\dots,3n \cdot 4^j}$ were replaced by the values $(\hat{g}_k^j)_{k=1,\dots,3n \cdot 4^j}$ according to a strategy known as *hard thresholding*. This consists of choosing a threshold $\varepsilon > 0$ and then setting

$$\hat{g}_k^j = \begin{cases} g_k^j, & \text{if } |g_k^j| \geq \varepsilon, \\ 0, & \text{otherwise.} \end{cases}$$

The ratio of the number of subsequent non-zero coefficients to the total number,

$$\frac{\sum_{j=0}^4 |\{k : \hat{g}_k^j \neq 0\}|}{3n \cdot 4^j},$$

will be referred to as the *compression rate*.

After the compression we performed the reconstruction, yielding an approximation with error \mathbf{e}^5 , $\mathbf{e}^5 = \mathbf{f}^5 - \hat{\mathbf{f}}^5$, where $\hat{\mathbf{f}}^5 = (\hat{f}_t^5)_{t \in \mathcal{U}^5}$ is the vector associated with the reconstructed function \hat{f}^5 . We have measured this error in several ways:

- the maximum error given by

$$\|\mathbf{e}^5\|_\infty = \max_{\eta \in \mathbb{S}^2} |\mathbf{e}^5(\eta)| = \max_{t \in \mathcal{U}^5} |\mathbf{e}_t^5|;$$

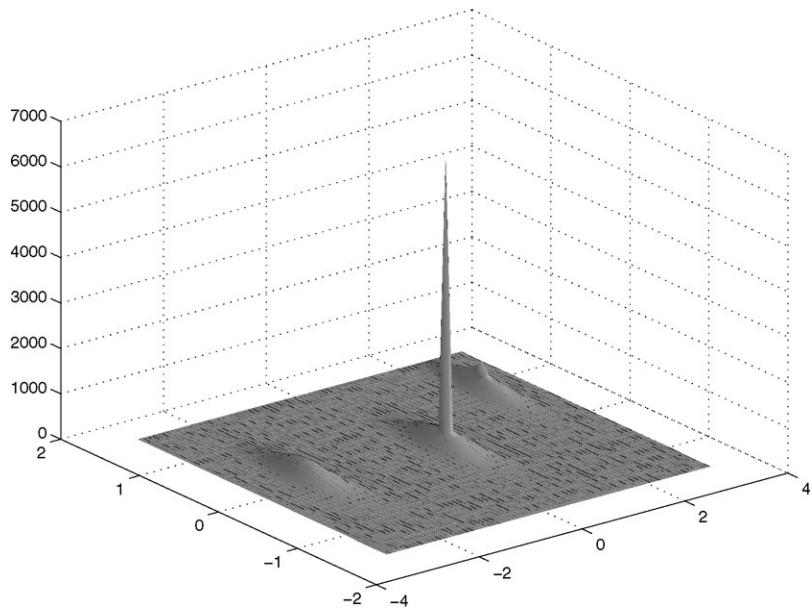


Fig. 7. The reconstructed function \hat{f}^5 for the compression rate 0.5.

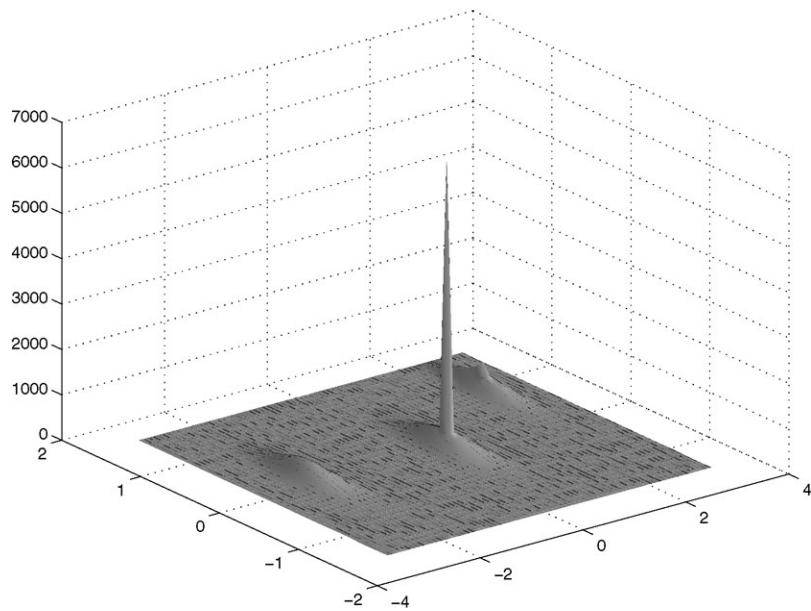


Fig. 8. The reconstructed function \hat{f}^5 for the compression rate 0.75.

- the 2-norm

$$\|\mathbf{e}^5\|_2 = \left(\sum_{t \in \mathcal{U}^5} |f_t^5 - \hat{f}_t^5|^2 \right)^{1/2};$$

- the mean absolute error over the triangles

$$\text{mean}(\mathbf{e}^5) = \frac{1}{n \cdot 4^j} \sum_{t \in \mathcal{U}^5} |\mathbf{e}_t^5|.$$

Figs. 6, 7 and 8 show the reconstructed functions \hat{f}^5 for different compression rates, and the errors are tabulated in Table 1.

Table 1. Reconstruction errors for some compression rates, with the wavelet $\frac{1}{6}[1, 1, 3, -5]$.

| comp. rate | no. of zero coeff. | $\ \mathbf{e}^5\ _\infty$ | $\ \mathbf{e}^5\ _2$ | mean(\mathbf{e}^5) |
|---------------|-----------------------|---------------------------|----------------------|------------------------|
| 0.05 | 7775 | 165.75 | 3122.10 | 29.40 |
| 0.1 | 7366 | 114.48 | 2715.90 | 25.13 |
| 0.25 | 6139 | 78.41 | 1855.40 | 15.48 |
| 0.5 | 4099 | 35.17 | 764.91 | 6.40 |
| 0.75 | 2047 | 19.24 | 242.26 | 1.53 |
| 0.8 | 1637 | 4.11 | 88.99 | 0.55 |
| 0.84 | 1228 | 0 | 0 | 0 |

Acknowledgement

This work was supported in part by the European Union research project “Multiresolution in Geometric Modelling (MINGLE)” under grant HPRN-CT-1999-00117.

References

1. Bonneau, G-P.: Optimal Triangular Haar Bases for Spherical Data. In: IEEE Visualization '99, San Francisco, USA (1999).
2. Nielson, G., Jung, I., Sung, J.: Haar Wavelets over Triangular Domains with Applications to Multiresolution Models for Flow over a Sphere. In: IEEE Visualization '97, 143-150 (1997).
3. Roșca, D.: Locally Supported Rational Spline Wavelets on the Sphere, submitted.
4. Schaeben, H., Potts, D., Prestin, J.: Spherical Wavelets with Application in Preferred Crystallographic Orientation, IAMG '2001, Cancun (2001).

Author Index

| | | | |
|-------------|---------------|-------------|--------------------|
| Alkalai | 231 | Iske | 319 |
| Alliez | 3 | Ivrißimtzis | 285 |
| Andujar | 339 | Kobbelt | 49, 245 |
| Barthe | 245, 259 | Levin | 301 |
| Belyaev | 143 | Magillo | 89, 101 |
| Beutel | 391 | Marinov | 301 |
| Brunet | 339 | Mesmoudi | 75 |
| Cebollada | 339 | Morando | 75 |
| Danovaro | 89 | Ohtake | 143 |
| De Floriani | 49, 75, 89 | Praun | 27 |
| Demaret | 319 | Prestin | 369 |
| Dodgson | 259, 271, 285 | Puppo | 49, 75, 101 |
| Dyn | 231, 301, 319 | Quak | 369 |
| Elber | 119 | Rössl | 187, 353 |
| Fairen | 339 | Roşca | 405 |
| Floater | 157, 319 | Sabin | 203, 245, 259, 285 |
| Gérot | 245, 259 | Seidel | 143, 187, 353 |
| Gotsman | 3 | Sokolovsky | 89 |
| Hahmann | 119 | Theisel | 353 |
| Hassan | 271 | Viaña | 101 |
| Hoppe | 27 | Zayer | 187 |
| Hormann | 157 | | |

Colour Plates

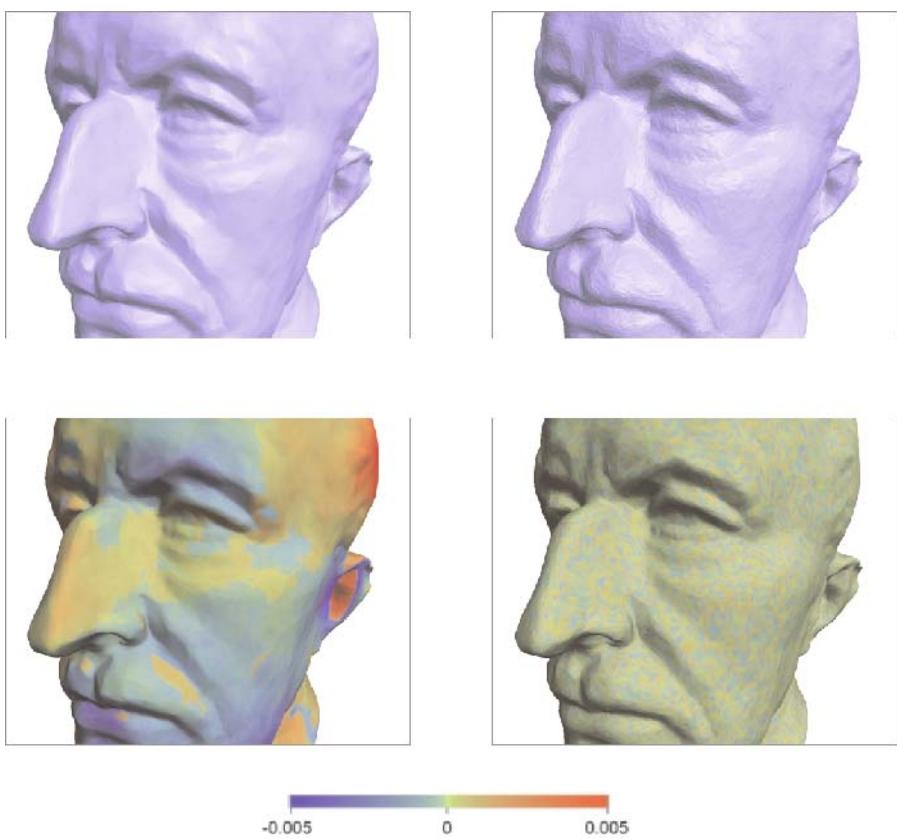


Plate 1. [Fig. 3 on p. 10.] The delta-coordinate quantisation to 5 bits/coordinate (left) introduces low-frequency errors to the geometry, whereas Cartesian coordinate quantisation to 11 bits/coordinate (right) introduces noticeable high-frequency errors. The upper rows shows the quantised model and the bottom figures use colour to visualise corresponding quantisation errors. Data courtesy O. Sorkine.

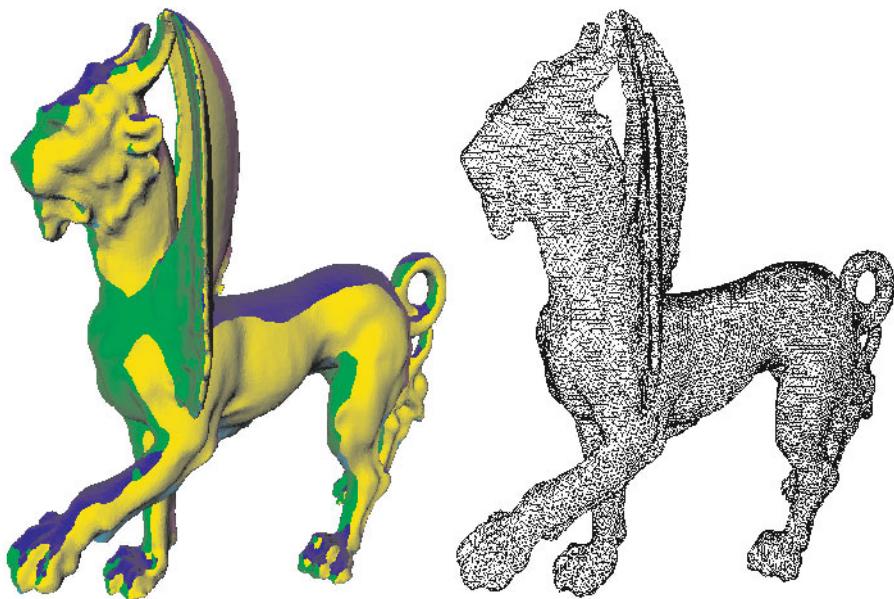


Plate 2. [Fig. 4 on p. 13.] Piecewise regular remeshing (data courtesy A. Szymczak).

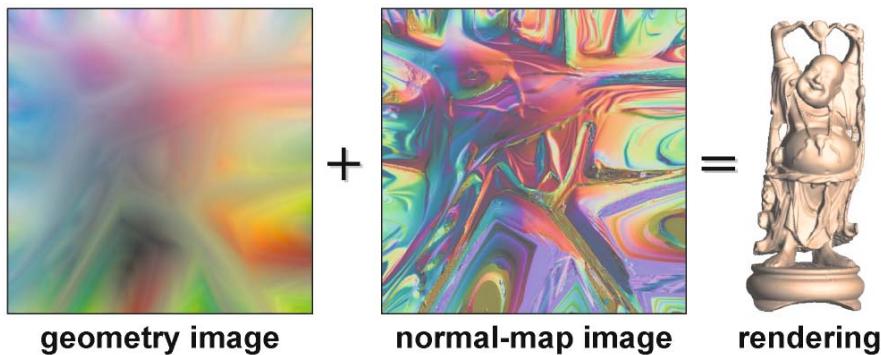


Plate 3. [Fig. 6 on p. 15.] Geometry image (data courtesy X. Gu).

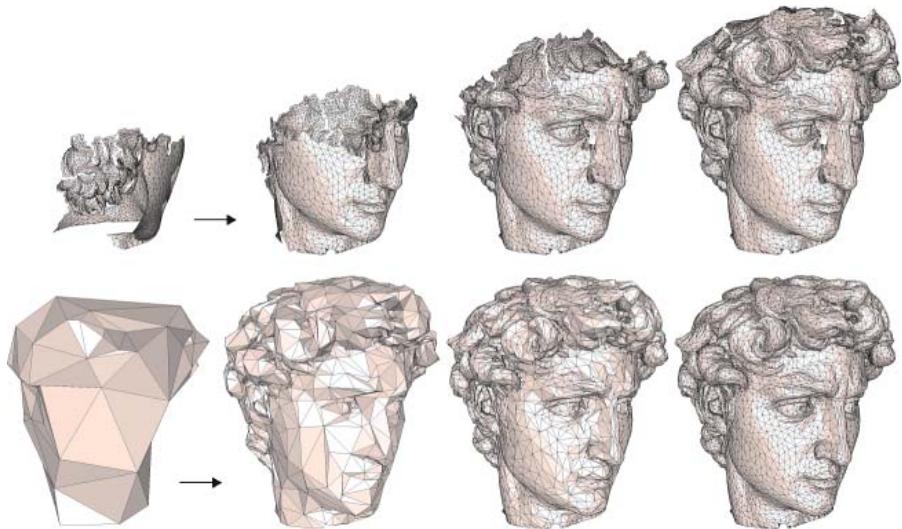


Plate 4. [Fig. 7 on p. 17.] Intermediate stages during the decoding of a mesh using a single-rate (top) or a progressive technique (bottom).

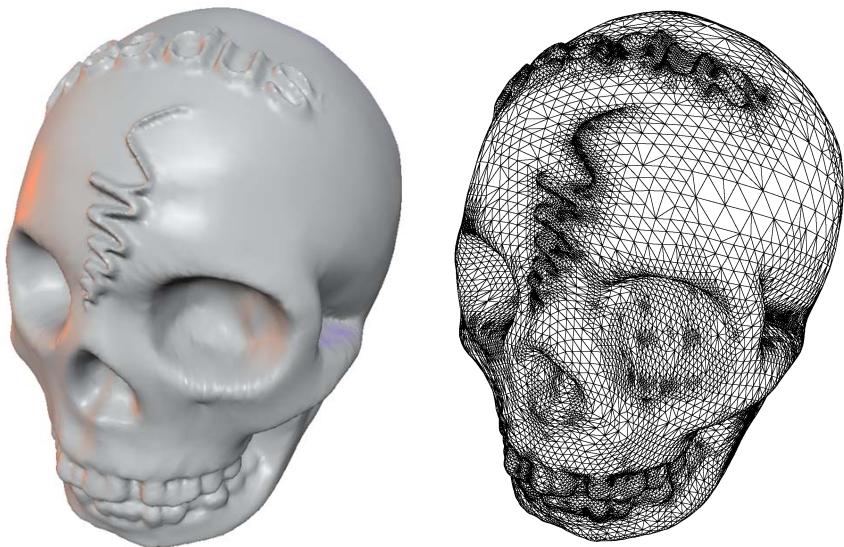


Plate 5. [Fig. 9 on p. 21.] Adaptive normal mesh for the skull model (data courtesy A.Khodakovsky).



Map of original mesh onto sphere, octahedron domain, and image.

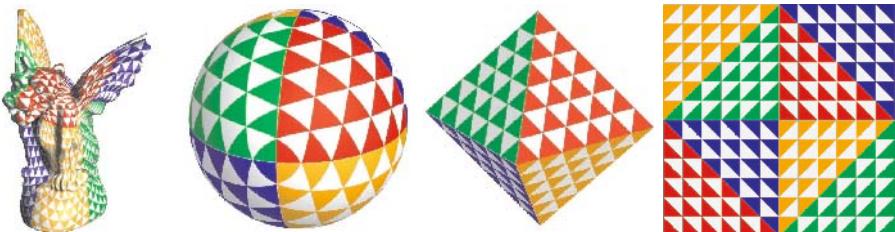


Illustration of the same map using image grid samples.

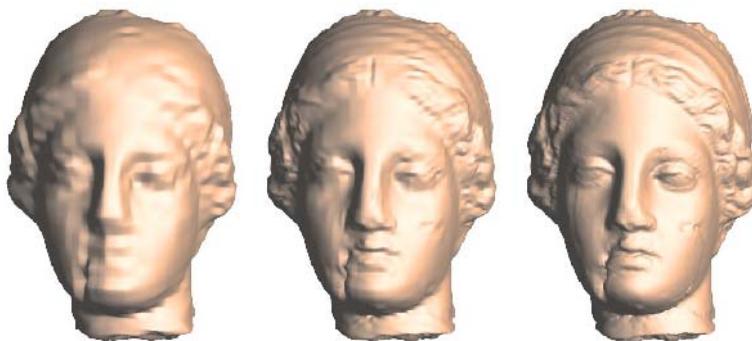
Map of original mesh onto sphere, *flat* octahedron domain, and image.

Illustration of the same map using image grid samples.

Plate 6. [Fig. 1 on p. 30.] Illustration of remeshing onto octahedron and flat octahedron domains.



1,445 bytes (61.6 dB) 2,949 bytes (67.0 dB) 11,958 bytes (75.7 dB)
Compression using spherical wavelets.



1,357 bytes (60.8 dB) 2,879 bytes (66.5 dB) 11,908 bytes (77.6 dB)
Compression using image wavelets.

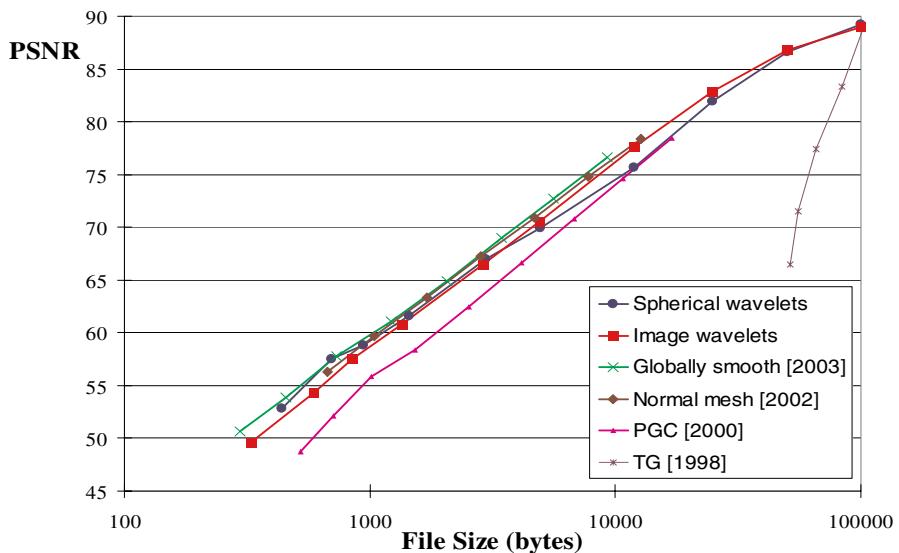


Plate 7. [Fig. 9 on p. 40.] Compression results on Venus model.



1,444 bytes (59.6 dB) 2,951 bytes (65.5 dB) 11,959 bytes (76.1 dB)
Compression using spherical wavelets.



1,367 bytes (60.5 dB) 2,889 bytes (66.2 dB) 11,915 bytes (77.5 dB)
Compression using image wavelets.

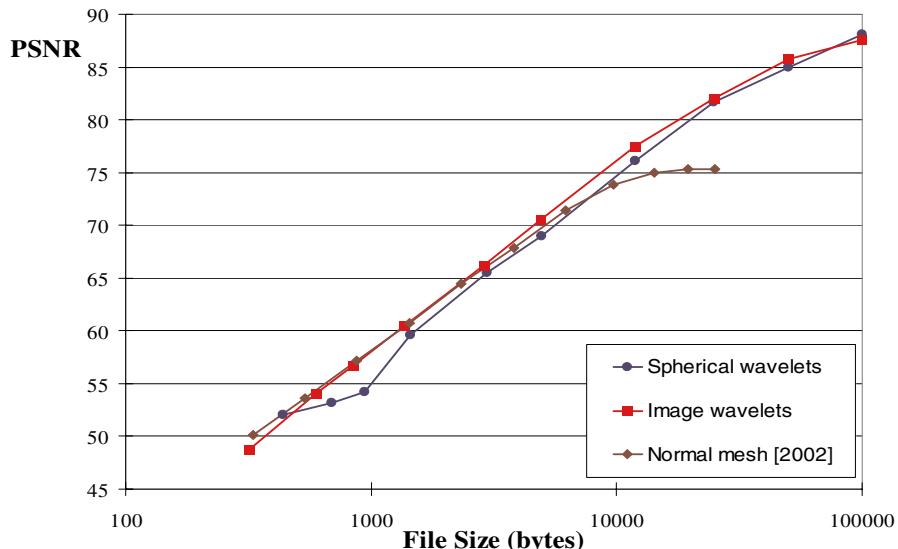


Plate 8. [Fig. 10 on p. 41.] Compression results on skull model.

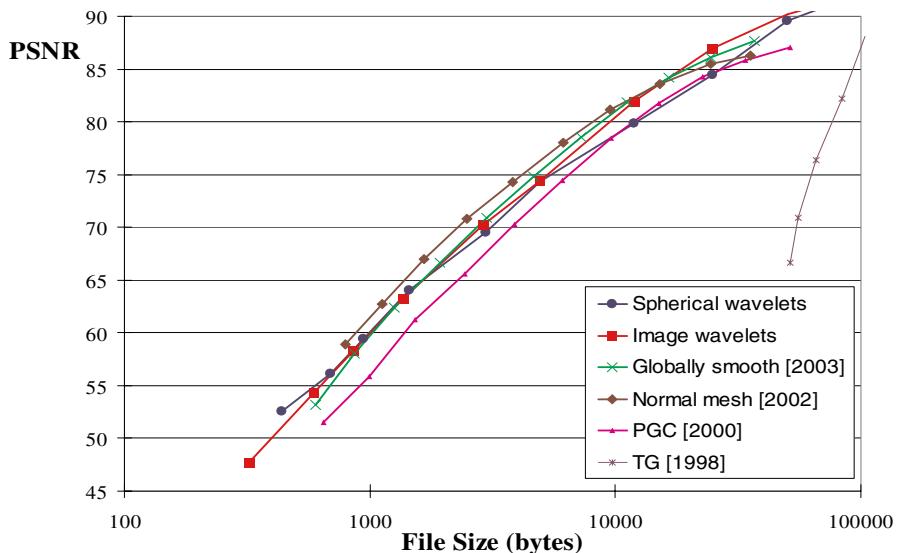
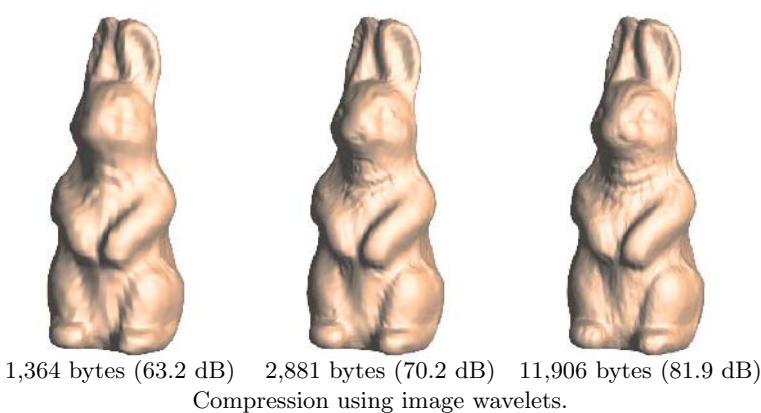
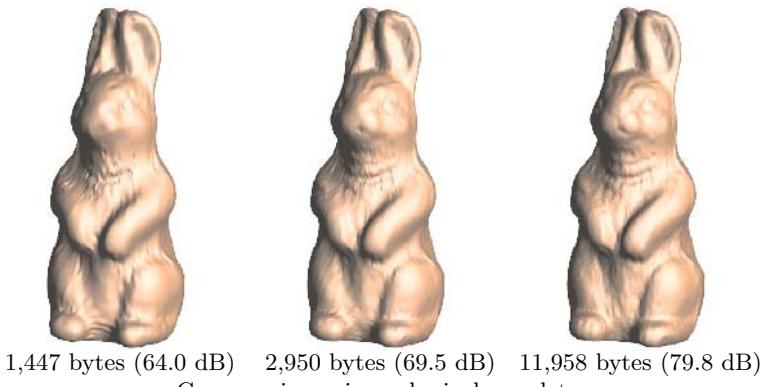
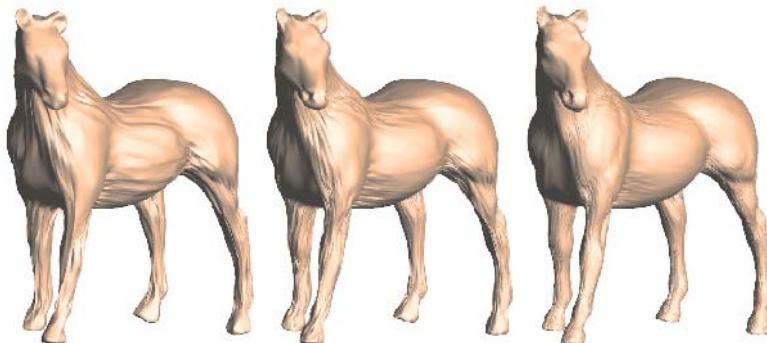
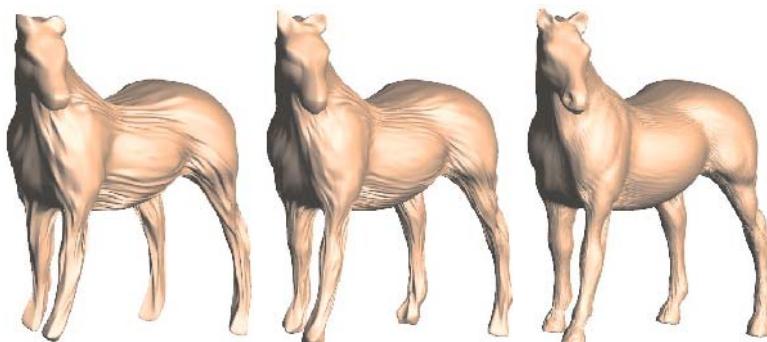


Plate 9. [Fig. 11 on p. 42.] Compression results on rabbit model.



1,456 bytes (55.7 dB) 2,961 bytes (57.6 dB) 11,968 bytes (69.7 dB)
Compression using spherical wavelets.



1,376 bytes (54.2 dB) 2,900 bytes (60.6 dB) 11,932 bytes (73.1 dB)
Compression using image wavelets.

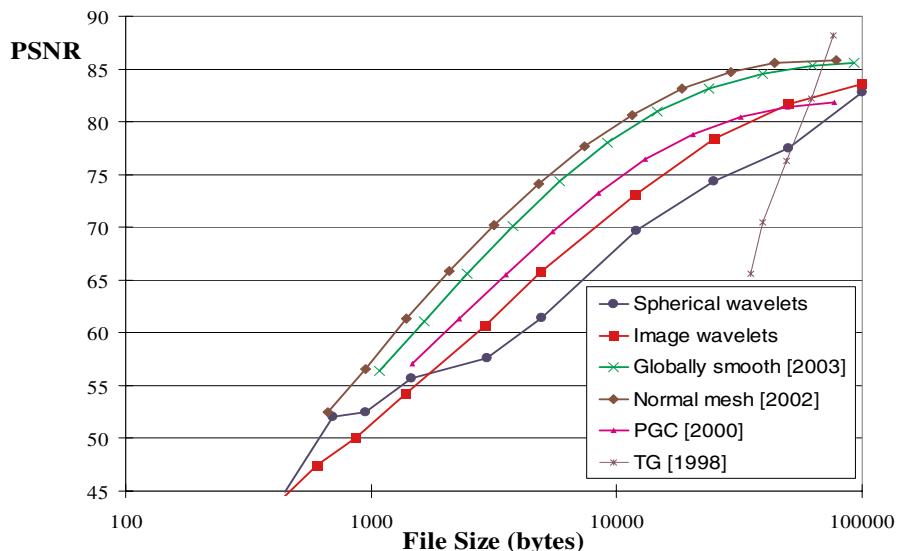


Plate 10. [Fig. 12 on p. 43.] Compression results on horse model.

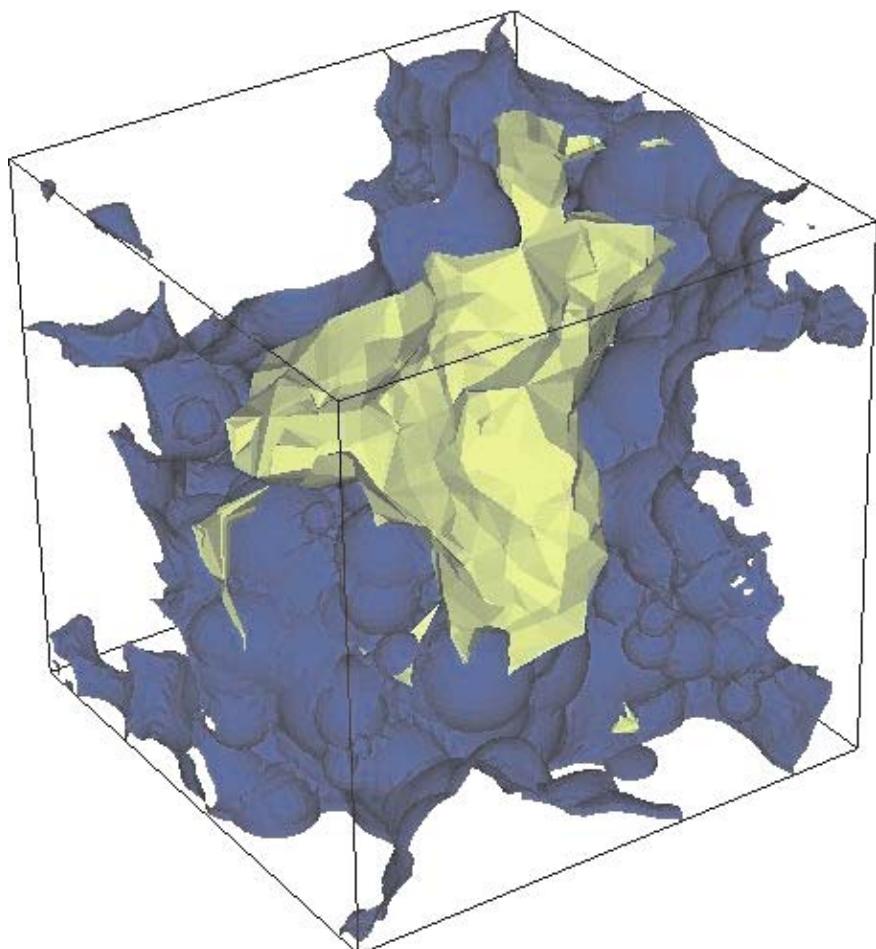


Plate 11. [Fig. 1 on p. 90.] Variable LOD based on field values: the isosurface with a field value equal to 1.27 (shown in blue) is extracted in high LOD. The second isosurface, with a field value equal to 1.45 (shown in green), illustrates the lower resolution of the mesh.

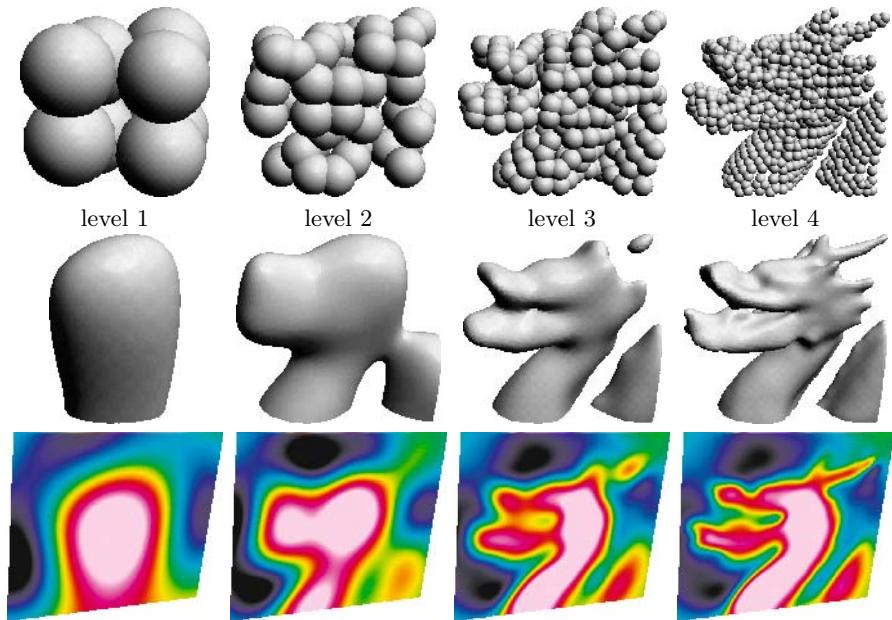


Plate 12. [Fig. 1 on p. 146.] Multi-scale interpolation of the Stanford dragon model. Top row: four first levels of the multi-scale hierarchy of points; the radii of the spheres at each level of the hierarchy are proportional to the support size of the RBFs used for the interpolation at that level. Middle row: zero level-sets of the interpolating functions. Bottom row: cross-sections of the interpolating functions.

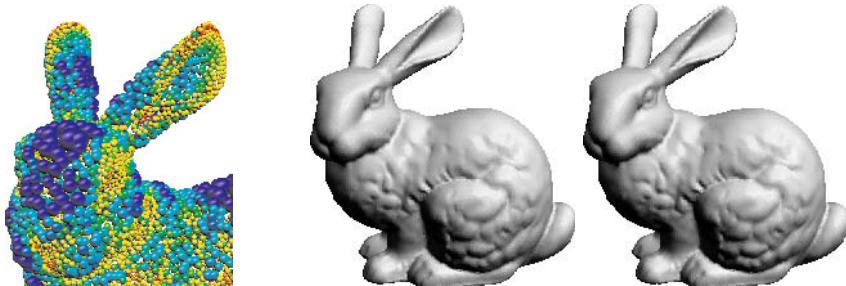


Plate 13. [Fig. 5 on p. 151.] Adaptive approximation of the Stanford bunny model with $\varepsilon_0 = 2.5 \times 10^{-4}$. Right: for $T = 1.5$ each approximation centre is visualised by a sphere of radius $\sigma_k/4$; the spheres are coloured according to their sizes which increases from red to blue. Middle: 8,504 RBF centres (and local approximations) are used if $T = 1.5$; L^2 error = 1.86×10^{-4} and L^∞ error = 2.76×10^{-3} ; computational time is 7 seconds. Right: 20,813 RBF centres are used if $T = 5$; L^2 error = 1.72×10^{-4} and L^∞ error = 2.57×10^{-3} ; computational time is 19 seconds.

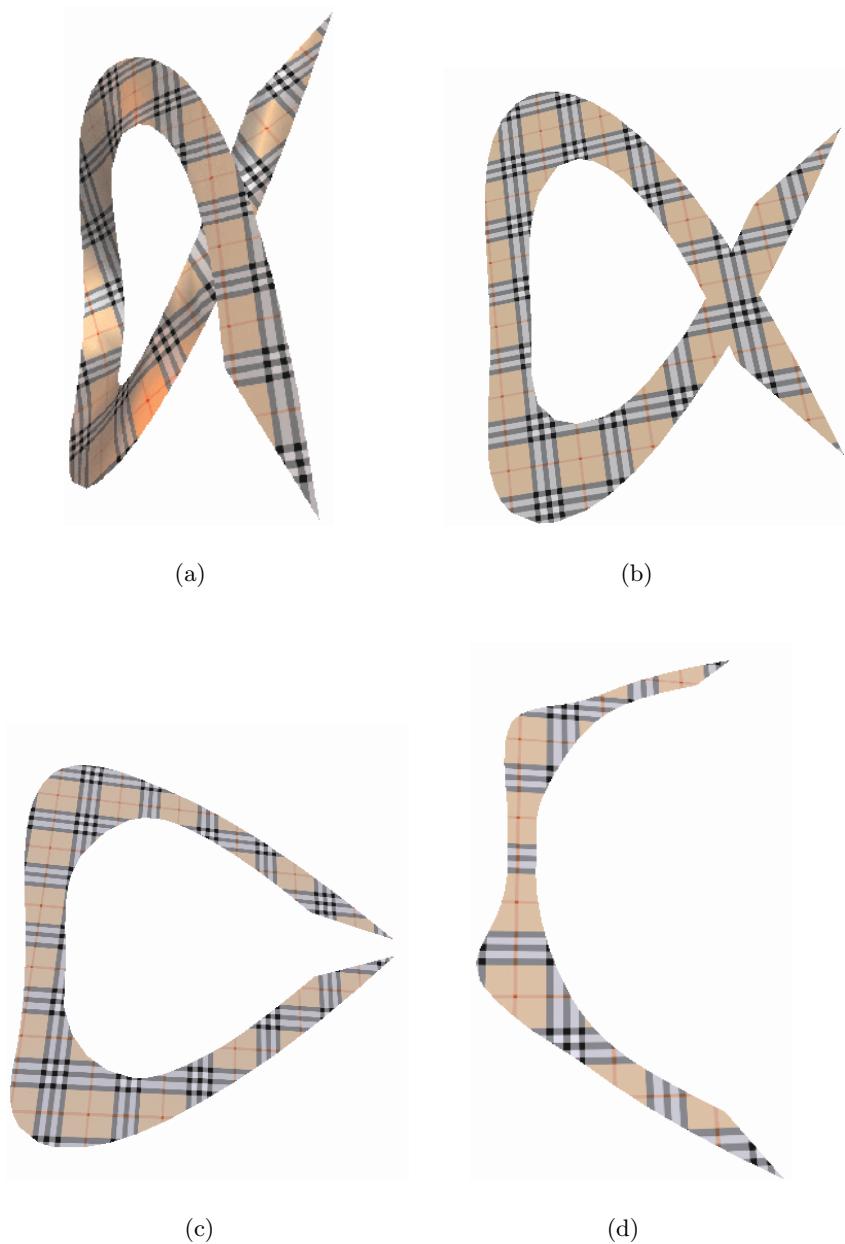


Plate 14. [Fig. 1 on p. 191.] Flattening an α -shaped model: (a) Original mesh. (b) The flattened mesh with boundary control coefficient $t = 2$, (c) $t = 1.1$, (d) $t = 1.03$. (The views are scaled differently.)

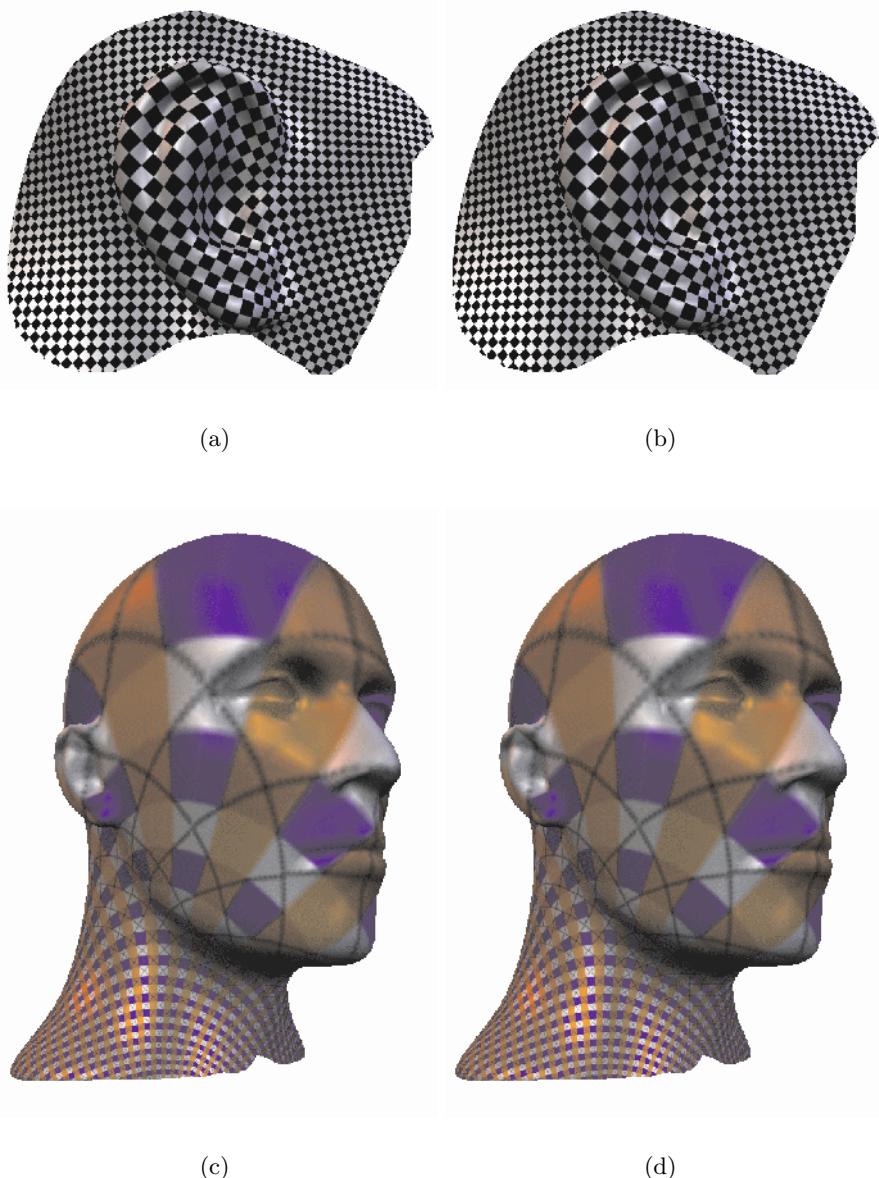
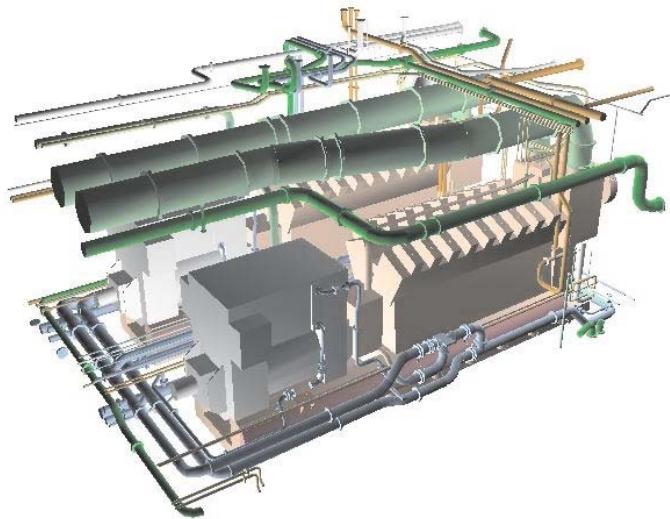


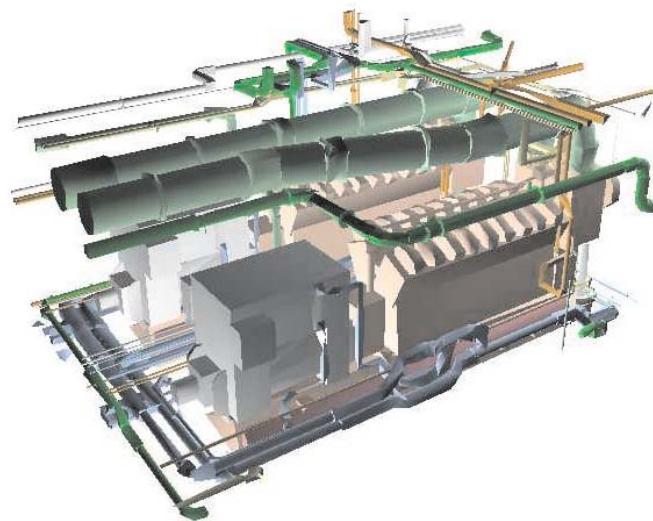
Plate 15. [Fig. 4 on p. 195.] Comparison of under-determined (a, c) and minimisation (b, d) solutions. The parameterization of the *Ear* and the *Mannequin* models is visualised by mapping regular textures.



Plate 16. [Fig. 5 on p. 196.] Visualisation of parameterizations from ABF by texture mapping different models. (a) *Clumpy*, (b) *Large ear*, (c) *Mechanical part*. Notice the quasi-conformality of the parameterization.

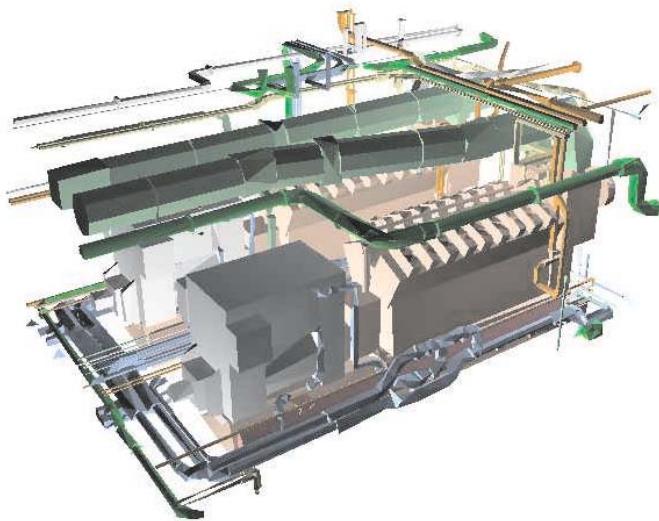


(a) Original model

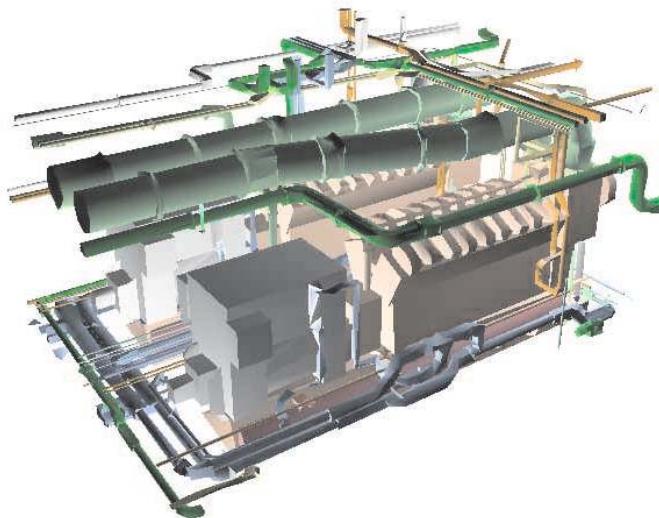


(b) Simplified with our method

Plate 17. [Figs. 5(a) and (b) on p. 348.] Results on the engine model.

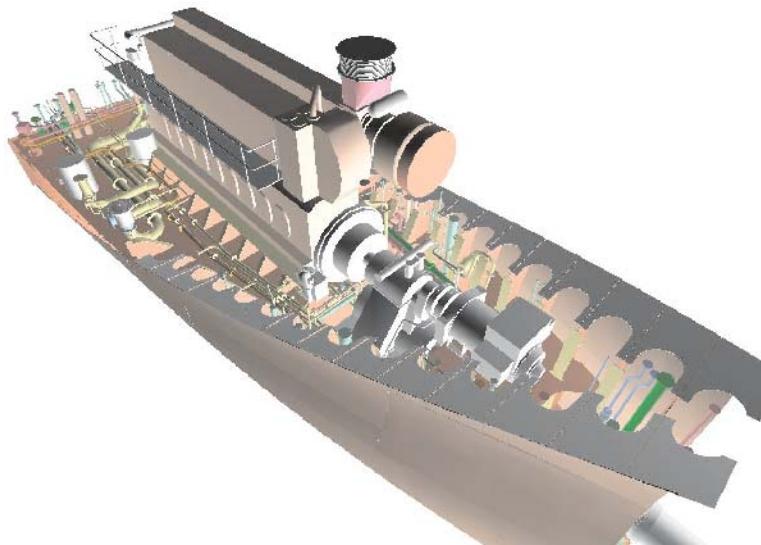


(a) With colour separation at the beginning

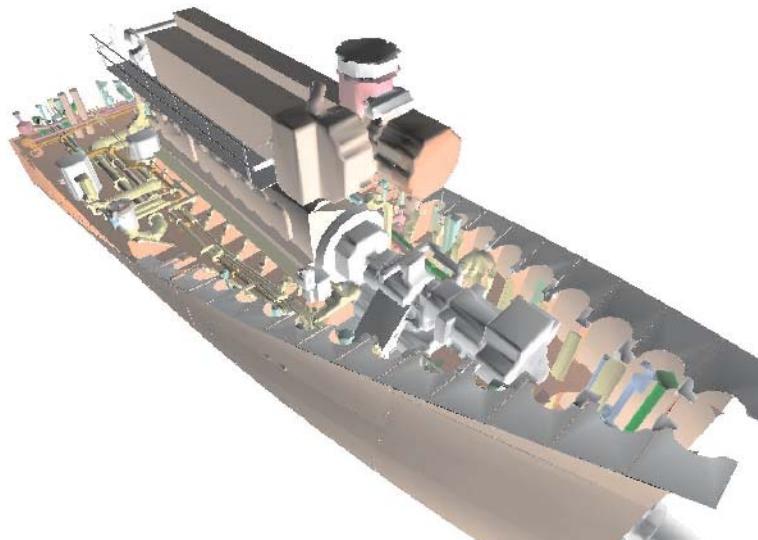


(b) Without colour separation

Plate 18. [Figs. 5(c) and (d) on p. 348.] Results on the engine model. N.B. (a) in this plate is Fig. 5(c), (b) in this plate is Fig. 5(d) on p. 348.



(a) Original image



(b) Simplified with our method

Plate 19. [Fig. 6 on p. 349.] Results on the oil tanker model.