

THE UNIVERSITY OF CALGARY

Constraining Wavelets for Multiresolution

by

Luke Jonathan Olsen

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

August, 2006

© Luke Jonathan Olsen 2006

THE UNIVERSITY OF CALGARY
FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled “Constraining Wavelets for Multiresolution” submitted by Luke Jonathan Olsen in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.

Dr. Faramarz Samavati
Department of Computer Science

Dr. Mario Costa Sousa
Department of Computer Science

Dr. Przemyslaw Prusinkiewicz.
Department of Computer Science

Dr. Laleh Behjat.
Department of Electrical and Computer Engineering

Date

Abstract

Multiresolution is a modeling technique in computer graphics for representing a model at different scales. Many graphical models do not result from a multiresolution process, and so a fundamental problem is to construct a multiresolution representation for such models. Subdivision is a method for increasing the resolution of a model and is used in the pipeline of many modeling tools. Thus there is an abundance of models that have the structure, or connectivity, created by subdivision.

In this thesis, we develop a framework for constructing multiresolution representations for models with subdivision connectivity. To have low-resolution representations that require no more storage than the original model, the model is partitioned into two independent sets: one that represents the coarse approximation, and one which will contain some additional information for reconstructing the original model. Then, a *wavelet constraint* is enforced so that the *entire* set of reconstruction information can be determined from the incomplete stored information. This constraint results in a set of rules for decomposing the model. To improve the quality of the low-resolution model, an optimization step is applied.

This framework is generic enough to be applied to a wide range of subdivision types. To demonstrate this applicability, it is used to develop multiresolution settings for several subdivision types: cubic B-spline curves, Loop surfaces, and Catmull-Clark surfaces. The resulting systems have desirable properties such as biorthogonality and compact support. Furthermore, the cubic B-splines and Loop systems are found to perform as well as existing systems, while the Catmull-Clark system represents the first true multiresolution system for that type of surface.

Acknowledgments

First and foremost, I would like to thank Dr. Faramarz Samavati. Without his influence, I wouldn't even be in a Masters program. Thanks for your excellent teaching skills that provided me with a solid graphics background in the undergrad program, which led to pursuing graphics as a research topic, for the idea that this thesis is based on, for all the guidance and support over the last two years, and for making my graduate experience so enjoyable and enriching.

Thanks also to Dr. Mario Costa Sousa, for accepting me for co-supervision; for providing very interesting and educational course experiences; for all the feedback on this work as well as the sketch-based work we did; and for keeping a light and fun atmosphere around the jungle.

I wouldn't be here without the help of two great professors I had during undergrad. Most recently, Dr. Jeff Boyd gave me an opportunity to do summer research under his tutelage, which led to me applying for the scholarship that ultimately persuaded me to pursue a Masters. I wish I could have done a Masters in vision as well! Thanks for all the support, encouragement, and apparently flattering reference letters. And taking a trip on the way-back machine, I owe a great debt to Con Ferris, the best instructor I've had in computer science, who is presumably still blessing Red Deer College with his talents.

I probably would have gone insane over the past two years, and past two months in particular, without a couple of great friends who provided the "play" in the work-play dichotomy... you know who you are! Oh, you need to see your name in print? Jeez, egotistical! Okay, huge thanks to Gabriel Becerra for, well, just being a great

friend, always willing to listen to even the most esoteric and ridiculous research and life problems. And thanks to Maxwell Sayles for all the fascinating conversations, helpful and often wacky advice, and always providing a laugh. You better go back to school though; your brain is rotting. I would throw an acknowledgment to Trevor Arsenault as well, but I kinda forget what he looks like at this point.

The most enjoyable part of grad studies so far has been the incredibly cool people that populate the graphics jungle. I've made a lot of great friends who have been invaluable rubber ducks, ski buddies, comic relief, and hockey-rant suppliers. So, huge thanks to all the junglers, in particular to Mahsa Eshraghi, Lakin Wecker, Adam Runions, John Brosz, Joseph Cherlin, Aaron Severn, Mitra Shirmomohomadi, Erwin de Groot, Mikolaj Cieslak, Fabricio Anastacio, and Robbie Timmer.

And finally, I couldn't have done it without the support of my family, especially my parents Everett and Sherry, and brothers Tuson and Jared. Thanks for all the support and encouragement, and also for stressing the importance of balance in life. That concept went out the window in summer aught-six, but I didn't forget about it!

Table of Contents

Approval Page	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	vi
1 Introduction	1
1.1 Motivation	3
1.2 Goals	6
1.3 Methodology	7
1.4 Contributions	9
1.5 Overview of Thesis	10
2 Background	11
2.1 Parametric Curves	11
2.2 Subdivision	13
2.2.1 Curves	13
2.2.2 Patches	17
2.2.3 Arbitrary-Connectivity Meshes	19
2.3 Multiresolution (MR)	23
2.3.1 Mesh Simplification & Remeshing	24
2.3.2 Wavelets for MR	27
2.3.3 Applications	38
2.4 Conclusion	43
3 Method Overview	45
4 Cubic B-spline Curves	51
4.1 Subdivision Filter	52
4.2 Trial Filters	53
4.3 Boundary Filters	55
4.4 Refinement	60
4.4.1 Width-3 Neighborhood	62
4.4.2 Width-5 Neighborhood	65
4.5 Partial Refinement	66
4.6 Closed-Form Filters	70

4.6.1	Width-7 Filter	72
4.6.2	Width-11 Filter	74
4.7	Conclusion	76
5	Loop Surfaces	78
5.1	Subdivision Filter	78
5.2	Trial Filters	80
5.3	Refinement	82
5.4	Partial Refinement	85
5.5	Boundary Filters	86
5.6	Closed Form	87
5.7	Conclusion	89
6	Catmull-Clark Surfaces	91
6.1	Subdivision Filter	92
6.2	Trial Filters	96
6.2.1	Valence-3 Vertices	98
6.3	Refinement	102
6.3.1	Valence-3 Vertices	104
6.4	Partial Refinement	104
6.5	Boundary Filters	105
6.6	Conclusion	107
7	Results	108
7.1	Cubic B-Splines	109
7.1.1	Analysis	110
7.1.2	Conclusion	117
7.2	Loop Surfaces	120
7.2.1	Analysis	121
7.2.2	Conclusion	131
7.3	Catmull-Clark Surfaces	131
7.3.1	Analysis	132
7.3.2	Conclusion	140
8	Conclusions & Future Work	142
	Bibliography	146
A	Notation	153

B	System Interface	158
B.1	Cubic B-Spline Curves & Patches	158
B.2	Loop & Catmull-Clark Surfaces	159
C	Implementation Details	166
C.1	Cubic B-Spline Curves & Patches	166
C.2	Loop & Catmull-Clark Surfaces	166
C.2.1	Class Structure	167
C.2.2	Half-Edge Data Structure	171
C.2.3	Semi-Regular Mesh Tagging	172

List of Figures

1.1	<i>Left:</i> A polygonal mesh represents an object as a set of vertices and a set of faces connecting the vertices. <i>Right:</i> Subdivision is a technique for smoothing such meshes.	2
1.2	Multiresolution editing in a commercial modeling application (ZBrush). (left to right) features on the high-resolution mesh are represented as offsets from a low-resolution base mesh; changes to the base mesh are reflected properly at the higher resolution. (Figure taken from ZBrush Quick Reference Guide, available at http://www.pixologic.com/zbrush/education/education-documentation.html .)	4
1.3	Multiresolution mesh editing: (a) the original mesh; (b) a lower resolution reached via multiresolution techniques; (c) a simple editing operation is applied at the low resolution; (d) the low-resolution edit is reflected when the original resolution is restored.	5
1.4	Multiresolution editing of a curve: (a) the original curve; (b) a low resolution approximation; (c) the low-resolution version is edited; (d) the edit is carried forward when the original resolution is restored.	6
1.5	The invertibility condition: in a multiresolution system, subdivision followed by decomposition should yield the original object.	8
1.6	The reconstructability condition: a multiresolution system should capture the high-frequency information that is lost by decomposition, to allow full reconstruction of the original surface.	8
1.7	The similitude condition: a multiresolution system should try to produce a low-resolution mesh that resembles the original object. The object in the center is decomposed with the Loop filters described in Sec. 5.2 (left) and 5.4 (right).	8
2.1	A parametric curve is defined by a small set of control vertices: (left) a curve defined by five vertices; moving a control vertex (center) creates an intuitive change in the resulting curve (right).	12
2.2	A cubic B-spline tensor product surface is defined by a regular grid of control vertices: (left) a control mesh; (right) the resulting surface.	13
2.3	Chaikin subdivision (left to right): the original control polygon; after one subdivision; after two subdivisions; the limit surface.	14
2.4	Cubic B-spline subdivision (left to right): the original control polygon; after one subdivision; after two subdivisions; the limit surface.	14

2.5	Open versus closed curves: for a closed curve (left), periodic indexing is used to “wrap” the control vertices around; for an open curve (right), the curve has definite start and end points, and the subdivision filters must be altered to provide endpoint interpolation.	16
2.6	A subdivision patch is, like a tensor product surface, defined by a regular grid of control vertices. The same control mesh of Fig. 2.2 (left) is subdivided first along the rows (middle) and then along the columns (right).	18
2.7	The regular arrangement in a quadrilateral mesh (left) yields two independent axes along which curve subdivision techniques can be used. A regular triangle mesh (right) has three dependent axes.	19
2.8	Mesh subdivision schemes allow any arbitrary polygonal object to be refined to a smooth surface. (Loop subdivision is used here.)	20
2.9	Doo-Sabin subdivision of a cube.	21
2.10	Catmull-Clark subdivision of a cube.	21
2.11	Loop subdivision of a cube.	21
2.12	Hoppe et al.’s approach to mesh simplification (left) is to minimize an energy function on the mesh by simple edge-based operations (right). (Figures taken from [22].)	25
2.13	The remeshing approach of Eck et al.: starting with an arbitrary mesh (top left) a base mesh (bottom center) is constructed and then subdivided to an approximation of the original mesh (bottom right). (Figures taken from [17].)	26
2.14	The Haar scaling functions for the function space V^2	31
2.15	The Haar scaling functions and wavelets for the function space V^1	31
2.16	Finkelstein and Salesin introduced feature transfer for MR curves. The characteristics of the curve in (a) are extracted and then applied to a different base curve (c), producing a new curve with the same characteristics (d). (Figure taken from [18].)	39
2.17	The cut-and-paste feature transfer system of Biermann et al. is a powerful application of MR to mesh editing. (Figure taken from [6].)	40
2.18	Zorin et al. developed a MR mesh editing system to facilitate both large-scale (the shrinking of the belly) and small-scale (the removal of the bellybutton). (Figure taken from [49].)	41
2.19	Lee et al. constructing equivalent base domains for each input mesh (left) to morph between them (right). (Figures taken from [29].)	43
2.20	View-dependent rendering of a progressive mesh: the original mesh (left) is rendered at full resolution only within a viewing frustum (center and right). (Figure taken from [21].)	44

3.1	Subdivision creates a natural vertex partitioning for MR: (left) an arbitrary mesh contains only “even” vertices; (center) after subdivision, even vertices are moved and “odd” vertices are created; (right) when the object is later decomposed, it is natural to replace even vertices with a coarse approximation, and odd vertices with a detail term. . .	46
3.2	The wavelet constraint: when an object (top) is decomposed, details are stored only at the odd vertices (bottom). To ensure reconstructability, a constraint is enforced such that the missing detail d_0^{k-1} can be computed from stored details d_{-1}^{k-1} and d_1^{k-1}	47
4.1	Local notation for cubic B-spline subdivision curves is centered about representative coarse vertex \tilde{c}_0 and fine vertex f_0 . During decomposition, even vertices (f_{-2} , f_0 , and f_2) are replaced with coarse vertices, while odd vertices f_{-1} and f_1 are replaced with details d_{-1} and d_1 . The wavelet constraint allows d_0 to be computed from d_{-1} and d_1 . . .	52
4.2	The boundary setting for cubic B-splines. Coarse vertices c_0, \dots, c_3 contribute to special boundary vertices f_0, \dots, f_4 during subdivision, which is different than the even-odd rhythm in the regular case. During decomposition, f_0 , f_1 , and f_3 are considered even, and f_2 is considered odd.	57
4.3	While the trial decomposition filter satisfies biorthogonality, successive applications of the filter causes high error in the coarse approximations. A vase object (left) is decomposed with the trial filter (left to right) once, twice, and three times, and then subdivided to the original’s resolution. After three levels of decomposition, the error has spiraled out of control.	61
4.4	Refinement of the trial vertices: (a) fine data \mathbf{F} ; (b) decomposition of \mathbf{F} produces $\tilde{\mathbf{C}}$ and \mathbf{D} ; (c) a refinement vector δ_0 is computed from a local set of details; (d) after refinement, the local error about \tilde{c}_0 is reduced.	64
4.5	Refining the trial filter (left) by Eqn. 4.19 (center) or Eqn. 4.21 (right) reduces the error introduced by decomposition.	65
4.6	Local vs. global effects of refinement (the dashed line represents the original fine data): if only the central vertex \tilde{c}_0 is displaced by δ_0 (a), then after refinement, the local error $E(\mathbf{PC}')$ is minimized (b); but, if all coarse vertices are refined (c), the non-independence of the refinements yields a non-minimal error $E(\mathbf{PC})$ (d).	67
4.7	Partial refinement softens each local refinement to account for the interdependence of each refinement.	69

5.1	Local notation for Loop subdivision is centered about representative coarse vertex c_0 and fine vertex f_0 . During decomposition, even vertex f_0 is replaced with coarse vertex \tilde{c}_0 , while odd vertices $f_{i \neq 0}$ are replaced with details.	79
5.2	The trial decomposition filter produces high error, especially under repeated applications. A simple model (left) is decomposed (left to right) once, twice, and three times, then subdivided without details to the original resolution.	83
5.3	Refinement improves the trial filter displacing each coarse vertex according to the direction and magnitude of the surrounding details. The original model \mathbf{C}^k of Fig. 5.2 is decomposed and then subdivided once (left), twice (center), and three times (right). However, because of the interdependence of the refinements, the error is actually higher than the trial filter error.	84
5.4	Partial refinement improves the trial filter even more by accounting for the interdependence of each local refinement.	86
5.5	Boundary vertices and edges in a Loop model can be decomposed with cubic B-spline filters.	86
5.6	The multiresolution framework with a refinement step included. After decomposing \mathbf{F} with $\tilde{\mathbf{A}}$, the refinement vectors Δ are added to produce \mathbf{C} . When reconstructing, the refinement vectors are first subtracted.	88
6.1	Catmull-Clark subdivision splits k -sided faces into k quadrilateral faces.	91
6.2	In Catmull-Clark subdivision, mesh \mathbf{C}^k (left) is subdivided to \mathbf{C}^{k+1} (right). \mathbf{C}^{k+1} has vertices corresponding to every vertex (yellow), edge (red), and face (blue).	92
6.3	The notation for the local neighborhood of a representative vertex-vertex v^k . At level k , the 1-ring of v^k consists of edge neighbors e_i^k and face neighbors $f_{i,j}^k$. At level $k+1$, the 1-ring of v^{k+1} contains edge neighbors e_i^{k+1} and face neighbors f_i^{k+1}	93
6.4	The trial filter produces high-error coarse approximations, especially under repeated applications. (left to right) the original model; after one decomposition with Eqn. 6.8; after two decompositions; after three decompositions.	98
6.5	The trial filter of Eqn. 6.8 fails on valence-3 vertices. However, decomposition can be performed in cases where the valence-3 vertex $v_{n=3}^k$ has a non-valence-3 neighbor $\tilde{v}_{n \neq 3}^k$ at the coarse level.	99

6.6	When a valence-3 vertex v^{k+1} has more than one coarse neighbor that has been decomposed, each neighbor nominates a position for \tilde{v}^k and \tilde{v}^k 's final position is taken as the average of the candidates: (a) a portion of a fine mesh around v^{k+1} ; (b) v^{k+1} 's coarse neighbors can be decomposed; (c) each neighbor nominates a position for \tilde{v}^k ; (d) the final position of \tilde{v}^k is the average of the candidates.	101
6.7	A refinement step is used to reduce the error of the trial filter.	104
6.8	Partial refinement reduces the per-vertex displacement to account for the interdependence of the refinements. Visually, the results are similar to the full refinement, but the error measurement shows improvement.	105
6.9	Boundary vertices and edges in a Catmull-Clark model can be decomposed with cubic B-spline filters.	106
7.1	Original fine data \mathbf{C}^k for evaluating the cubic B-spline MR system of Chapter 4.	111
7.2	A car model is decomposed two (left column) and three (right column) levels and then subdivided the same number of times.	112
7.3	A wolf model is decomposed one (left column) and two (right column) levels and then subdivided the same number of times.	113
7.4	A face model is decomposed one (left column), two (middle), and three (right) levels and then subdivided the same number of times.	114
7.5	A tree model is decomposed one (left column) and two (right column) levels and then subdivided the same number of times.	115
7.6	A terrain patch is decomposed one (left column), two (middle), and three (right) levels and then subdivided the same number of times.	116
7.7	A plot of least-squares error for the car model of Fig. 7.2.	117
7.8	A plot of least-squares error for the wolf model of Fig. 7.3.	118
7.9	A plot of least-squares error for the face model of Fig. 7.4.	118
7.10	A plot of least-squares error for the tree model of Fig. 7.5.	119
7.11	A plot of least-squares error for the terrain model of Fig. 7.6.	119
7.12	Notation for the refinement mask of Li et al. [31]. Our refinement step is based on the edge details d_i^e , while their refinement also takes the siding details d_i^s into account.	121
7.13	Fine data \mathbf{C}^k for testing the Loop MR system of Chapter 5.	122
7.14	The alien model of Fig. 7.13 is decomposed several times in succession.	123
7.15	The pawn model of Fig. 7.13 is decomposed several times in succession.	125
7.16	The hand model of Fig. 7.13 is decomposed several times in succession.	126
7.17	The wolf model of Fig. 7.13 is decomposed several times in succession.	127
7.18	A plot of least-squares error for the alien model of Fig. 7.14.	128

7.19	A plot of least-squares error for the pawn model of Fig. 7.15.	128
7.20	A plot of least-squares error for the hand model of Fig. 7.16.	129
7.21	A plot of least-squares error for the wolf model of Fig. 7.17.	129
7.22	A plot of least-squares error for a vase model.	130
7.23	A plot of least-squares error for a head model.	130
7.24	Original fine data \mathbf{C}^k for evaluating the Catmull-Clark MR system of Chapter 6.	133
7.25	A semi-regular Catmull-Clark muffin model is decomposed with the trial and refined filters.	134
7.26	A plot of least-squares error for the muffin model of Fig. 7.25.	134
7.27	A semi-regular Catmull-Clark donut model is decomposed with the trial and refined filters.	135
7.28	A plot of least-squares error for the donut model of Fig. 7.27.	135
7.29	A semi-regular Catmull-Clark bullet model is decomposed with the trial and refined filters.	136
7.30	A plot of least-squares error for the bullet model of Fig. 7.29.	136
7.31	A semi-regular Catmull-Clark seat model is decomposed with the trial and refined filters.	137
7.32	A plot of least-squares error for the seat model of Fig. 7.31.	138
7.33	A semi-regular Catmull-Clark dog model is decomposed with the trial and refined filters.	138
7.34	A plot of least-squares error for the dog model of Fig. 7.33.	139
7.35	A plot of least-squares error for a teddy bear model.	139
7.36	Terrain model decomposed with the trial and partially-refined Catmull-Clark decomposition filters.	140
7.37	A plot of least-squares error for the terrain model of Fig. 7.36.	141
B.1	The main window of the cubic B-spline curve editor.	159
B.2	System window.	160
B.3	Four different rendering modes are available: wireframe, outline, flat shaded, and smooth shaded.	161
B.4	Three multiresolution schemes were implemented: Catmull-Clark, (Chapter 6), Loop (Chapter 5), and LQS Loop (from Li et al. [31]).	162
B.5	Several options are available to control the behavior of each multiresolution system.	162
B.6	Editing by translation: the selected vertices are moved in a plane perpendicular to the camera direction.	164
B.7	Scaling: the selected vertices are scaled uniformly about a user-defined origin point. A visual guide is shown to indicate the amount of scaling being applied.	164

B.8	When rotating, the selected vertices are rotated about an axis pointing out of the screen, centered at the user-defined origin point. A helpful user interface helps the user to control the amount of rotation.	165
C.1	Class diagram.	168
C.2	The half-edge data structure. Each edge is split into two half-edges. A half-edge (red) stores pointers (green) to the next half-edge in the face, the opposite half-edge, and optionally to the previous half-edge. Each vertex (yellow) contains a pointer to some half-edge rooted at it. A face (blue) also requires only a single pointer to a half-edge in the face.	171
C.3	Vertex traversal in a half-edge structure: starting from any half-edge h rooted at v , the neighbor vertex is given by $h->next->vertex$. To get to the next neighbor, h can be replaced by $h->opposite->next$. . .	173
C.4	A face can be traversed starting from h by following the <i>next</i> pointers until h is reached again.	173
C.5	Vertex tagging for Loop surfaces: (a) The algorithm is initialized with a known v -vertex (or a guess); (b) each neighbor of a v -vertex is an edge vertex; (c) each edge vertex has six neighbors, in the sequence $v-e-e-v-e-e$; (d) a correct tagging is reached after visiting all vertices.	177
C.6	Vertex tagging precedes decomposition: (a) The algorithm is initialized with a known v -vertex (or a guess); (b) each neighbor of a v -vertex is an edge vertex; (c) each edge vertex has four neighbors, in the sequence $v-f-v-f$; (d) after visiting all vertices, we have a correct tagging.	178

Chapter 1

Introduction

Multiresolution modeling is an important tool in computer graphics. In the broadest sense, multiresolution can refer to any technique that distinguishes between features at different scales. By adding or removing features from these scales, the amount of perceived detail or quality of the model can be manipulated. Such techniques are important because they respect the macro- and microscopic nature of objects in the real world. For example, the handle of an ax held by a character in an interactive video game need only represent the general shape and can hint at surface properties through texturing; the same ax handle in a motion picture will have to stand up to much more scrutiny, and should carry more fine-scale geometric features.

Polygonal meshes, such as the one shown in Fig. 1.1(a), are the most common representations of rigid objects in computer graphics. A mesh is made up of *vertices*, *faces*, and *edges*; *valence* refers to either the number of edges incident to a vertex, or the number of vertices in a face. Vertices and faces alone are enough for representing and rendering a mesh, but the concept of edges is necessary when discussing subdivision. Subdivision is a linear-time technique for smoothing such object representations (Fig. 1.1(b)); subdivision increases the smoothness of a mesh by inserting new vertices and displacing old vertices.

Due to the popularity of meshes and subdivision techniques for them, mesh-centric multiresolution techniques are intensely researched [49, 38, 7, 31, 4, 28]. There are three broad classes of meshes: meshes with regularly arranged vertices, those with

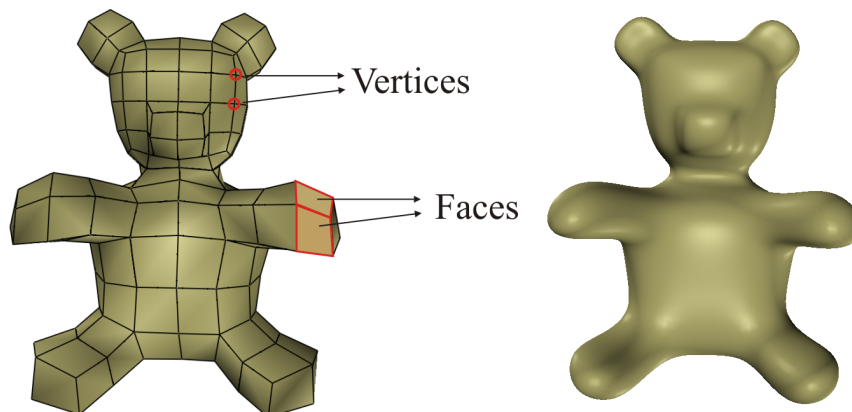


Figure 1.1: *Left:* A polygonal mesh represents an object as a set of vertices and a set of faces connecting the vertices. *Right:* Subdivision is a technique for smoothing such meshes.

subdivision connectivity (semi-regular), and those without any connectivity pattern (irregular). For subdivision curves and regular meshes (subdivision patches), the classical approach for building multiresolution systems is wavelet analysis, which casts subdivision into more established signal-processing terms [41]. More recently, the linear nature of subdivision has been exploited to apply least squares optimizations for constructing multiresolution systems for curves and patches [35, 3, 38].

Both wavelet and local least squares approaches are difficult to extend to mesh schemes. For certain subdivision schemes such as Loop [4, 31] and Doo-Sabin [38], wavelet systems have been successfully constructed, yet some popular schemes like Catmull-Clark still lack a true multiresolution system.

Multiresolution techniques for irregular meshes are exceedingly complex, particularly when compared with the local, linear multiresolutions that result from semi-regular or regular meshes. Irregular mesh schemes typically accept an arbitrary mesh as input and then attempt to reconstruct a low-error approximation of the surface

by repeatedly performing simple operations such as vertex or edge removal. Recent approaches [30, 42, 7] tend to decimate the input mesh to a very coarse representation that has the same topology as the input, and then use subdivision techniques to build a low-error semi-regular representation.

This thesis presents a new technique for constructing multiresolution systems from existing subdivision schemes. Such systems can be used to reduce the resolution of regular and semi-regular meshes in a fast and efficient manner. In conjunction with existing remeshing techniques that construct semi-regular meshes from irregular ones, multiresolution systems open the door to many possible applications, from mesh editing to level-of-detail rendering. The strengths of the technique are that it can be naturally applied to a broad range of subdivision types, and that it performs quantitatively as well as or better than existing techniques.

1.1 Motivation

The key motivation for undertaking this research is to have a more complete set of modeling tools. Many modeling packages include subdivision interfaces for increasing the resolution of a model, but the lack of robust multiresolution methods for semi-regular meshes creates a one-way modeling pipeline.

Consider Z-Brush [34], a commercial modeling tool that offers a very powerful painting metaphor, allowing features to be brushed on interactively. Underneath, Catmull-Clark subdivision is used to increase the resolution and create features. However, once fine-scale features have been created, the lack of a true multiresolution filter set makes it difficult to make changes to the lower-resolution features; in

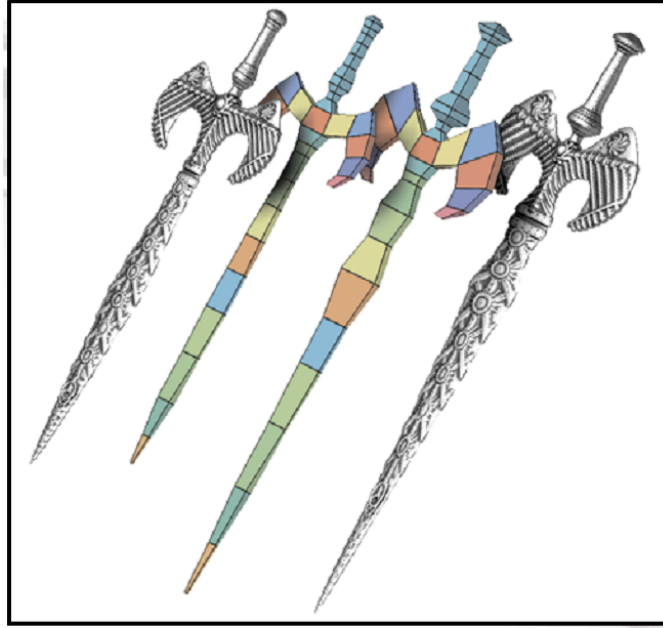


Figure 1.2: Multiresolution editing in a commercial modeling application (ZBrush). (left to right) features on the high-resolution mesh are represented as offsets from a low-resolution base mesh; changes to the base mesh are reflected properly at the higher resolution. (Figure taken from ZBrush Quick Reference Guide, available at <http://www.pixologic.com/zbrush/education/education-documentation.html>.)

practice, the whole mesh hierarchy must be cached (Fig. 1.2).

With a true multiresolution system, a semi-regular mesh can be downsampled algorithmically on demand, thus enabling more contextual editing. When working with fine-scale features, a high-resolution representation can be edited; to make broad or macroscopic changes, a coarse representation can be edited. While there is no technical barrier to making broad changes at the high resolution, such edits would require substantially more effort.

Figures 1.3 and 1.4 illustrate this concept for editing meshes and curves, respectively. In (a), there is a simple mesh with some high-frequency (non-smooth) details. Via a multiresolution method described like those presented in this thesis, a lower

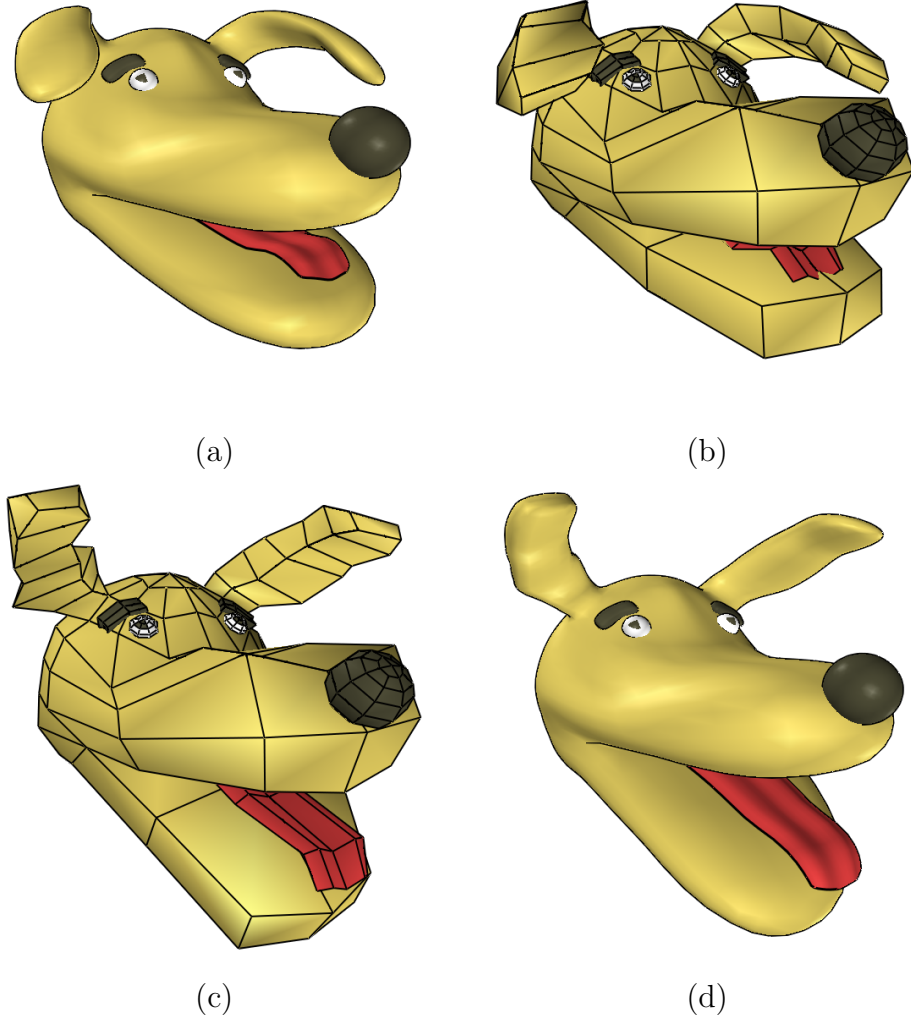


Figure 1.3: Multiresolution mesh editing: (a) the original mesh; (b) a lower resolution reached via multiresolution techniques; (c) a simple editing operation is applied at the low resolution; (d) the low-resolution edit is reflected when the original resolution is restored.

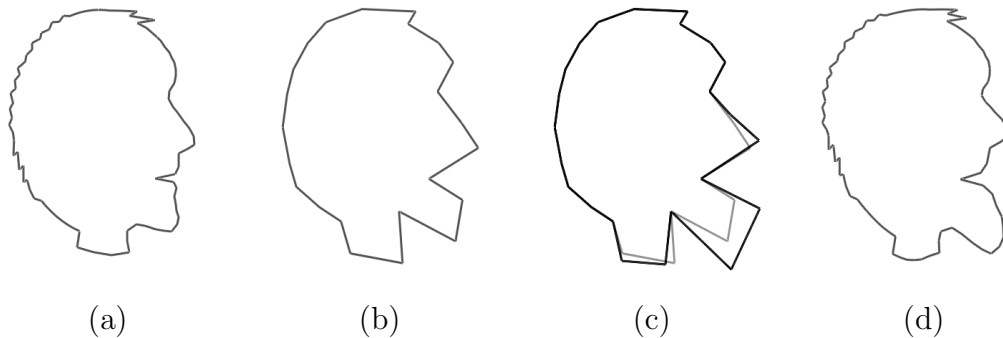


Figure 1.4: Multiresolution editing of a curve: (a) the original curve; (b) a low resolution approximation; (c) the low-resolution version is edited; (d) the edit is carried forward when the original resolution is restored.

resolution can be reached easily, producing the mesh shown in (b). When the mesh is then altered at the low resolution, as in (c), the changes are reflected intuitively at the higher resolution (d). These edits can of course be performed on the full-resolution model of (a). However, to produce the figure in (d) the user would have to move hundreds or thousands of vertices. By editing at a coarser level, only a few vertices need to be moved.

1.2 Goals

In this thesis, a new framework is introduced for constructing wavelet systems for a wide range of curve and surface subdivision schemes. The resulting multiresolution systems are designed with the following goals in mind:

- *Invertibility* When a model is subdivided and then decomposed (lowered in resolution) with the multiresolution system, the original model is expected as output (Fig. 1.5).

- *Reconstructability* When a surface is decomposed, some information is necessarily lost. The multiresolution system should capture this detail information so that the original surface can be reconstructed. See Fig. 1.6.
- *Storage constraint* The low-resolution representation of a model plus the extra information required for reconstruction should require no more storage than the full-resolution representation of the model. This is typically the most difficult condition to satisfy.
- *Locality & Linearity* Subdivision techniques are popular in graphics largely because of their linear running time, which is made possible by the fact that subdivision is based on *linear* operations on a *local* subset of data. An efficient multiresolution system should also involve local and linear operations.
- *Similitude* When a surface that is not the product of subdivision – but that has semi-regular connectivity – is decomposed, the multiresolution system should produce a model that “looks like” the original object. Without this condition, the multiresolution system would be poorly suited to the most important applications: editing and compression. See Fig. 1.7.

1.3 Methodology

When discussing subdivision, the terms *coarse* and *fine* are used to refer to the relative smoothness before and after subdivision, respectively. Multiresolution systems augment subdivision by offering a decrease in resolution, constructing coarse data from fine data. To be able to return to the fine data from the coarse, some extra

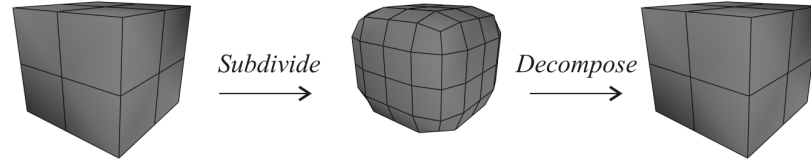


Figure 1.5: The invertibility condition: in a multiresolution system, subdivision followed by decomposition should yield the original object.

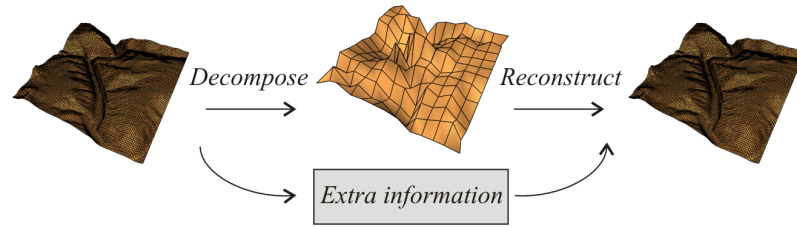


Figure 1.6: The reconstructability condition: a multiresolution system should capture the high-frequency information that is lost by decomposition, to allow full reconstruction of the original surface.

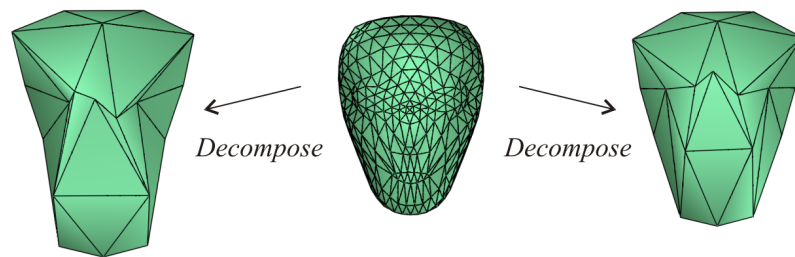


Figure 1.7: The similitude condition: a multiresolution system should try to produce a low-resolution mesh that resembles the original object. The object in the center is decomposed with the Loop filters described in Sec. 5.2 (left) and 5.4 (right).

information called the *details* must also be determined.

To construct multiresolution systems that satisfy the goals of Sec. 1.2, the storage constraint serves as the starting point. That is, the vertices in a fine model that will be replaced with coarse approximations are identified, and then a condition requiring that the extra information for reconstructing the coarse vertices (the *details*) can be computed from a local neighborhood of other details is enforced. Because the detail terms represent coefficients of a wavelet basis (Sec. 2.3.2), this enforced condition is called *constraining the wavelets*.

Having this constraint immediately satisfies the storage constraint, as well as reconstructability (because the entire set of details is either stored or computable). Fortunately, the mathematical formulation of the wavelet constraint yields a method for computing the positions of coarse vertices and ultimately a biorthogonal system.

Finally, the similitude condition is addressed by an optimization step. By analyzing the magnitude of the wavelet coefficients surrounding a coarse vertex, each vertex can be displaced in a way that pushes the coarse mesh closer to the shape and dimensions of the original surface.

1.4 Contributions

The main contribution of this work is a framework for constructing multiresolution systems for a wide range of subdivision schemes. Traditional wavelet methods describe multiresolution in mathematically elegant and robust terms, but falter when it comes to actually constructing multiresolution systems for mesh subdivision schemes. The inherently structured geometric nature of subdivision can be exploited for con-

structing multiresolution systems that are intuitive to understand and implement, yet at the same mathematically sound and indeed able to be cast back into traditional wavelet terms.

Using the framework proposed herein, two key contributions to specific subdivision schemes are made. First, a full multiresolution system for Loop subdivision is derived that delivers results in line with earlier methods while offering easier implementation. Second, the framework is used to develop a full multiresolution system for Catmull-Clark subdivision, not only addressing an unsolved problem in graphics, but addressing it in an elegant fashion.

1.5 Overview of Thesis

This thesis is organized as follows. Chapter 2 discusses the related work in the areas of subdivision, multiresolution, mesh simplification, and mesh editing. Chapter 3 gives an overview of the proposed framework for constructing multiresolution systems. Chapter 4 applies the method to B-spline-based curve subdivision schemes, cubic B-splines in particular; the simplicity and clean notation of curve schemes allows for more in-depth analysis than arbitrary-mesh schemes. Chapters 5 and 6 extend the method to Loop and Catmull-Clark mesh subdivision schemes, respectively. Chapter 7 presents results and analysis for each of these applications. Finally, Chapter 8 concludes the thesis and offers some directions for future investigation. Appendix A summarizes the notation used herein, while Appendices B and C discuss the implementation and interface of the software written for this work.

Chapter 2

Background

Multiresolution is a broad term, applied to all types of graphical models from polygonal meshes to implicit surfaces and involved in both modeling and rendering applications. The primary interest of this work is multiresolution as it relates to subdivision surfaces and wavelets. Parametric curves are an important precursor to subdivision curves and surfaces, so they are presented briefly in Sec. 2.1. Subdivision schemes for curves, regular surfaces, and arbitrary meshes are discussed in Sec. 2.2. Finally, Sec. 2.3 covers multiresolution techniques, in particular as they relate to subdivision methods.

2.1 Parametric Curves

Computer modeling as we know it today is a vastly complex domain. Creation of the models used in motion pictures and even interactive video games requires highly trained artists and technical staff. A crucial aspect of modeling software is intuitiveness: users are generally more comfortable manipulating an object directly than they are manipulating an underlying mathematical representation.

Parametric curves are an early exemplification of this concept. As illustrated in Fig. 2.1, a parametric curve is defined by a set of control vertices. To manipulate the entire continuous curve, a user need only move the control vertices.

Formally, a parametric curve $Q(u)$ is defined by a set of *control vertices*, $\mathbf{V} =$

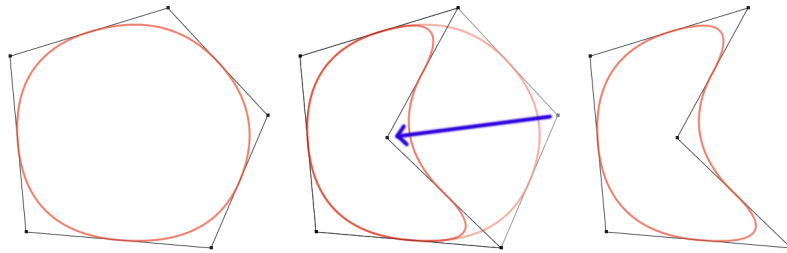


Figure 2.1: A parametric curve is defined by a small set of control vertices: (left) a curve defined by five vertices; moving a control vertex (center) creates an intuitive change in the resulting curve (right).

$\{P_0, \dots, P_m\}$; the curve itself is constructed by evaluating linear combinations of the control vertices over the range of the parameter u . A set of *basis functions* $B_i(u)$ define how the control vertices of a curve are combined to produce a point on the curve: $Q(u) = \sum_i B_i(u)P_i$.

B-spline curves [19, 1] are an important type of parametric curve, especially as they relate to subdivision. The B-spline basis functions – usually denoted $N_{i,k}(u)$ for an order- k curve – are defined recursively; the base case is $k = 1$. An interesting property of the B-spline basis functions – one that makes them suitable for subdivision – is that $N_{i,k}(u)$ is only non-zero for a subset of the range of u values; this is typically referred to as *local control*, because it means that moving a control vertex will only impact the curve in a local area. B-splines are a popular modeling tool for both curves and surfaces; many popular subdivision schemes, including those discussed in this thesis, are extensions of B-spline curves.

The formulation of a parametric curve can be naturally extended to higher-dimensional objects such as 2D surfaces, 3D volumes, and so on. Consider a surface patch defined by a regular 2D grid of control vertices (Fig. 2.2 (left)). The regularity of the vertices allows the patch to be split into two dimensions, usually denoted as

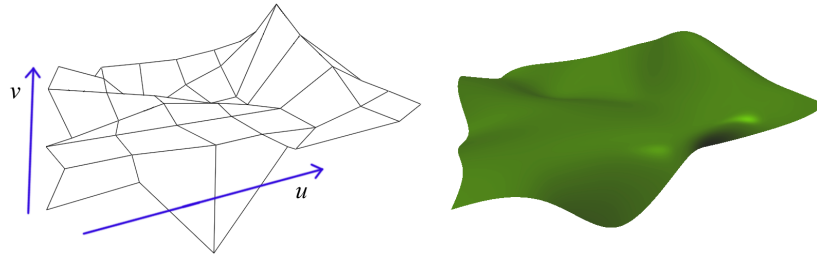


Figure 2.2: A cubic B-spline tensor product surface is defined by a regular grid of control vertices: (left) a control mesh; (right) the resulting surface.

the u and v directions. A *tensor-product surface* can be defined over these vertices as a bivariate function $Q(u, v) = \sum_i B_i(u) \sum_j B_j(v) P_{i,j}$.

2.2 Subdivision

Subdivision techniques in graphics can be broken into three basic categories. Subdivision curves are the elemental type (Section 2.2.1); these include extensions of Beziér [5] and B-spline curves. Subdivision curves, like parametric curves, can be naturally extended to higher dimensions, such as surfaces (2D) and volumes (3D); Section 2.2.2 briefly covers these cases, primarily the surface case that is most relevant to my research. In Section 2.2.3 subdivision schemes that operate on meshes with arbitrary (or very loosely constrained) connectivity to produce semi-regular meshes are discussed.

2.2.1 Curves

Even with optimization techniques, the rendering of parametric curves and surfaces is expensive because they must be evaluated at thousands of parameter values. Sub-

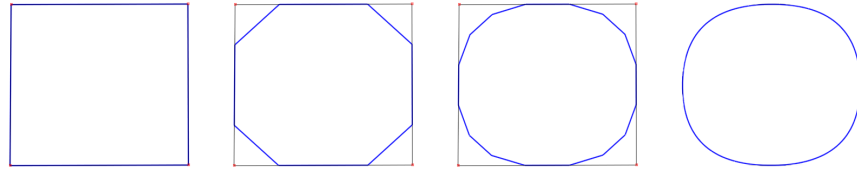


Figure 2.3: Chaikin subdivision (left to right): the original control polygon; after one subdivision; after two subdivisions; the limit surface.

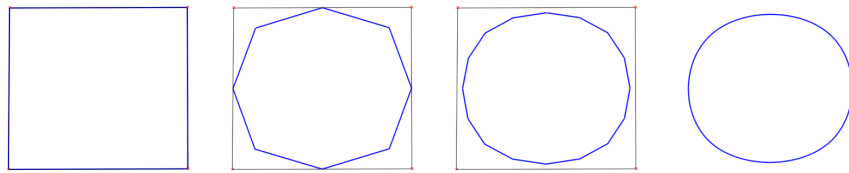


Figure 2.4: Cubic B-spline subdivision (left to right): the original control polygon; after one subdivision; after two subdivisions; the limit surface.

division curves avoid evaluating the curve across the parameter range and instead represent a curve or surface as simply a piecewise-linear connection of the control vertices. Obviously such a rendering would be of poor quality if the original control vertices were used. Subdivision is a process that refines an initial set of control vertices \mathbf{V} into a new set \mathbf{V}' such that the object defined by \mathbf{V}' is equivalent to that defined by \mathbf{V} , yet the piecewise-linear object defined by \mathbf{V}' is “closer” to the limit curve than \mathbf{V} .

This is best illustrated with an example. The first curve subdivision scheme is credited to Chaikin [10], dating back to 1974. When a curve is subdivided with this scheme, every line segment is contracted about its center by a factor of one-half, and the endpoints of these contracted segments are joined to form the new curve.

Equivalently, point P_i is replaced by two points on the line segment between P_i and P_{i+1} : one at $\frac{1}{4}P_i + \frac{3}{4}P_{i+1}$ and another at $\frac{3}{4}P_i + \frac{1}{4}P_{i+1}$. The values $(\frac{1}{4}, \frac{3}{4}, \frac{3}{4}, \frac{1}{4})$ are known as *filter values*. Figure 2.3 illustrates Chaikin's subdivision process being applied to a simple control polygon.

In 1975, that Lane and Riesenfeld [27] showed that Chaikin's "corner-cutting" scheme actually generates quadratic (degree-2) B-spline curves in the limit. They further showed that the filter values for any k^{th} -order B-spline curve can be derived from Pascal's triangle.

Cubic (3^{rd} -degree or 4^{th} -order) B-spline curves are an important entity in computer graphics. The filter values for cubic B-spline subdivision are $(\frac{1}{8}, \frac{1}{2}, \frac{3}{4}, \frac{1}{2}, \frac{1}{8})$. These filter values mean that point P_i is replaced by two points: $\frac{1}{8}P_{i-1} + \frac{3}{4}P_i + \frac{1}{8}P_{i+1}$ and $\frac{1}{2}P_i + \frac{1}{2}P_{i+1}$. Fig. 2.4 shows this scheme applied to a simple control polygon.

Subdivision curves (as well as parametric curves) can be either *open* or *closed*, as illustrated in Fig. 2.5. Closed curves are the easier setting, as the control vertices simply need to be indexed periodically; that is, if a curve is defined by vertices P_1, \dots, P_m , then periodic indexing will wrap the vertices around by setting $P_{1-j} \equiv P_{m+1-j}$ and $P_{m+j} \equiv P_j$. An open curve, in contrast, requires a special set of filters to provide a smooth limit curve while also interpolating the first and last vertices.

Matrix Form of Subdivision

Subdivision is characterized by two important properties: locality and linearity. That is, a vertex produced by subdivision is a linear combination of a local neighborhood of vertices from the original object. Because the operations are linear, subdivision can be expressed in terms of matrices (although the locality of subdivi-

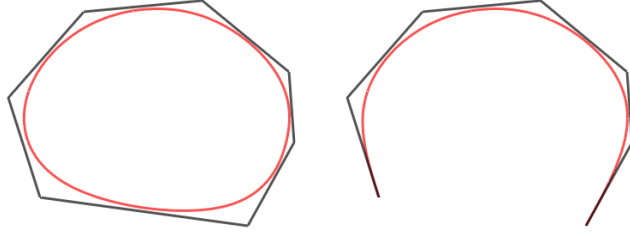


Figure 2.5: Open versus closed curves: for a closed curve (left), periodic indexing is used to “wrap” the control vertices around; for an open curve (right), the curve has definite start and end points, and the subdivision filters must be altered to provide endpoint interpolation.

sion means that such a representation is inefficient).

Consider the previous example of cubic B-spline subdivision, with the filter values of $\left(\frac{1}{8}, \frac{1}{2}, \frac{3}{4}, \frac{1}{2}, \frac{1}{8}\right)$. These filter values represent a regular, repeating column of the subdivision matrix, usually denoted \mathbf{P} . If the control polygon of Fig. 2.4 is represented by four control vertices $\mathbf{C}^k = \{c_0^k, c_1^k, c_2^k, c_3^k\}$, then the subdivision operation will produce 8 control vertices $\mathbf{C}^{k+1} = \{c_0^{k+1}, \dots, c_7^{k+1}\}$. The entire operation can be encapsulated in matrix form as $\mathbf{C}^{k+1} = \mathbf{P}\mathbf{C}^k$, or

$$\begin{bmatrix} c_0^{k+1} \\ c_1^{k+1} \\ \vdots \\ c_7^{k+1} \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & \frac{1}{8} & 0 & \frac{1}{8} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{8} & 0 & \frac{1}{8} & \frac{3}{4} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} c_0^k \\ c_1^k \\ c_2^k \\ c_3^k \end{bmatrix}. \quad (2.1)$$

The subdivision matrix \mathbf{P} is also referred to as a *filter*.

The matrix form of subdivision is important to the analysis of subdivision, especially for schemes that are not based on B-spline curves. For instance, the eigenvalues of the subdivision matrix (more specifically, eigenvalues of a square submatrix of \mathbf{P} such as \mathbf{S} from Eqn. 2.9) indicate whether the scheme converges and with what level of continuity. Matrix notation is also beneficial for constructing and analyzing multiresolution systems, as exemplified by Chapter 4.

In the previous section, subdivision curves were said to be either open or closed. For a closed curve, the subdivision matrix will appear as in Eqn. 2.1: the columns and rows wrap around. For open curves, special filters must be used at the boundary vertices. Following the notation of Samavati & Bartels [36], let \mathbf{P}_s and \mathbf{P}_e represent special subdivision matrices for the start and end of the curve respectively. For interior points, the subdivision filter is applied as usual; denote the subdivision matrix for this regular portion of the curve as \mathbf{P}_r . Together, \mathbf{P}_s , \mathbf{P}_e , and \mathbf{P}_r define the subdivision matrix for an open curve, and the entire matrix can be written in block-matrix form as

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_s \\ \mathbf{P}_r \\ \mathbf{P}_e \end{bmatrix}.$$

2.2.2 Patches

As with parametric curves, the curve subdivision schemes discussed in Sec. 2.2.1 can be easily extended to multidimensional objects, provided the control vertices have some implied connectivity, i.e. regularity.

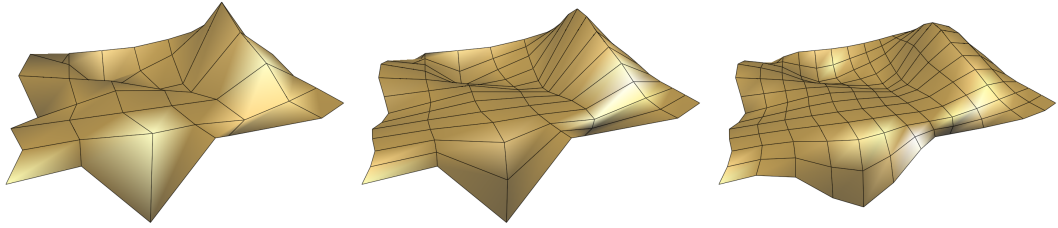


Figure 2.6: A subdivision patch is, like a tensor product surface, defined by a regular grid of control vertices. The same control mesh of Fig. 2.2 (left) is subdivided first along the rows (middle) and then along the columns (right).

The most common extension is to 2D surfaces (commonly referred to as *patches*) or 3D volumes. For instance, a 2D surface can be defined by a regular grid of control vertices, $\mathbf{V} = \{P_{0,0}, \dots, P_{0,m}, \dots, P_{m',0}, \dots, P_{m',m}\}$. To subdivide the surface, a curve subdivision scheme is applied first to the rows of vertices, and then to the columns. (Note that this row-column sequence is analogous to the u - v distinction in the parametric form.) Figure 2.6 shows an example of a subdivision patch.

Note that a “regular” mesh is equivalent to a mesh with only quadrilateral faces (or a volume with cubic voxels, and so on), and in which all vertices have a valence of 4 (Fig. 2.7 (left)). For these kinds of meshes, there is a clear delineation of two independent directions – call them u and v – and subdivision techniques can be applied independently along each direction.

Another possible regular arrangement of faces and vertices is a triangle mesh (Fig. 2.7 (right)). In such a mesh there are three directions: the same u and v directions, and a third direction w that is not independent. Because they are dependent, it does not make sense to apply a curve subdivision technique along each one independently. So although a triangle mesh also has a regular arrangement, it is not as straightforward as a regular quad mesh. Samavati & Bartels [37] recently considered

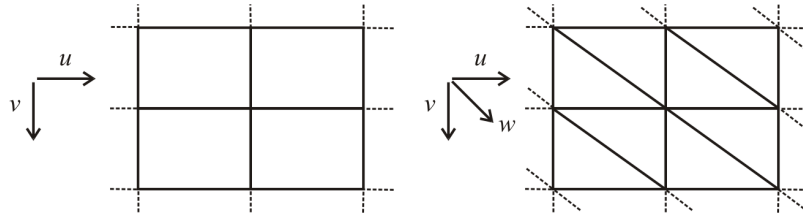


Figure 2.7: The regular arrangement in a quadrilateral mesh (left) yields two independent axes along which curve subdivision techniques can be used. A regular triangle mesh (right) has three dependent axes.

multiresolution settings for regular triangle meshes.

2.2.3 Arbitrary-Connectivity Meshes

Subdivision patches with regular connectivity can be used to model quite complex objects, everything from teapots to vehicles. However, many separate patches must be used to create complex models, and aligning these patches to ensure smoothness at the boundary is a difficult task. Thus regular subdivision patches are perhaps not an ideal model representation.

While subdivision patches or volumes have implicit connectivity – in a curve $\{p_0, \dots, p_m\}$, there is an edge between p_i and p_{i+1} – the connections between vertices in a polygonal mesh have to be explicitly represented by a face structure. Because of the more complex connectivity, subdivision schemes for meshes are typically represented by *masks* rather than filters; a mask dictates how to compute a fine vertex position based on a local neighborhood of coarse vertices.

In 1978, Catmull & Clark [8] and Doo & Sabin [14, 15] published subdivision methods that could operate on meshes with non-regular connectivity. Figure 2.8 shows a polygonal model representing a chess piece; after only three applications of subdivision, a smooth, visually appealing surface is reached. The ability to subdi-

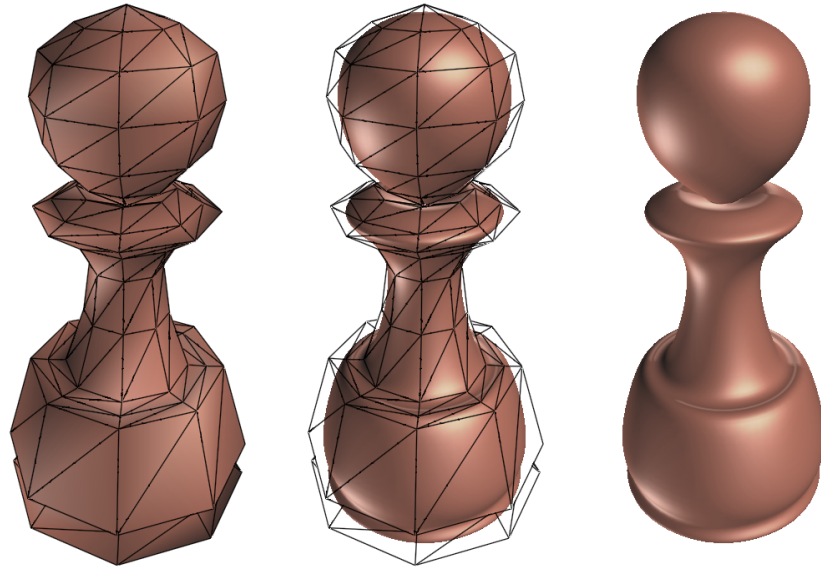


Figure 2.8: Mesh subdivision schemes allow any arbitrary polygonal object to be refined to a smooth surface. (Loop subdivision is used here.)

vide any polygonal mesh to a smooth surface represented a major breakthrough in modeling for computer graphics. Because each subdivision scheme has a structured way of creating vertices and faces, meshes that result from subdivision are said to be *semi-regular*. In fact, any mesh that has the vertex connectivity, but not necessarily the geometry, produced by subdivision is said to be semi-regular. Semi-regularity of an arbitrary mesh is typically determined by a graph-traversal-like process; see Appendix C.

Doo & Sabin’s method is a generalization of 2^{nd} -degree (Chaikin) B-spline patches; like the Chaikin curve subdivision scheme, Doo-Sabin subdivision has a “corner-cutting” effect (see Fig. 2.9). Catmull & Clark’s subdivision method, meanwhile, is an extension of 3^{rd} -degree (cubic) B-splines. The result of applying Catmull-Clark subdivision to a cube is shown in Fig. 2.10.

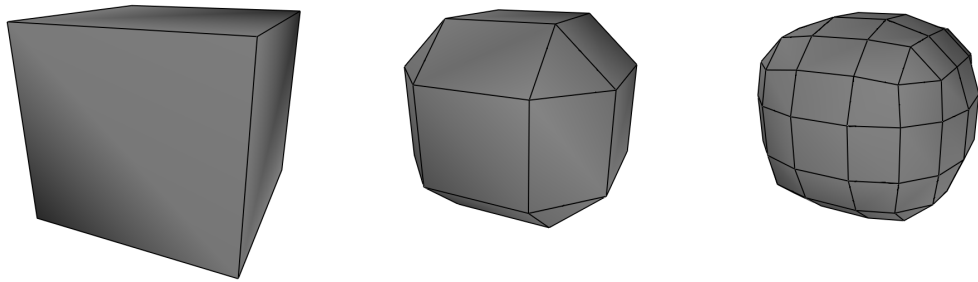


Figure 2.9: Doo-Sabin subdivision of a cube.

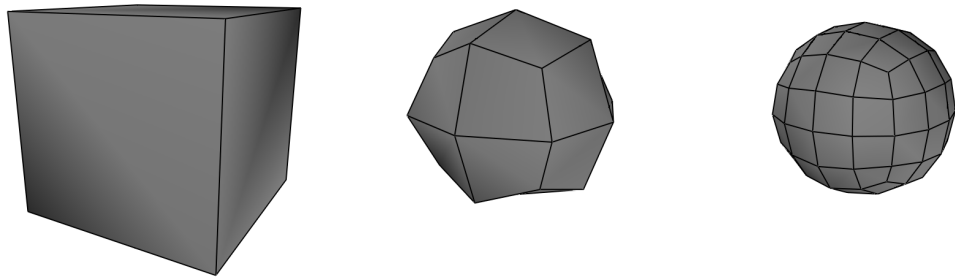


Figure 2.10: Catmull-Clark subdivision of a cube.

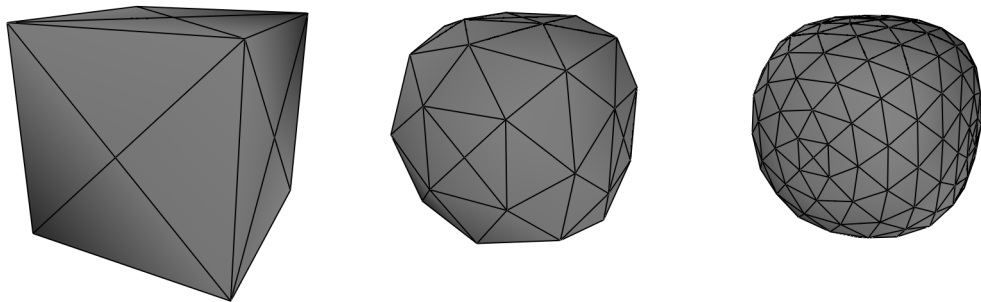


Figure 2.11: Loop subdivision of a cube.

Doo-Sabin subdivision *contracts* (shrinks) each face in the original mesh, while Catmull-Clark subdivision *splits* each face into several new faces. Face- (or edge-) splitting subdivision schemes are referred to as *primal* schemes, while face-contracting schemes are known as *dual*, or vertex-split, schemes [47].

In 1987, Loop [33] published a subdivision scheme that, like Catmull-Clark, is based on cubic B-splines. His scheme operates only on meshes with triangular faces; this is not much of a limitation though, because any arbitrary mesh can easily be converted to a triangular mesh (see Sec. 2.3.1). Loop’s scheme is based on *quartic box splines* [19]. An iteration of Loop subdivision splits each triangle into four triangles by creating new vertices at each edge and displacing old vertices. Figure 2.11 shows the application of Loop subdivision to a cube mesh.

There are several other subdivision methods for arbitrary meshes [16, 48, 32], but they are not integral to this research. It suffices to say that the underlying mathematics of each scheme is very similar, but that each scheme has certain properties (continuity, interpolation, etc.) that suit it to particular modeling domains. The combination of C^2 continuity and the popularity of triangle- and quadrilateral-based meshes make Loop and Catmull-Clark subdivision the most popular schemes in practice.

Subdivision surfaces are used extensively in both commercial and in-house modeling packages. For instance, Pixar Animation Studios © popularized the use of Catmull-Clark surfaces in computer animation [13]. ZBrush [34] is a commercial modeling package that provides a sculpting paradigm for modeling, and subdivision is used to increase the resolution of a model when finer-scale features are being sculpted. Other modeling applications such as Maya, 3DSMax, and the open-source

Blender also provide subdivision surfaces as a basic tool.

2.3 Multiresolution (MR)

Subdivision is a powerful tool for increasing the resolution of a model, but unfortunately it is a one-way process: resolution is only increased, not decreased. In cases such as Fig. 2.8, where a mesh is simply subdivided a number of times for rendering purposes, this is a minor problem. The original mesh can be cached at the beginning to avoid having to store the high-resolution version. Or, because most subdivision rules can be reversed directly, the original mesh could be returned to algorithmically. Thus, undoing subdivision is not the goal of multiresolution.

The purpose of multiresolution (**MR**) is to take a high-resolution object that is not the product of subdivision and reduce the resolution in a way that satisfies the goals of invertibility, reconstructability, minimal error, and the storage constraint. That is, an MR process should: a) produce a mesh that “looks like” the original mesh; b) compute and store additional information for reconstruction of the original mesh; and c) require no more storage for the approximation and detail information than the original mesh required.

In general, it is easy to invert a subdivision scheme, i.e. construct MR filters that satisfy invertibility. For instance, consider cubic B-spline subdivision as illustrated by Eqn. 2.1. Specifically, consider $c_0^{k+1} = \frac{1}{8}c_4^k + \frac{3}{4}c_0^k + \frac{1}{8}c_1^k$. To construct an invertible MR system, an expression for c_0^k (representative of all coarse vertices) is needed. An easy way to determine this is to express c_0^{k+1} in terms of c_0^k and other *fine* vertices. By noting that $c_7^{k+1} = \frac{1}{2}c_4^k + \frac{1}{2}c_0^k$ and $c_1^{k+1} = \frac{1}{2}c_0^k + \frac{1}{2}c_1^k$, the original mask can be

rewritten as $c_0^{k+1} = \frac{1}{4}c_7^{k+1} + \frac{1}{2}c_0^k + \frac{1}{4}c_1^{k+1}$. From this form, level k can be returned to from $k + 1$ by isolating c_0^k as $c_0^k = 2c_0^{k+1} - \frac{1}{2}c_7^{k+1} - \frac{1}{2}c_1^{k+1}$.

This “trick” of rewriting the subdivision mask to use fine vertices from level $k + 1$ rather than coarse vertices from level k works for a wide range of subdivision schemes, including complex ones like Catmull-Clark. But unfortunately, having an invertible MR system does not automatically satisfy all other goals. In practice, this straightforward reversal of the subdivision rule produces high-error approximations. And equally problematic, such a construction gives no indication as to how to satisfy the storage constraint.

Building MR systems that satisfy all of the stated goals is a challenging problem. In the following sections, some existing approaches will be looked at. In Sec. 2.3.1, mesh simplification approaches that do not require a subdivision surface as a starting point are considered. In Sec. 2.3.2, some theory and applications of wavelet systems are presented. Finally, Sec. 2.3.3 discusses some practical applications of MR to modeling and rendering.

2.3.1 Mesh Simplification & Remeshing

A problem that is closely related to MR is mesh simplification. Given an input polygonal mesh, the goal of mesh simplification is to produce a new mesh “of the same topological type as [the input mesh], that fits the data well and has a small number of vertices” [22].

Schroeder et al. [39] approach the remeshing problem by considering a vertex removal and retriangulation process. On each iteration, vertices that satisfy some local geometric criteria, such as low local curvature, are removed and the resulting

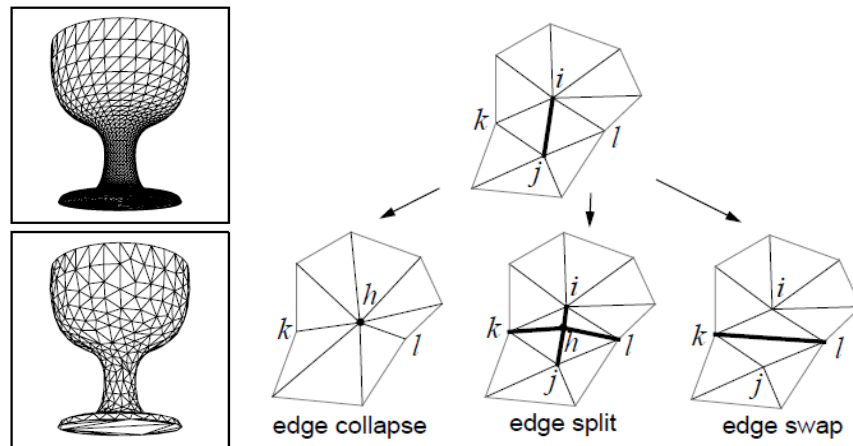


Figure 2.12: Hoppe et al.'s approach to mesh simplification (left) is to minimize an energy function on the mesh by simple edge-based operations (right). (Figures taken from [22].)

hole is retriangulated. After several iterations, a very coarse mesh can be reached. The amount of simplification that results is controlled by setting thresholds on the vertex removal criterion.

Hoppe et al. [22] also consider the problem of mesh simplification, with the goal of reducing the number of vertices in a dense or poorly sampled triangle mesh. Their approach is to define an energy function on the mesh, with terms representing the competing goals of accuracy and conciseness. They define a simple set of edge-based operations for simplifying a mesh (Fig. 2.12), which are iteratively applied in a way that satisfies the energy function.

The idea of mesh simplification was taken a step further by Eck et al. [17]. Rather than just decimating a dense mesh, they create a coarse base mesh and then use subdivision to create a semi-regular approximation of the original mesh. Their approach is based on computing a Voronoi tiling of the original mesh via harmonic maps; the computation of these maps is a bottleneck of the algorithm, however.

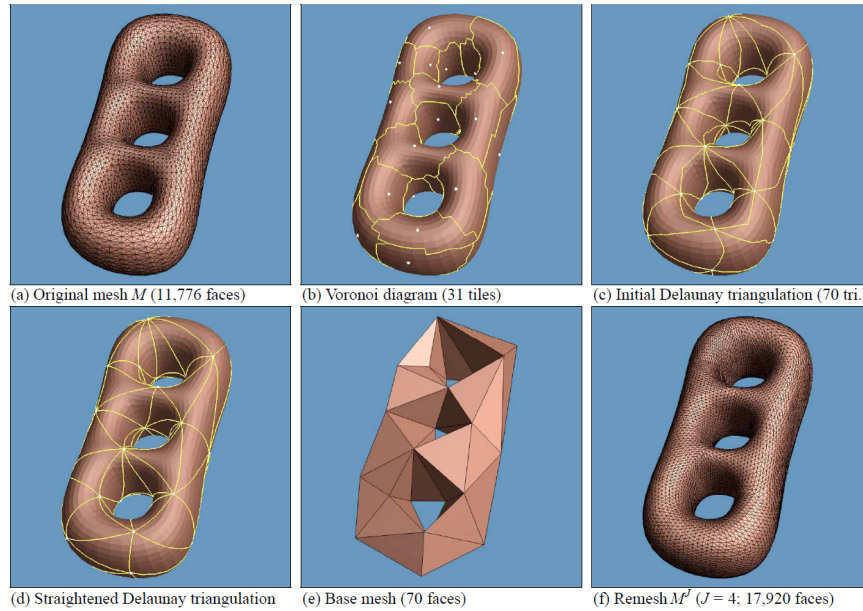


Figure 2.13: The remeshing approach of Eck et al.: starting with an arbitrary mesh (top left) a base mesh (bottom center) is constructed and then subdivided to an approximation of the original mesh (bottom right). (Figures taken from [17].)

Their algorithm yields a hierarchy of mesh resolutions and allows them to leverage the benefits of MR on arbitrary meshes: compression, powerful editing, and level-of-detail. This approach is known as *remeshing*.

Lee et al. [30] also explore remeshing for subdivision connectivity. They use atomic mesh simplification steps including the edge-based operations of Hoppe et al. and also adding vertex-based operations. Their simplification and remeshing approach attempts to address the inefficiency of Eck et al.'s harmonic map approach. By returning to the atomic simplification steps of Hoppe et al., they can quickly reach a base domain; because the base mesh is only necessary to build the approximation mesh, its quality is relatively unimportant.

Hoppe [20] later presented the concept of *progressive meshes* as an alternative to

MR meshes. This work is an extension of the earlier mesh optimization work [22] to an approach more akin to remeshing, i.e. the original mesh is reached in the limit. The progressive mesh approach is well suited as a preprocess applied to a mesh before rendering, as the intermediate meshes are good representations of the input mesh. However, as the *progressive* moniker implies, a progressive mesh provides many intermediate resolutions between a coarse and fine mesh, but without a local, linear set of operations for moving between them; this makes them well suited to level-of-detail applications, because the transition between resolutions is not jarring. The discretized resolutions provided by MR systems for semi-regular meshes are more suitable for editing applications.

There are other remeshing approaches that operate under different criteria than subdivision connectivity, such as maximizing the number of regular vertices (valence-6 for triangle meshes) [42] or equalizing the area of triangles [7]. However, our approach is to build MR systems for meshes with subdivision connectivity.

2.3.2 Wavelets for MR

Wavelets, “a mathematical tool for hierarchically decomposing functions” [40], have a history spanning many scientific domains, primarily engineering and signal processing [45]. Whereas the Fourier transform decomposes a function into frequency components, wavelets decompose a function into a coarse approximation of the function plus some detail coefficients. Thus wavelets are well-suited to computer graphics because the decomposed function is an approximation of the input function, and the correspondence can be evaluated visually.

The following section describes wavelet systems formally according to the nota-

tion and style of Stollnitz et al. [41], and then wavelet classifications are discussed.

Wavelet Theory

In multiresolution theory, a signal that exists within a linear function space V^k is represented as a vector $\mathbf{C}^k = [c_0^k \ c_1^k \ \dots]$, where c_i^k is a coefficient of basis function i of V^k . For instance, in a B-spline curve the control points are coefficients of the B-spline basis functions $N_{i,k}(u)$, and the “signal” is the curve defined by those control points. In wavelet literature, the basis functions of V^k are referred to as *scaling functions*, denoted $\phi_i^k(u)$. The set of all scaling functions for a given V^k is denoted by $\Phi^k(u) = [\phi_0^k(u) \ \phi_1^k(u) \ \dots]$; a $v(k)$ -dimensional space will have $v(k)$ basis functions.

In Sec. 2.2.1, subdivision was defined as an operation that takes a set of control points \mathbf{C}^{k-1} and produces a new set \mathbf{C}^k such that both sets of control points define the same limit curve. In terms of scaling functions, this means

$$\Phi^{k-1}(u)\mathbf{C}^{k-1} = \Phi^k(u)\mathbf{C}^k . \quad (2.2)$$

Because \mathbf{C}^k is defined by a subdivision operation, it can be replaced with $\mathbf{P}^k\mathbf{C}^{k-1}$ in Eqn. 2.2 yields

$$\begin{aligned} \Phi^{k-1}(u)\mathbf{C}^{k-1} &= \Phi^k(u)\mathbf{P}^k\mathbf{C}^{k-1} \\ \Phi^{k-1}(u) &= \Phi^k(u)\mathbf{P}^k . \end{aligned} \quad (2.3)$$

Equation 2.3 captures the relationship between the scaling functions at level $k-1$ and k ; subdivision really represents a change of representation, embedding an object in a higher-dimensional function space.

Multiresolution analysis requires additional function spaces called wavelet spaces, denoted W^k . A $w(k)$ -dimensional wavelet space W^k is the complement of V^k in V^{k+1} ,

meaning that any function in V^{k+1} can be written as the sum of a unique function from each of V^k and W^k . For example, in the Haar wavelets of Figs. 2.14 and 2.15, $\phi_0^2 = \phi_0^1 + \psi_0^1$.

Though the term “wavelets” is often used to describe wavelet systems in general, the term specifically refers to the basis functions $\Psi^k(u) = [\psi_0^k(u) \psi_1^k(u) \cdots]$ of the wavelet space W^k . When an object is reduced in resolution, from level k to $k - 1$, some high-frequency information may be lost; if this information can be represented as coefficients of the wavelets, then the original object can be recovered.

Similar to Eqn. 2.3, the wavelets at level $k - 1$ are related to the scaling functions at level k by a matrix \mathbf{Q}^k :

$$\Psi^{k-1}(u) = \Phi^k(u) \mathbf{Q}^k, \quad (2.4)$$

where \mathbf{Q}^k is a $v(k) \times w(k - 1)$ matrix. Equations 2.3 and 2.4 can be combined as

$$[\Phi^{k-1} | \Psi^{k-1}] = \Phi^k [\mathbf{P}^k | \mathbf{Q}^k]. \quad (2.5)$$

For 1D wavelet systems with no boundaries (such as closed curves), the matrices \mathbf{P}^k and \mathbf{Q}^k can be entirely described by sequences $(\dots, p_{-1}, p_0, p_1, \dots)$ and $(\dots, q_{-1}, q_0, q_1, \dots)$, which represent regular columns of the respective matrices; these sequences are called *filter values*. For instance, the subdivision filter values of regular cubic B-spline curves are $(\frac{1}{8}, \frac{1}{2}, \frac{3}{4}, \frac{1}{2}, \frac{1}{8})$.

The matrices \mathbf{P}^k and \mathbf{Q}^k perform a transition from level $k - 1$ to level k by refining the scaling functions and wavelets at level $k - 1$. For multiresolution systems, we also want to transition from level k to level $k - 1$, i.e. lower the resolution. This transition requires Φ^k to be decomposed into its complementary basis functions Φ^{k-1}

and Ψ^{k-1} , which can be represented by two matrices \mathbf{A}^k and \mathbf{B}^k as

$$\left[\Phi^{k-1} | \Psi^{k-1} \right] \begin{bmatrix} \mathbf{A}^k \\ \mathbf{B}^k \end{bmatrix} = \Phi^k . \quad (2.6)$$

These matrices are referred to as *decomposition filters*.

The decomposition filter \mathbf{A}^k downsamples an object \mathbf{C}^k by $\mathbf{C}^{k-1} = \mathbf{A}^k \mathbf{C}^k$. In certain cases, the detail lost by downsampling can be captured as another vector \mathbf{D}^{k-1} by $\mathbf{D}^{k-1} = \mathbf{B}^k \mathbf{C}^k$. Together, these operations are called *decomposition*.

The inverse of decomposition is called *reconstruction*, and is accomplished by \mathbf{P}^k and \mathbf{Q}^k as $\mathbf{C}^k = \mathbf{P}^k \mathbf{C}^{k-1} + \mathbf{Q}^k \mathbf{D}^{k-1}$. Reconstruction thus involves subdivision of the downsampled data, followed by some interpretation of the detail information.

Together, the four matrices – \mathbf{A}^k , \mathbf{B}^k , \mathbf{P}^k , and \mathbf{Q}^k – form a multiresolution or wavelet system. By Eqns. 2.5 and 2.6, the four matrices must be related as

$$\begin{bmatrix} \mathbf{A}^k \\ \mathbf{B}^k \end{bmatrix} = \left[\mathbf{P}^k | \mathbf{Q}^k \right]^{-1} ,$$

or equivalently

$$\begin{bmatrix} \mathbf{A}^k \\ \mathbf{B}^k \end{bmatrix} \left[\mathbf{P}^k | \mathbf{Q}^k \right] = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \mathbf{I} . \quad (2.7)$$

This implies that each block matrix must be invertible, which in turn implies that decomposition is the inverse of reconstruction and vice versa.

Example: Haar Wavelets

These concepts are best illustrated with a simple example, Haar wavelets, following the structure used by Stollnitz et al. [41]. The scaling functions in the Haar

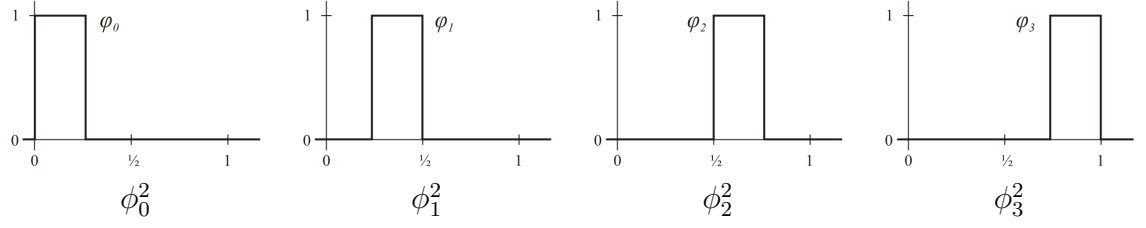


Figure 2.14: The Haar scaling functions for the function space V^2 .

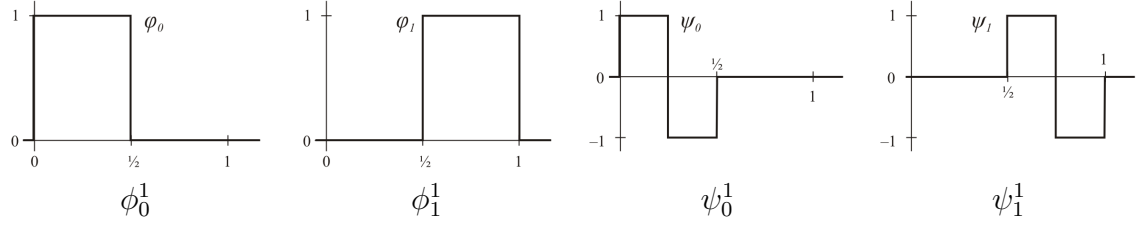


Figure 2.15: The Haar scaling functions and wavelets for the function space V^1 .

wavelet system are piecewise-constant box functions of height 1 (Fig. 2.14). For instance, consider a “signal” \mathbf{C}^k that represents a row of pixels in an image:

$$\mathbf{C}^2 = \begin{bmatrix} 12 & 10 & 13 & 19 \end{bmatrix}^T.$$

The 2 superscript indicates that this is the third level of resolution for the function, the first level being one pixel and the second level two pixels. Because the Haar scaling functions are box functions, the pixel values *are* the scaling function coefficients.

Haar decomposition represents each pair of coefficients by their average, and the difference between the average and original values is stored as a detail term. Thus for the 4-pixel signal example, the decomposition filters are

$$\mathbf{A}^2 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad \text{and} \quad \mathbf{B}^2 = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}.$$

The decomposition of \mathbf{C}^2 is

$$\begin{aligned}\mathbf{C}^1 &= \mathbf{A}^2 \mathbf{C}^2 = \begin{bmatrix} 11 & 16 \end{bmatrix}^T, \\ \mathbf{D}^1 &= \mathbf{B}^2 \mathbf{C}^2 = \begin{bmatrix} 1 & -3 \end{bmatrix}^T.\end{aligned}$$

The function can again be decomposed to

$$\begin{aligned}\mathbf{C}^0 &= \mathbf{A}^1 \mathbf{C}^1 = \begin{bmatrix} 13.5 \end{bmatrix}^T, \\ \mathbf{D}^0 &= \mathbf{B}^1 \mathbf{C}^1 = \begin{bmatrix} -2.5 \end{bmatrix}^T.\end{aligned}$$

The sequence $[\mathbf{C}^0 \ \mathbf{D}^0 \ \mathbf{D}^1]$ represents the original function \mathbf{C}^2 in the Haar basis: $[13.5 \ -2.5 \ 1 \ -2]$.

To reconstruct the original function, a \mathbf{P} and \mathbf{Q} are required for each level. Since decomposition is based on an averaging and differencing, reconstruction can be performed by first duplicating the coefficients and then adding in the difference or its negation. For instance, to return to \mathbf{C}^1 from \mathbf{C}^0 , the pixel is first duplicated,

$$\begin{bmatrix} 13.5 \end{bmatrix} \rightarrow \begin{bmatrix} 13.5 & 13.5 \end{bmatrix},$$

and then the difference (plus or minus) is added,

$$\begin{bmatrix} 13.5 & 13.5 \end{bmatrix} + \begin{bmatrix} -2.5 & -(-2.5) \end{bmatrix} = \begin{bmatrix} 11 & 16 \end{bmatrix}.$$

As mentioned above, the wavelets at level k are related to the scaling functions at level $k + 1$ by a matrix \mathbf{Q}^k . During reconstruction, \mathbf{Q}^k also encapsulates the conversion of the detail terms from the wavelet space W^k to the function space V^{k+1} . Each detail term contributes to two samples at the higher resolution level, with weights of 1 and -1 ; this operation is represented by a step function that has

height 1 for half of its active domain and -1 for the other half. The Haar wavelets at level 1 are depicted, along with the scaling functions, in Fig. 2.15.

In matrix form, the transition from level 1 to 2 is given by

$$\mathbf{P}^2 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{Q}^2 = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}.$$

Then, the reconstruction of level 2 is performed by

$$\begin{aligned} \mathbf{C}^2 &= \mathbf{P}^2 \mathbf{C}^1 + \mathbf{Q}^2 \mathbf{D}^1 \\ &= \begin{bmatrix} 11 & 11 & 16 & 16 \end{bmatrix} + \begin{bmatrix} 1 & -1 & -3 & -(-3) \end{bmatrix} \\ &= \begin{bmatrix} 12 & 10 & 13 & 19 \end{bmatrix}. \end{aligned}$$

Wavelet Classifications

Wavelet systems are classified according to the relationship between the wavelets and the scaling functions. There are three classifications: orthogonal, semi-orthogonal, and biorthogonal. For clarity, the superscript k will be omitted from matrix notation; all reconstruction and decomposition filters are assumed to be of the proper size to operate on their data.

Orthogonal wavelets require that “the scaling functions are orthogonal to one another, the wavelets are orthogonal to one another, and each of the wavelets is orthogonal to every coarser scaling function” [41]; two functions $f(u)$ and $g(u)$ are orthogonal if $\int_u f(u) \times g(u) = 0$. In such a setting, the determination of the MR filters is quite easy, because $[\mathbf{P}|\mathbf{Q}]^{-1} = [\mathbf{P}|\mathbf{Q}]^T$. Therefore, $\mathbf{A} = \mathbf{P}^T$ and $\mathbf{B} = \mathbf{Q}^T$. Due to this easy determination of all MR filters, orthogonality is a nice condition to

have. Unfortunately, orthogonality is difficult to satisfy for all but the most trivial bases (such as Haar). Furthermore, Haar wavelets are the only system that are at once orthogonal, compact, and symmetric [41].

Haar wavelets are not the only orthogonal wavelet system used in graphics. Daubechies wavelets [11] are an orthonormal and compactly supported system for signals on the infinite real line. The scaling functions and wavelets of Daubechies are asymmetric and nowhere differentiable. However, Daubechies wavelets offer few benefits over Haar and are interesting mainly from a theoretical standpoint.

Semiorthogonal wavelets relax the orthogonality conditions greatly, only requiring that each wavelet is orthogonal to all coarser scaling functions. Relaxing the orthogonality constraints on the wavelets allows for MR systems to be constructed with more desirable properties, such as compactness and symmetry. The drawback of semiorthogonal wavelets is that while \mathbf{P} and \mathbf{Q} will be sparse and banded matrices (meaning that reconstruction can be done in linear time), it often turns out that the decomposition filters are full matrices, meaning that decomposition would take quadratic instead of linear time.

There are many more practical instances of semi-orthogonal wavelet systems for graphics than orthogonal wavelets. Finkelstein and Salesin [18] follow a traditional wavelet approach to construct a \mathbf{Q}^k that lies in the nullspace of a particular matrix \mathbf{M}^k , where \mathbf{M}^k is derived from the subdivision filter \mathbf{P}^k and the scaling functions Φ^k . The decomposition filters are then found by solving a sparse linear system.

Finally, there are *biorthogonal* wavelets that have many of the properties of semi-orthogonal wavelets but enforce no orthogonality conditions. The only condition in a biorthogonal setting is that $[\mathbf{P}|\mathbf{Q}]$ is invertible, which by Equation 2.7 implies that

the decomposition filters \mathbf{A} and \mathbf{B} exist. Clearly this is the minimum condition to satisfy. Biorthogonal wavelets allow a lot of freedom in the selection of MR filters, so it is usually possible to have a full set of sparse filters and therefore linear-time reconstruction *and* decomposition.

A general approach to constructing biorthogonal wavelets is the *lifting* scheme of Sweldens [43, 44]. In a lifting approach, simple and often poor-quality (in terms of similitude) MR filters \mathbf{P} , \mathbf{Q} , \mathbf{A} , and \mathbf{B} are constructed. Then, a lifting matrix \mathbf{L} modifies the original scheme as:

$$\begin{aligned}\mathbf{Q}_{lift} &= \mathbf{A} + \mathbf{L}\mathbf{B} \\ \mathbf{A}_{lift} &= \mathbf{Q} - \mathbf{P}\mathbf{L}\end{aligned}\tag{2.8}$$

The remaining matrices – \mathbf{P} and \mathbf{B} – are unaltered by lifting. If the original set of filters is biorthogonal, then the lifted set of filters is also biorthogonal for any choice of \mathbf{L} . As will be shown in Sec. 4.6, our method can be viewed as a lifting-like process.

Most MR systems begin from an existing subdivision scheme, meaning that the \mathbf{P} matrix is pre-determined, and the remaining three matrices need to be defined to complete the system. In the literature on MR, there are two basic approaches to finding \mathbf{A} , \mathbf{B} , and \mathbf{Q} given some subdivision matrix \mathbf{P} . In wavelet methods, \mathbf{Q} is defined by the relationship between the wavelets and the scaling functions in the same manner as \mathbf{P} ; the decomposition filters \mathbf{A} and \mathbf{B} are then constructed to satisfy Eqn. 2.7. Thus the filters are derived in the sequence

$$\mathbf{P} \Rightarrow \mathbf{Q} \Rightarrow \mathbf{A}, \mathbf{B} .$$

The local least squares approach of Samavati & Bartels [35, 3] first derives \mathbf{A} from \mathbf{P} , and \mathbf{Q} and \mathbf{B} follow from linear algebra techniques. The filter derivation sequence

in this approach is

$$\mathbf{P} \Rightarrow \mathbf{A} \Rightarrow \mathbf{Q}, \mathbf{B} .$$

Note that in our framework, described in Chapter 3, the filters are derived in the sequence

$$\mathbf{P} \Rightarrow \mathbf{A} , \mathbf{B} \Rightarrow \mathbf{Q} .$$

The order that the filters are derived is important because it is easier to design filters with particular properties earlier in the sequence. For instance, if \mathbf{A} is chosen first, it can be designed with similitude in mind.

Samavati & Bartels [35, 3] approach MR from a linear algebra perspective. The idea of their method is to derive \mathbf{A} by minimizing the local error in the coarse approximation. In particular, they choose \mathbf{A} to minimize the least-squares error $\|\mathbf{C}^{k+1} - \mathbf{PAC}^{k+1}\|^2$. Of course this would be minimized by $\mathbf{A} = \mathbf{P}^{-1}$, but \mathbf{P} is not invertible (it is not square). To have a tractable approach, they instead consider the *local* least squares error. That is, rather than dealing with a full subdivision matrix \mathbf{P}^k of size $v(k) \times v(k-1)$, they consider a local subdivision matrix \mathbf{S} that represents all possible interactions of \mathbf{P}^k . For example, the local subdivision matrix for cubic B-spline subdivision would be

$$\mathbf{S} = \begin{bmatrix} \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \end{bmatrix} . \quad (2.9)$$

The decomposition filter \mathbf{A} is chosen so that its local interaction with \mathbf{S} will satisfy

$\mathbf{AP} = \mathbf{I}$:

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{8} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{3}{4} & \frac{1}{2} & \frac{1}{8} & 0 & 0 & 0 \\ 0 & \frac{1}{8} & \frac{1}{2} & \frac{3}{4} & \frac{1}{2} & \frac{1}{8} & 0 \\ 0 & 0 & 0 & \frac{1}{8} & \frac{1}{2} & \frac{3}{4} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} a_{-3} \\ a_{-2} \\ a_{-1} \\ a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}. \quad (2.10)$$

For instance, one possible solution to this system is $(a_{-3}, \dots, a_3) = (0, 0, -\frac{1}{2}, 2, -\frac{1}{2}, 0, 0)$, which can be compactly expressed as

$$\frac{a_0}{2} \begin{bmatrix} a_{\pm 1} \\ -\frac{1}{2} \end{bmatrix}. \quad (2.11)$$

Solutions with wider support, or more non-zero entries in the \mathbf{A} filter, often provide better results; some of the wider filters from Samavati and Bartels are considered in Chapter 7. The remaining filters – \mathbf{B} and \mathbf{Q} – are chosen by more elaborate methods to satisfy $\mathbf{AQ} = \mathbf{0}$, $\mathbf{BP} = \mathbf{0}$, and $\mathbf{BQ} = \mathbf{I}$. Bartels & Samavati [3] were able to apply this approach to B-spline subdivision curves; the resulting wavelet systems are semi-orthogonal systems under a non-standard inner product definition.

Samavati et al. [38] later applied a similar local least squares method to Doo-Sabin subdivision surfaces. The derived MR filters are sparse (and therefore efficient) and produce a locally optimal coarse approximation of the fine data as quantified by the least squares error.

Methods for constructing semi-orthogonal wavelets are generally difficult to extend to mesh subdivision schemes, because the interactions between vertices are

much more complex and thus matrix forms are unwieldy. As such, mesh MR systems are often only biorthogonal. Recently, Bertram [4] and Li et al. [31] constructed fairly stable biorthogonal wavelets for Loop subdivision. Their approach is based on rewriting the subdivision rules with some additional free parameters, such that regular subdivision is unchanged but an inversion of the rules produces a multiresolution system. Each researcher pursues a different method to determine the free parameters, but each results in a large and unwieldy set of constants to handle different vertex valences. In Chapter 5, an MR system is constructed by our method that provides competitive performance and satisfies all goals of MR, but our system has far fewer parameters and offers a more streamlined implementation.

Lanquetin and Neveu [28] recently published a reversal method for Catmull-Clark subdivision. They derive a decomposition \mathbf{A} filter by setting up a linear system of equations based on the subdivision masks. Their approach is lacking in several ways. First, their method for deriving the \mathbf{A} filter is unnecessarily complex; by the filter rewriting trick discussed at the beginning of Sec. 2.3, the decomposition mask follows immediately without the need to solve any system. More critically, their approach does not define an MR system; there are no filters for the wavelet coefficients, nor is the storage constraint considered. Finally, as will be shown in Chapter 6, their simplistic decomposition mask often does not satisfy the similitude condition.

2.3.3 Applications

A true MR system is biorthogonal (at least) and computes and stores detail information for reconstruction. These detail coefficients can be employed in many interesting ways.

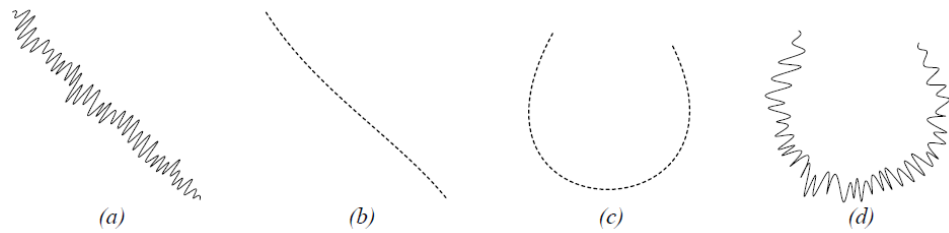


Figure 2.16: Finkelstein and Salesin introduced feature transfer for MR curves. The characteristics of the curve in (a) are extracted and then applied to a different base curve (c), producing a new curve with the same characteristics (d). (Figure taken from [18].)

One powerful use of MR is *feature transfer*. When a signal is decomposed, the detail coefficients represent the “characteristics” of the original signal at varying resolutions or frequencies. These characteristics can be applied to other objects by using them in the subdivision/reconstruction process of a different base signal.

Finkelstein and Salesin [18] developed an early implementation of this idea for MR curves. By decomposing an input curve to a coarse representation and some characteristic details, the details can then be applied to a new base curve. Figure 2.16 demonstrates the feature transfer allowed by their system.

By a similar technique, Biermann et al. [6] developed a feature transfer algorithm for meshes. Mesh techniques are necessarily more complex, but the spirit is the same and their results are impressive (Fig. 2.17). Their method could benefit from a true MR system for Catmull-Clark surfaces, though: they represent a high-resolution mesh as a base mesh plus a complete set of details for all levels of the subdivision hierarchy. With a true MR system for Catmull-Clark surfaces, the storage requirement of such a mesh could be greatly reduced, and the transitions between levels would be more efficient as well.

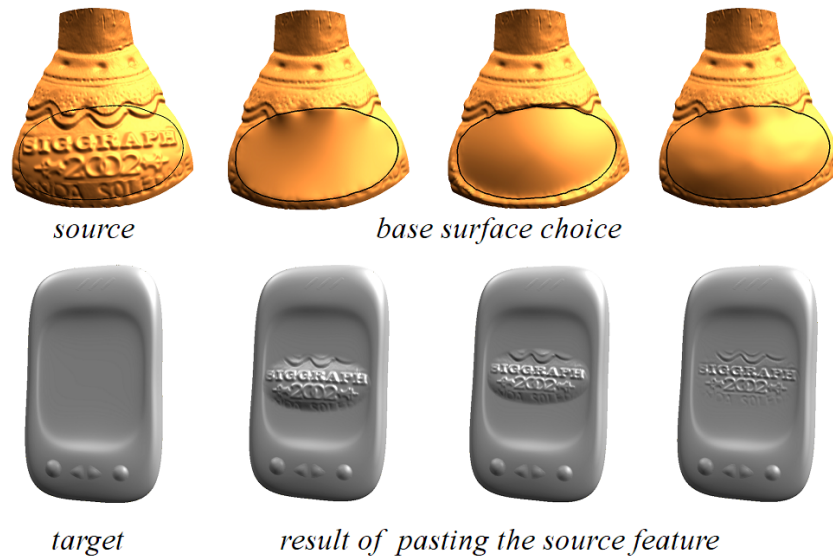


Figure 2.17: The cut-and-paste feature transfer system of Biermann et al. is a powerful application of MR to mesh editing. (Figure taken from [6].)

Feature transfer also has applications in non-graphical fields such as biometrics. Wecker et al. [46] use B-spline curve wavelets to extract the characteristics from iris images. These characteristics are then combined with those of other irises to create synthetic iris images. The problem of data synthesis is a very important one in biometrics, so extracting and transferring characteristics via MR is a great boon to synthesizing unique but plausible synthetic data.

Feature transfer is one way to edit a curve or mesh, by imbuing it with certain characteristics from another object. Other possibilities include creating details on an object interactively, or retaining high-resolution features of an object while making macroscopic changes at a low resolution. MR systems naturally facilitate these sorts of operations.

The power of MR editing was demonstrated early by Zorin et al. [49]. They



Figure 2.18: Zorin et al. developed a MR mesh editing system to facilitate both large-scale (the shrinking of the belly) and small-scale (the removal of the bellybutton). (Figure taken from [49].)

recognized that moving vertices at a coarse level creates macroscopic changes to a mesh, while moving vertices at a fine level creates microscopic changes. Together, these operations can provide a powerful editing system, as illustrated by Fig. 2.18. However, their method is not based on the reversing of subdivision rules, so the magnitudes of the details can often become large.

Kobbelt et al. [26] describe a system for similar editing operations, while losing the restriction of subdivision connectivity in the high-resolution mesh. The drawback of each of these methods is that the transitions between levels of the mesh hierarchy require complex computations; by a reverse subdivision/MR approach, the transitions between levels are much more efficient and easy to implement.

An important application of MR is compression. Decomposition produces a coarse model plus a set of detail coefficients; these details have a pure geometric interpretation, in that they quantify the difference between a subdivided (and there-

fore smooth) object and the detailed (non-smooth) object. However, a mesh may have regions that are approximately smooth and therefore have very small detail coefficients. By decomposing an object and discarding details below a certain threshold, the storage requirement of the object is reduced.

Khodakovsky et al. [25] describe a compression algorithm for irregular meshes. Their method is based on the observation that of the geometry, parameter, and connectivity information represented by a mesh, only the geometry information is important in terms of error reduction. By remeshing to a mesh with subdivision connectivity, the parameter and connectivity information is implied and can be eliminated except at the base level.

MR also has applications in rendering. Certain et al. [9] describe a dynamic surface viewer that uses MR techniques to balance between geometric detail, texture detail, and rendering efficiency. However, their method does not use low-error decomposition and therefore the low-resolution geometries produced by their system are of poor quality.

The remeshing approach of Lee et al. [30] was extended to the problem of morphing between two topologically equivalent meshes [29]. The remeshing stage is used to construct equivalent based domains for each input mesh (Fig. 2.19). Once a single base domain is established, intermediate meshes can be computed by combining the details from each source mesh.

An MR mesh representation allows for efficient level-of-detail control – both global changes as an object moves around a scene, and local changes as the visible part of an object changes over time. For instance, Hoppe [21] extended his progressive mesh technique to view-dependent rendering; see Fig. 2.20. The precom-

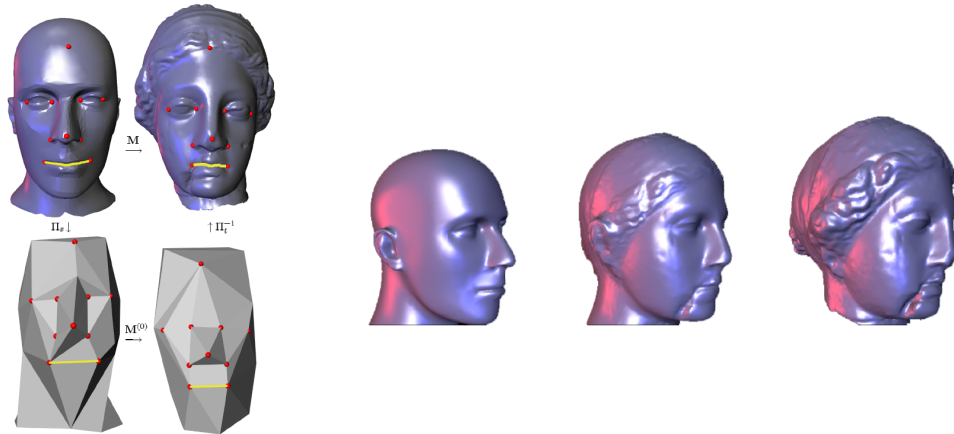


Figure 2.19: Lee et al. constructing equivalent base domains for each input mesh (left) to morph between them (right). (Figures taken from [29].)

puted progressive mesh data structure is well-suited to level-of-detail control, as the cost of increasing the resolution of a mesh is quite low.

In a similar vein, Azuma et al. [2] present a view-dependent rendering approach for meshes with Loop subdivision connectivity. From such a mesh, a wavelet decomposition is performed to reach a coarse mesh and wavelet coefficients. The wavelet coefficients are then added back in per-frame according to a screen space error metric that respects both geometry and texture quality. Their method is more efficient than Hoppe's, but has the additional semi-regular mesh requirement.

2.4 Conclusion

Multiresolution is a broad domain in computer graphics. It has applications in many different forms of modeling and rendering. As MR specifically relates to subdivision, many interesting and important uses have been demonstrated: feature transfer, hier-



Figure 2.20: View-dependent rendering of a progressive mesh: the original mesh (left) is rendered at full resolution only within a viewing frustum (center and right). (Figure taken from [21].)

archical editing, compression, level-of-detail, and morphing. However, many of these applications rely on computationally expensive algorithms for remeshing or building a mesh hierarchy. If robust, low-error wavelet approaches were available for a broader range of semi-regular meshes, the power of these applications would increase substantially.

In this thesis, a method for building MR systems for many subdivision schemes is presented. The work of Bartels and Samavati [3], Bertram [4], Li et al. [31], and Lanquetin and Neveu [28] are most related to the method and results presented in this work, and thus serve as valuable touchstones for evaluation and contextualizing our results.

Chapter 3

Method Overview

An intransigent property of subdivision and multiresolution is that subdivision increases the number of vertices in an object while decomposition reduces the number. To satisfy the storage constraint – which states that a coarsened mesh plus the detail terms requires no more storage space than the associated fine mesh – for semi-regular meshes, the inherent structure of the problem dictates the size of the detail “budget.” For example, Catmull-Clark subdivision of a cube (8 vertices) produces an object with 26 vertices. When the subdivided object is decomposed, exactly 8 of the 26 vertices must be replaced with coarse approximations. Thus there is a budget of 18 detail terms to spend. But how should these vertices and details be allocated?

The failing of many multiresolution approaches is that the storage constraint is ignored or left unsatisfied. Yet there is an inherent structure in subdivision schemes that can be exploited. In primal schemes, subdivision displaces existing (even) vertices and creates new (odd) vertices for each edge and possibly each face (Fig. 3.1). When such a mesh is then decomposed, the most obvious approach is to replace each even vertex with a coarse approximation. Then, the storage constraint indicates that the odd vertices should be replaced with detail terms.

For dual schemes such as Chaikin curves or Doo-Sabin surfaces, the distinction between even and odd vertices is less clear because each vertex of valence n is split into n new vertices during subdivision. However, if one of the split vertices is arbitrarily labeled as even (say, the left vertex in a Chaikin curve) and the remaining

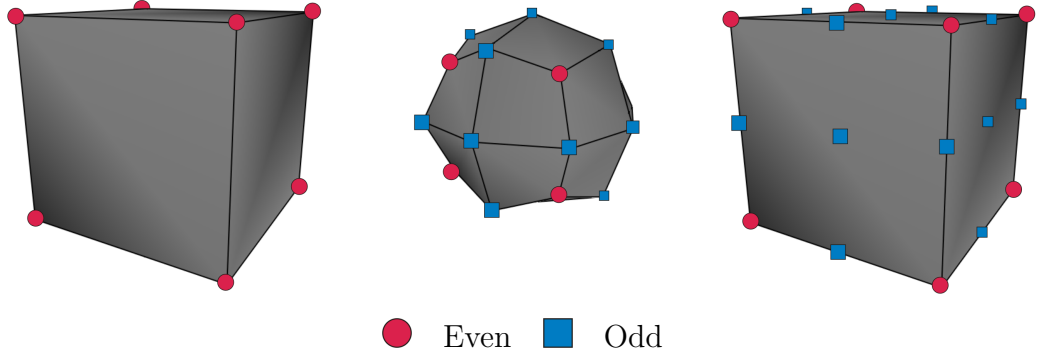


Figure 3.1: Subdivision creates a natural vertex partitioning for MR: (left) an arbitrary mesh contains only “even” vertices; (center) after subdivision, even vertices are moved and “odd” vertices are created; (right) when the object is later decomposed, it is natural to replace even vertices with a coarse approximation, and odd vertices with a detail term.

vertices as odd, a decomposition strategy of replacing even vertices with a coarse vertex and odd vertices with a detail term can again be employed.

The method proposed herein is based on this observation: for semi-regular meshes, the MR system will replace vertices labeled as “even” with a coarse approximation, and vertices labeled as “odd” with a detail term. With this strategy, the often-elusive storage constraint is satisfied immediately, but simultaneously a new question is raised: how can the original surface be reconstructed when only a partial set of details is being stored? More specifically, how can the detail term at an even vertex be determined from the stored details?

The answer to this question is the crux of the method: by *constraining the wavelets* (detail terms) so that the missing (even) wavelet coefficients are computable from stored (odd) coefficients. Furthermore, in the linear and local spirit of subdivision, the wavelet coefficient at an even vertex should be computable by a linear combination of a local neighborhood of the stored wavelets coefficients (Fig. 3.2).

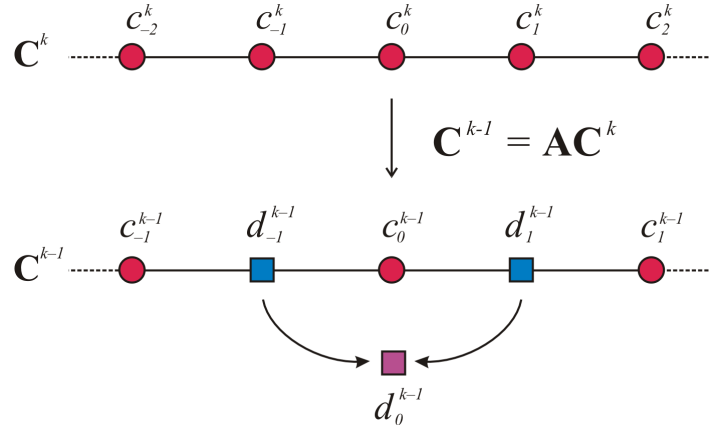


Figure 3.2: The wavelet constraint: when an object (top) is decomposed, details are stored only at the odd vertices (bottom). To ensure reconstructability, a constraint is enforced such that the missing detail d_0^{k-1} can be computed from stored details d_{-1}^{k-1} and d_1^{k-1} .

At this time, the goals stated in Sec. 1.2 should be revisited and framed in more precise terms. The original goals were: invertibility, reconstructability, the storage constraint, similitude, and locality/linearity. However, the concept of biorthogonality introduced in Sec. 2.3.2 can take the place of several of these goals. Recall that biorthogonality simply implies that $[\mathbf{P}|\mathbf{Q}]$ is invertible; equivalently, there exists an \mathbf{A} and \mathbf{B} such that Eqn. 2.7 is satisfied. So, a biorthogonal wavelet system must satisfy $\mathbf{AP} = \mathbf{I}$, $\mathbf{AQ} = \mathbf{0}$, $\mathbf{BP} = \mathbf{0}$, and $\mathbf{BQ} = \mathbf{I}$. Together, these four equations imply that a biorthogonal system satisfies invertibility (because $\mathbf{AP} = \mathbf{I}$), reconstructability (a full set of matrices exists), and the storage constraint, and linearity. Therefore, the goals to keep in mind when constructing an MR system are:

1. biorthogonality,
2. locality, and
3. similitude.

As will be evident in the following chapters, the outcome of the wavelet constraint is a biorthogonal and local set of *trial* filters. The final goal of similitude is not satisfied by the trial filters, however. To address this goal, the geometric nature of the detail terms is exploited. In broad terms, the details represent the difference between a fine mesh and the subdivision of a coarse approximating mesh, i.e. $\mathbf{D} = \mathbf{C}^k - \mathbf{P}\mathbf{C}^{k-1}$. Thus the magnitude and direction of a detail vector indicates how “far” the subdivision of a coarse approximation is from the original surface.

To reduce the error in the coarse approximation produced by the trial filters, the local neighborhood of details about a given coarse vertex is used to displace the vertex. The result of displacement is that the subdivision of the coarse vertices is closer than the original trial approximation.

Our method for constructing local multiresolution filters from a given subdivision scheme can be summarized as follows.

1. Enforce a wavelet constraint of the form

$$d_{even} = \alpha \sum_i d_i, \text{ for } i \in \text{ odd neighbors } . \quad (3.1)$$

The wavelet constraint is designed to be symmetric so that the resulting filters are symmetric.

2. Solve the wavelet constraint for α . This yields a form for computing even details from neighboring odd details, thus facilitating reconstruction and satisfying the storage constraint. This also defines the trial decomposition $\tilde{\mathbf{B}}$ filter.
3. Rearrange the wavelet constraint to express an even coarse vertex in terms of fine vertices, which defines a trial decomposition filter $\tilde{\mathbf{A}}$, Together with the $\tilde{\mathbf{B}}$

filter from the previous step, $\widetilde{\mathbf{Q}}$ can then be determined in such a way as to define a biorthogonal wavelet system.

4. Perform an optimization step to increase the similitude of the trial filters. If the trial filter produces a set of coarse points $\widetilde{\mathbf{C}}$, then the optimization step produces a set of displacement vectors Δ , such that

$$\mathbf{C}^{k-1} = \widetilde{\mathbf{C}}^{k-1} + \Delta . \quad (3.2)$$

This process is referred to as a *refinement* of the coarse data.

To elucidate this method, it will be applied to several subdivision schemes over the following chapters. In Chapter 4, a set of MR filters is constructed for cubic B-spline curve subdivision. In Chapter 5 the method is applied to Loop subdivision surfaces. Finally, Chapter 6 shows how to construct an MR system for semi-regular Catmull-Clark meshes.

Evaluating MR

Qualitatively, an MR system can be evaluated by looking at whether or not it satisfies the goals stated in Sec. 1.2. The goals of invertibility (really, biorthogonality), reconstructability, and the storage constraint can be evaluated qualitatively: the goal is either satisfied, or it isn't.

The similitude goal, however, can be satisfied to varying degrees. Because MR systems operate on graphical entities, their quality can be assessed subjectively by asking, does the coarse object “look like” the original object? (Or more precisely, does the subdivision of the coarse object look like the original object?) Such an

evaluation can distinguish a decent filter from a very poor one, but a more objective criteria is necessary when comparing filters of similar quality.

To objectively evaluate the quality of an MR system, the least-squares error metric is used to measure the error introduced by decomposition. More precisely, if a fine mesh \mathbf{C}^k is decomposed j times to a coarse mesh \mathbf{C}^{k-j} , then the error in \mathbf{C}^{k-j} is quantified by the difference between \mathbf{C}^k and $\mathbf{P}^j \mathbf{C}^{k-j}$ (\mathbf{C}^{k-j} *subdivided* – not reconstructed – j times). Formally, the least-squares error $E(\mathbf{C}^{k-j})$ is defined as

$$E(\mathbf{C}^{k-j}) = \sqrt{\|\mathbf{C}^k - \mathbf{P}^j \mathbf{C}^{k-j}\|^2} . \quad (3.3)$$

If $\mathbf{P}^j \mathbf{C}^{k-1} = \mathbf{C}^k$, then $E(\mathbf{C}^{k-j}) = 0$.

Chapter 4

Cubic B-spline Curves

The framework of Chapter 3 is best illustrated with subdivision curves. Many surface subdivisions schemes, such as Loop and Catmull-Clark, are extensions of subdivision curves, so the structure of the problems are similar. But curve schemes are easier to illustrate, have clean matrix representations, and have a large body of previous work to compare against.

Cubic B-spline subdivision curves, introduced in Sec. 2.2.1, are defined by the filter values $(\frac{1}{8}, \frac{1}{2}, \frac{3}{4}, \frac{1}{2}, \frac{1}{8})$. It is a primal (edge-split) scheme, with even vertices displaced and odd vertices created at the midpoint of each edge; see Fig. 2.4 for an illustration of this scheme.

To build an MR system for these curves, the strategy laid out in Chapter 3 is followed. After briefly revisiting the cubic B-spline subdivision filter in Sec. 4.1, in Sec. 4.2 the trial filters are built by setting up and solving the wavelet constraint. Section 4.3 deals with the extraordinary boundary cases. The error-reducing refinement step is discussed in Sec. 4.4 and 4.5. Finally, Sec. 4.6 collects the trial filters and refinements into single filters, and Sec. 4.7 summarizes the key points of the chapter.

For clarity of notation, the fine data will be denoted by $\mathbf{F} = \{f_0, f_1, \dots\}$ (rather than \mathbf{C}^k), and tilde notation will be used to denote the trial decomposition $\tilde{\mathbf{C}} = \{\tilde{c}_0, \tilde{c}_1, \dots\}$ of \mathbf{F} , as well as the subdivision $\tilde{\mathbf{F}} = \mathbf{P}\tilde{\mathbf{C}} = \{\tilde{f}_0, \tilde{f}_1, \dots\}$ of $\tilde{\mathbf{C}}$.

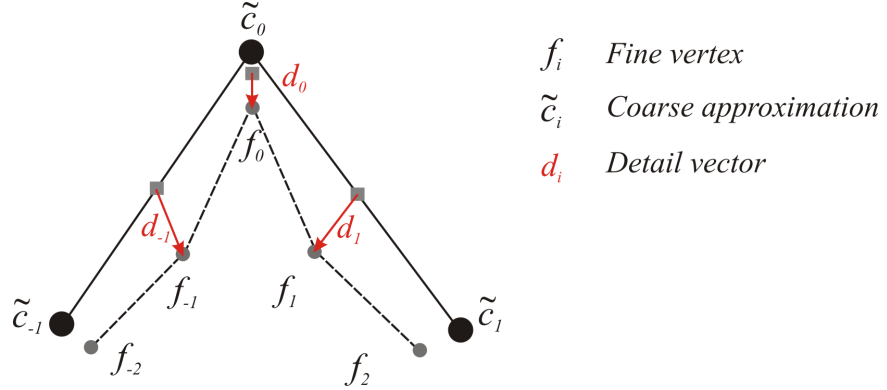


Figure 4.1: Local notation for cubic B-spline subdivision curves is centered about representative coarse vertex \tilde{c}_0 and fine vertex f_0 . During decomposition, even vertices (f_{-2} , f_0 , and f_2) are replaced with coarse vertices, while odd vertices f_{-1} and f_1 are replaced with details d_{-1} and d_1 . The wavelet constraint allows d_0 to be computed from d_{-1} and d_1 .

4.1 Subdivision Filter

For implementation purposes, it is best to have the subdivision mask expressed strictly in terms of the coarse vertices. For cubic B-spline, the coarse mask is

$$f_{2i} = \frac{3}{4}c_i + \frac{1}{8}(c_{i-1} + c_{i+1}) , \quad (4.1)$$

$$f_{2i+1} = \frac{1}{2}(c_i + c_{i+1}) . \quad (4.2)$$

It is useful to rewrite Eqn. 4.1 in a form that uses only c_i from the coarse level, eliminating $c_{i\pm 1}$. This is easily done by replacing c_{i-1} and c_{i+1} via Eqn. 4.2:

$$\begin{aligned} f_{2i-1} &= \frac{1}{2}(c_{i-1} + c_i) \rightarrow c_{i-1} = 2f_{2i-1} - c_i \\ f_{2i+1} &= \frac{1}{2}(c_i + c_{i+1}) \rightarrow c_{i+1} = 2f_{2i+1} - c_i \end{aligned}$$

These expressions for c_{i-1} and c_{i+1} can be used in Eqn. 4.1, yielding

$$f_{2i} = \frac{3}{4}c_i + \frac{1}{8}(2f_{2i-1} - c_i) + \frac{1}{8}(2f_{2i+1} - c_i)$$

$$f_{2i} = \frac{1}{2}c_i + \frac{1}{4}(f_{2i-1} + f_{2i+1}) . \quad (4.3)$$

4.2 Trial Filters

The first step of constructing an MR system is to set up the wavelet constraint, as per Eqn. 3.1. In this case, the representative detail vector d_0 , should be computable from the immediate odd neighbors, d_{-1} and d_1 (Fig. 4.1). Thus the wavelet constraint is expressed as

$$d_0 = \alpha(d_{-1} + d_1) . \quad (4.4)$$

Finding α will simultaneously satisfy the storage constraint and produce a trial filter.

A detail d_i is defined as the difference $f_i - \tilde{f}_i$ between the original fine data f_i and the subdivided coarse data, $\tilde{f} = \mathbf{P}\tilde{c}_i$. Using the mask of Eqn. 4.3, the details are

$$\begin{aligned} d_0 &= f_0 - \left(\frac{1}{4}\tilde{f}_{-1} + \frac{1}{2}\tilde{c}_0 + \frac{1}{4}\tilde{f}_1 \right) \\ d_{-1} &= f_{-1} - \tilde{f}_{-1} \\ d_1 &= f_1 - \tilde{f}_1 . \end{aligned}$$

Substituting these expressions into the wavelet constraint of Eqn. 4.4 yields

$$f_0 - \frac{1}{4}\tilde{f}_{-1} - \frac{1}{2}\tilde{c}_0 - \frac{1}{4}\tilde{f}_1 = \alpha(f_{-1} - \tilde{f}_{-1} + f_1 - \tilde{f}_1) . \quad (4.5)$$

A decomposition filter computes a coarse vertex position from a local neighborhood of fine vertices. In Eqn. 4.5, there are some terms that represent reconstructed data, the \tilde{f} terms. So choosing α to eliminate these terms will leave a form suitable

for a decomposition filter, while simultaneously satisfying the wavelet constraint. Both \tilde{f}_{-1} and \tilde{f}_1 appear in Eqn. 4.5 with coefficients of α on the left side and $\frac{1}{4}$ on the right; so, by inspection, setting

$$\alpha = \frac{1}{4} \quad (4.6)$$

will cancel these terms, leaving

$$f_0 - \frac{1}{2}\tilde{c}_0 = \frac{1}{4}(f_{-1} + f_1) ,$$

which simplifies to

$$\tilde{c}_0 = 2f_0 - \frac{1}{2}(f_{-1} + f_1) . \quad (4.7)$$

Equation 4.7 expresses coarse point \tilde{c}_0 in terms of fine points f_{-1} , f_0 , and f_1 . In other words, it represents a regular row of the trial $\widetilde{\mathbf{A}}$ filter. Treating f_0 as a representative fine vertex, Eqn. 4.7 can be compactly expressed as a vector $\{\dots, a_{-1}, a_0, a_1, \dots\}$ of decomposition coefficients forming a regular row of the trial filter $\widetilde{\mathbf{A}}$:

$$\frac{a_0}{2} \frac{a_{\pm 1}}{-\frac{1}{2}} \quad (4.8)$$

This 3-element filter is the same one as arrived at, via different means, by Bartels and Samavati [3].

Knowing α , Eqn. 4.4 determines $\widetilde{\mathbf{Q}}$, the contribution of the details \mathbf{D} to fine data \mathbf{F} . For even points, the two neighboring odd details determine the even detail based, so the row in $\widetilde{\mathbf{Q}}$ will contain $(\frac{1}{4}, \frac{1}{4})$. For odd points the detail is stored, i.e. the corresponding element in $\widetilde{\mathbf{Q}}$ should be 1. Thus a regular column of $\widetilde{\mathbf{Q}}$ will be $(\frac{1}{4}, 1, \frac{1}{4})$.

Since $\widetilde{\mathbf{A}}$ is specified, determining $\widetilde{\mathbf{B}}$ is straightforward. For a representative detail vector d_1 , Eqn. 4.7 can be employed to express the detail in terms of fine vertices f_i .

$$\begin{aligned}
 d_1 &= f_1 - \widetilde{f}_1 \\
 &= f_1 - \left(\frac{1}{2}\widetilde{c}_0 + \frac{1}{2}\widetilde{c}_1 \right) \\
 &= f_1 - \frac{1}{2} \left(2f_0 - \frac{1}{2}f_{-1} - \frac{1}{2}f_1 \right) - \frac{1}{2} \left(2f_2 - \frac{1}{2}f_1 - \frac{1}{2}f_3 \right) \\
 d_1 &= \frac{1}{4}f_{-1} - f_0 + \frac{3}{2}f_1 - f_2 + \frac{1}{4}f_3 .
 \end{aligned}$$

This forms a regular row of $\widetilde{\mathbf{B}}$. Since only the details at odd points need to be computed, $\widetilde{\mathbf{B}}$ is fully determined.

In the next section the matrix forms of $\widetilde{\mathbf{Q}}$, $\widetilde{\mathbf{A}}$, and $\widetilde{\mathbf{B}}$ will be summarized, including a treatment of the boundary cases for open curves.

4.3 Boundary Filters

To have a complete set of multiresolution filters applicable to either open or closed curves, a set of special-case filters for boundaries should be developed. The block matrix notation introduced in Sec. 2.2.1 is be used here to differentiate between regular and boundary filters.

For open cubic B-spline curves, the \mathbf{P} filter is

$$\begin{bmatrix} \mathbf{P}_s \\ \mathbf{P}_r \\ \mathbf{P}_e \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \dots \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & \dots \\ 0 & \frac{3}{4} & \frac{1}{4} & 0 & 0 & 0 & \dots \\ 0 & \frac{3}{16} & \frac{11}{16} & \frac{1}{8} & 0 & 0 & \dots \\ \hline 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & \dots \\ 0 & 0 & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & 0 & \dots \\ & & & \vdots & & & \\ \hline \dots & 0 & 0 & \frac{1}{8} & \frac{11}{16} & \frac{3}{16} & 0 \\ \dots & 0 & 0 & 0 & \frac{1}{4} & \frac{3}{4} & 0 \\ \dots & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \dots & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

To build boundary filters, the wavelet constraint can again be used. Let f_0 , f_1 , f_2 , and f_3 represent the four fine vertices created by the boundary subdivision mask,

$$\begin{aligned} f_0 &= c_0 , \\ f_1 &= \frac{1}{2}c_0 + \frac{1}{2}c_1 , \\ f_2 &= \frac{3}{4}c_1 + \frac{1}{4}c_2 , \\ f_3 &= \frac{3}{16}c_1 + \frac{11}{16}c_2 + \frac{1}{8}c_3 . \end{aligned}$$

Unfortunately these vertices have no clear distinction between even and odd vertices. For this discussion f_0 , f_1 , and f_3 will be considered as even and f_2 as odd. Figure 4.2 illustrates this notation.

Each of these boundary vertices is considered in order. Because $f_0 = c_0$, the best

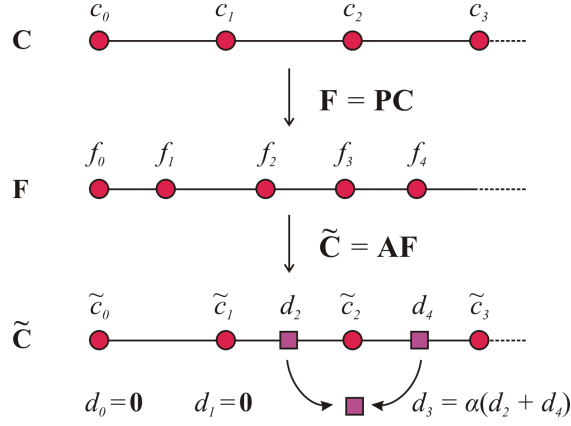


Figure 4.2: The boundary setting for cubic B-splines. Coarse vertices c_0, \dots, c_3 contribute to special boundary vertices f_0, \dots, f_4 during subdivision, which is different than the even-odd rhythm in the regular case. During decomposition, f_0, f_1 , and f_3 are considered even, and f_2 is considered odd.

coarse approximation is $\tilde{c}_0 = f_0$. After subdivision, $\tilde{f}_0 = \tilde{c}_0 = f_0$; therefore there is no need to store or compute a detail for this vertex, as $d_0 = \tilde{f}_0 - f_0 = \mathbf{0}$.

The best coarse approximation \tilde{c}_1 for f_1 also has a simple form due to f_0 not moving during decomposition or reconstruction. If we want to satisfy $d_1 = \mathbf{0}$, then it should be the case that $\frac{1}{2}(\tilde{c}_0 + \tilde{c}_1) = f_1$. This can be rearranged to a decomposition mask:

$$\begin{aligned} \frac{1}{2}(\tilde{c}_0 + \tilde{c}_1) &= f_1 \\ \tilde{c}_1 &= 2f_1 - \tilde{c}_0 \\ &= 2f_1 - f_0 . \end{aligned}$$

Again there is no need to store or compute a detail, because

$$\begin{aligned} d_1 &= f_1 - \tilde{f}_1 \\ &= f_1 - \frac{1}{2}\tilde{c}_0 - \frac{1}{2}\tilde{c}_1 \end{aligned}$$

$$\begin{aligned}
&= f_1 - \frac{1}{2}f_0 - \frac{1}{2}(2f_1 - f_0) \\
&= \mathbf{0} .
\end{aligned}$$

The third boundary vertex, f_2 , is designated as an odd vertex, so it will be replaced with a detail vector $d_2 = f_2 - \tilde{f}_2$.

The final boundary case for decomposition is f_3 . Unlike f_0 and f_1 , this is a non-trivial case. To find a decomposition filter, a new instance of the wavelet constraint is required. Using the notation of Fig. 4.2, f_3 will be replaced by coarse vertex \tilde{c}_2 , and its associated detail vector is d_3 . During decomposition, the left and right neighbors f_2 and f_4 are replaced by details d_2 and d_4 respectively. Thus the wavelet constraint for this case is

$$d_3 = \alpha'(d_2 + d_4) , \quad (4.9)$$

where

$$\begin{aligned}
d_2 &= f_2 - \left(\frac{3}{4}\tilde{c}_1 + \frac{1}{4}\tilde{c}_2 \right) , \\
d_3 &= f_3 - \left(\frac{3}{16}\tilde{c}_1 + \frac{11}{16}\tilde{c}_2 + \frac{1}{8}\tilde{c}_3 \right) , \\
d_4 &= f_4 - \left(\frac{1}{2}\tilde{c}_2 + \frac{1}{2}\tilde{c}_3 \right) .
\end{aligned}$$

Using these expressions, Eqn. 4.9 becomes

$$\begin{aligned}
f_3 - \frac{3}{16}\tilde{c}_1 - \frac{11}{16}\tilde{c}_2 - \frac{1}{8}\tilde{c}_3 &= \alpha'(f_2 - \frac{3}{4}\tilde{c}_1 - \frac{1}{4}\tilde{c}_2) + \alpha'(f_4 - \frac{1}{2}\tilde{c}_2 - \frac{1}{2}\tilde{c}_3) \\
&= \alpha'(f_2 + f_4) - \alpha'\frac{3}{4}\tilde{c}_1 - \alpha'\frac{3}{4}\tilde{c}_2 - \alpha'\frac{1}{8}\tilde{c}_3 . \quad (4.10)
\end{aligned}$$

The goal is an expression for \tilde{c}_2 in terms of f_i s, so α' is chosen to eliminate \tilde{c}_1 and \tilde{c}_3 . For \tilde{c}_1 , Eqn. 4.10 indicates

$$\frac{3}{16} - \alpha'\frac{3}{4} = 0 \quad \rightarrow \quad \alpha' = \frac{1}{4} ,$$

while cancellation of \tilde{c}_3 dictates that

$$\frac{1}{8} - \alpha' \frac{1}{2} = 0 \quad \rightarrow \quad \alpha' = \frac{1}{4}.$$

Fortunately, both yield the same value for α' . Using this value in Eqn. 4.10 cancels \tilde{c}_1 and \tilde{c}_2 , leaving

$$\begin{aligned} f_3 - \frac{11}{16}\tilde{c}_2 &= \frac{1}{4}(f_2 + f_4) - \frac{1}{4} \cdot \frac{3}{4}\tilde{c}_2 \\ \left(\frac{11}{16} - \frac{3}{16}\right)\tilde{c} &= f_3 - \frac{1}{4}(f_2 + f_4) \\ \tilde{c}_2 &= 2f_3 - \frac{1}{2}(f_2 + f_4) \end{aligned} \tag{4.11}$$

Interestingly, the boundary wavelet constraint yields the same results as the regular case, for both α and the decomposition filter.

As a final step, d_2 can be expressed in terms of fine vertices f_i .

$$\begin{aligned} d_2 &= f_2 - \left(\frac{3}{4}\tilde{c}_1 - \frac{1}{4}\tilde{c}_2\right) \\ &= f_2 - \frac{3}{4}(2f_1 - f_0) - \frac{1}{4}\left(\frac{1}{2}f_2 + 2f_3 - \frac{1}{2}f_4\right) \\ &= \frac{3}{4}f_0 - \frac{3}{2}f_1 + \frac{9}{8}f_2 - \frac{1}{2}f_3 + \frac{1}{8}f_4 \end{aligned}$$

The result of this analysis is summarized in the block matrices below.

$$\begin{bmatrix} \widetilde{\mathbf{A}}_s \\ \widetilde{\mathbf{A}}_r \\ \widetilde{\mathbf{A}}_e \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \cdots \\ -1 & 2 & 0 & 0 & 0 & 0 & \cdots \\ \hline 0 & \frac{-1}{2} & 2 & \frac{-1}{2} & 0 & 0 & \cdots \\ & & & \vdots & & & \\ \hline \cdots & 0 & 0 & 0 & 0 & 2 & -1 \\ \cdots & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.12}$$

$$\begin{bmatrix} \widetilde{\mathbf{B}}_s \\ \widetilde{\mathbf{B}}_r \\ \widetilde{\mathbf{B}}_e \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & \frac{-3}{2} & \frac{9}{8} & \frac{-1}{2} & \frac{1}{8} & 0 & \dots \\ \dots & \frac{1}{4} & -1 & \frac{3}{2} & -1 & \frac{1}{4} & \dots \\ & & & \vdots & & & \\ \dots & 0 & \frac{1}{8} & \frac{-1}{2} & \frac{9}{8} & \frac{-3}{2} & \frac{3}{4} \end{bmatrix} \quad (4.13)$$

$$\begin{bmatrix} \widetilde{\mathbf{Q}}_s \\ \widetilde{\mathbf{Q}}_r \\ \widetilde{\mathbf{Q}}_e \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 1 & 0 & 0 & 0 & \dots \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 & \dots \\ & & \vdots & & \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \end{bmatrix} . \quad (4.14)$$

4.4 Refinement

The trial filter of Eqn. 4.7 is biorthogonal, i.e. Eqn. 2.7 is satisfied. However, when an object \mathbf{F} that is *not* the product of subdivision is decomposed with the trial filter $\widetilde{\mathbf{A}}$, the coarse approximation often bears little resemblance to the original object; see Fig. 4.3. More formally, the least-squares error $E(\mathbf{C})$ becomes large, especially under repeated application of the decomposition filter. While there is no choice of \mathbf{A} for which $E = 0$ (except for trivial cases where $\mathbf{F} = \mathbf{PC}$ for some \mathbf{C}), there are some choices that are better than others.

To improve the trial filter, a local refinement of the coarse vertices is considered, such that after refinement the local error is reduced. That is, a set of refinement vectors $\Delta = \{\delta_0, \dots, \delta_n\}$ representing the per-vertex displacements of $\widetilde{\mathbf{C}} = \{\widetilde{c}_0, \dots, \widetilde{c}_n\}$

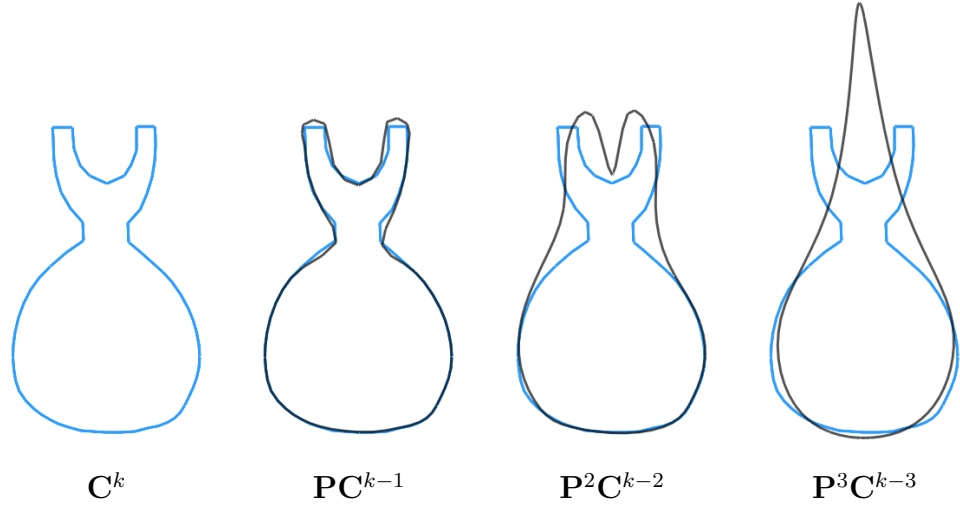


Figure 4.3: While the trial decomposition filter satisfies biorthogonality, successive applications of the filter causes high error in the coarse approximations. A vase object (left) is decomposed with the trial filter (left to right) once, twice, and three times, and then subdivided to the original's resolution. After three levels of decomposition, the error has spiraled out of control.

needs to be determined. Figure 4.4 illustrates the refinement procedure. Δ should be chosen such that

$$E(\tilde{\mathbf{C}} + \Delta) < E(\tilde{\mathbf{C}}) .$$

This expresses the *global* error in the coarse points. However, enforcing a global reduction in error is both difficult and incongruous with the local nature of subdivision methods.

Instead, the impact of refinement is considered only on a local neighborhood about \tilde{c}_0 . For cubic B-splines, this means that δ_0 , the refinement applied to \tilde{c}_0 , influences a 5-element (denoted by $5n$) neighborhood of the fine points, due to the length of the subdivision filter. To simplify the problem, analysis is first limited to a 3-element neighborhood (denoted $3n$).

4.4.1 Width-3 Neighborhood

When \tilde{c}_0 is refined to $c_0 = \tilde{c}_0 + \delta_0$, there is a corresponding change at the finer level.

Consider $\tilde{f}_0 = \frac{3}{4}\tilde{c}_0 + \frac{1}{8}(\tilde{c}_{-1} + \tilde{c}_1)$. After refinement, this becomes

$$\begin{aligned}\tilde{f}'_0 &= \frac{3}{4}(\tilde{c}_0 + \delta_0) + \frac{1}{8}(\tilde{c}_{-1} + \tilde{c}_1) \\ &= \tilde{f}_0 + \frac{3}{4}\delta_0 .\end{aligned}$$

Similarly, the left and right fine vertices are impacted by δ_0 according to the subdivision weights.

$$\begin{aligned}\tilde{f}'_{-1} &= \tilde{f}_{-1} + \frac{1}{2}\delta_0 \\ \tilde{f}'_1 &= \tilde{f}_1 + \frac{1}{2}\delta_0 .\end{aligned}$$

The goal of refining \tilde{c}_0 is to reduce the local least squares error. This error metric is determined by the differences $f_i - \tilde{f}_i$, so it should be the case that – locally – $f_i - \tilde{f}'_i < f_i - \tilde{f}_i$. By reducing the local error of each coarse vertex, a global reduction in error is expected. Recalling that $d_i = f_i - \tilde{f}_i$, then the local error $E(\delta_0)$ after refinement is expressed as

$$\begin{aligned}E(\delta_0) &= \left\| f_{-1} - (\tilde{f}_{-1} + \frac{1}{2}\delta_0) \right\|^2 + \left\| f_0 - (\tilde{f}_0 + \frac{3}{4}\delta_0) \right\|^2 + \left\| f_1 - (\tilde{f}_1 + \frac{1}{2}\delta_0) \right\|^2 \\ &= \left\| d_{-1} - \frac{1}{2}\delta_0 \right\|^2 + \left\| d_0 - \frac{3}{4}\delta_0 \right\|^2 + \left\| d_1 - \frac{1}{2}\delta_0 \right\|^2 .\end{aligned}\tag{4.15}$$

Equation 4.15 can be simplified by recalling that $\|\mathbf{v}\|^2 = \mathbf{v}^T \mathbf{v}$. Therefore

$$\|d_i - y\delta_0\|^2 = \|d_i\|^2 - 2y(d_i)^T \delta_0 + y^2 \|\delta_0\|^2 .\tag{4.16}$$

By Eqn. 4.16, Eqn. 4.15 simplifies to

$$\begin{aligned}E(\delta_0) &= \frac{17}{16}\|\delta_0\|^2 - \left(d_{-1} + \frac{3}{2}d_0 + d_1\right)^T \delta_0 + \|d_{-1}\|^2 + \|d_0\|^2 + \|d_1\|^2 \\ &= a\|\delta_0\|^2 - \mathbf{v}^T \delta_0 + b ,\end{aligned}\tag{4.17}$$

where

$$\begin{aligned} a &= \frac{17}{16} , \\ \mathbf{v} &= d_{-1} + \frac{3}{2}d_0 + d_1 , \\ b &= \|d_{-1}\|^2 + \|d_0\|^2 + \|d_1\|^2 . \end{aligned}$$

Because Eqn. 4.17 is a simple quadratic function of δ_0 , the optimization can be solved analytically. At a minimum of the function, the first derivative is zero, or

$$E'(\delta_0) = 2a\delta_0 - \mathbf{v} = 0$$

Therefore

$$\begin{aligned} 2a\delta_0 - \mathbf{v} &= 0 \\ \delta_0 &= \frac{\mathbf{v}}{2a} . \end{aligned} \tag{4.18}$$

Note that this is a minimum, because $E''(\delta_0) = 2a > 0$. In fact, it is a unique global minimum, which follows from the fact that the Hessian $\nabla^2 E$ is positive-definite [12].

Substituting the proper values of \mathbf{v} and a yields a closed-form expression for δ_0 .

$$\begin{aligned} \delta_0 &= \frac{\mathbf{v}}{2a} \\ &= \frac{1}{2} \cdot \frac{16}{17} \left(d_{-1} + \frac{3}{2}d_0 + d_1 \right) \\ &= \frac{8}{17} \left(d_{-1} + \frac{3}{2}d_0 + d_1 \right) \\ \delta_0^{3n} = \delta_0 &= \frac{11}{17} (d_{-1} + d_1) , \end{aligned} \tag{4.19}$$

The final simplification step, which allows δ_0 to be computed from the stored detail information, follows from Eqn. 4.4.

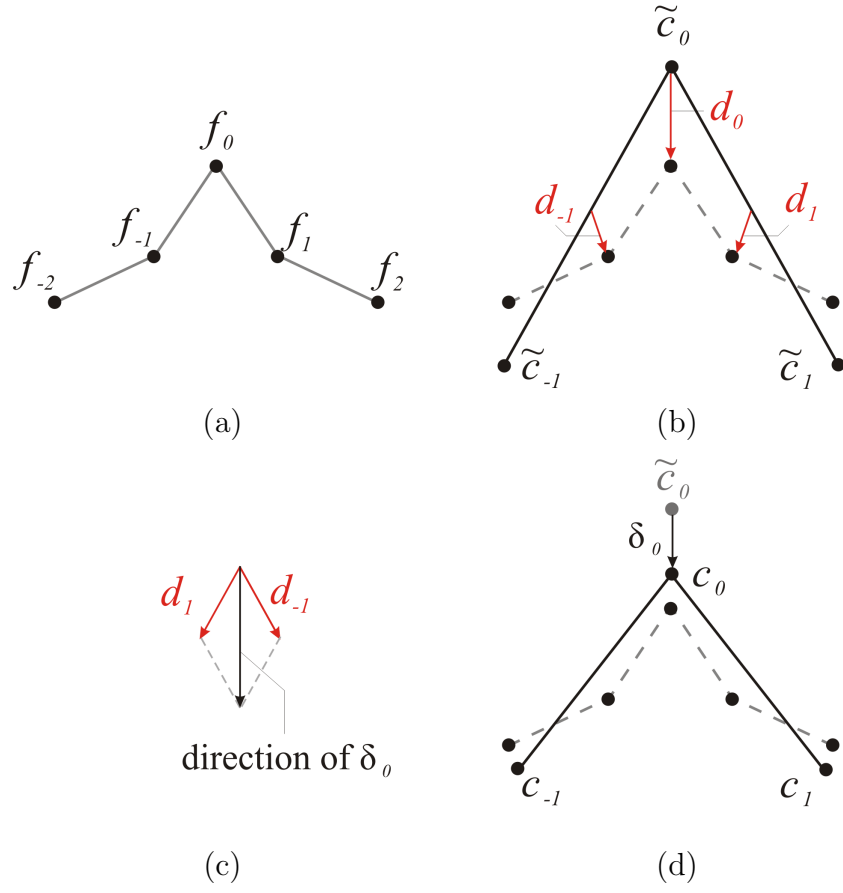


Figure 4.4: Refinement of the trial vertices: (a) fine data \mathbf{F} ; (b) decomposition of \mathbf{F} produces $\tilde{\mathbf{C}}$ and \mathbf{D} ; (c) a refinement vector δ_0 is computed from a local set of details; (d) after refinement, the local error about \tilde{c}_0 is reduced.

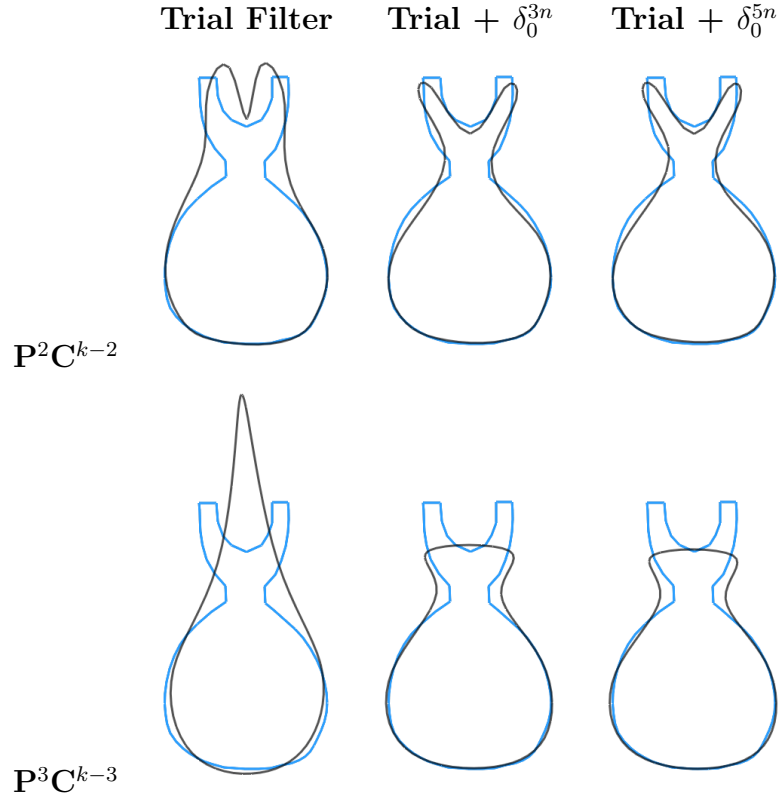


Figure 4.5: Refining the trial filter (left) by Eqn. 4.19 (center) or Eqn. 4.21 (right) reduces the error introduced by decomposition.

Note that δ_0 points in the direction of $d_{-1} + d_1$. Because the detail terms, roughly speaking, point in the direction of the reconstructed object relative to the subdivision of the coarse object, this result fits with intuition. Figure 4.4 illustrates the geometric intuition behind refinement.

4.4.2 Width-5 Neighborhood

If the impact of δ_0 is considered in a wider, 5-element neighborhood, then the local error function becomes

$$E(\delta_0) = \left\| d_{-2} - \frac{1}{8}\delta_0 \right\|^2 + \left\| d_{-1} - \frac{1}{2}\delta_0 \right\|^2 + \left\| d_0 - \frac{3}{4}\delta_0 \right\|^2 + \left\| d_1 - \frac{1}{2}\delta_0 \right\|^2 + \left\| d_2 - \frac{1}{8}\delta_0 \right\|^2 \quad (4.20)$$

Again this can be simplified to the form $E(\delta_0) = a\|\delta_0\|^2 - \mathbf{v}^T\delta_0 + b$, where

$$\begin{aligned} a &= \frac{35}{32} , \\ \mathbf{v} &= \frac{3}{2}d_0 + (d_{-1} + d_1) + \frac{1}{4}(d_{-2} + d_2) , \\ b &= \|d_{-2}\|^2 + \|d_{-1}\|^2 + \|d_0\|^2 + \|d_1\|^2 + \|d_2\|^2 . \end{aligned}$$

As before, the solution is $\delta_0 = \frac{\mathbf{v}}{2a}$, or

$$\begin{aligned} \delta_0 &= \frac{1}{2} \cdot \frac{32}{35} \left(\frac{3}{2}d_0 + (d_{-1} + d_1) + \frac{1}{4}(d_{-2} + d_2) \right) \\ &= \frac{16}{35} \left(\frac{3}{2}d_0 + (d_{-1} + d_1) + \frac{1}{4}(d_{-2} + d_2) \right) \\ \delta_0^{5n} = \delta_0 &= \frac{1}{35} (d_{-3} + 23d_{-1} + 23d_1 + d_3) . \end{aligned} \tag{4.21}$$

The final step replaces the even details d_0 , d_{-2} , and d_2 with their odd neighbors, according to Eqn. 4.4.

As shown in Fig. 4.5, both refinement steps – δ_0^{3n} and δ_0^{5n} – dramatically improve the similitude of the trial filter.

4.5 Partial Refinement

In developing the refinement vectors, one important aspect has been neglected: the refinements are not independent. The computation of an optimal displacement for each coarse vertex assumes that all other coarse vertices will remain unchanged, i.e. will not be displaced. This assumption is not true though: every coarse point will be displaced by its own refinement vector.

For an illustration of why this is problematic, consider Fig. 4.6. In (a), a small section of a fine curve \mathbf{F} is shown, and the corresponding trial vertices in $\tilde{\mathbf{C}}$; when

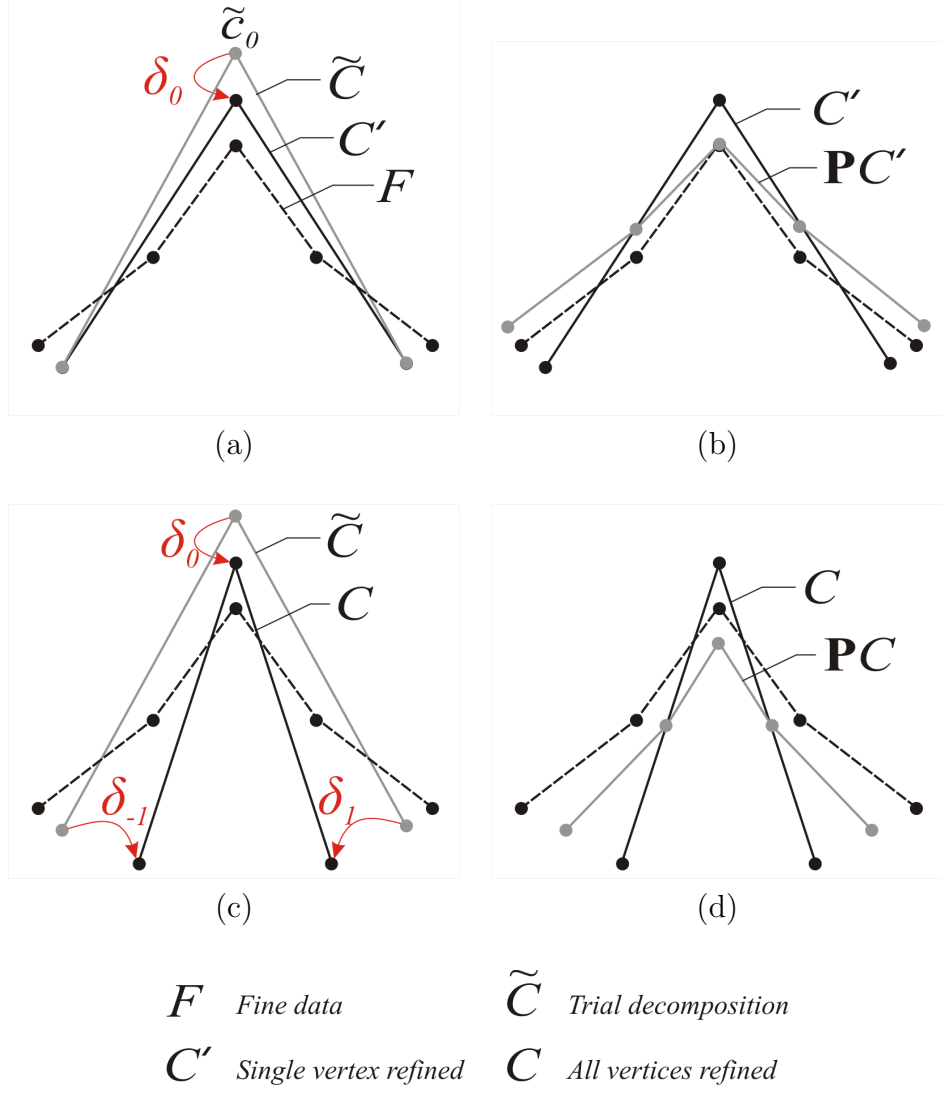


Figure 4.6: Local vs. global effects of refinement (the dashed line represents the original fine data): if only the central vertex \tilde{c}_0 is displaced by δ_0 (a), then after refinement, the local error $E(\mathbf{PC}')$ is minimized (b); but, if all coarse vertices are refined (c), the non-independence of the refinements yields a non-minimal error $E(\mathbf{PC})$ (d).

these coarse vertices are subdivided without detail information, the error in $\mathbf{P}\tilde{\mathbf{C}}$ is quite large. Now consider how the refinement step would impact this curve. In (b), the effect of a single refinement step on $\tilde{c}_0 \in \tilde{\mathbf{C}}$ is shown. If the neighbors \tilde{c}_{-1} and \tilde{c}_1 are indeed stationary, then the subdivision of the vertices in (b) will result in a truly minimized error, as shown in (c). However, in reality \tilde{c}_{-1} and \tilde{c}_1 will be refined along with \tilde{c}_0 , as shown in (d), resulting in \mathbf{C} . When \mathbf{C} is subdivided, the error is no longer minimized because the neighbors of each coarse vertex have been displaced, as in (e).

Intuitively, the interdependence of the refinements will cause a net “overshooting” effect, i.e. too much displacement. This is visible in Fig. 4.6: the subdivision of the trial points, shown in (a), lies on one side of the fine data, while the subdivision of the refined points, shown in (e), lies on the opposite side. Therefore, a partial refinement strategy is considered; instead of $c_0 = \tilde{c}_0 + \delta_0$, each refinement should be softened by setting $c_0 = \tilde{c}_0 + \mu\delta_0$, where $0 \leq \mu \leq 1$ is some scalar to be determined.

It is not clear how μ might be chosen, outside of trial-and-error. In general, a perfect optimization would involve a different value of μ for each coarse vertex, allowing for sensitivity to the local feature scale. However, this would lead to a decomposition mask that changes from vertex to vertex, which does not lead to a very efficient algorithm. Thus a single value for μ is a necessary simplification. An analytic approach to the selection of μ might be considered, but any local approach would still suffer from interdependence problems. Most importantly, however, it is necessary to have a method that can be extended to mesh schemes.

Here a voting strategy is considered, based on the observation that \tilde{c}_0 wants to take the full refinement step, while neighbors \tilde{c}_{-1} and \tilde{c}_1 want \tilde{c}_0 to not move at all so

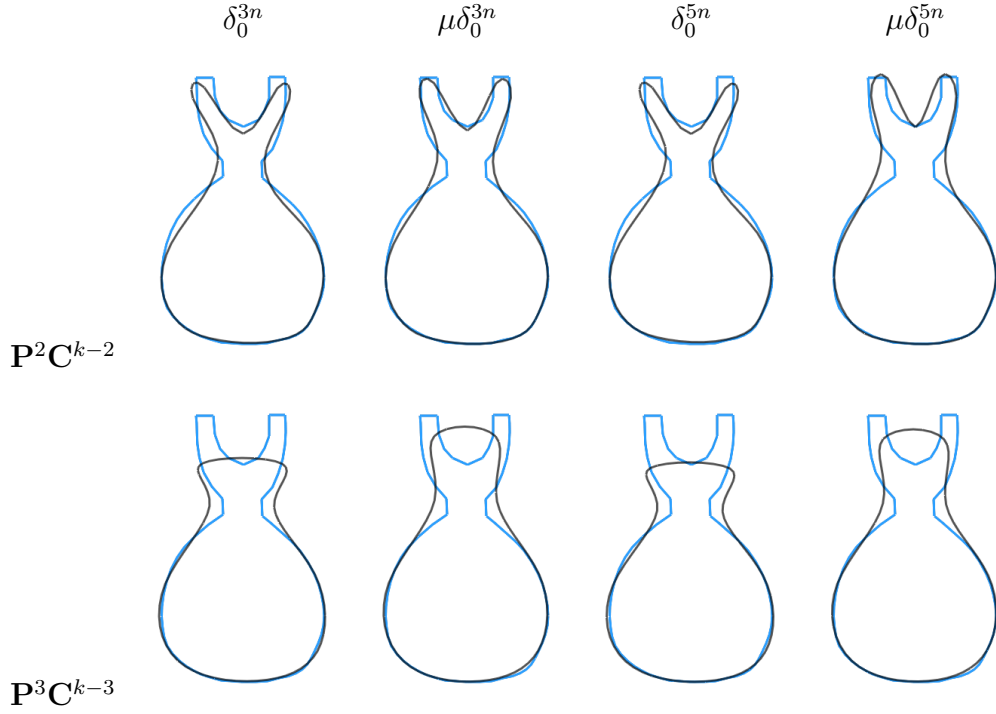


Figure 4.7: Partial refinement softens each local refinement to account for the interdependence of each refinement.

that they can take their own full refinement step. In other words, the central vertex would vote for $\mu = 1$, while its neighbors would vote for $\mu = 0$. However, their votes should not carry equal weight, because the position of c_0 is much more important to \tilde{c}_0 than to its neighbors. In fact, based on the subdivision filter \mathbf{P} , \tilde{c}_0 contributes $\frac{3}{4}$ to f_0 , and only $\frac{1}{8}$ to \tilde{c}_{-1} and \tilde{c}_1 . If these weights are adopted in the voting scheme, then

$$\mu = \left(\frac{3}{4}\right) 1 + 2 \left(\frac{1}{8}\right) 0 = \frac{3}{4}.$$

Note that μ is equal to the central weight of the subdivision filter.

This value of μ is supported by empirical data. For a sample of around ten curves \mathbf{F}_i of varying complexity, a global least squares solution for \mathbf{C}_i was found by casting

the refinement process as an iterative method [23]. The globally optimal solutions were then interpreted as refinements Δ_i of the trial decompositions $\tilde{\mathbf{C}}_i$, and over the samples it was found that the global solution implied a partial refinement with $\mu \approx 0.724$.

Using $\mu = \frac{3}{4}$, c_0 becomes $c_0 = \tilde{c}_0 + \frac{3}{4}\delta_0$. For the 3-element neighborhood, Eqn. 4.19 defines the full refinement step, and the partial refinement is

$$\delta_0^{3n} \leftarrow \frac{3}{4}\delta_0^{3n} = \frac{33}{68}(d_{-1} + d_1) . \quad (4.22)$$

For the wider neighborhood, where δ_0 is given by Eqn. 4.21, the same partial refinement weight can be used, yielding

$$\delta_0^{5n} \leftarrow \frac{3}{4}\delta_0^{5n} = \frac{3}{140}(d_{-3} + d_{-1} + d_1 + d_3) . \quad (4.23)$$

Figure 4.7 shows the improvement realized by partial refinement. And, as shown in Sec. 7.1, the partially refined filters perform as well as the near-minimum norm filters of Bartels and Samavati [3]. So although the voting scheme is not particularly rigorous, the results give credence to the method.

4.6 Closed-Form Filters

There is a question of where the refinement process fits into the usual multiresolution framework of decomposing with \mathbf{A} and \mathbf{B} and reconstructing with \mathbf{P} and \mathbf{Q} . One possibility is to treat the refinement process as a separate step in decomposition, which is later undone before the normal reconstruction process.

Recall, however, that throughout development of the trial filters and refinement vectors, local and linear formulations have been sought. The linearity of the trial

filter allows it to be encapsulated by $\widetilde{\mathbf{A}}$.

In the refinement stage, the displacements δ_0 are computed from a linear combination of a local set of details. The wavelet constraint allows δ_0 to be computed directly from the odd details stored in \mathbf{D} . Thus there is some matrix \mathbf{L} such that the set of refinement vectors is defined by $\Delta = \mathbf{LD}$.

This fact allows us to encapsulate decomposition by $\widetilde{\mathbf{A}}$ and refinement by Δ in a single operation:

$$\begin{aligned}
 \mathbf{C} &= \widetilde{\mathbf{C}} + \Delta \\
 &= \widetilde{\mathbf{A}}\mathbf{F} + \mathbf{LD} \\
 &= \widetilde{\mathbf{A}}\mathbf{F} + \mathbf{L}\widetilde{\mathbf{B}}\mathbf{F} \\
 &= (\widetilde{\mathbf{A}} + \mathbf{L}\widetilde{\mathbf{B}})\mathbf{F} \\
 &= \mathbf{A}\mathbf{F} ,
 \end{aligned}$$

where

$$\mathbf{A} = \widetilde{\mathbf{A}} + \mathbf{L}\widetilde{\mathbf{B}} \tag{4.24}$$

is a closed-form decomposition filter.

The undoing of refinement and subsequent reconstruction by \mathbf{P} and \mathbf{Q} can similarly be combined into a single operation.

$$\begin{aligned}
 \mathbf{F} &= \mathbf{P}\widetilde{\mathbf{C}} + \widetilde{\mathbf{Q}}\mathbf{D} \\
 &= \mathbf{P}(\mathbf{C} - \Delta) + \widetilde{\mathbf{Q}}\mathbf{D} \\
 &= \mathbf{P}(\mathbf{C} - \mathbf{LD}) + \widetilde{\mathbf{Q}}\mathbf{D} \\
 &= \mathbf{PC} + (\widetilde{\mathbf{Q}} - \mathbf{PL})\mathbf{D} \\
 &= \mathbf{PC} + \mathbf{QD} ,
 \end{aligned}$$

where

$$\mathbf{Q} = \widetilde{\mathbf{Q}} - \mathbf{P}\mathbf{L} \quad (4.25)$$

is the closed-form reconstruction filter. \mathbf{Q} incorporates the undoing of the refinement, $\widetilde{\mathbf{C}} = \mathbf{C} - \Delta$. Note that it is unnecessary to alter $\widetilde{\mathbf{B}}$ filter to reflect the refinement, i.e. the details do not need to be recomputed as $\mathbf{D} = \mathbf{F} - \mathbf{P}\mathbf{C}$. Instead \mathbf{Q} is responsible for correctly interpreting the unaltered details.

The matrix \mathbf{L} *lifts* the trial filters, just as in Sweldens' lifting scheme (Eqn. 2.8). Thus the refinement procedure can be viewed as a way to determine the lifting matrix. Furthermore, because the trial filters are biorthogonal, the refined filters must also be biorthogonal [44].

4.6.1 Width-7 Filter

For the 3-element neighborhood, the matrix \mathbf{L} is defined by Eqn. 4.22

$$\begin{bmatrix} \mathbf{L}_s \\ \mathbf{L}_r \\ \mathbf{L}_e \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ \hline \frac{33}{68} & \frac{33}{68} & 0 & 0 & \dots \\ 0 & \frac{33}{68} & \frac{33}{68} & 0 & \dots \\ & & \vdots & & \\ \hline \dots & 0 & 0 & 0 & 0 \\ \dots & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Note that it is unnecessary to lift the boundary points, because their details are always zero.

By the forms of $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ given in Eqn. 4.12 and 4.13 respectively, \mathbf{L} lifts $\widetilde{\mathbf{A}}$ to $\mathbf{A} = \widetilde{\mathbf{A}} + \mathbf{L}\widetilde{\mathbf{B}}$:

$$\begin{bmatrix} \mathbf{A}_s \\ \mathbf{A}_r \\ \mathbf{A}_e \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \frac{99}{272} & \frac{-99}{136} & \frac{91}{544} & \frac{173}{136} & \frac{157}{544} & \frac{-33}{68} & \frac{33}{272} & 0 & 0 & \dots \\ \hline 0 & 0 & \frac{33}{272} & \frac{-33}{68} & \frac{95}{272} & \frac{35}{34} & \frac{95}{272} & \frac{-33}{68} & \frac{33}{272} & \dots \\ & & & \vdots & & & & & & \\ \hline \dots & 0 & 0 & \frac{33}{272} & \frac{-33}{68} & \frac{157}{544} & \frac{173}{136} & \frac{91}{544} & \frac{-99}{136} & \frac{99}{272} \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -1 \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Note that the regular filter has widened width-3 trial filter to a width-7 filter, which can be compactly expressed by a filter vector.

$$\begin{matrix} a_0 & a_{\pm 1} & a_{\pm 2} & a_{\pm 3} \\ \frac{35}{34} & \frac{95}{272} & \frac{-33}{68} & \frac{33}{272} \end{matrix} . \quad (4.26)$$

Similarly, \mathbf{L} lifts $\widetilde{\mathbf{Q}}$ (Eqn. 4.14) to $\mathbf{Q} = \widetilde{\mathbf{Q}} - \mathbf{P}\mathbf{L}$:

$$\begin{bmatrix} \mathbf{Q}_s \\ \mathbf{Q}_r \\ \mathbf{Q}_e \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ \frac{239}{272} & \frac{-33}{272} & 0 & 0 & \dots \\ \frac{-91}{1088} & \frac{-157}{1088} & \frac{-33}{544} & 0 & \dots \\ \hline \frac{-33}{136} & \frac{35}{68} & \frac{-33}{136} & 0 & \dots \\ \frac{-33}{544} & \frac{-95}{544} & \frac{-95}{544} & \frac{-33}{544} & \dots \\ & & \vdots & & \\ \hline \dots & 0 & \frac{-33}{544} & \frac{-157}{1088} & \frac{-91}{1088} \\ \dots & 0 & 0 & \frac{-33}{272} & \frac{239}{272} \\ \dots & 0 & 0 & 0 & 0 \\ \dots & 0 & 0 & 0 & 0 \end{bmatrix}.$$

4.6.2 Width-11 Filter

For the 5-element neighborhood, \mathbf{L} is defined by Eqn. 4.21.

$$\begin{bmatrix} \mathbf{L}_s \\ \mathbf{L}_r \\ \mathbf{L}_e \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \frac{69}{140} & \frac{69}{140} & \frac{3}{140} & 0 & 0 & 0 & \dots \\ \hline \frac{3}{140} & \frac{69}{140} & \frac{69}{140} & \frac{3}{140} & 0 & 0 & \dots \\ 0 & \frac{3}{140} & \frac{69}{140} & \frac{69}{140} & \frac{3}{140} & 0 & \dots \\ & & \vdots & & & & \\ \hline \dots & 0 & 0 & 0 & \frac{3}{140} & \frac{69}{140} & \frac{69}{140} \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Using \mathbf{L} to lift $\widetilde{\mathbf{A}}$ produces

$$\begin{bmatrix} \mathbf{A}_s \\ \mathbf{A}_r \\ \mathbf{A}_e \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \frac{207}{560} & \frac{-207}{280} & \frac{199}{1120} & \frac{353}{280} & \frac{49}{160} & \frac{-18}{35} & \frac{87}{560} & \frac{-3}{140} & \frac{3}{560} & 0 & 0 & 0 & 0 & \dots \\ \frac{9}{560} & \frac{-9}{280} & \frac{33}{224} & \frac{-141}{280} & \frac{409}{1120} & \frac{71}{70} & \frac{103}{280} & \frac{-18}{35} & \frac{87}{560} & \frac{-3}{140} & \frac{3}{560} & 0 & 0 & \dots \\ \hline 0 & 0 & \frac{3}{560} & \frac{-3}{140} & \frac{87}{560} & \frac{-18}{35} & \frac{103}{280} & \frac{71}{70} & \frac{103}{280} & \frac{-18}{35} & \frac{87}{560} & \frac{-3}{140} & \frac{3}{560} & \dots \\ & & & & & & & \vdots & & & & & & \\ \hline \dots & 0 & 0 & \frac{3}{560} & \frac{-3}{140} & \frac{87}{560} & \frac{-18}{35} & \frac{103}{280} & \frac{71}{70} & \frac{409}{1120} & \frac{-141}{280} & \frac{33}{224} & \frac{-9}{280} & \frac{9}{560} \\ \dots & 0 & 0 & 0 & 0 & \frac{3}{560} & \frac{-3}{140} & \frac{87}{560} & \frac{-18}{35} & \frac{49}{160} & \frac{353}{280} & \frac{199}{1120} & \frac{-207}{280} & \frac{207}{560} \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -1 \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Here the larger support of \mathbf{L} has widened the width-3 trial filter to a width-11 filter.

Similarly, $\widetilde{\mathbf{Q}}$ is lifted to

$$\begin{bmatrix} \mathbf{Q}_s \\ \mathbf{Q}_r \\ \mathbf{Q}_e \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \frac{491}{560} & \frac{-69}{560} & \frac{-3}{560} & 0 & 0 & 0 & \dots \\ \frac{-41}{448} & \frac{-337}{2240} & \frac{-171}{2240} & \frac{-3}{1120} & 0 & \dots & \\ \frac{-87}{1120} & \frac{-103}{560} & \frac{-103}{560} & \frac{-87}{1120} & \frac{-3}{1120} & 0 & \dots \\ \hline \frac{-3}{280} & \frac{-9}{35} & \frac{71}{140} & \frac{-9}{35} & \frac{-3}{280} & 0 & \dots \\ \frac{-3}{1120} & \frac{-87}{1120} & \frac{-103}{560} & \frac{-103}{560} & \frac{-87}{1120} & \frac{-3}{1120} & \dots \\ \vdots & & & & & & \\ \hline \dots & 0 & \frac{-3}{1120} & \frac{-87}{1120} & \frac{-103}{560} & \frac{-103}{560} & \frac{-87}{1120} \\ \dots & 0 & 0 & \frac{-3}{1120} & \frac{-171}{2240} & \frac{-337}{2240} & \frac{-41}{448} \\ \dots & 0 & 0 & 0 & \frac{3}{560} & \frac{-69}{560} & \frac{491}{560} \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The associated filter vector for \mathbf{A} in the 5-element neighborhood is

$$\begin{bmatrix} a_0 & a_{\pm 1} & a_{\pm 2} & a_{\pm 3} & a_{\pm 4} & a_{\pm 5} \\ \frac{71}{70} & \frac{103}{280} & -\frac{18}{35} & \frac{87}{560} & -\frac{3}{140} & \frac{3}{560} \end{bmatrix}. \quad (4.27)$$

4.7 Conclusion

Following the procedure outlined in Chapter 3, three multiresolution systems were developed for regular cubic B-spline curves and surfaces. The trial filter, characterized by its width-three decomposition filter (Eqn. 4.8), satisfies most of the goals of multiresolution: biorthogonality, reconstructability, and the storage constraint. However, the very important goal of producing a low error coarse approximation is not

satisfied, especially under repeated applications of the filter.

To reduce the error produced by the trial filter, a refinement procedure was investigated whereby each coarse vertex was displaced according to the magnitude and direction of neighboring detail vectors. To have a tractable approach, the error before and after refinement is considered on a local scale. The width of the subdivision filter dictates that refinement impacts a local neighborhood of 5 vertices. In this case, δ_0 depends on four stored details (Eqn. 4.21) and a width-11 closed-form decomposition filter results in the regular case (Eqn. 4.27). When a smaller 3-element neighborhood is used, δ_0 depends on only two details (Eqn. 4.19) and has an associated width-7 decomposition filter (Eqn. 4.26).

After the coarse vertices are refined, an MR system that produces satisfactorily low-error coarse approximations is reached. And as proved by Sweldens [44], the linear nature of refinement and the biorthogonal nature of trial filter implies that the closed-form refined filters are also orthogonal. Thus the final, refined construction satisfies each of the goals stated in Chapter 3.

Chapter 5

Loop Surfaces

Now that the framework of Chapter 3 has been demonstrated with a simple curve subdivision scheme, it can be extended to a more complex surface subdivision scheme. Loop subdivision is a scheme that operates strictly on triangle meshes and has C^2 continuity at regular (valence 6) vertices. Loop subdivision is quite popular for triangle meshes because of its smoothness and also its relative ease of implementation.

To build an MR system for Loop subdivision, trial filters are constructed by setting up and solving the wavelet constraint (Sec. 5.2). In Sec. 5.3 and 5.4, the error of the trial filters is reduced by refining the coarse vertices. Section 5.5 discusses how to handle boundary cases, while Sec. 5.6 briefly discusses implementation considerations. Finally, Sec. 5.7 summarizes the key results of the chapter.

5.1 Subdivision Filter

Loop subdivision is a face-splitting scheme for triangle meshes [33]. A new (odd) vertex is created along each edge, and each existing (even) vertex is displaced. Consider a vertex c_0 of valence n , as depicted in Fig. 5.1. The subdivision filter for a representative even vertex c_0 is

$$f_0 = (1 - n\beta)c_0 + \beta \sum_{i=1}^n c_i , \quad (5.1)$$

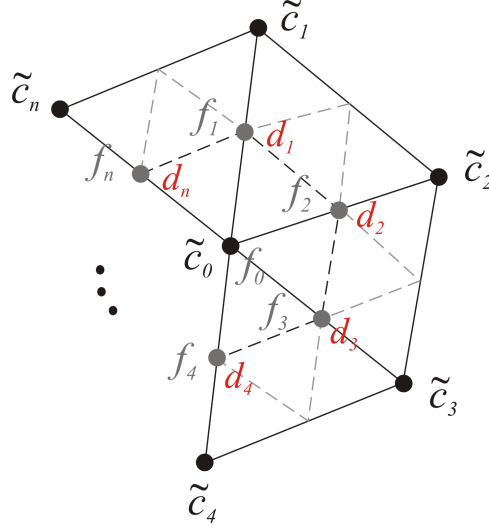


Figure 5.1: Local notation for Loop subdivision is centered about representative coarse vertex c_0 and fine vertex f_0 . During decomposition, even vertex f_0 is replaced with coarse vertex \tilde{c}_0 , while odd vertices $f_{i \neq 0}$ are replaced with details.

where

$$\beta = \frac{1}{n} \left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4} \cos \left(\frac{2\pi}{n} \right) \right)^2 \right) .$$

Odd vertices are defined by

$$f_i = \frac{3}{8}(c_0 + c_i) + \frac{1}{8}(c_{i-1} + c_{i+1}) , \quad j = 1, 2, \dots ,$$

where periodic indexing is used (that is, $c_{1-1} = c_n$ and $c_{n+1} = c_1$). Together, these filters define the \mathbf{P} matrix for Loop subdivision. However, matrix notation for mesh subdivision schemes is not as elegant as the curve case. Because of the irregular and complex connections between vertices, the subdivision matrix will typically not have a banded structure (i.e. a structure in which each column is a shifted version of the first column), although the local nature of subdivision will produce a sparse matrix (that is, one with mostly zero entries).

Solving the wavelet constraint is easier when the subdivision filter (Eqn. 5.1) for c_0 is expressed in terms of fine vertices f_i rather than coarse vertices c_i . The general form of such a filter is $f_0 = Ac_0 + B \sum f_i$; A and B can be determined from the regular subdivision filters.

$$\begin{aligned}
f_0 &= Ac_0 + B \sum_i f_i \\
&= Ac_0 + B \sum \left(\frac{3}{8}c_0 + \frac{3}{8}c_i + \frac{1}{8}c_{i-1} + \frac{1}{8}c_{i+1} \right) \\
&= Ac_0 + B \left(\frac{3}{8} \sum c_0 + \frac{3}{8} \sum c_i + \frac{1}{8} \sum c_{i-1} + \frac{1}{8} \sum c_{i+1} \right) \\
&= Ac_0 + \frac{3}{8}nBc_0 + \frac{5}{8}B \sum c_i \\
&= \left(A + \frac{3}{8}nB \right) c_0 + \frac{5}{8}B \sum c_i .
\end{aligned} \tag{5.2}$$

Equation 5.1 dictates that the weight $\frac{5}{8}B$ applied to $\sum c_i$ should be β , or

$$\frac{5}{8}B = \beta \quad \rightarrow \quad B = \frac{8}{5}\beta .$$

Similarly, the weight $A + \frac{3}{8}nB$ of c_0 should be equal to $1 - n\beta$, so

$$A + \frac{3}{8}nB = 1 - n\beta \quad \rightarrow \quad A = 1 - \frac{8}{5}n\beta .$$

Therefore, the subdivision mask for even vertices based on fine odd vertices is

$$f_0 = \left(1 - \frac{8}{5}n\beta \right) c_0 + \frac{8}{5}\beta \sum f_i . \tag{5.3}$$

5.2 Trial Filters

The local neighborhood of a vertex c_0 in a Loop mesh is usually referred to as an n -ring, where n denotes the number of edge traversals to get to a vertex from c_0 . For instance, the 2-ring of c_0 is the set of all vertices $c_i \neq c_0$ that are at most two edge

traversals from c_0 . The smallest local neighborhood is the 1-ring. For example, in Fig. 5.1, $\{f_1, \dots, f_n\}$ constitute the 1-ring of f_0 .

During decomposition, the 1-ring of an even vertex f_0 will be replaced with details $\{d_1, \dots, d_n\}$; to satisfy the storage constraint, detail d_0 must be computable from this neighborhood of stored details. Thus the wavelet constraint for Loop subdivision is

$$d_0 = \alpha(d_1 + d_2 + \dots + d_n) . \quad (5.4)$$

Let the *subdivided* (not reconstructed) position of a fine vertex f_i be denoted by \tilde{f}_i . Then, using the fine-vertex subdivision mask of Eqn. 5.3, the details for the local neighborhood can be expressed by

$$\begin{aligned} d_0 &= f_0 - \left(\left(1 - \frac{8}{5}n\beta\right) \tilde{c}_0 + \frac{8}{5}\beta \sum_{i=1}^n \tilde{f}_i \right) , \\ d_1 &= f_1 - \tilde{f}_1 , \\ &\vdots \\ d_n &= f_n - \tilde{f}_n . \end{aligned}$$

Equation 5.4 then becomes

$$f_0 - \left(1 - \frac{8}{5}n\beta\right) \tilde{c}_0 - \frac{8}{5}\beta \sum_{i=1}^n \tilde{f}_i = \alpha \sum_{i=1}^n f_i - \alpha \sum_{i=1}^n \tilde{f}_i .$$

A proper selection of α will eliminate the \tilde{f}_i terms. Equating the coefficient from the left side with the coefficient from the right,

$$\begin{aligned} \frac{8}{5}\beta \sum \tilde{f}_i &= \alpha \sum \tilde{f}_i \\ \alpha &= \frac{8}{5}\beta . \end{aligned} \quad (5.5)$$

Eliminating the \tilde{f}_i terms leaves

$$f_0 - \left(1 - \frac{8}{5}n\beta\right) \tilde{c}_0 = \alpha \sum f_i$$

$$\begin{aligned}
f_0 - (1 - n\alpha) \tilde{c}_0 &= \alpha \sum f_i \\
\tilde{c}_0 &= \frac{1}{1 - n\alpha} f_0 - \frac{\alpha}{1 - n\alpha} \sum_{i=1}^n f_i .
\end{aligned} \tag{5.6}$$

Equation 5.6 represents the trial decomposition filter $\tilde{\mathbf{A}}$.

With $\alpha = \frac{8}{5}\beta$ known, the wavelet constraint of Eqn. 5.4 becomes

$$d_0 = \frac{8}{5}\beta \sum_{i=1}^n d_i . \tag{5.7}$$

From Eqn. 5.7, the contribution of \mathbf{D} to \mathbf{F} , i.e. $\tilde{\mathbf{Q}}$, is defined. For even vertices, the details d_i surrounding a vertex should be summed and scaled by α ; for odd vertices, the associated detail is explicitly stored.

The final MR filter is $\tilde{\mathbf{B}}$, which computes a set of details $\mathbf{D} = \tilde{\mathbf{B}}\mathbf{F}$. By the wavelet constraint details need to be computed and stored for odd vertices. Consider a representative odd detail, d_1 :

$$\begin{aligned}
d_1 &= f_1 - \tilde{f}_1 \\
&= f_1 - \frac{3}{8}(\tilde{c}_0 + \tilde{c}_1) - \frac{1}{8}(\tilde{c}_2 + \tilde{c}_n) .
\end{aligned} \tag{5.8}$$

It is feasible to find an expression for d_1 that depends only on fine data f_i by replacing each \tilde{c}_i according to Eqn. 5.6. However, that would involve enumerating the 1-ring of four different vertices and determining where they overlap. It is more practical to simply use the form above to compute the odd details after computing the coarse data.

5.3 Refinement

The trial filter of Eqn. 5.6 is biorthogonal and satisfies the storage constraint. However, as Fig. 5.2 illustrates, the trial filter produces high-error coarse approximations.

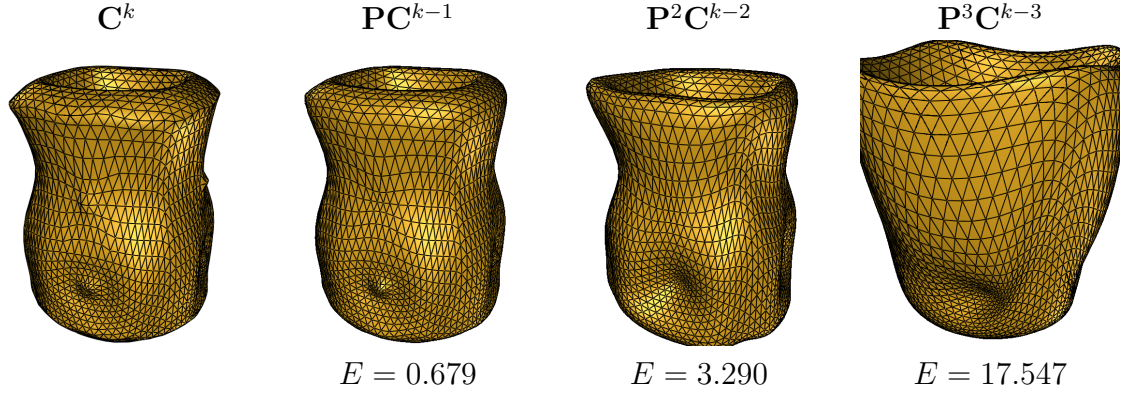


Figure 5.2: The trial decomposition filter produces high error, especially under repeated applications. A simple model (left) is decomposed (left to right) once, twice, and three times, then subdivided without details to the original resolution.

To alleviate this problem, the local error of \tilde{c}_0 can be reduced by a refinement procedure, $c_0 = \tilde{c}_0 + \delta_0$. Then, δ_0 should minimize

$$E(\delta_0) = \|d_0 - (1 - n\beta)\delta_0\|^2 + \left\|d_1 - \frac{3}{8}\delta_0\right\|^2 + \dots + \left\|d_n - \frac{3}{8}\delta_0\right\|^2. \quad (5.9)$$

The form of $E(\delta_0)$ given by Eqn. 5.9 follows from the Loop subdivision mask: c_0 contributes $1 - n\beta$ to \tilde{f}_0 and $\frac{3}{8}$ to $\tilde{f}_1, \dots, \tilde{f}_n$.

Simplifying Eqn. 5.9 yields a form that can be solved analytically.

$$E(\delta_0) = a\|\delta_0\|^2 - \mathbf{v}^T\delta_0 + b,$$

where

$$\begin{aligned} a &= (1 - n\beta)^2 + \frac{9}{64}n, \\ \mathbf{v} &= 2(1 - n\beta)d_0 + 2 \cdot \frac{3}{8} \sum_{i=1}^n d_i, \\ b &= \sum_{i=0}^n \|d_i\|^2. \end{aligned}$$

The minimum of $E(\delta_0)$ occurs at

$$\delta_0 = \frac{\mathbf{v}}{2a}$$

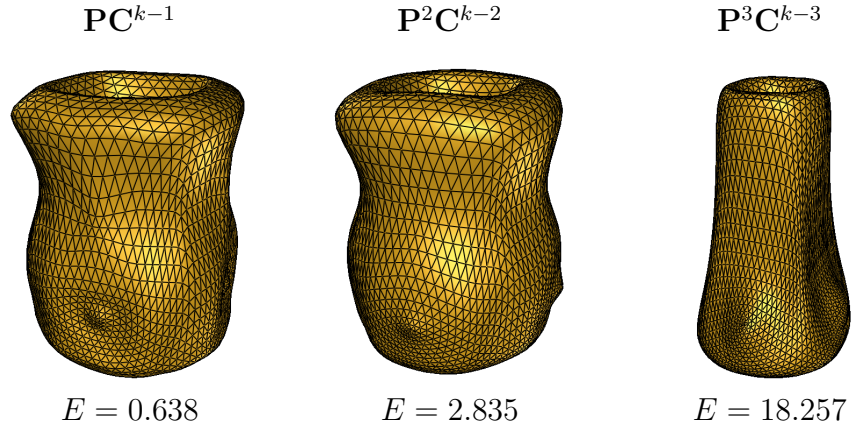


Figure 5.3: Refinement improves the trial filter displacing each coarse vertex according to the direction and magnitude of the surrounding details. The original model \mathbf{C}^k of Fig. 5.2 is decomposed and then subdivided once (left), twice (center), and three times (right). However, because of the interdependence of the refinements, the error is actually higher than the trial filter error.

$$\begin{aligned}
 &= \frac{2(1 - n\beta)d_0 + 2 \cdot \frac{3}{8} \sum_{i=1}^n d_i}{2 \left((1 - n\beta)^2 + \frac{9}{64}n \right)} \\
 &= \frac{(1 - n\beta)\alpha \sum_{i=1}^n d_i + \frac{3}{8} \sum_{i=1}^n d_i}{(1 - n\beta)^2 + \frac{9}{64}n} \\
 \delta_0 &= \kappa \sum_{i=1}^n d_i, \tag{5.10}
 \end{aligned}$$

where

$$\kappa = \frac{(1 - n\beta)\alpha + \frac{3}{8}}{(1 - n\beta)^2 + \frac{9}{64}n}.$$

Equation 5.7 is used to eliminate rewrite d_0 in terms of neighbors $d_{i \neq 0}$.

Figure 5.3 shows the marked improvement of the trial filter realized by refinement. The improvement is especially noticeable after two applications of the filter: where the trial filter blew up, the refined filter maintains a shape reasonably close to the original.

5.4 Partial Refinement

In Sec. 4.5, it was observed that each local refinement minimizes the local error only if no other refinements are performed. When all coarse vertices are refined, the assumption of stationary neighbors is violated and the local refinements are no longer optimal. So, although refinement does reduce the global error, the refinement steps are often too large.

To account for the interdependence of each refinement, a voting scheme was considered, in which each vertex affected by a local refinement “voted” for their preferred refinement step size. In practice, this produced filters that behaved very closely with established near-minimum-norm filters.

Using the voting strategy to dampen the refinement of Eqn. 5.10, \tilde{c}_0 wants to take the full refinement step (i.e. votes for $1 \times \delta_0$), while its neighbors $\tilde{c}_{i \neq 0}$ want \tilde{c}_0 to not move at all (i.e. vote for $0 \times \delta_0$). The central vertex votes for $\mu = 1$ with a weight of $1 - n\beta$, while the first-ring neighbors vote for $\mu = 0$ with a weight of β . Thus

$$\mu = (1 - n\beta) 1 + (n\beta) 0 = 1 - n\beta ,$$

and δ_0 becomes

$$\delta_0 = (1 - n\beta) \kappa \sum_{i=1}^n d_i . \quad (5.11)$$

Figure 5.4 compares the full refinement of Eqn. 5.10 with the partial refinement of Eqn. 5.11. While the full refinement improves a great deal on the trial filter, partial refinement offers an even greater reduction in error.

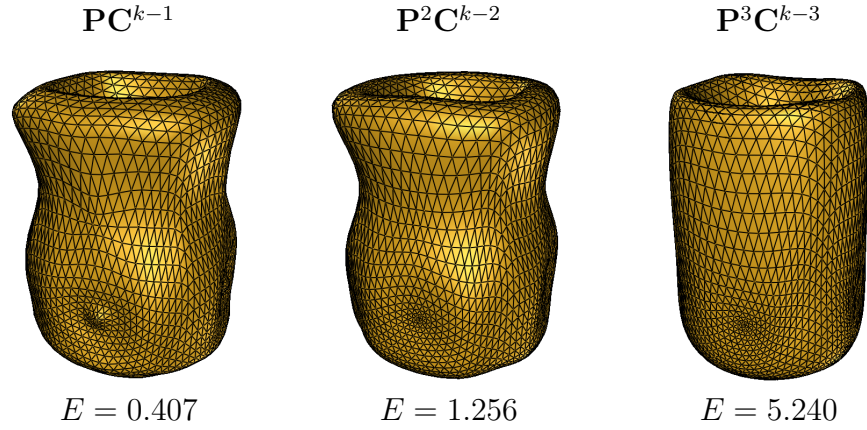


Figure 5.4: Partial refinement improves the trial filter even more by accounting for the interdependence of each local refinement.

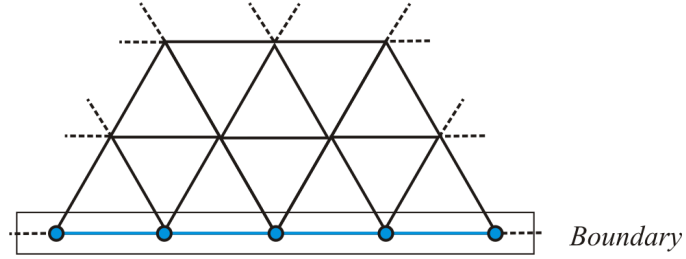


Figure 5.5: Boundary vertices and edges in a Loop model can be decomposed with cubic B-spline filters.

5.5 Boundary Filters

A subdivision mask can only be applied when a complete neighborhood exists. For interior vertices, a full neighborhood is always defined, as well as for interior edges. If there is a boundary in a mesh, however, then vertices and edges that make up the boundary do not have full neighborhoods. Special boundary masks must be defined to handle such cases. See Fig. 5.5.

Because Loop subdivision surfaces are a generalization of cubic B-spline curves, boundary vertices are subdivided according to cubic B-spline subdivision. Thus for

multiresolution boundary filters, a cubic B-spline multiresolution should be used, such as the systems described in Sec. 4.6. In particular, the regular filters \mathbf{A}_r , \mathbf{B}_r , and \mathbf{Q}_r are used along any continuous boundary, while the boundary cases $\mathbf{A}_{s/e}$, $\mathbf{B}_{s/e}$, and $\mathbf{Q}_{s/e}$ can be applied to corner points (although corner points are difficult to automatically detect in a mesh).

5.6 Closed Form

The partial refinement vectors are computed as a linear combination of the odd details surrounding a representative vertex. So it is possible to write all refinement vectors as a vector Δ such that $\Delta = \mathbf{L}\mathbf{D}$ for some matrix \mathbf{L} , where \mathbf{L} is determined by the refinement stage (Eqn. 5.11). \mathbf{L} can then be used to lift $\widetilde{\mathbf{A}}$ to find the closed-form decomposition filter.

However, a side-effect of refinement is a widening of our filter's closed form. Recall that the width-3 trial filter for cubic B-splines became filters of width 7 and 11 after lifting $\widetilde{\mathbf{A}}$ by \mathbf{L} (Sec. 4.6). This widening arises because computation of the refinement vector incorporates information from surrounding details, which themselves incorporate information from a wider neighborhood around the representative vertex.

For surface subdivision schemes, the concept of a subdivision *filter* is replaced by the concept of a subdivision *mask*. This is necessary because the interactions between vertices is much more complicated than a simple curve. It is possible to build a subdivision matrix \mathbf{P} to apply Loop subdivision to a mesh, but it would not be regular or banded like in the curve case: the weights applied to each vertex

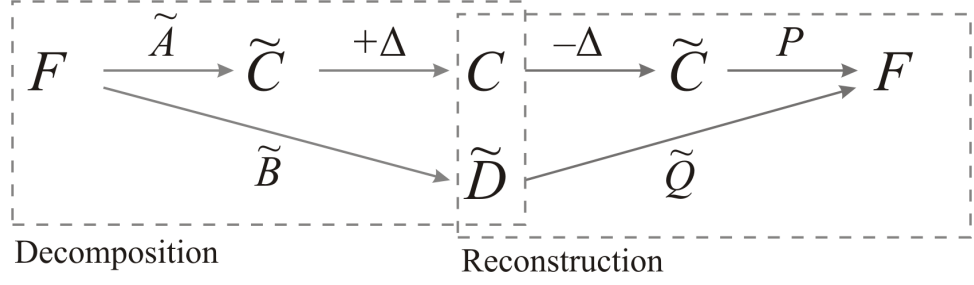


Figure 5.6: The multiresolution framework with a refinement step included. After decomposing \mathbf{F} with $\tilde{\mathbf{A}}$, the refinement vectors Δ are added to produce \mathbf{C} . When reconstructing, the refinement vectors are first subtracted.

depend on the valence, and there is no regularity in the vertex connectivity. One could also build \mathbf{A} , \mathbf{B} , and \mathbf{Q} matrices and then lift them by some refinement matrix \mathbf{L} , but the matrices would be specific to a certain mesh, and extracting a set of lifted masks from the matrices would be very difficult.

In fact, matrix notation here is infeasible and unnecessary. A closed-form of the Loop trial filter plus refinement would require a 3-ring of neighbors about the central vertex. Even if all vertices are regular (valence-6), enumerating a vertex's 3-ring is a challenging problem; in a non-regular setting, it is intractable. Thus for Loop surfaces – and mesh schemes in general – a stepwise procedure (Fig. 5.6) for decomposition and reconstruction is more practical.

With a refinement stage, decomposition becomes a four-step process. The first two steps are part of the normal decomposition procedure, while the latter two steps are for refinement.

1. Compute the trial coarse approximation $\tilde{\mathbf{C}} = \tilde{\mathbf{A}}\mathbf{F}$ according to Eqn. 5.6.
2. Compute the details $\tilde{\mathbf{D}} = \mathbf{B}\mathbf{F}$ according to Eqn. 5.8.

3. Compute the refinement vectors, Δ , by Eqn. 5.11.
4. Add the refinement vectors to the trial vertices: $\mathbf{C} = \tilde{\mathbf{C}} + \Delta$.

Reconstruction also requires a couple of additional steps to undo the refinement before the traditional reconstruction can occur.

1. Compute the refinement vectors, Δ .
2. Undo the refinement: $\tilde{\mathbf{C}} = \mathbf{C} - \Delta$.
3. Reconstruct the fine data from the coarse data and details: $\mathbf{F} = \mathbf{P}\tilde{\mathbf{C}} + \tilde{\mathbf{Q}}\tilde{\mathbf{D}}$.

5.7 Conclusion

The method described in Chapter 3 was successfully employed to build a complete MR system for semi-regular Loop surfaces. The wavelet constraint leads to a trial decomposition filter (Eqn. 5.6), and also dictates how to compute even (Eqn. 5.7) and odd (Eqn. 5.8) details. These operations together account for the $\tilde{\mathbf{A}}$, \mathbf{B} , and $\tilde{\mathbf{Q}}$ matrices in the MR system.

The trial decomposition filter creates high-error coarse approximations (Fig. 5.2), so a refinement procedure was used to reduce the error in the coarse vertices. The locally optimal displacement of each vertex is given by Eqn. 5.10; however, this optimization assumes that only the representative vertex will be displaced, when in reality all coarse vertices will be refined. By partially refining each vertex (Eqn. 5.11), a more optimal coarse mesh was reached (Fig. 5.4).

Due to the semi-regular mesh setting, it is difficult to encapsulate both the trial filters and refinement procedure into a single operation. For implementation pur-

poses a step-wise decomposition procedure is used, refining the vertices after the usual decomposition steps. Before applying the regular reconstruction steps, the refinement must first be undone.

Chapter 6

Catmull-Clark Surfaces

Catmull-Clark subdivision is a C^2 -continuous scheme extended from cubic B-splines. Unlike Loop subdivision – which operates only on triangle meshes – Catmull-Clark subdivision can operate on any mesh, placing no restrictions on the valence of faces or vertices. During subdivision, a k -sided face is split into k quadrilateral (quad) faces, so a semi-regular Catmull-Clark mesh consists of only quads. Figure 6.1 illustrates the face-splitting structure.

To build an MR system for Catmull-Clark subdivision, the wavelet constraint is again the starting point; in Sec. 6.2, the wavelet constraint is solved to produce a trial set of MR filters. In Sec. 6.3 and 6.4, a refinement step is applied to increase the similitude of the trial filter. Section 6.5 discusses how to handle boundary cases. Finally, Sec. 6.6 summarizes the key results of the chapter.

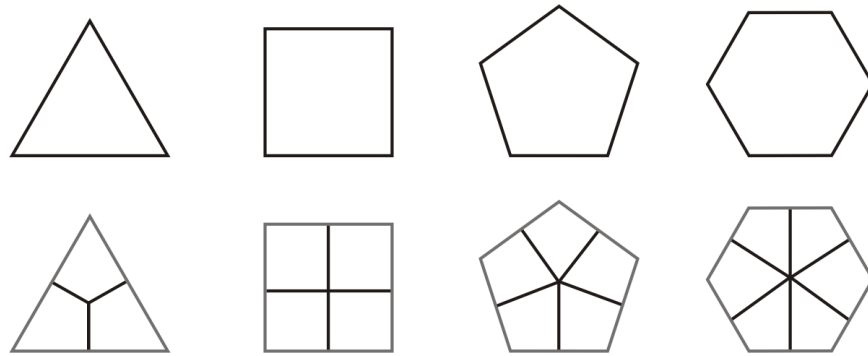


Figure 6.1: Catmull-Clark subdivision splits k -sided faces into k quadrilateral faces.

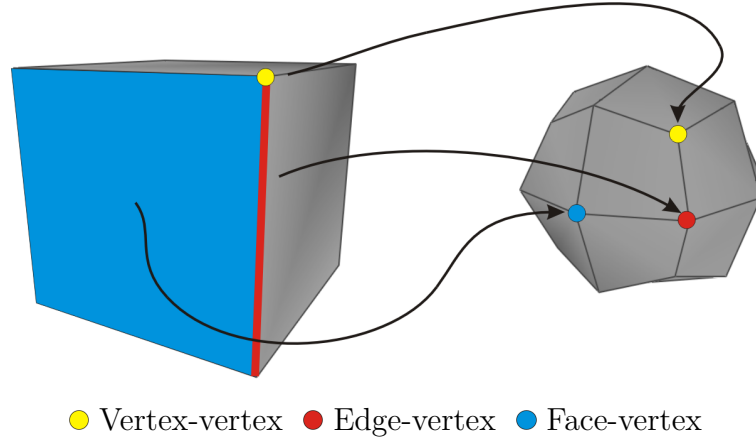


Figure 6.2: In Catmull-Clark subdivision, mesh \mathbf{C}^k (left) is subdivided to \mathbf{C}^{k+1} (right). \mathbf{C}^{k+1} has vertices corresponding to every vertex (yellow), edge (red), and face (blue).

6.1 Subdivision Filter

Catmull-Clark subdivision operates on arbitrary meshes, and consequently the subdivision mask is more complex than Loop and other schemes. Where Loop subdivision creates two types of vertices – vertex (even) and edge (odd) – Catmull-Clark also creates new (odd) vertices at the centroid of every face (Fig. 6.2).

After subdivision, a semi-regular Catmull-Clark mesh has three types of vertices. *Vertex-vertices* are vertices from the previous level of subdivision and are displaced by subdivision; these can also be categorized as *even* vertices. *Edge-vertices* are vertices at level $k + 1$ that correspond to edges from level k , while *face-vertices* correspond to faces from level k ; both edge- and face-vertices can be categorized as *odd* vertices. The even-odd distinction is necessary for building an MR system, as it dictates which vertices will be coarsened and which will be replaced with details. However, because not all “odd” vertices use the same subdivision mask, the more granular vertex-edge-face vertex characterization is necessary.

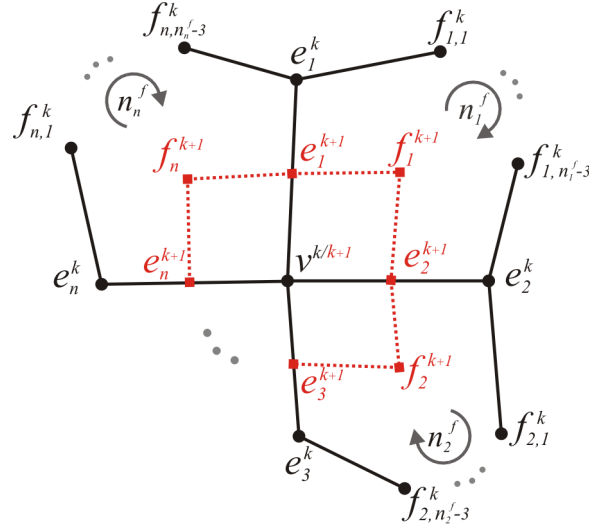


Figure 6.3: The notation for the local neighborhood of a representative vertex-vertex v^k . At level k , the 1-ring of v^k consists of edge neighbors e_i^k and face neighbors $f_{i,j}^k$. At level $k+1$, the 1-ring of v^{k+1} contains edge neighbors e_i^{k+1} and face neighbors f_i^{k+1} .

The added complexity of Catmull-Clark subdivision necessitates a change of notation. In previous chapters, coarse vertices have been denoted by c_i and fine vertices by f_i , and the even-odd distinction was clear because of the simple structure of the vertex neighborhoods. Catmull-Clark subdivision involves more complex neighborhoods: a vertex of valence n is shared by n faces f_1, \dots, f_n , each having an arbitrary valence n_1^f, \dots, n_n^f .

The notation for the local neighborhood of v in a Catmull-Clark mesh is illustrated in Fig. 6.3. Level k is not assumed to have subdivision connectivity, while level $k+1$ results from the subdivision of level k . The notation of level k is therefore a bit messy, because the valence of faces connected to v^k is arbitrary. Vertices e_i^k and e_i^{k+1} share an edge with v^k and v^{k+1} respectively, while vertices $f_{i,j}^k$ and f_i^{k+1} share a face but not an edge with v^k and v^{k+1} respectively.

Catmull-Clark subdivision is usually represented by the following masks:

$$f_i^{k+1} = \frac{1}{n_f} \left(v^k + e_i^k + e_{i+1}^k + \sum_j f_{i,j}^k \right), \quad (6.1)$$

$$e_i^{k+1} = \frac{1}{4} \left(v^k + e_i^k + f_{i-1}^{k+1} + f_i^{k+1} \right), \quad (6.2)$$

$$v^{k+1} = \frac{n-2}{n} v^k + \frac{1}{n^2} \sum_i e_i^k + \frac{1}{n^2} \sum_i f_i^{k+1}. \quad (6.3)$$

This form of the mask is often confusing because the vertex mask (Eqn. 6.3) is expressed in terms of both coarse (the e_i^k terms) and fine (f_i^{k+1}) vertices. To make construction and analysis easier, masks that are expressed in terms of vertices from only one level of subdivision are preferred: either only coarse vertices, as subdivision masks are usually represented, or strictly in terms of fine vertices, as done previously in Sec. 4.1 and Sec. 5.1.

Coarse Masks

Subdivision masks most often express the level- $k+1$ position of a vertex (even or odd) in terms of level- k neighbors. The subdivision mask for face-vertices, given by Eqn. 6.1, is already in this form. The masks for edge- and vertex-vertices, however, need to be rewritten. By replacing the f_i^{k+1} terms in Eqn. 6.2 according to Eqn. 6.1, the coarse edge mask is expressed as

$$\begin{aligned} e_i^{k+1} &= \frac{1}{4} \left(v^k + e_i^k + f_{i-1}^{k+1} + f_i^{k+1} \right) \\ &= \frac{1}{4} \left(v^k + e_i^k + \frac{1}{n_{f-1}} (v^k + e_{i-1}^k + e_i^k + \sum_j f_{i-1,j}^k) + \frac{1}{n_f} (v^k + e_i^k + e_{i+1}^k + \sum_j f_{i,j}^k) \right) \\ e_i^{k+1} &= \frac{1}{4} \left(\left(1 + \frac{1}{n_{f-1}} + \frac{1}{n_f} \right) (v^k + e_i^k) + \frac{1}{n_{f-1}} (e_{i-1}^k + \sum_j f_{i-1,j}^k) + \frac{1}{n_f} (e_{i+1}^k + \sum_j f_{i,j}^k) \right) \end{aligned} \quad (6.4)$$

Similarly, replacing the f_i^{k+1} terms in Eqn. 6.3 produces a coarse vertex mask:

$$v^{k+1} = \frac{n-2}{n} v^k + \frac{1}{n^2} \sum_i e_i^k + \frac{1}{n^2} \sum_i f_i^{k+1}$$

$$\begin{aligned}
&= \frac{n-2}{n}v^k + \frac{1}{n^2} \sum_i e_i^k + \frac{1}{n^2} \sum_i \frac{1}{n_i^f} \left(v^k + e_i^k + e_{i+1}^k + \sum_j f_{i,j}^k \right) \\
&= \frac{n-2}{n}v^k + \frac{1}{n^2} \sum_i e_i^k + \frac{1}{n^2} \sum_i \frac{1}{n_i^f} v^k + \frac{1}{n^2} \sum_i \frac{1}{n_i^f} e_i^k + \frac{1}{n^2} \sum_i \frac{1}{n_i^f} e_{i+1}^k + \\
&\quad \frac{1}{n^2} \sum_i \frac{1}{n_i^f} \sum_j f_{i,j}^k \\
&= \left(\frac{n-2}{n} + \frac{1}{n^2} \sum_i \frac{1}{n_i^f} \right) v^k + \frac{1}{n^2} \sum_i e_i^k \frac{1}{n^2} \sum_i \frac{1}{n_i^f} e_i^k + \frac{1}{n^2} \sum_i \frac{1}{n_{i-1}^f} e_i^k + \\
&\quad \frac{1}{n^2} \sum_i \frac{1}{n_i^f} \sum_j f_{i,j}^k \\
v^{k+1} &= \left(\frac{n-2}{n} + \frac{1}{n^2} \sum_i \frac{1}{n_i^f} \right) v^k + \frac{1}{n^2} \sum_i \frac{1}{n_i^f} \sum_j f_{i,j}^k + \\
&\quad \frac{1}{n^2} \sum_i \left(1 + \frac{1}{n_{i-1}^f} + \frac{1}{n_i^f} \right) e_i^k . \tag{6.5}
\end{aligned}$$

Fine Masks

As in previous chapters, the solution of the wavelet constraint is made easier by having a subdivision mask for v^k expressed in terms of fine neighbors e_i^{k+1} and f_i^{k+1} , rather than coarse neighbors e_i^k and f_i^k .

To have such a mask, e_i^{k+1} should replace e_i^k in Eqn. 6.3. This can be done by rewriting Eqn. 6.2 as $e_i^k = 4e_i^{k+1} - v^k - f_{i-1}^{k+1} - f_i^{k+1}$. Then Eqn. 6.3 can be rewritten as

$$\begin{aligned}
v^{k+1} &= \frac{n-2}{n}v^k + \frac{1}{n^2} \sum_i e_i^k + \frac{1}{n^2} \sum_i f_i^{k+1} \\
&= \frac{n-2}{n}v^k + \frac{1}{n^2} \sum_i (4e_i^{k+1} - v^k - f_{i-1}^{k+1} - f_i^{k+1}) + \frac{1}{n^2} \sum_i f_i^{k+1} \\
&= \frac{n-2}{n}v^k + \frac{4}{n^2} \sum_i e_i^{k+1} - \frac{1}{n^2} \sum_i v^k - \frac{1}{n^2} \sum_i f_{i-1}^{k+1} - \frac{1}{n^2} \sum_i f_i^{k+1} + \frac{1}{n^2} \sum_i f_i^{k+1} \\
&= \frac{n-2}{n}v^k - \frac{n}{n^2}v^k + \frac{4}{n^2} \sum_i e_i^{k+1} - \frac{2}{n^2} \sum_i f_i^{k+1} + \frac{1}{n^2} \sum_i f_i^{k+1} \\
v^{k+1} &= \frac{n-3}{n}v^k + \frac{4}{n^2} \sum_i e_i^{k+1} - \frac{1}{n^2} \sum_i f_i^{k+1} . \tag{6.6}
\end{aligned}$$

6.2 Trial Filters

The wavelet constraint is used to construct a trial filter and satisfy the storage constraint. In previous chapters, this has taken the form of $d_{even} = \alpha \sum d_{odd}$ (Eqns. 4.4 and 5.4). For Catmull-Clark surfaces the same strategy is adopted, but because of the heterogeneity of odd vertices, two free parameters are needed: one for edge-vertex details, and one for face-vertex details. If d^v represents the detail term for a vertex-vertex, and d^e and d^f represent edge- and face-vertex details respectively, then the wavelet constraint is

$$d^v = \alpha_e \sum_i d_i^e + \alpha_f \sum_i d_i^f, \quad (6.7)$$

based on the local neighborhood of v^k depicted in Fig. 6.3. Each detail term represents the difference between the original data at level $k+1$ and the subdivision of the coarser data from level k . Using the fine Catmull-Clark vertex mask (Eqn. 6.6), the details of interest are expressed as

$$\begin{aligned} d^v &= v^{k+1} - \left(\frac{n-3}{n} \tilde{v}^k + \frac{4}{n^2} \sum_i \tilde{e}_i^{k+1} - \frac{1}{n^2} \sum_i \tilde{f}_i^{k+1} \right), \\ d_i^e &= e_i^{k+1} - \tilde{e}_i^{k+1}, \\ d_i^f &= f_i^{k+1} - \tilde{f}_i^{k+1}. \end{aligned}$$

To solve for α_e and α_f , Eqn. 6.7 can be rewritten as

$$\begin{aligned} d^v &= \alpha_e \sum_i d_i^e + \alpha_f \sum_i d_i^f \\ d^v &= \alpha_e \sum_i (e_i^{k+1} - \tilde{e}_i^{k+1}) + \alpha_f \sum_i (f_i^{k+1} - \tilde{f}_i^{k+1}) \\ v^{k+1} - \frac{n-3}{n} \tilde{v}^k &= \frac{4}{n^2} \sum_i \tilde{e}_i^{k+1} + \frac{1}{n^2} \sum_i \tilde{f}_i^{k+1} \\ &= \alpha_e \sum_i e_i^{k+1} - \alpha_e \sum_i \tilde{e}_i^{k+1} + \alpha_f \sum_i f_i^{k+1} - \alpha_f \sum_i \tilde{f}_i^{k+1}. \end{aligned}$$

Setting $\alpha_e = \frac{4}{n^2}$ and $\alpha_f = -\frac{1}{n^2}$ will cancel the $\sum_i \tilde{e}_i^{k+1}$ and $\sum_i \tilde{f}_i^{k+1}$ terms from each side, leaving

$$\begin{aligned} v^{k+1} - \frac{n-3}{n} \tilde{v}^k &= \frac{4}{n^2} \sum_i e_i^{k+1} - \frac{1}{n^2} \sum_i f_i^{k+1} \\ \frac{n-3}{n} \tilde{v}^k &= v^{k+1} - \frac{4}{n^2} \sum_i e_i^{k+1} + \frac{1}{n^2} \sum_i f_i^{k+1} \\ \tilde{v}^k &= \frac{n}{n-3} v^{k+1} - \frac{4}{n(n-3)} \sum_i e_i^{k+1} + \frac{1}{n(n-3)} \sum_i f_i^{k+1} . \end{aligned} \quad (6.8)$$

Equation 6.8 represents the trial decomposition filter **A** for Catmull-Clark surfaces.

This filter is the same as the final result of Lanquetin and Neveu [28].

From the wavelet constraint, the remaining MR filters are also determined easily. The **B** filter must replace every odd vertex with a detail vector. A closed for could be derived to express the details only in terms of fine-level vertices, but from an implementation standpoint it is more practical to simply compute the details after computing the coarse vertices, via

$$\begin{aligned} d_i^e &= e_i^{k+1} - \tilde{e}_i^{k+1} , \\ d_i^f &= f_i^{k+1} - \tilde{f}_i^{k+1} . \end{aligned}$$

The **Q** filter is responsible for interpreting the wavelet coefficients that result from decomposition. In this case, the stored details represent the difference between an odd vertex after subdivision and the original fine data. Thus for odd vertices, the corresponding detail can be located and added to the subdivided vertex:

$$\begin{aligned} e_i^{k+1} &= \tilde{e}_i^{k+1} + d_i^e , \\ f_i^{k+1} &= \tilde{f}_i^{k+1} + d_i^f . \end{aligned}$$

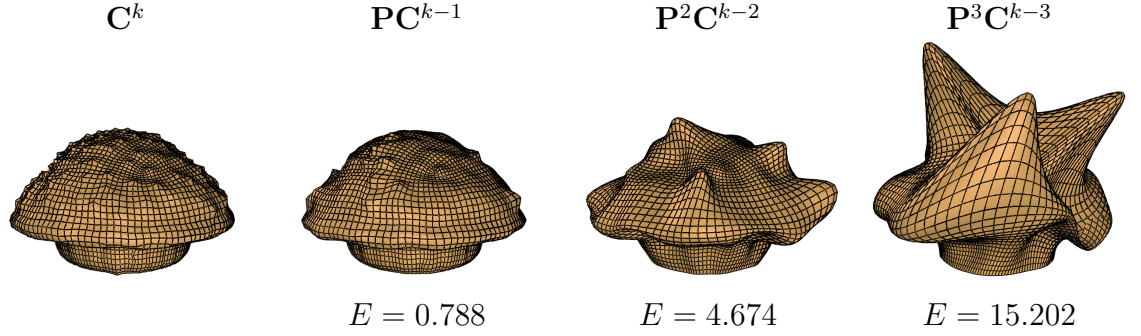


Figure 6.4: The trial filter produces high-error coarse approximations, especially under repeated applications. (left to right) the original model; after one decomposition with Eqn. 6.8; after two decompositions; after three decompositions.

For even vertices, the correct detail can be computed by Eqn. 6.7:

$$d^v = \frac{4}{n^2} \sum_i d_i^e - \frac{1}{n^2} \sum_i d_i^f .$$

Together, all MR filters are determined by the wavelet constraint, and a biorthogonal system results.

Figure 6.4 shows the performance of the trial filter on a simple object. After three decompositions, the error has become quite large, turning the muffin into a featureless blob. While decomposition followed by subdivision will always cause high-frequency details to be lost, perhaps more of the low-frequency information can be retained.

6.2.1 Valence-3 Vertices

Note that Eqn. 6.8 is undefined when $n = 3$ because the denominator of each coefficient is zero. Therefore, the trial filter cannot decompose valence-3 vertices. This result is not unexpected when Eqn. 6.6 is examined: when $n = 3$, the central vertex v^k contributes nothing to its subdivided position v^{k+1} ; thus there is no chance to

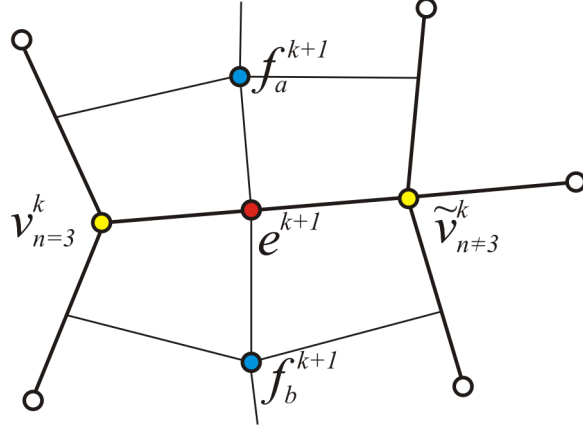


Figure 6.5: The trial filter of Eqn. 6.8 fails on valence-3 vertices. However, decomposition can be performed in cases where the valence-3 vertex $v_{n=3}^k$ has a non-valence-3 neighbor $\tilde{v}_{n \neq 3}^k$ at the coarse level.

recover v^k from v^{k+1} by simply rearranging that equation. This is an unfortunate result, but a decomposition mask that works for most valence-3 vertices can be found by an alternate method.

Consider a valence-3 vertex $v_{n=3}^k$ with at least one non-valence-3 coarse neighbor $v_{n \neq 3}^k$, as depicted in Fig. 6.5. At level k these vertices are connected by an edge, while at level $k+1$ the edge is bijected by edge-vertex e^{k+1} ; at level $k+1$, the faces on either side of the edge are split by face-vertices f_a^{k+1} and f_b^{k+1} .

Recall the regular edge subdivision mask of Eqn. 6.2. For the local neighborhood depicted in Fig. 6.5, e^{k+1} is computed by the mask

$$e^{k+1} = \frac{1}{4} \left(v_{n=3}^k + v_{n \neq 3}^k + f_a^{k+1} + f_b^{k+1} \right) .$$

When decomposing level $k+1$, e^{k+1} , f_a^{k+1} , and f_b^{k+1} are known while $\tilde{v}_{n=3}^k$ and $\tilde{v}_{n \neq 3}^k$ are to be determined. However, Eqn. 6.8, $v_{n \neq 3}^{k+1}$ can be decomposed to $\tilde{v}_{n \neq 3}^k$, leaving

$$e^{k+1} \approx \frac{1}{4} \left(v_{n=3}^k + \tilde{v}_{n \neq 3}^k + f_a^{k+1} + f_b^{k+1} \right) .$$

This is only an approximation because e^{k+1} may not result from subdivision (i.e. there may be some error introduced by decomposition). Since $v_{n=3}^k$ is the only unknown, this expression can be written as

$$v_{n=3}^k \approx 4e^{k+1} - \tilde{v}_{n \neq 3}^k - f_a^{k+1} - f_b^{k+1} . \quad (6.9)$$

In the above formulation it was implied that $v_{n \neq 3}^k$ was not a valence-3 vertex. In fact, Eqn. 6.9 allows a valence-3 vertex to be decomposed just in case it has *any* coarse-level neighbor that has been decomposed.

Unfortunately, decomposing a vertex with something other than the trial filter violates the wavelet constraint. In other words, $v_{n=3}^k$ is an even vertex, but Eqn. 6.7 cannot be used to compute its detail from neighboring odd details; a detail term must be explicitly computed and stored for each of these vertices. Because valence-3 vertices are also being replaced with coarse approximations, a single vertex at level $k + 1$ becomes a vertex and a vector at level k ; this violates the storage constraint locally and globally.

Equation 6.9 represents a valid inversion of subdivision: if vertices at level $k + 1$ represent the subdivision $\mathbf{C}^{k+1} = \mathbf{P}\mathbf{C}^k$ of some mesh \mathbf{C}^k , then decomposition with the trial filter and this special-case filter will return the original mesh. However, in general the MR system will be tasked with decomposing a mesh that does not result from subdivision. In such cases Eqn. 6.9 may produce a very poor approximation for $v_{n=3}^{k+1}$, because it has no contribution from $v_{n=3}^{k+1}$ itself and thus is totally insensitive to high-frequency information.

To alleviate this issue somewhat, an approach in which each already-decomposed neighbor of $v_{n=3}^{k+1}$ nominates a “candidate” position for $\tilde{v}_{n=3}^k$ based on Eqn. 6.9 is used.

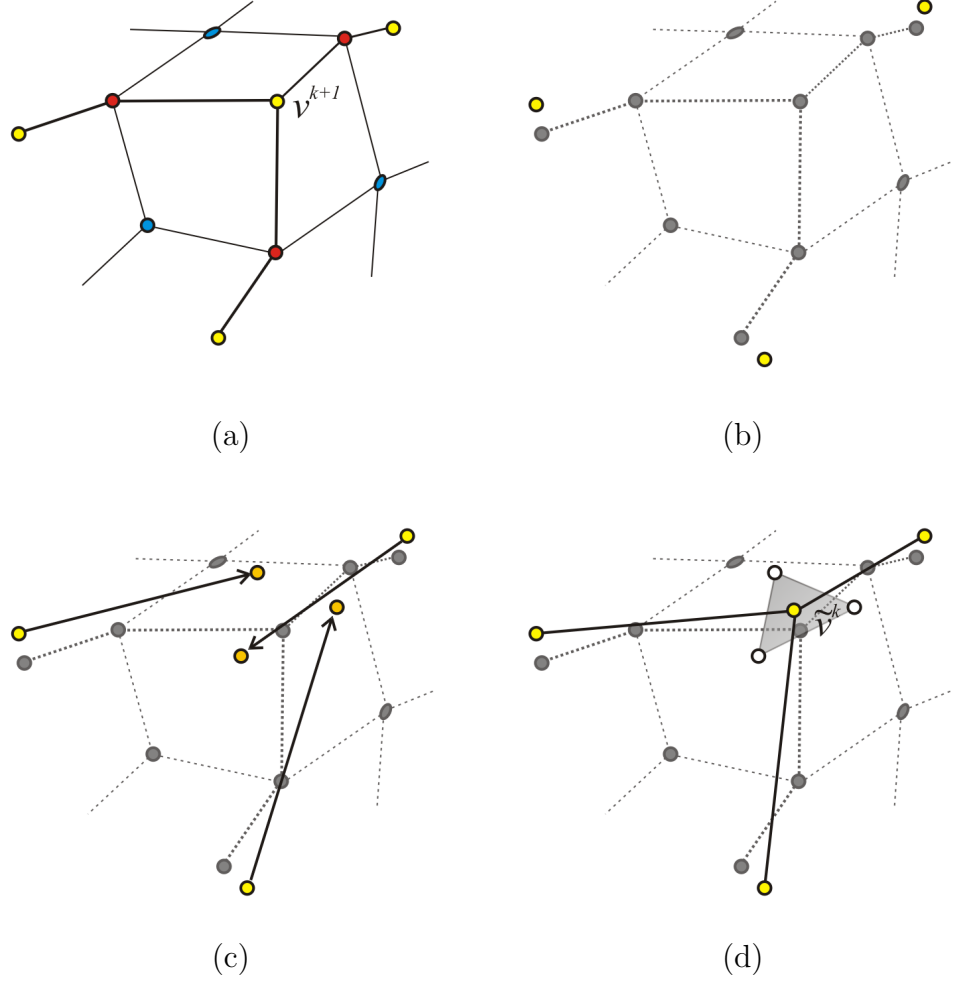


Figure 6.6: When a valence-3 vertex v^{k+1} has more than one coarse neighbor that has been decomposed, each neighbor nominates a position for \tilde{v}^k and \tilde{v}^k 's final position is taken as the average of the candidates: (a) a portion of a fine mesh around v^{k+1} ; (b) v^{k+1} 's coarse neighbors can be decomposed; (c) each neighbor nominates a position for \tilde{v}^k ; (d) the final position of \tilde{v}^k is the average of the candidates.

Then, the final position of $\tilde{v}_{n=3}^k$ is taken to be the average of all such candidates. Figure 6.6 illustrates this approach. This candidate-averaging approach is similar in spirit to the decomposition strategy used by Samavati et al. [38] for Doo-Sabin surfaces.

6.3 Refinement

Notice that in the trial decomposition mask (Eqn. 6.8), the weight applied to v^{k+1} is $\frac{n}{n-3} > 1$; for regular valence-4 vertices, the weight would be 4. This large central weight causes an amplification of errors under repeated applications of the mask, as illustrated in Fig. 6.4.

A refinement step that replace \tilde{v}^k by $v^k = \tilde{v}^k + \delta$ can reduce the error incurred by the trial filter. To select a δ that will reduce the error, the error about v^k is considered on a local scale.

Before refinement, the local error of \tilde{v}^k is $E_{\tilde{v}^k} = \|d^v\|^2 + \sum \|d_i^e\|^2 + \sum \|d_i^f\|^2$. To find the error $E(\delta)$ after refinement, the coarse masks of Eqns. 6.1 – 6.4 and 6.5 – can be used. These masks indicate how much v^k contributes to f_i^{k+1} , e_i^{k+1} , and v^{k+1} during subdivision: v^k contributes r to v^{k+1} , s_i to e_i^{k+1} , and t_i to f_i^{k+1} , where

$$\begin{aligned} r &= \frac{n-2}{n} + \frac{1}{n^2} \sum_i \frac{1}{n_i^f}, \\ s_i &= \frac{1}{4} \left(1 + \frac{1}{n_i^f} + \frac{1}{n_{i-1}^f} \right), \\ t_i &= \frac{1}{n_i^f}. \end{aligned}$$

When \tilde{v}^k is refined by δ , the subdivision of the coarse data is impacted according to

these weights. Thus the local error after refinement can be written as

$$E(\delta) = \|d^v - r\delta\|^2 + \sum \|d_i^e - s_i\delta\|^2 - \sum \|d_i^f - t_i\delta\|^2 . \quad (6.10)$$

To improve the trial filter, δ should be selected to minimize this quantity.

Recalling that $\|x\|^2 = x \cdot x$, and therefore $\|d - x\delta\|^2 = \|d\|^2 - 2xd \cdot \delta + x^2\|\delta\|^2$, Eqn. 6.10 simplifies to

$$\begin{aligned} E(\delta) = & \left(\|d^v\|^2 - 2rd^v \cdot \delta + r^2\|\delta\|^2 \right) + \sum \left(\|d_i^e\|^2 - 2s_id_i^e \cdot \delta + s_i^2\|\delta\|^2 \right) + \\ & \sum \left(\|d_i^f\|^2 - 2t_id_i^f \cdot \delta + t_i^2\|\delta\|^2 \right) . \end{aligned}$$

After grouping of like terms, this further simplifies to

$$E(\delta) = a\|\delta\|^2 - \mathbf{v} \cdot \delta + b , \quad (6.11)$$

where

$$\begin{aligned} a &= r^2 + \sum_i s_i^2 + \sum_i t_i^2 , \\ \mathbf{v} &= 2 \left(rd^v + \sum_i s_id_i^e + \sum_i t_id_i^f \right) , \text{ and} \\ b &= \|d^v\|^2 + \sum \|d_i^e\|^2 + \sum \|d_i^f\|^2 . \end{aligned}$$

Equation 6.11, the error function $E(\delta)$ that we want to minimize, is quadratic in δ , and can be solved analytically by differentiating and finding a zero-crossing:

$$E(\delta)' = 2a\delta - \mathbf{v} = 0 \implies \delta = \frac{\mathbf{v}}{2a} .$$

This is a minimum, because $E(\delta)'' = 2a > 0$. Therefore

$$\delta = \frac{\mathbf{v}}{2a}$$

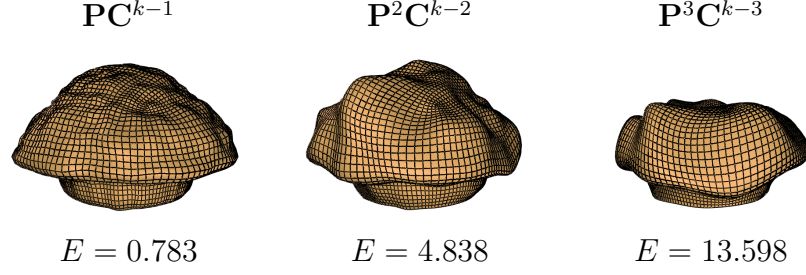


Figure 6.7: A refinement step is used to reduce the error of the trial filter.

$$\begin{aligned}
 &= \frac{2(rd^v + \sum s_i d_i^e + \sum t_i d_i^f)}{2(r^2 + \sum s_i^2 + \sum t_i^2)} \\
 &= \frac{r(\alpha_e \sum d_i^e + \alpha_f \sum d_i^f) + \sum s_i d_i^e + \sum t_i d_i^f}{r^2 + \sum s_i^2 + \sum t_i^2} \\
 \delta &= \frac{\sum(r\alpha_e + s_i)d_i^e + \sum(r\alpha_f + t_i)d_i^f}{r^2 + \sum s_i^2 + \sum t_i^2} .
 \end{aligned} \tag{6.12}$$

The final simplification step is made possible by Eqn. 6.7.

6.3.1 Valence-3 Vertices

It is unclear whether the special-case valence-3 vertices discussed in Sec. 6.2.1 should be refined or not. In practice, we have observed that not refining these vertices produces better coarse approximations.

6.4 Partial Refinement

Using the voting strategy to dampen the refinement of Eqn. 6.12, \tilde{v}^k wants to take the full refinement step, while its neighbors \tilde{e}_i^k do not want \tilde{v}^k to move at all. Each vote is weighted according to the subdivision mask: \tilde{v}^k votes for $\mu = 1$ with a weight

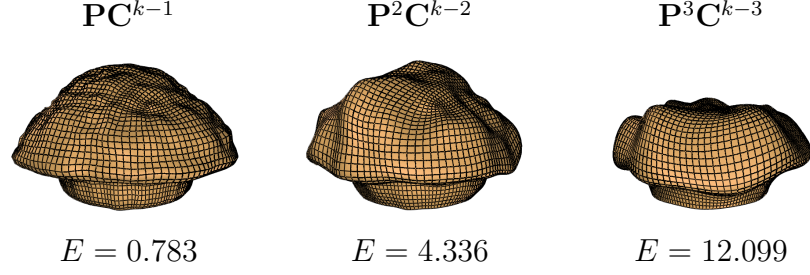


Figure 6.8: Partial refinement reduces the per-vertex displacement to account for the interdependence of the refinements. Visually, the results are similar to the full refinement, but the error measurement shows improvement.

of r , while \tilde{e}_i^k vote for $\mu = 0$ with a weight of $\frac{1}{n^2}(1 + \frac{1}{n_i^f} + \frac{1}{n_{i-1}^f})$. So

$$\mu = r(1) + \left(\frac{1}{n^2} \left(1 + \frac{1}{n_i^f} + \frac{1}{n_{i-1}^f} \right) \right) (0) = r .$$

Using this value to dampen δ , the per-vertex refinement vector becomes

$$\begin{aligned} \delta &= r \frac{\sum (r\alpha_e + s_i)d_i^e + \sum (r\alpha_f + t_i)d_i^f}{r^2 + \sum s_i^2 + \sum t_i^2} \\ &= \frac{\sum (r^2\alpha_e + s_i)d_i^e + \sum (r^2\alpha_f + t_i)d_i^f}{r^2 + \sum s_i^2 + \sum t_i^2} . \end{aligned} \quad (6.13)$$

Figure 6.8 illustrates the difference between refining with Eqn. 6.12 and partially refining with Eqn. 6.13. The differences are subtle in this instance, but careful inspection reveals that the partially refined mesh retains more of the muffin's bulge after three decompositions.

6.5 Boundary Filters

For the Catmull-Clark subdivision masks to be applied, an entity (vertex, face, or edge) must have a complete neighborhood. For a vertex of valence n , a full neighborhood requires n edge neighbors and n faces; if the number of edge neighbors is

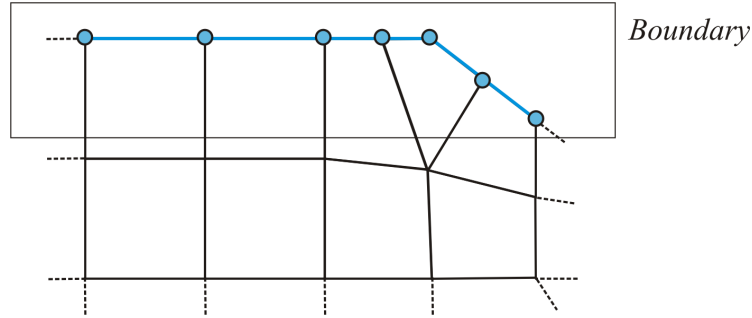


Figure 6.9: Boundary vertices and edges in a Catmull-Clark model can be decomposed with cubic B-spline filters.

greater than the number of connected faces, then the vertex is on a boundary and does not have a full neighborhood. For an edge, the mask requires two adjacent faces; if the edge has only one adjacent face, then it is on a boundary and does not have a full neighborhood. The subdivision mask for a face requires only the vertices in the face, and so a face always has a full neighborhood. Thus, only boundary vertices and boundary edges require special treatment during subdivision, and by extension, require special treatment during decomposition and reconstruction.

Like Loop surfaces, Catmull-Clark surfaces are a generalization of cubic B-spline curves, and so the cubic B-spline filter is used to subdivide boundary vertices and edges. For boundary MR filters, therefore, a cubic B-spline multiresolution can be used during decomposition and reconstruction, such as those presented in Sec. 4.6. In particular, the regular filters \mathbf{A}_r and \mathbf{Q}_r can be used along any continuous boundary, while the boundary cases $\mathbf{A}_{s/e}$ and $\mathbf{Q}_{s/e}$ should be used for corner points.

6.6 Conclusion

In this chapter, the first complete MR system for Catmull-Clark surfaces has been developed. In Sec. 6.2 a trial decomposition filter resulted from the wavelet constraint. While the trial filter (Eqn. 6.8) is equivalent to the filter proposed by Lanquetin and Neveu, our construction simultaneously satisfies the storage constraint. This represents a substantial benefit of our work, because it opens the door to applications such as mesh compression.

An additional refinement procedure is developed to reduce the error in the trial coarse approximation. The refinement (Eqn. 6.12) is based on minimizing the local error around a coarse vertex. However, minimality is only satisfied when other coarse vertices in the local neighborhood are not displaced; in practice, then, the refinements are too large because elements of the local neighborhood are also being displaced by refinement. By dampening each local refinement (Eqn. 6.13), the trial filter is improved even more.

One drawback of the MR system developed in this chapter is that valence-3 vertices must be handled by special filters, and in certain pathological cases cannot be decomposed at all. Most valence-3 vertices can be decomposed by the procedure described in Sec. 6.2.1; only in the pathological case where all even vertices are valence-3, such as a cube, could the method fail. Because Catmull-Clark subdivision produces mostly regular vertices (all edge vertices are valence-4), the method could only possibly fail at the topmost level of the mesh hierarchy, and again only in pathological cases. Thus, the valence-3 issue is not a significant problem in practice.

Chapter 7

Results

In this chapter we will quantitatively analyze the MR systems constructed in Chaps. 4–6. For each type of curve or surface, the performance of our filters – both the trial and refined – will be compared to that of existing filters.

The procedure outlined Sec. ?? is used to evaluate each MR system. That is, given a semi-regular object \mathbf{C}^k , the object is decomposed j times with \mathbf{A} ,

$$\mathbf{C}^{k-j} = \mathbf{A}^{k-j} \dots \mathbf{A}^{k-1} \mathbf{A}^k \mathbf{C}^k ,$$

and then subdivided *without details* back to level k ,

$$\tilde{\mathbf{C}}^k = \mathbf{P}^{k-1} \mathbf{P}^{k-2} \dots \mathbf{P}^{k-j} \mathbf{C}^{k-j} .$$

The least-squares error $E(\tilde{\mathbf{C}}^k)$ is then measured as

$$E(\tilde{\mathbf{C}}^k) = \sqrt{\|\mathbf{C}^k - \tilde{\mathbf{C}}^k\|^2} .$$

The $\tilde{\mathbf{C}}^k$ notation cannot distinguish how many times a reconstructed object was initially decomposed. As an alternative and more precise notation, the reconstruction of \mathbf{C}^{k-j} will be denoted as $\mathbf{P}^j \mathbf{C}^{k-j}$.

This error metric has no absolute meaning; the numerical value of E will depend on the coordinate system of the particular object. Relatively, however, a “better” filter will generally produce a lower error value.

7.1 Cubic B-Splines

In Chapter 4, an MR system was constructed for cubic B-spline subdivision curves. A trial filter of width-3 resulted from the wavelet constraint:

$$\frac{a_0}{2} \frac{a_{\pm 1}}{-\frac{1}{2}} .$$

To refine this filter, the error was considered in two differently-sized local neighborhoods, one of width 3 and the other of width 5. The 3-element neighborhood produced a closed-form filter of width 7, the *W-7* filter:

$$\frac{a_0}{\frac{35}{34}} \frac{a_{\pm 1}}{\frac{95}{272}} - \frac{a_{\pm 2}}{\frac{33}{68}} \frac{a_{\pm 3}}{\frac{33}{272}} .$$

To evaluate this filter in a broader context, it is compared against a minimum-norm width-7 filter from Bartels and Samavati [3],

$$\frac{a_0}{\frac{52}{49}} \frac{a_{\pm 1}}{\frac{9}{28}} - \frac{a_{\pm 2}}{\frac{23}{49}} \frac{a_{\pm 3}}{\frac{23}{196}} ,$$

which will be referred to as the *BS-7* filter.

When considering the error in a larger 5-element neighborhood, the trial filter widened even further to a width-11 decomposition filter, the *W-11* filter:

$$\frac{a_0}{\frac{71}{70}} \frac{a_{\pm 1}}{\frac{103}{280}} - \frac{a_{\pm 2}}{\frac{18}{35}} \frac{a_{\pm 3}}{\frac{87}{560}} - \frac{a_{\pm 4}}{\frac{3}{140}} \frac{a_{\pm 5}}{\frac{3}{560}} .$$

Bartels and Samavati's 11-element *BS-11* filter, given by

$$\frac{a_0}{\frac{5714}{6019}} \frac{a_{\pm 1}}{\frac{4479}{12038}} - \frac{a_{\pm 2}}{\frac{2024}{6019}} - \frac{a_{\pm 3}}{\frac{141}{926}} \frac{a_{\pm 4}}{\frac{1138}{6019}} - \frac{a_{\pm 5}}{\frac{569}{12038}} ,$$

is used for comparison

7.1.1 Analysis

Figure 7.1 shows several curves and a tensor product surface that together represent the high-resolution test data \mathbf{C}^k .

Figure 7.2 shows each filter applied twice (left column) and three times (right) to the car object of Fig. 7.1(a). Below each figure, the error $E(\tilde{\mathbf{C}}^k)$ is listed. As expected, the trial filter performs much worse than all other filters, especially after several decompositions. The W7 filter performs near-equivalently to Bartels and Samavati's, which is a strong validation of our method. In the 11-element filters, Bartels and Samavati's filter reduces the error even further than the 7-element filter, as one would expect; curiously, however, the W11 filter does not perform as well and actually produces higher errors (though visually there is little difference).

These trends continue for Figs. 7.3–7.6. The trial filter performs quite poorly, particularly under repeated decompositions. The W-7 filter performs very near to the BS-7 filter, while the wider W-11 filter performs reasonably well but always worse than the W-7 filter. Filter BS-11 is consistently the best performer.

Overall, these results are in line with expectations. That the W-7 filter performs essentially the same as a minimum-norm filter is validation of the refinement procedure and also of the partial refinement strategy.

The one unforeseen result is the relatively poor performance of our W-11 filter, both in comparison to the BS-11 filter and to the narrower width-7 filters. Looking at the filter vectors for each filter, one observation is that both width-7 filters have a central weight $a_0 > 1$; the W-11 filter also has $a_0 > 1$, while the central filter for BS-11 is $a_0 < 1$. From this observation, it seems that the refinement that lead to

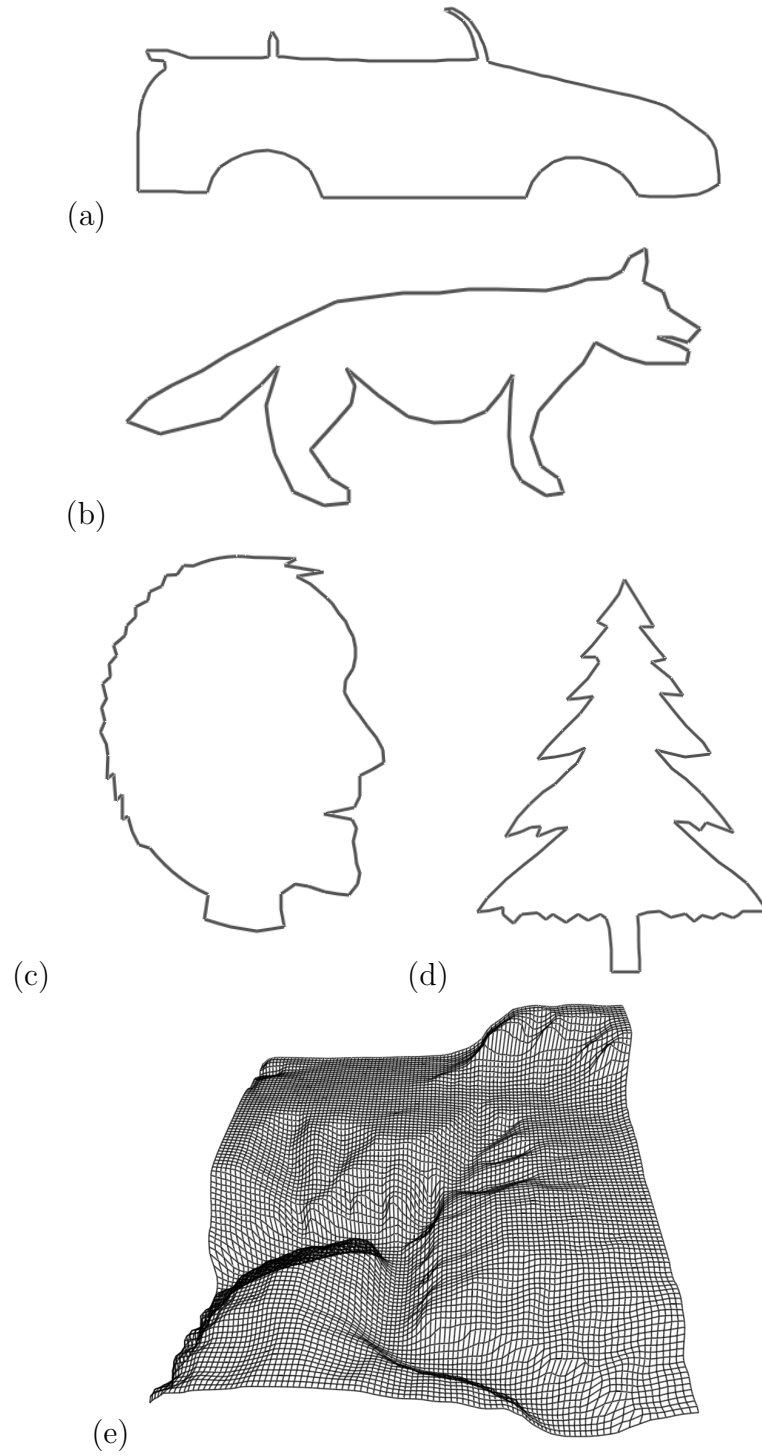


Figure 7.1: Original fine data \mathbf{C}^k for evaluating the cubic B-spline MR system of Chapter 4.

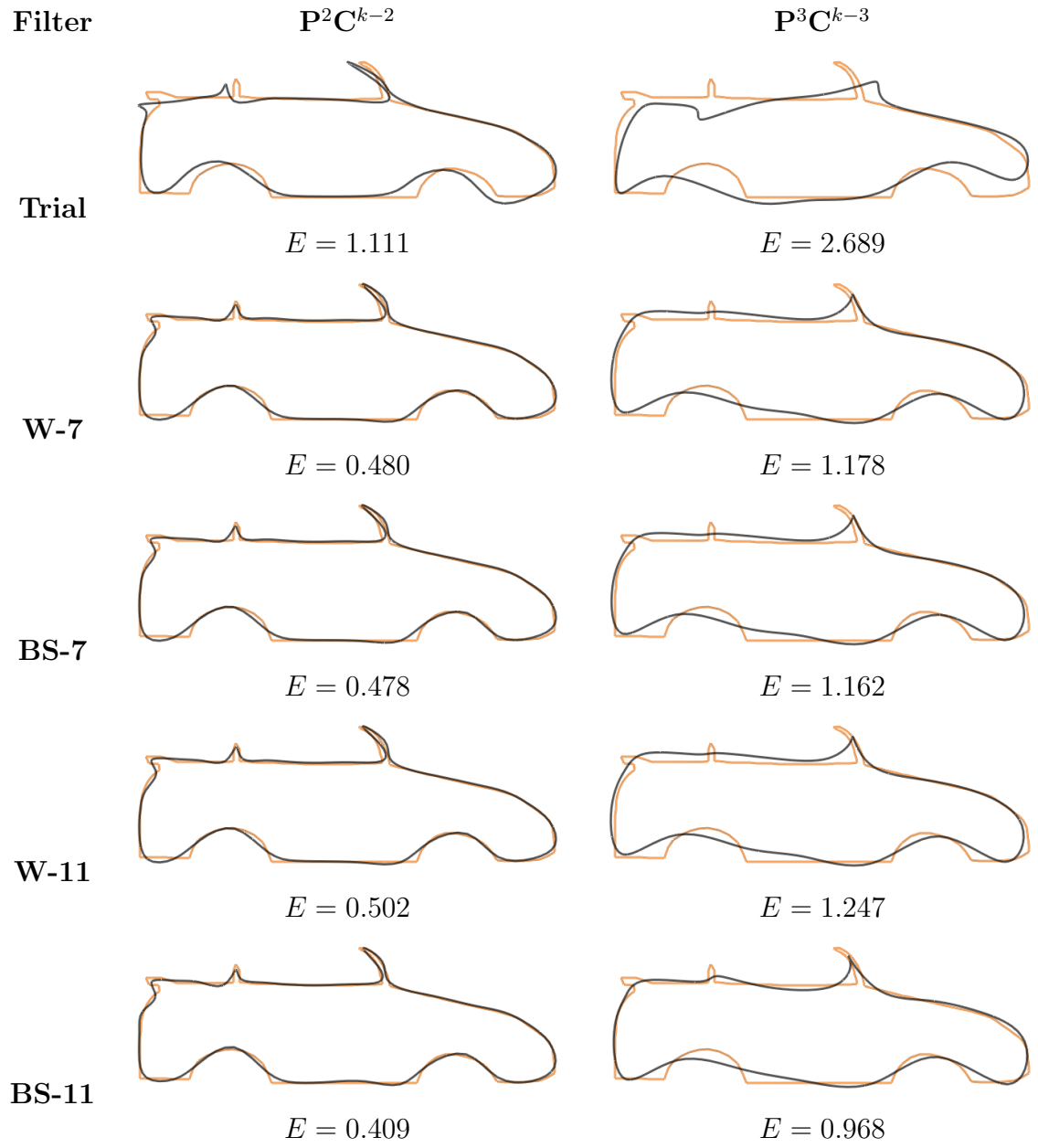


Figure 7.2: A car model is decomposed two (left column) and three (right column) levels and then subdivided the same number of times.

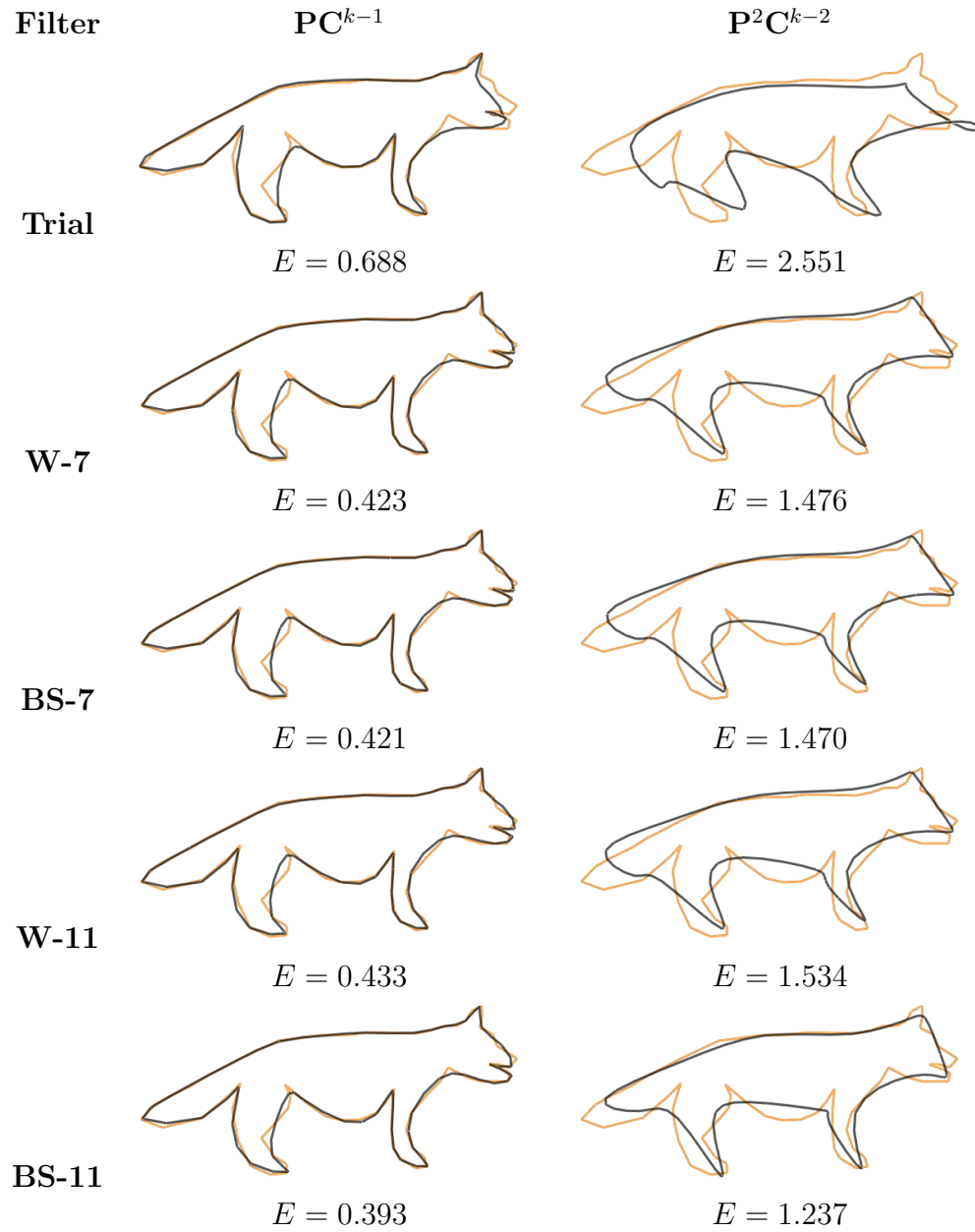


Figure 7.3: A wolf model is decomposed one (left column) and two (right column) levels and then subdivided the same number of times.

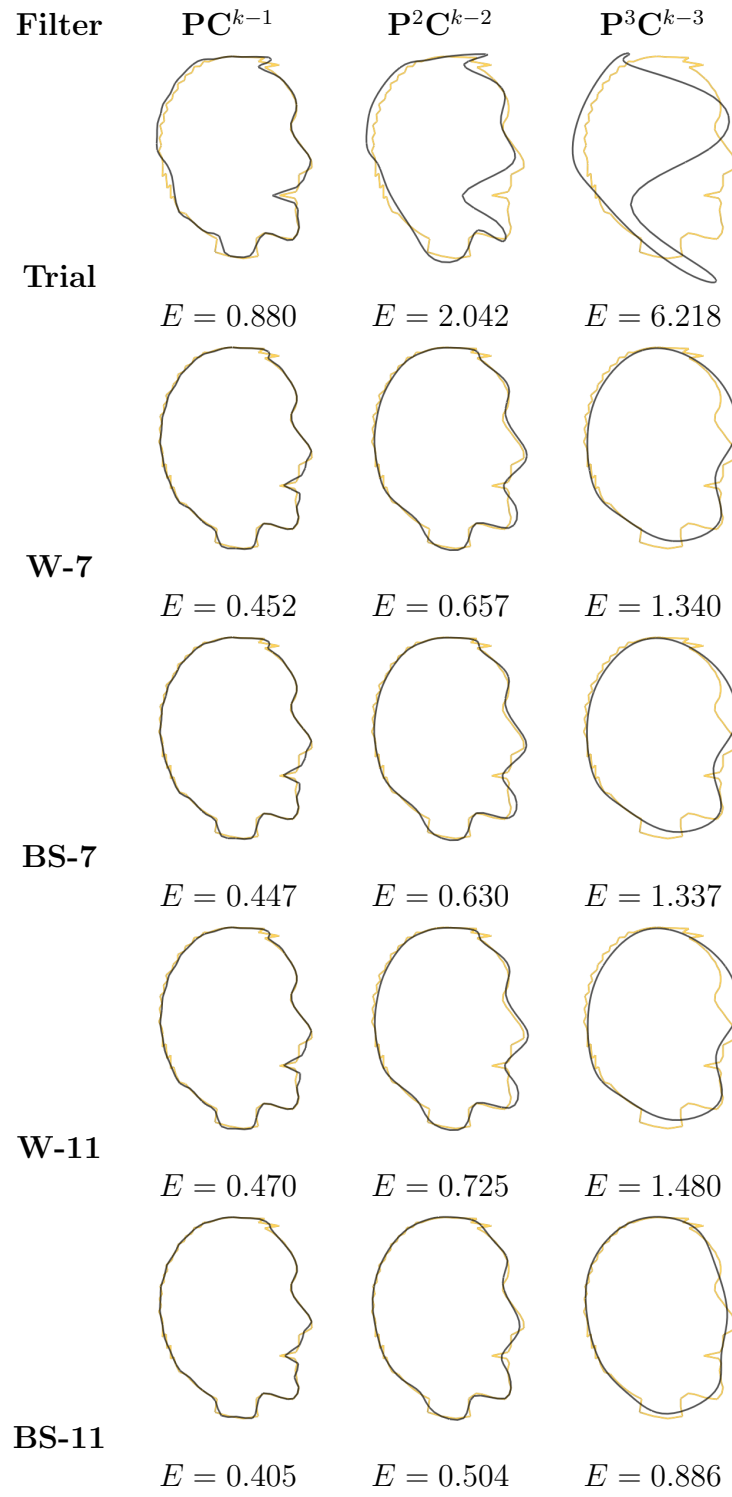


Figure 7.4: A face model is decomposed one (left column), two (middle), and three (right) levels and then subdivided the same number of times.











Filter	\mathbf{PC}^{k-1}	$\mathbf{P}^2\mathbf{C}^{k-2}$
Trial	 $E = 0.843$	 $E = 3.933$
W-7	 $E = 0.450$	 $E = 1.323$
BS-7	 $E = 0.448$	 $E = 1.282$
W-11	 $E = 0.464$	 $E = 1.475$
BS-11	 $E = 0.403$	 $E = 0.953$

Figure 7.5: A tree model is decomposed one (left column) and two (right column) levels and then subdivided the same number of times.

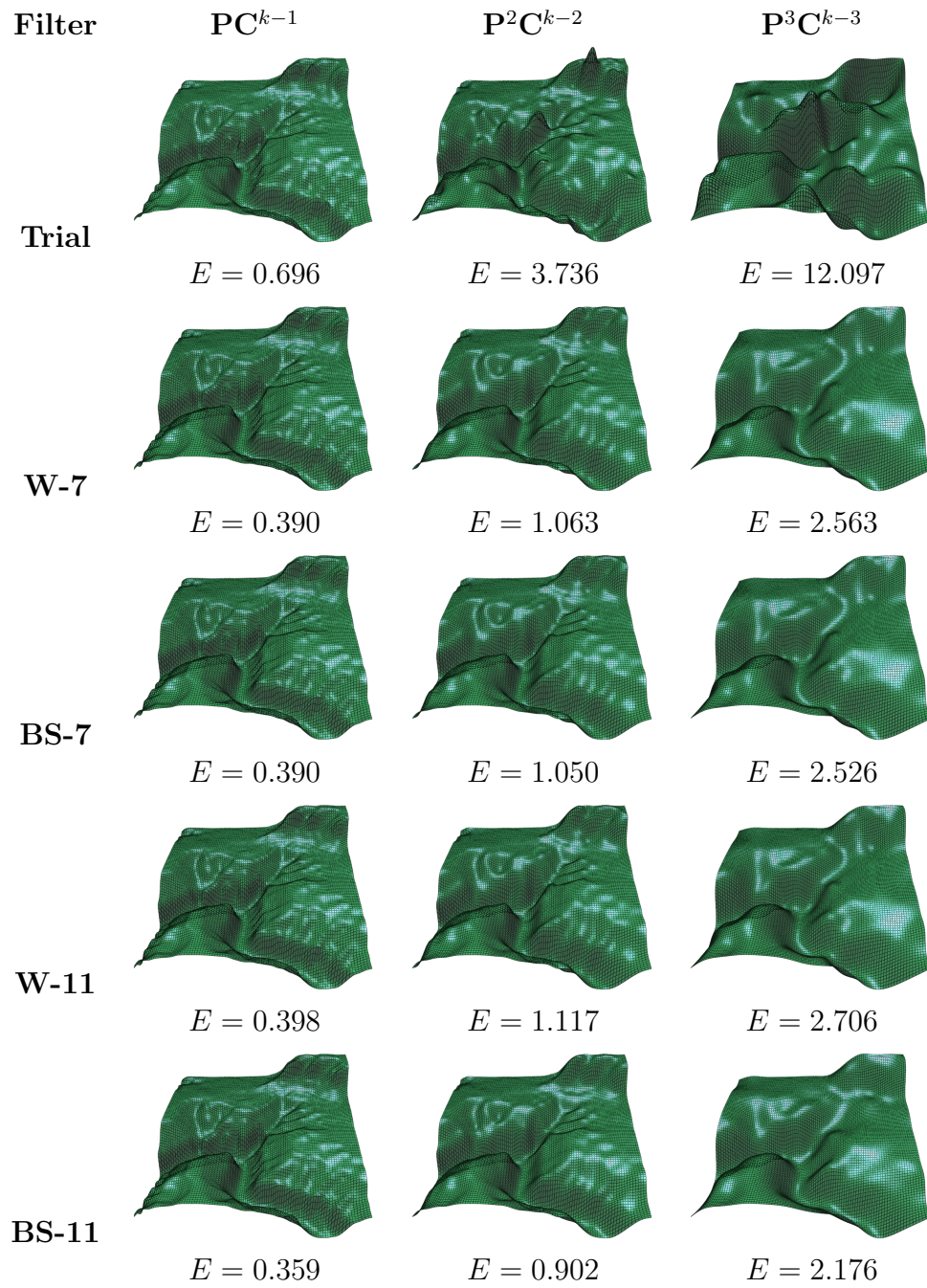


Figure 7.6: A terrain patch is decomposed one (left column), two (middle), and three (right) levels and then subdivided the same number of times.

f

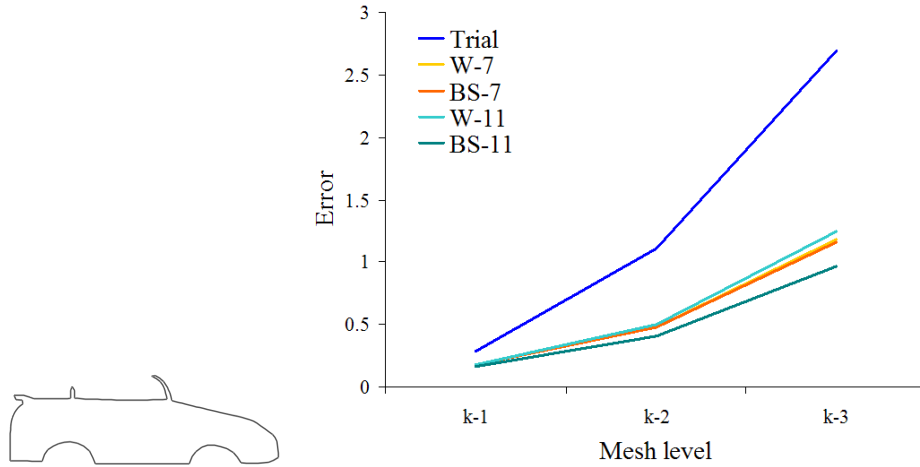


Figure 7.7: A plot of least-squares error for the car model of Fig. 7.2.

the W-11 filter did not lessen the impact of the central vertex to the same extent as BS-11. The \mathbf{L} matrices support this, as the weights of d_{-1} and d_1 are $\frac{33}{68} \approx 0.48$ for the W-7 filter and $\frac{69}{140} \approx 0.49$ for the W-11 filter. Thus a refinement step that produced a lower central weight would likely produce a better result for the wider 5-element neighborhood.

7.1.2 Conclusion

The results indicate that the trial filter is unsuitable for producing low-error coarse data. Meanwhile, comparisons of the W-7 and W-11 filters with established near-minimum-norm filters of the same widths – BS-7 and BS-11 – indicate that our MR construction method is able to create high-quality filters within an extensible framework that other constructions are lacking.

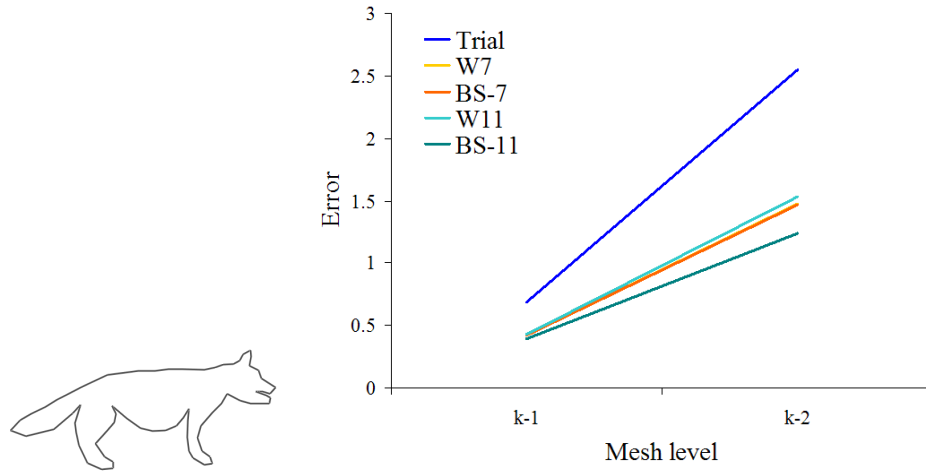


Figure 7.8: A plot of least-squares error for the wolf model of Fig. 7.3.

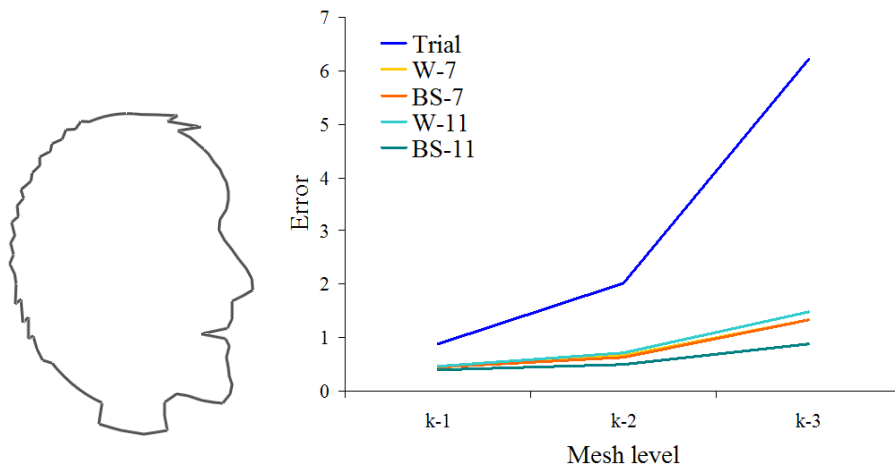


Figure 7.9: A plot of least-squares error for the face model of Fig. 7.4.

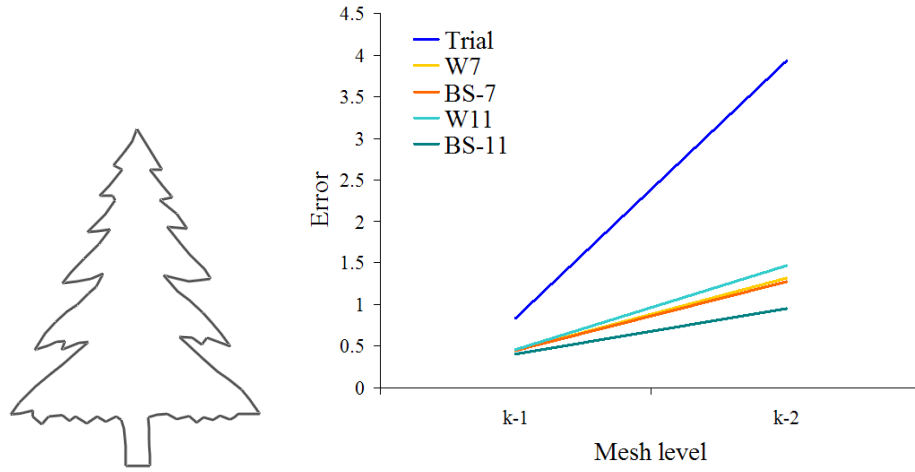


Figure 7.10: A plot of least-squares error for the tree model of Fig. 7.5.

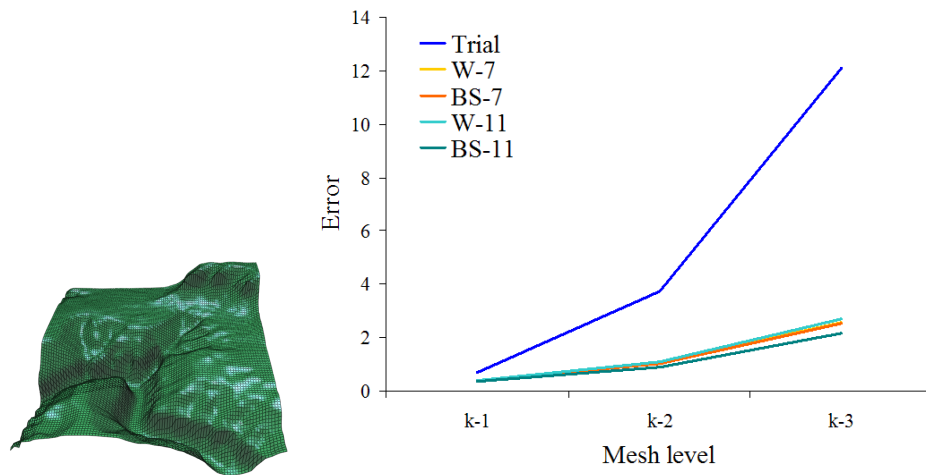


Figure 7.11: A plot of least-squares error for the terrain model of Fig. 7.6.

7.2 Loop Surfaces

In Chapter 5 an MR system for Loop subdivision surfaces was developed. From the wavelet constraint, a trial filter was reached, for which the decomposition mask $\widetilde{\mathbf{A}}$ is given by

$$\tilde{c}_0 = \frac{1}{1 - n\alpha} f_0 - \frac{\alpha}{1 - n\alpha} \sum f_i .$$

To increase the quality of the coarse meshes produced by the trial filter, a partial refinement step was developed: each coarse vertex \tilde{c}_0 is displaced by δ_0 , where

$$\delta_0 = (1 - n\beta)\kappa \sum d_i .$$

This filter will be referred to as the *refined* filter (though it is in fact the partially refined filter of Sec. 5.4).

Recently, Li et al. developed a Loop MR scheme [31]. Their method is based on rewriting the subdivision rules with some free parameters such that when the rules are inverted and the free parameters defined, an MR system is induced. Their construction does not indicate how details for even vertices are computed, so their construction appears to not satisfy the storage constraint. Nevertheless, the decomposition process that results from the inverted rules is essentially the same as our trial-refined sequence. When viewed this way; their refinement step is

$$\delta_0 = \alpha_{LQS} \sum d_i^e + \beta_{LQS} \sum d_i^s , \tag{7.1}$$

where α_{LQS} and β_{LQS} are the free parameters of their scheme, and d_i^s are details from “sidewing” vertices (Fig. 7.12). This filter is referred to as the *LQS filter*.

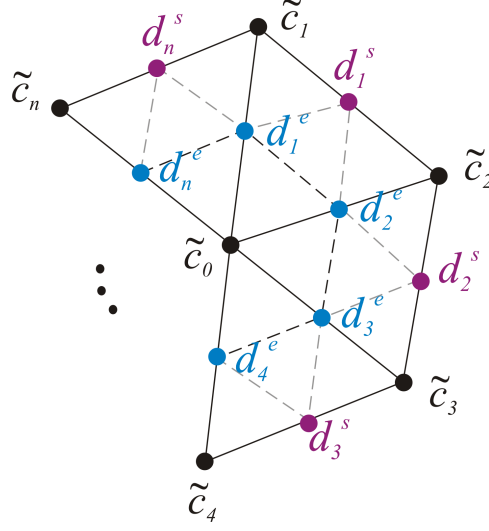


Figure 7.12: Notation for the refinement mask of Li et al. [31]. Our refinement step is based on the edge details d_i^e , while their refinement also takes the sidewing details d_i^s into account.

7.2.1 Analysis

The set of semi-regular Loop meshes shown in Fig. 7.13 were used for testing. In Fig. 7.14, the performance of each filter is shown. In the top row, the mesh produced by the trial filter is quite poor after two decompositions and after three applications the error has exploded. The refined filter performs much better, and even after three decompositions the general shape is preserved. The LQS filter performs about the same as the refined filter for the first two decompositions, but after three decompositions the resulting mesh looks better than the refined version. Figure 7.18 summarizes the error values from each filter.

The pawn model depicted in Fig. 7.15 again shows that the trial filter produces high-error approximations, in particular over several levels of decomposition. The refined filter does a much better job of keeping the error at a reasonable level. Visu-

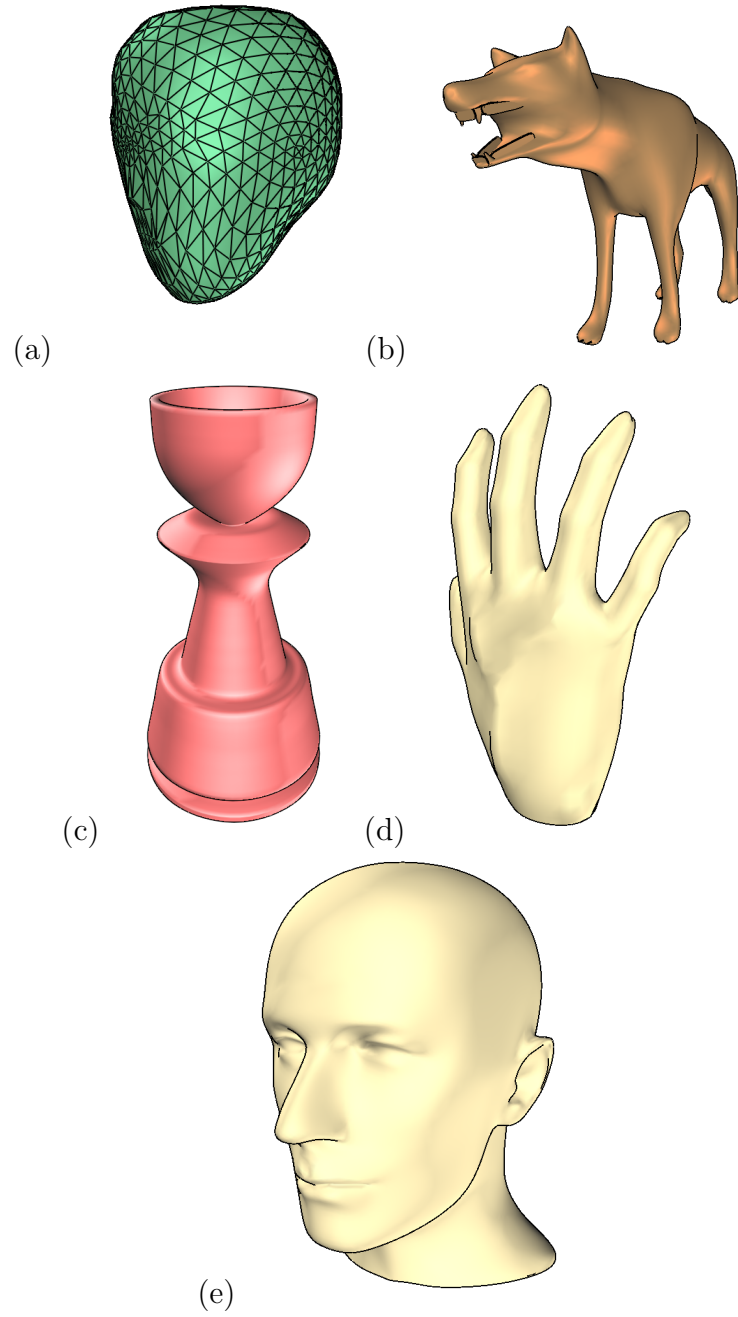


Figure 7.13: Fine data \mathbf{C}^k for testing the Loop MR system of Chapter 5.

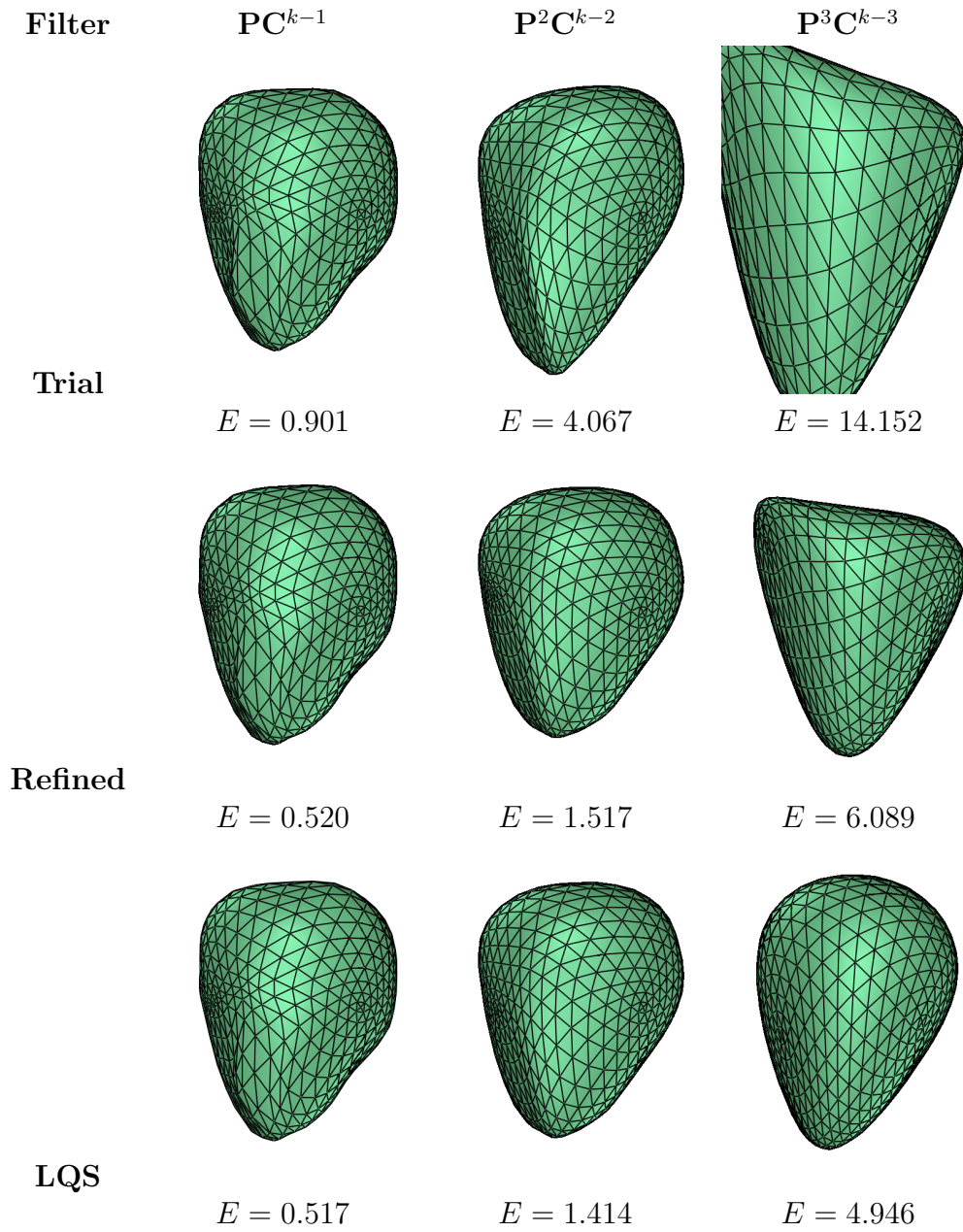


Figure 7.14: The alien model of Fig. 7.13 is decomposed several times in succession.

ally and numerically, there is little difference between the refined filter and the LQS filter. Figure 7.19 summarizes the errors from each filter for this model.

Figure 7.16 shows some fairly subtle differentiation of the filters when a hand model is decomposed. After one decomposition, each filter produces a low-error approximation, but after two decompositions the trial filter shows some noticeable deformation around the knuckles of the hand. Again, the refined filter is noticeably better than the trial filter, and about the same as the LQS filter. The results are recapped in Fig. 7.20.

The wolf model of Fig. 7.17 represents a difficult challenge for a decomposition filter, particularly in the sharp teeth. Subdivision will always round off sharp features (unless special sharp-feature masks are used), so a low-error approximation of such sharp features is tough to satisfy. The trial filter performs nearly as well as the refined filters after one decomposition, but the teeth area blows up after two decompositions; there is also some noticeable deviation around the ears. The LQS filter also has difficulty, though with less drastic results, with the teeth region of the wolf. In this case, the refined filter is remarkably able to handle the sharp teeth and produce a good coarse approximation after two decompositions. The errors of each filter are summarized in Fig. 7.21.

Figures 7.22 summarizes the results from the vase model from Chapter 5, while Fig. 7.23 shows the results of applying the filters to a head model. In each case, the refined and LQS filters perform essentially the same.

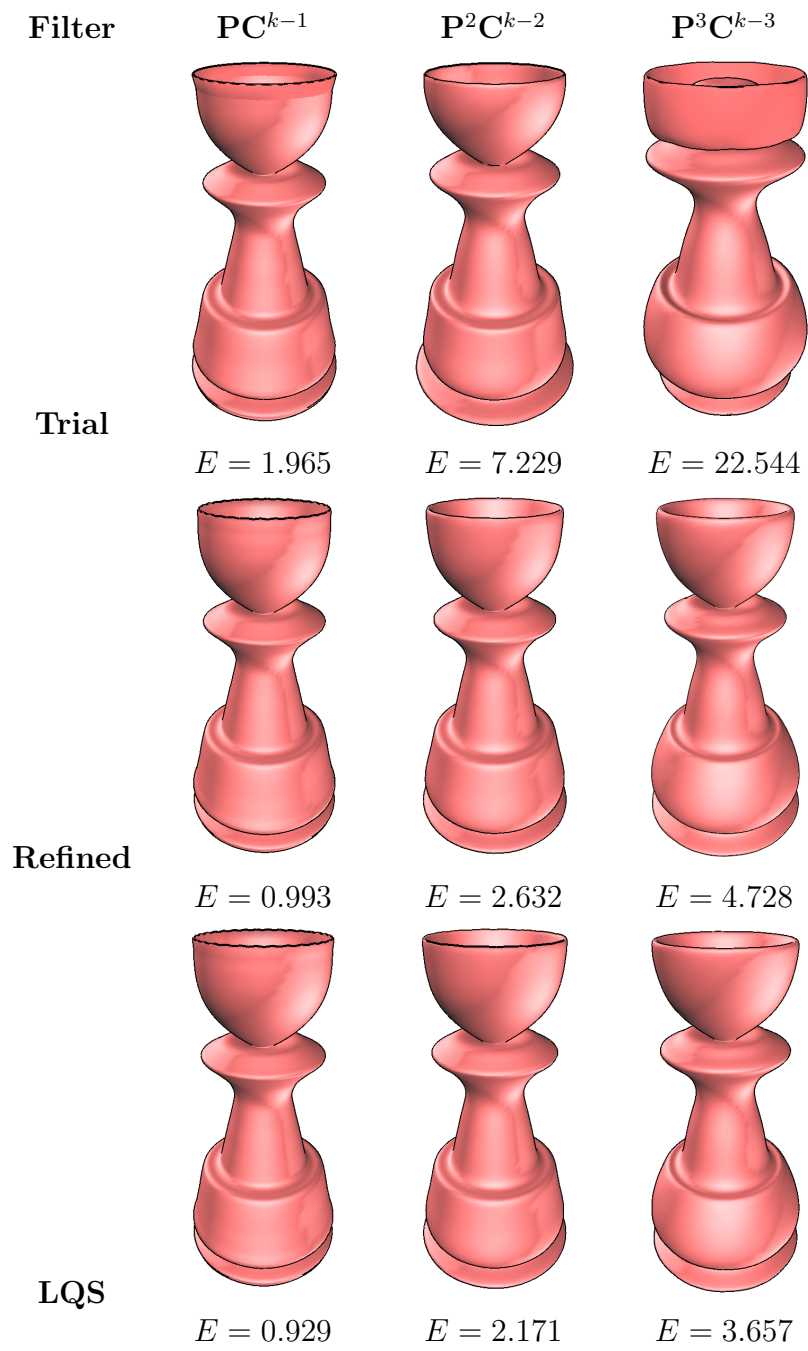


Figure 7.15: The pawn model of Fig. 7.13 is decomposed several times in succession.

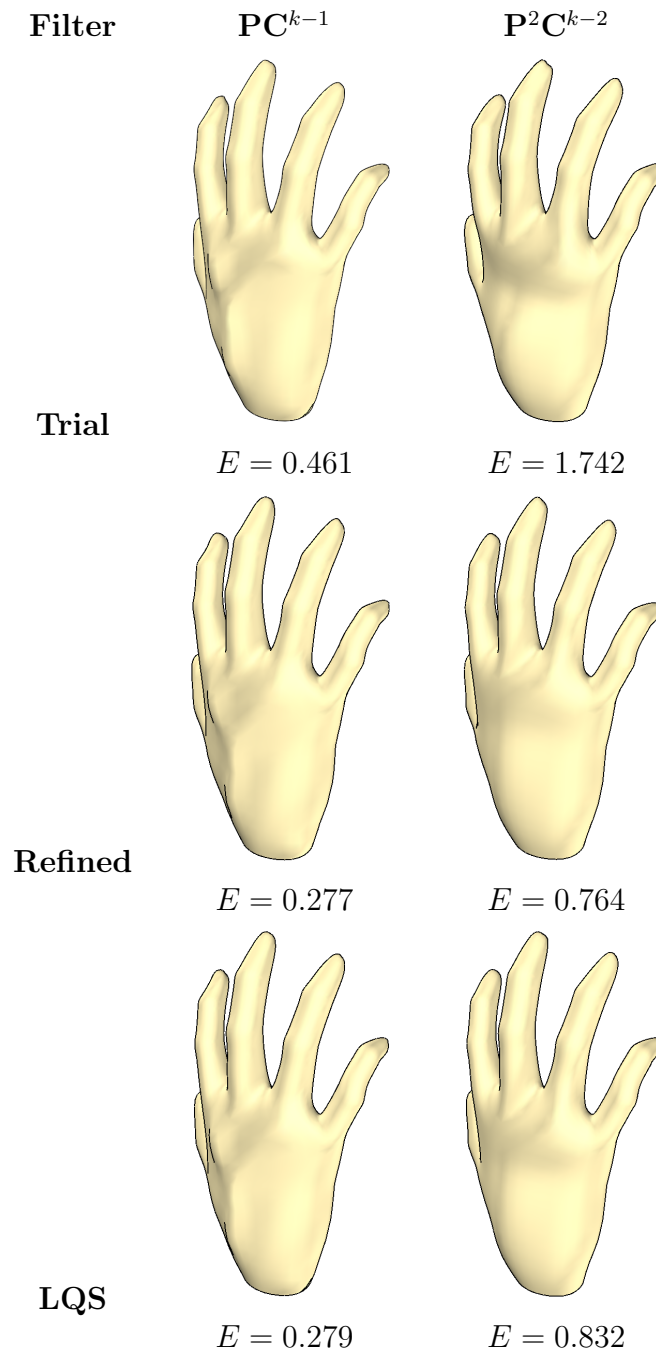


Figure 7.16: The hand model of Fig. 7.13 is decomposed several times in succession.

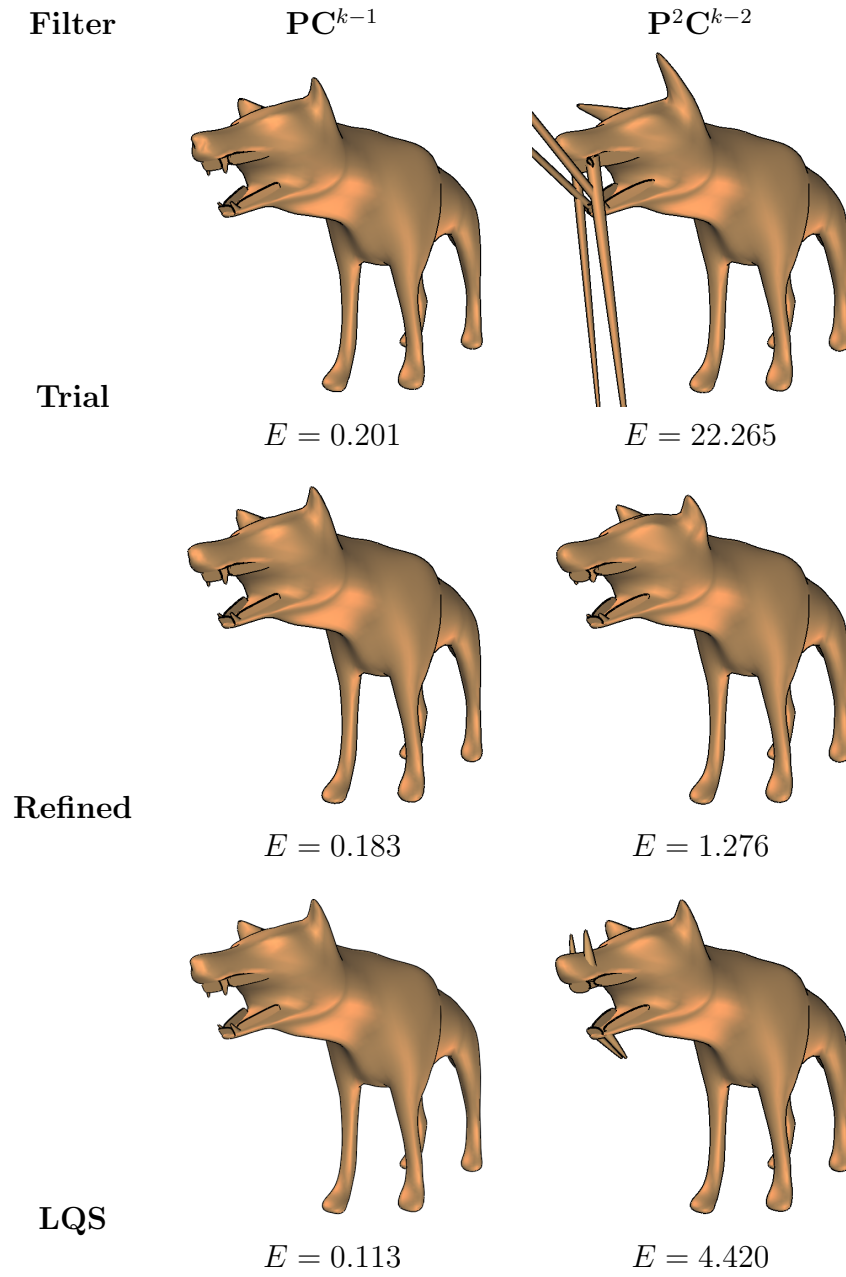


Figure 7.17: The wolf model of Fig. 7.13 is decomposed several times in succession.

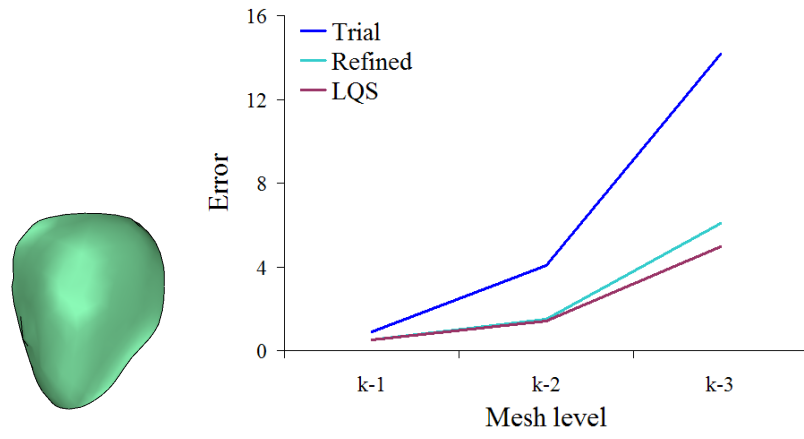


Figure 7.18: A plot of least-squares error for the alien model of Fig. 7.14.

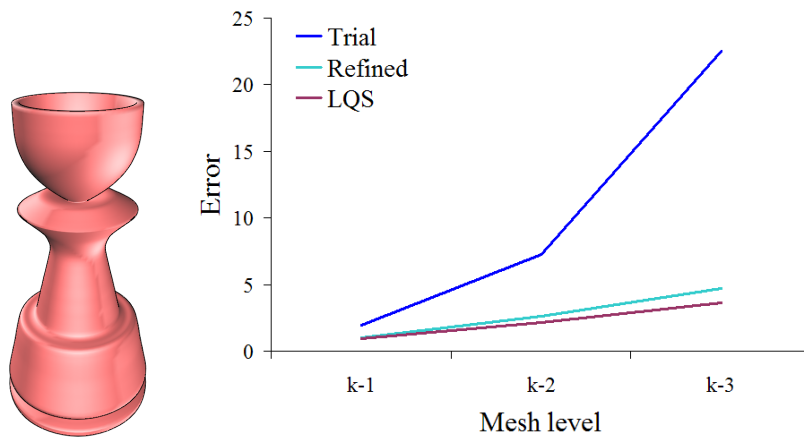


Figure 7.19: A plot of least-squares error for the pawn model of Fig. 7.15.

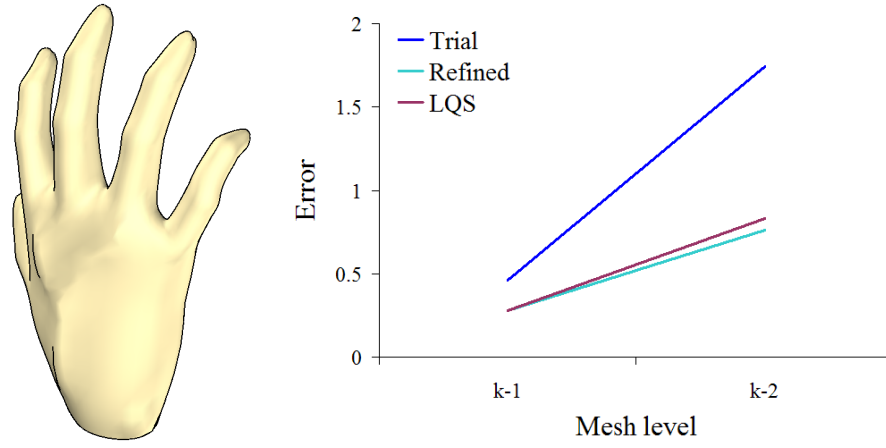


Figure 7.20: A plot of least-squares error for the hand model of Fig. 7.16.

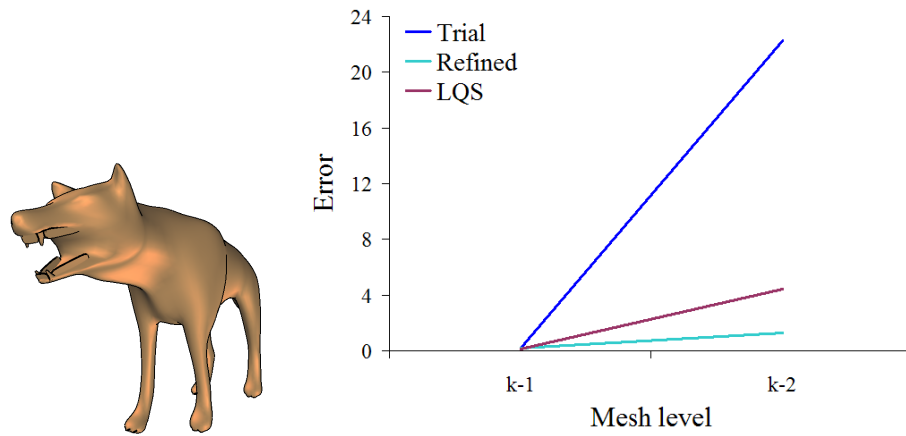


Figure 7.21: A plot of least-squares error for the wolf model of Fig. 7.17.

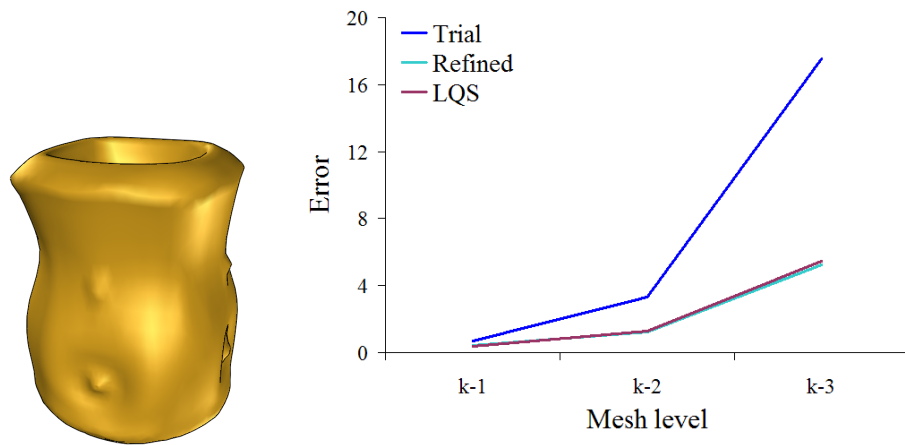


Figure 7.22: A plot of least-squares error for a vase model.

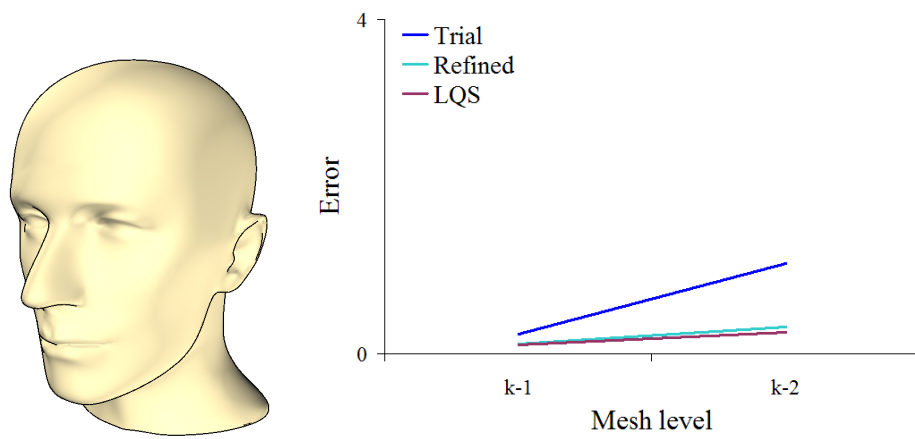


Figure 7.23: A plot of least-squares error for a head model.

7.2.2 Conclusion

As indicated by the error-versus-number of decompositions graphs of Figs. 7.18–7.23, the refined filter represents a large improvement over the trial filter, and more importantly it is competitive with the established LQS filter.

These results – both quantitative error measurements and qualitative visual evaluation of the meshes – further validate our method. As Fig. 7.12 depicts, the refinement step of Li et al. considers a wider neighborhood of details than our refinement; we would expect the wider support to yield a more optimal displacement. The fact that the refined filter produces equivalent quality with narrower support is even more gratifying.

7.3 Catmull-Clark Surfaces

In Chapter 6, a complete MR system for Catmull-Clark subdivision surfaces was built by the method described in Chapter 3. Though the subdivision masks for this scheme are more complicated than others, the wavelet constraint was able to derive a trial filter with only slight alterations. The resulting *trial* filter,

$$\tilde{v}^k = \frac{n}{n-3}v^{k+1} - \frac{4}{n(n-3)}\sum_i e_i^{k+1} + \frac{1}{n(n-3)}\sum_i f_i^{k+1},$$

is the same decomposition filter as Lanquetin and Neveu [28].

After satisfying the storage constraint with the trial filter, a refinement process is applied to improve it. In the *refined* filter, every coarse vertex \tilde{v}^k is displaced by δ , where δ is determined by a local neighborhood of details:

$$\delta = \frac{\sum (r^2\alpha_e + s_i) d_i^e + \sum (r^2\alpha_f + t_i) d_i^f}{r^2 + \sum s_i^2 + \sum t_i^2}.$$

In Figs. 7.25–7.36, the high-resolution models of Fig. 7.24 are decomposed several times with both the trial and refined filters.

7.3.1 Analysis

In the muffin model of Fig. 7.25, both filters produce a reasonable results after one level of decomposition. After two levels, the trial filter is beginning to show some large errors while the refined filter is keeping the overall shape in tact. By the third decomposition, the error in the trial filter has blown up, while the refined filter has lost some of the original shape dimensions but has retained the overall shape. Figure 7.26 plots the error from each filter for this model.

Fig. 7.27 depicts the donut model of Fig. 7.24 depicted after several decompositions. After one and two decompositions, both filters have reasonably low errors. Visually, however, the refined filter looks much more faithful to the original object; one might say that the refined filter spends its error budget more wisely. After three decompositions, the refined filter outperforms the trial filter both visually and numerically. These error values, as well as after a fourth level of decomposition, are shown in the graph of Fig. 7.28.

Figure 7.29 shows each of the filters applied to a bullet model. As summarized in Fig. 7.30, the errors produced by each filter are numerically close. However, a visual inspection of the resulting meshes shows that the refined filter again spends its error budget more wisely: unlike the trial filter, the refined filter preserves the overall bullet-like shape of the mesh even after several decompositions.

A bicycle seat was decomposed with each filter, and the results are summarized in Figs. 7.31 and 7.32. For the first level, the trial and refined filters perform equally. For

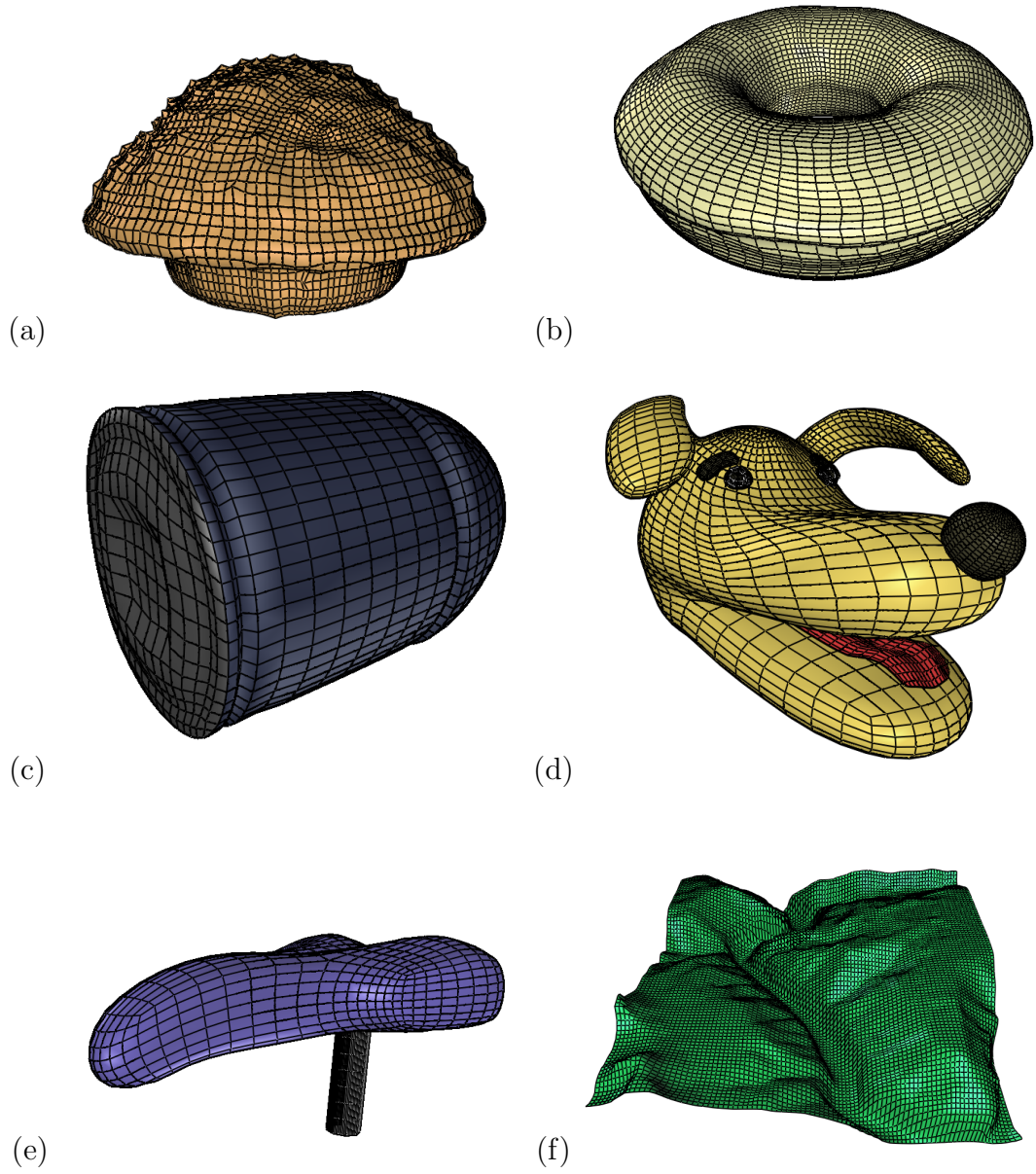


Figure 7.24: Original fine data \mathbf{C}^k for evaluating the Catmull-Clark MR system of Chapter 6.

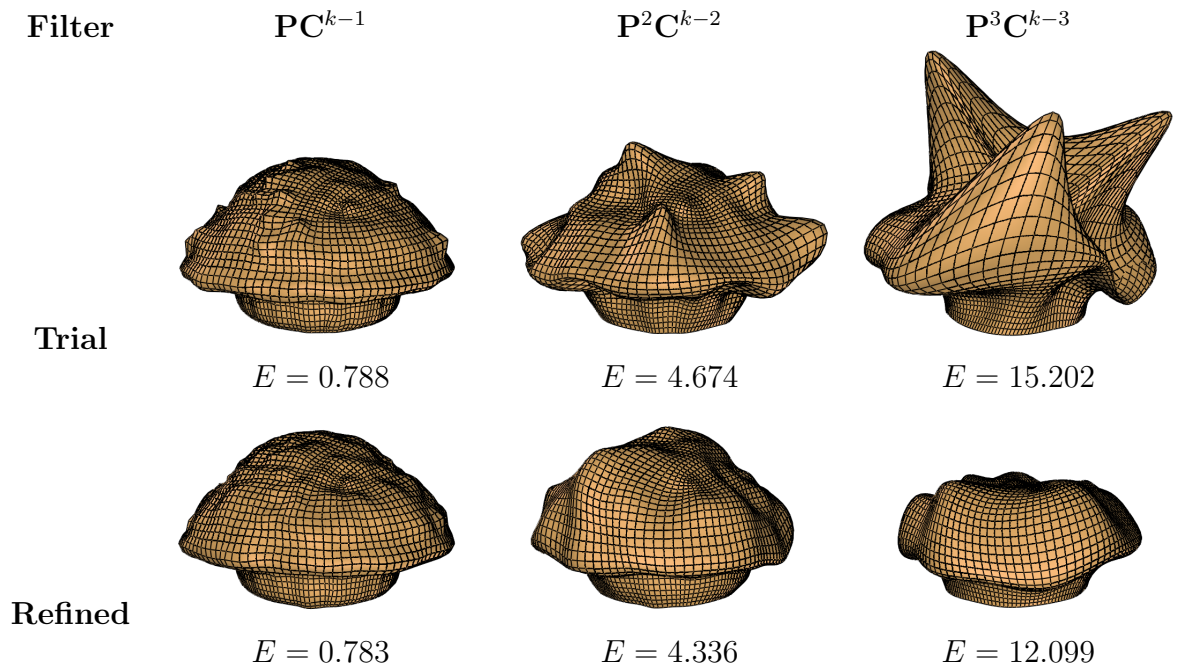


Figure 7.25: A semi-regular Catmull-Clark muffin model is decomposed with the trial and refined filters.

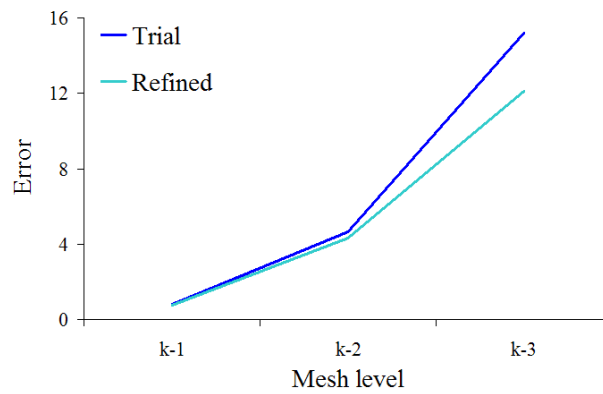


Figure 7.26: A plot of least-squares error for the muffin model of Fig. 7.25.

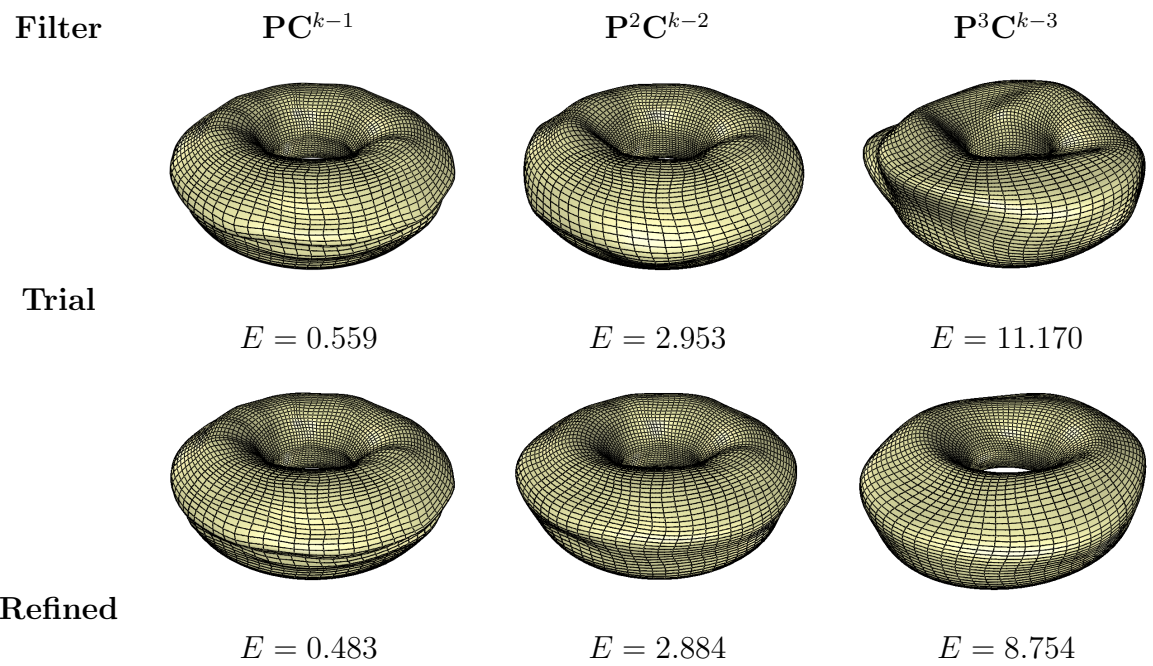


Figure 7.27: A semi-regular Catmull-Clark donut model is decomposed with the trial and refined filters.

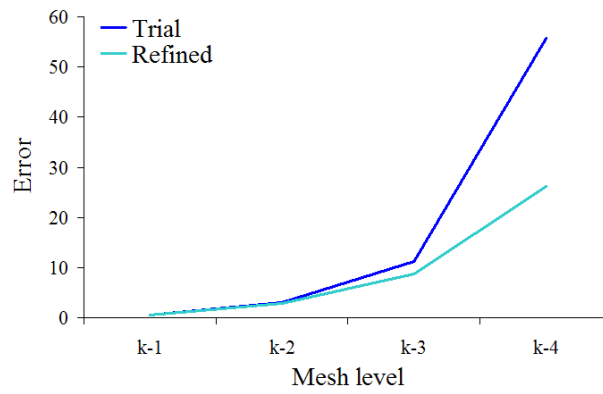


Figure 7.28: A plot of least-squares error for the donut model of Fig. 7.27.

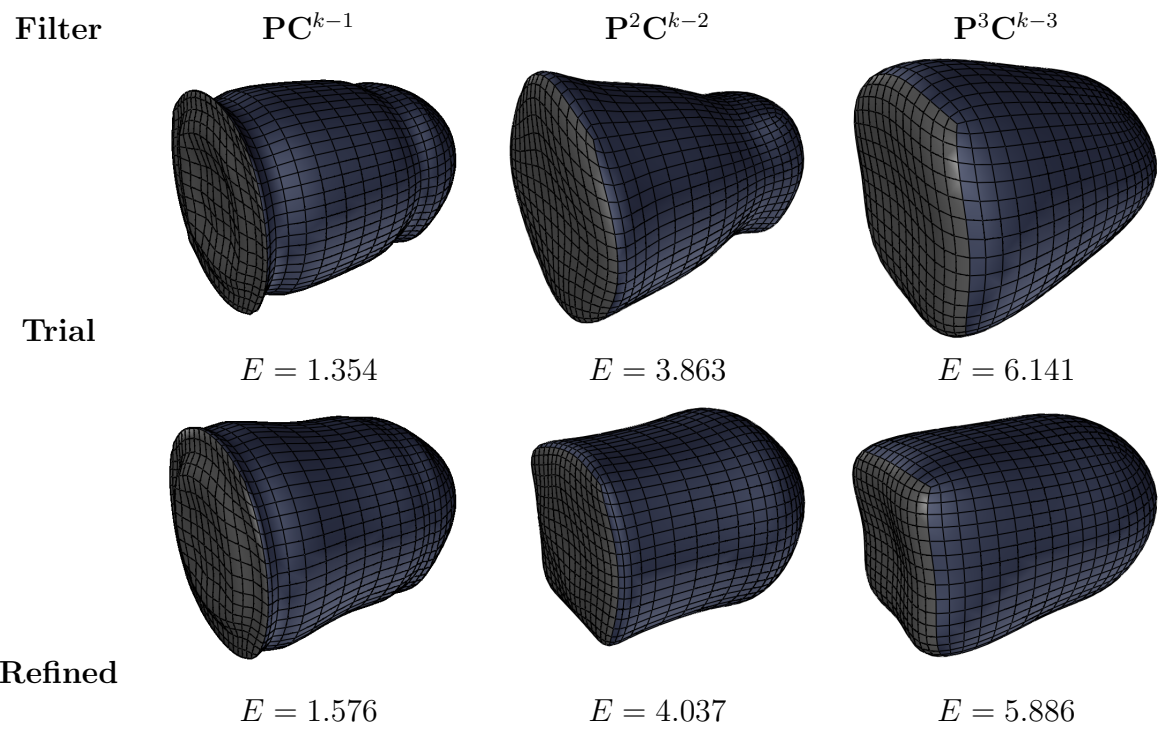


Figure 7.29: A semi-regular Catmull-Clark bullet model is decomposed with the trial and refined filters.

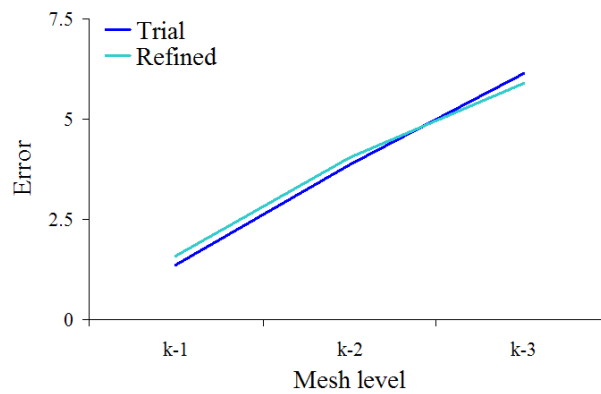


Figure 7.30: A plot of least-squares error for the bullet model of Fig. 7.29.

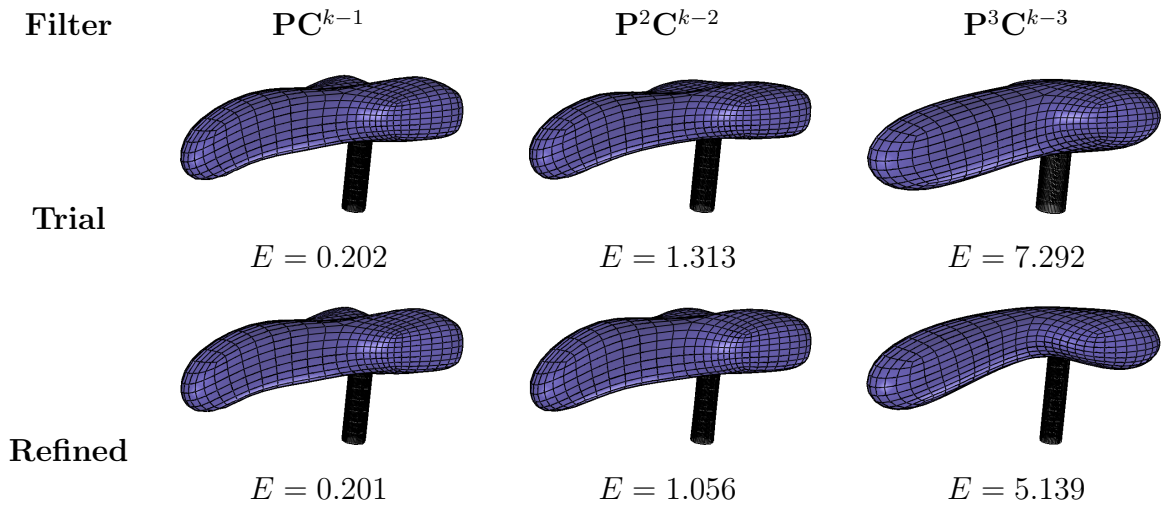


Figure 7.31: A semi-regular Catmull-Clark seat model is decomposed with the trial and refined filters.

more decompositions, the refined filter pulls ahead both visually and quantitatively.

The dog model first shown in Chapter 1 is decomposed with both the trial and refined filters; Fig. 7.33 shows the results. After one level of decomposition, the results are hard to distinguish. But after two levels the refined filter has significantly lower error and, equally important, produces a better-looking mesh. compare the ears of the dog under each filter; the refined filter is clearly more able to preserve the shape and dimensionality of that feature.

Figure 7.35 summarizes the results of applying each filter to a teddy bear model. The interesting aspect of this model is that the refined model performs slightly worse than the trial filter. The error of each filter is relatively low, but the refined filter should still be able to outperform the trial filter.

Finally, Fig. 7.36 shows the decomposition of a terrain patch with the Catmull-Clark filters. After one level, the difference between the trial and refined filters is negligible. By the second decomposition, the trial filter is beginning to show some

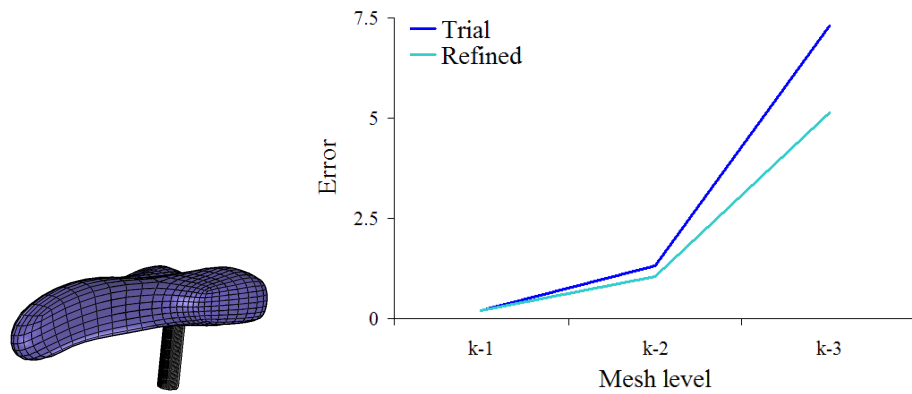


Figure 7.32: A plot of least-squares error for the seat model of Fig. 7.31.

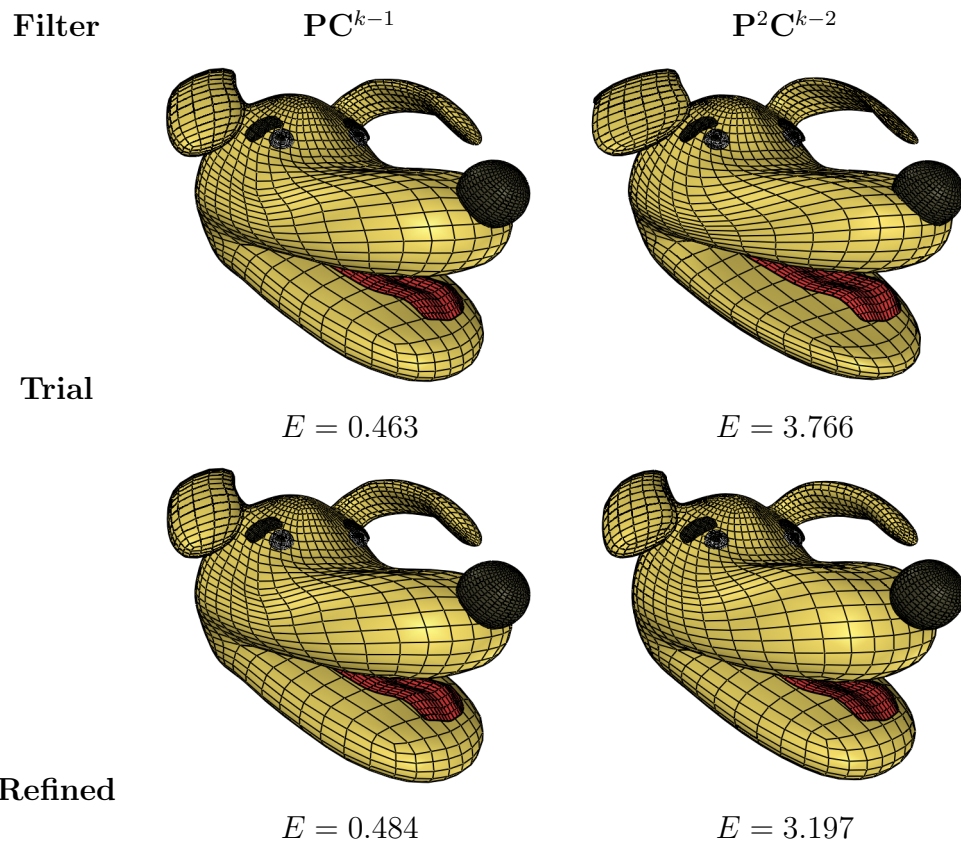


Figure 7.33: A semi-regular Catmull-Clark dog model is decomposed with the trial and refined filters.

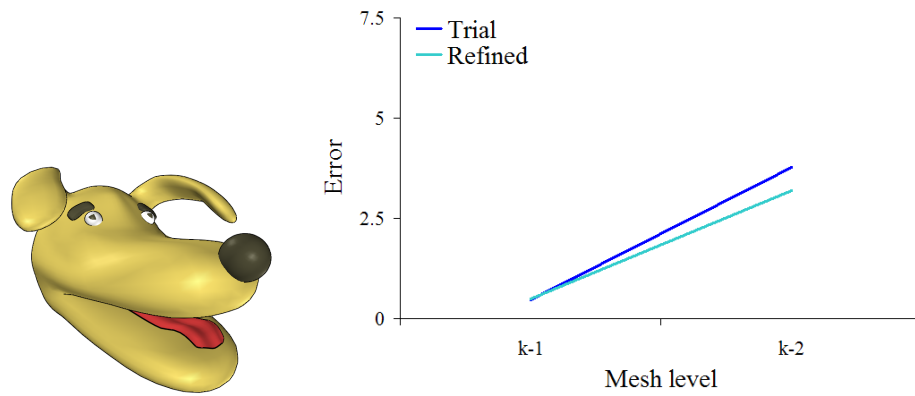


Figure 7.34: A plot of least-squares error for the dog model of Fig. 7.33.

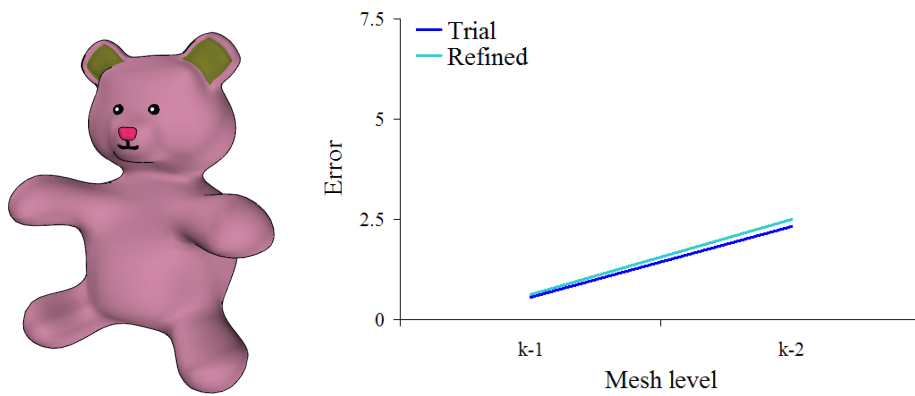


Figure 7.35: A plot of least-squares error for a teddy bear model.

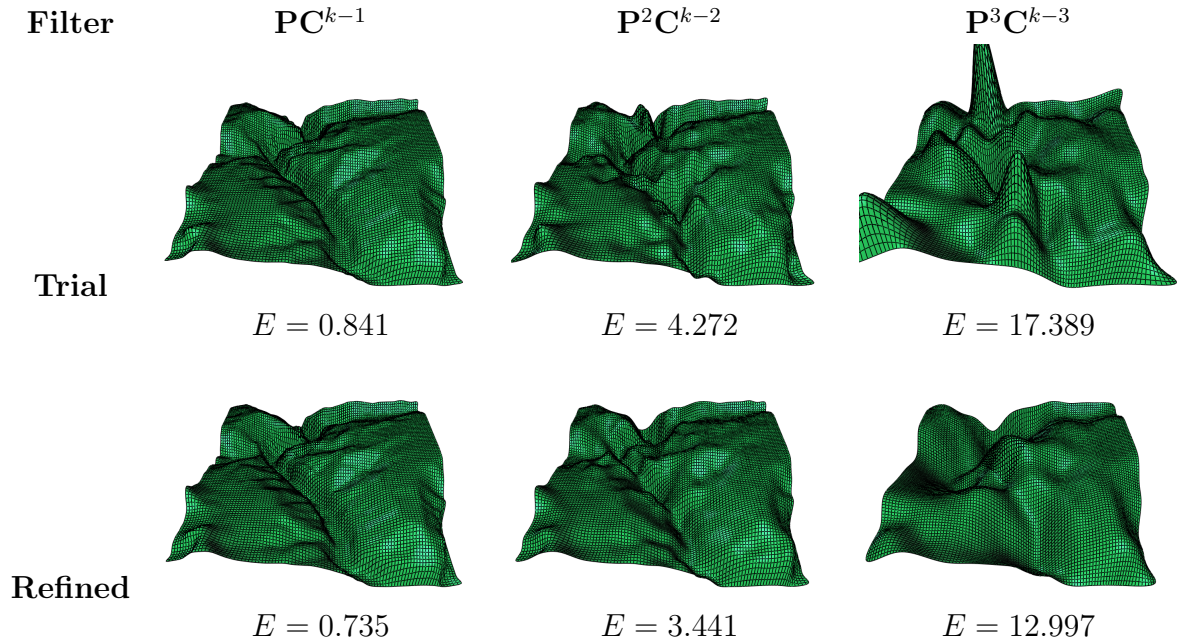


Figure 7.36: Terrain model decomposed with the trial and partially-refined Catmull-Clark decomposition filters.

error amplification, and by the third level the terrain bears little resemblance to the original model. The refined filter is much better at maintaining the overall shape of the terrain, even after drastically reducing the mesh resolution by three levels. Figure 7.37 compares the results of the two filters, including a fourth level of decomposition not shown.

7.3.2 Conclusion

The results presented here indicate that the trial decomposition filter produces large errors in general, as well as amplifying errors from previous levels. This result is expected, due to the narrow support of the trial filter. The refinement step greatly improves upon the trial filter, especially when the results are evaluated visually: where the trial filter produces wavy or irregular meshes, the refined filter respects

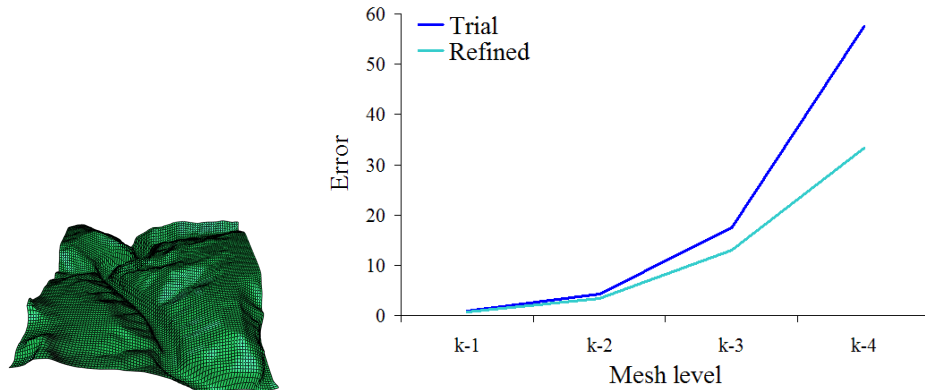


Figure 7.37: A plot of least-squares error for the terrain model of Fig. 7.36.

the original shape and dimensions of an object.

Curiously, the numerical results are not as strong as the visuals would suggest. The error reduction realized by refinement is not as dramatic as with previous schemes, and in several cases the refined filter actually produces higher error according to the least-squares metric. Refinement still constitutes a major improvement because regardless of what the measured error is, visual faithfulness to the original model is of the utmost importance – and the refined filter is clearly better in this respect. Nonetheless, the numerical results seem to indicate that an even greater improvement to the trial filter could be made.

Chapter 8

Conclusions & Future Work

In this thesis, a general framework for constructing MR systems from existing subdivision schemes has been presented. The construction exploits the inherent even/odd-old/new vertex structure that all subdivision schemes share to categorize fine vertices into two groups: those that will be coarsened, and those that will be replaced with detail terms. This classification leads to the wavelet constraint, a way of satisfying the storage constraint by having missing details be computable from stored details.

The solution of the wavelet constraint results in a set of trail filters, which is a fully biorthogonal MR system but that typically produces high-error coarse approximations. Therefore, a local optimization is applied to reduce the error; because the wavelet terms represent geometric information, a local neighborhood of details is used to decide which direction and by how much to displace each coarse vertex.

In Chapters 4–6, this method was followed to produce MR systems for cubic B-spline curves, Loop subdivision surfaces, and Catmull-Clark surfaces. When compared against previous MR systems for these types of subdivision (Chapter 7), the generic construction is found to produce very competitive filters.

In Chapter 4, two different MR systems were developed, primarily distinguished by the width of the decomposition **A** filter. The narrower width-7 filter performed in line with the near-minimum-norm filter of Samavati & Bartels [3], but the wider width-11 filter was slightly outmatched by the wider filter of Samavati & Bartels. This result seems to indicate that the wider neighborhood of details used in the W-11

refinement are less indicative of a good displacement direction than the smaller W-7 neighborhood. Since the wider neighborhood analysis of W-11 is difficult to extend to mesh schemes, however, the important point is that the W-7 filter performed very well.

For Loop surfaces, the filters constructed in Chapter 5 generally performed well in comparison to the filters of Li et al.; there were instances where our filters performed better (Fig. 7.17, for instance), and also instances where their filter produced more pleasing results (Fig. 7.14). This underscores the difficulty of constructing an MR system that can perform equally well on all types of meshes.

The filters derived for Catmull-Clark surfaces in Chapter 6 represent a major improvement over the earlier work (Lanquetin and Neveu [28]) for two reasons. First, the storage constraint is satisfied, whereas previous work simply provides coarsening masks and assumes that detail terms will be stored for all vertices. Second, the refinement step produces much more stable coarse approximations than the trial filter.

Future Work

There are several possible directions to take this work in the future. An immediate goal is to find a more elegant solution to the valence-3 vertex issue in the Catmull-Clark filters (Sec. 6.2.1). First, the candidate positions suggested by Eqn. 6.9 are insensitive to high-frequency data because the position of the valence-3 vertex is not considered in the mask. Second, and more importantly, the special-case mask does not satisfy the wavelet constraint, and therefore the storage constraint cannot be

satisfied at these vertices.

In Chapter 4, two filters were reached: a width-7 filter and a width-11 filter. The W-11 filter was expected to perform better than the W-7 filter, because it takes into account a larger set of data to construct the coarse approximation. But in practice, the W-11 filter was generally worse than the narrower filters. However, in the comparison filters BS-7 and BS-11, the wider width-11 filter is consistently better. To reconcile these results, it would be interesting to recast the Samavati & Bartels filters as refinements of the trial filters and see what kind of refinement their filters implied. This would perhaps indicate a different refinement strategy that could be extended to mesh schemes to improve the quality of their MR systems.

There are some more general questions that arise from our construction. For instance, is the voting scheme proposed for the partial refinements even close to optimal? The results from cubic B-spline curves and Loop surfaces are encouraging, but there seems to be room for improvement in our Catmull-Clark filters. Perhaps the voting idea could be carried further into more game theory-like schemes in which each vertex would vote less selfishly and recognize that a more optimal solution is for every vertex to move a little bit. Ultimately, more investigation is needed to determine if and how much the filters could be improved.

In the longer term, having a full multiresolution system for Catmull-Clark opens the door to many interesting applications. Polygonal mesh representations of objects are always going to be popular because of their natural integration into the rendering pipeline; commodity graphics hardware is highly optimized to render polygons, not implicit surfaces or other types of object representations. Thus the creation of intuitive and powerful modeling tools is an important problem that needs to be

addressed in computer graphics.

A recent research direction in graphics is sketch-based modeling. Sketch-based modeling systems attempt to leverage a person's natural ability to draw or otherwise convey the important features of an object. This is a difficult problem for 3D modeling because sketching is a 2D process; the 3rd dimension must be inferred, and often the results are not what the user would expect. As well, most sketch-based systems produce produce irregular and imprecise meshes as output. To truly replace the traditional control-point based modeling paradigms, sketch-based systems must produce high-quality meshes. In the context of the techniques discussed in this thesis, a sketch-based system that produced semi-regular meshes that could be decomposed would be a great benefit.

Bibliography

- [1] E. Angel. *Interactive Computer Graphics*. Addison Wesley, Toronto, Ontario, third edition, 2003.
- [2] D.I. Azuma, D.N. Wood, B. Curless, T. Duchamp, D.H. Salesin, and W. Stuetzle. View-dependent refinement of multiresolution meshes with subdivision connectivity. In *AFRIGRAPH '03: Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa*, pages 69–78, New York, NY, USA, 2003. ACM Press.
- [3] R. Bartels and F.F. Samavati. Reversing Subdivision Rules: Local Linear Conditions and Observations on Inner Products. *Journal of Computational and Applied Mathematics*, 119:29–67, 2000.
- [4] M. Bertram. Biorthogonal Loop-Subdivision Wavelets. *Computing*, 72(1-2):29–39, 2004.
- [5] P. Beziér. Mathematical and Practical Possibilities of UNISURF. In *Computer Aided Geometric Design*, New York, 1974. Academic Press.
- [6] H. Biermann, I. Martin, F. Bernardini, and D. Zorin. Cut-and-paste editing of multiresolution surfaces. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 312–321, New York, NY, USA, 2002. ACM Press.
- [7] M. Botsch and L. Kobbelt. A remeshing approach to multiresolution modeling. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium*

- sium on Geometry processing*, pages 185–192, New York, NY, USA, 2004. ACM Press.
- [8] E. Catmull and J. Clark. Recursively Generated B-spline Surfaces on Arbitrary Topological Surfaces. *Computer-Aided Design*, 10(6):350–355, 1978.
 - [9] A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle. Interactive multiresolution surface viewing. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 91–98, New York, NY, USA, 1996. ACM Press.
 - [10] G. Chaikin. An Algorithm for High Speed Curve Generation. *Computer Graphics and Image Processing*, 3(4):346–349, 1974.
 - [11] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41(7):909–996, 1988.
 - [12] J. Dennis and R. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.
 - [13] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 85–94, New York, NY, USA, 1998. ACM Press.
 - [14] D. Doo. A Subdivision Algorithm for Smoothing Down Irregularly Shaped Polyhedrons. In *Proc. of International Conference on Interactive Techniques in Computer Aided Design*, pages 157–165, 1978.

- [15] D. Doo and M. Sabin. Behaviour of Recursive Subdivision Surfaces Near Extraordinary Points. *Computer-Aided Design*, 10(6):356–260, 1978.
- [16] N. Dyn, D. Levine, and J. Gregory. A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control. *ACM Trans. Graph.*, 9(2), 1990.
- [17] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 173–182, New York, NY, USA, 1995. ACM Press.
- [18] A. Finkelstein and D.H. Salesin. Multiresolution curves. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 261–268, New York, NY, USA, 1994. ACM Press.
- [19] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Publishing Company, Don Mills, Ontario, second edition, 1993.
- [20] H. Hoppe. Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108, New York, NY, USA, 1996. ACM Press.
- [21] H. Hoppe. View-dependent refinement of progressive meshes. In *Proceedings of the 24th International Conference on Computer Graphics and Interactive Techniques*, pages 189–198, 1997.
- [22] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh

- optimization. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 19–26, New York, NY, USA, 1993. ACM Press.
- [23] C.T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. Society for Industrial and Applied Math, 1995.
- [24] Lutz Kettner. Designing a data structure for polyhedral surfaces. In *SCG '98: Proceedings of the fourteenth annual symposium on Computational geometry*, pages 146–154, New York, NY, USA, 1998. ACM Press.
- [25] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive Geometry Compression. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 271–278, 2000.
- [26] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive Multi-resolution Modeling on Arbitrary Meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 105–114. ACM Press, 1998.
- [27] J. Lane and R. Riesenfeld. A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1):35–46, 1975.
- [28] S. Lanquetin and M. Neveu. Reverse catmull-clark subdivision. In *WSCG'2006 Conference Proceedings*, Plzen, Czech Republic, 2006.

- [29] A. Lee, D. Dobkin, W. Sweldens, and P. Schröder. Multiresolution mesh morphing. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 343–350, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.
- [30] A. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. Maps: multiresolution adaptive parameterization of surfaces. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 95–104, New York, NY, USA, 1998. ACM Press.
- [31] D. Li, K. Qin, and H. Sun. Unlifted Loop Subdivision Wavelets. In *12th Pacific Conference on Computer Graphics and Applications*, 2004.
- [32] N. Litke, A. Levin, and P. Schröder. Fitting Subdivision Surfaces. In *IEEE Visualization 2001*, pages 319–324, 2001.
- [33] C. Loop. Smooth Subdivision Surfaces Based on Triangles. Master’s thesis, Department of Mathematics, University of Utah, 1987.
- [34] Pixologic. Zbrush quick reference guide, July 2006. Available at <http://www.pixologic.com/zbrush/education/education-documentation.html>.
- [35] F.F. Samavati and R.H. Bartels. Multiresolution Curve and Surface Representation by Reversing Subdivision Rules. *Computer Graphics Forum*, 18(2):97–120, 1999.
- [36] F.F. Samavati and R.H. Bartels. Local filters of b-spline wavelets. In *Biometrics 2004*, 2004.

- [37] F.F. Samavati and R.H. Bartels. Diagrammatic tools for generating biorthogonal multiresolutions. *International Journal of Shape Modeling*, 12(1), 2006.
- [38] F.F. Samavati, N. Mahdavi-Amiri, and R. Bartels. Multiresolution Surfaces having Arbitrary Topologies by a Reverse Doo Subdivision Method. *Computer Graphics Forum*, 21(2):121–136, 2002.
- [39] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen. Decimation of triangle meshes. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 65–70, New York, NY, USA, 1992. ACM Press.
- [40] E. Stollnitz, T. DeRose, and D. Salesin. Wavelets for Computer Graphics: A Primer, Part 1. *IEEE Computer Graphics and Applications*, 15(3):76–84, 1995.
- [41] E. Stollnitz, T. DeRose, and D. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1996.
- [42] V. Surazhsky and C. Gotsman. Explicit surface remeshing. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 20–30, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
- [43] W. Sweldens. Wavelets and the Lifting Scheme: A 5 Minute Tour. *Z. Angew. Math. Mech.*, 76 (Suppl. 2):41–44, 1996.

- [44] W. Sweldens. The Lifting Scheme: A Construction of Second Generation Wavelets. *SIAM J. Math. Anal.*, 29(2):511–546, 1997.
- [45] W. Sweldens and P. Schröder. Building Your Own Wavelets At Home. In *Wavelets in Computer Graphics*, pages 15–87. ACM SIGGRAPH Course notes, 1996.
- [46] L. Wecker, F.F. Samavati, and M. Gavrilova. Iris synthesis: A reverse subdivision application. In *Proceedings of Graphite 2005, in association with ACM SIGGRAPH*, pages 121–125, Dunedin, New Zealand, 2005.
- [47] D. Zorin and P. Schröder. Subdivision for Modeling and Animation. In *SIGGRAPH 2000 Course Notes*, 2000.
- [48] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 189–192, New York, NY, USA, 1996. ACM Press.
- [49] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 259–268, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

Appendix A

Notation

Multiresolution systems exist within a nested set of linear function spaces $V^0 \subset V^1 \subset \dots$, and a complementary set of wavelet spaces $\{W^0, W^1, \dots\}$, such that $W^k + V^k = V^{k+1}$.

The basis functions $\Phi^k = [\phi_0^k(x) \dots \phi_{v(k)-1}^k(x)]$ of V^k are called *scaling functions*. The scaling functions defined by a subdivision procedure define a nested function space, and vice versa. In such a setting, the scaling functions are said to be *refinable*. *Wavelets* $\Psi^k = [\psi_0^k(x) \dots \psi_{w(k)-1}^k(x)]$ are the basis functions of the wavelet spaces W^k .

The subdivision matrix \mathbf{P}^k represents the refinement of the scaling functions: $\Phi^{k+1} = \Phi^k \mathbf{P}^k$. Similarly, the matrix \mathbf{Q}^k matrix quantifies the relationship between wavelets at level $k-1$ and the scaling functions of level k : $\Psi^{k+1} = \Phi^k \mathbf{Q}^k$. Together, \mathbf{P}^k and \mathbf{Q}^k are called *reconstruction filters*.

Multiresolution systems encapsulate two operations: reconstruction (the increasing of resolution), and decomposition (decreasing resolution). Decomposition is represented by two matrices, \mathbf{A}^k and \mathbf{B}^k , which are related to the scaling functions and wavelets as

$$[\Phi^{k-1} | \Psi^{k-1}] \begin{bmatrix} \mathbf{A}^k \\ \mathbf{B}^k \end{bmatrix} = \Phi^k .$$

\mathbf{A}^k and \mathbf{B}^k are *decomposition filters*.

The object to be operated on by a multiresolution system, be it a signal, curve,

or mesh, is represented as a vector of scaling function coefficients, \mathbf{C}^k . The terms *coarse* and *fine* are relative terms referring to the resolution at which an object representation exists.

When an object \mathbf{C}^k is downsampled or decomposed to \mathbf{C}^{k-1} , some information is lost. The missing information \mathbf{D}^{k-1} can be embedded in the wavelet space W^{k-1} as coefficients of the wavelets. These wavelet coefficients are informally referred to as the *details*. When the coarse data is then subdivided, the results is an approximation, denoted $\tilde{\mathbf{C}}^k$, of the original data \mathbf{C}^k .

For simplicity of notation, the superscripts are often omitted. In that case, the original fine data is represented as \mathbf{F} , while the coarse data after one decomposition is denoted \mathbf{C} . When \mathbf{C} is subdivided, an approximation $\tilde{\mathbf{F}}$ results.

In the development of multiresolution filters, the local nature of subdivision allows the analysis to be carried out for a single representative vertex. For cubic B-splines, the representative vertex $f_0 \in \mathbf{F}$ is neighbored by vertices f_{-1}, f_{-2}, \dots on the left and f_1, f_2, \dots on the right. When the fine data is coarsened, f_0 is replaced by a coarse approximation $c_0 \in \mathbf{C}$; fine vertices with even indexes are also coarsened. Meanwhile, fine vertices with odd indices, $f_{\pm 1}, f_{\pm 2}, \dots$, are replaced by detail terms $d_{\pm 1}, d_{\pm 2}, \dots$.

For Loop surfaces, the representative vertex is $f_0 \in \mathbf{F}$, which is replaced by coarse approximation $c_0 \in \mathbf{C}$ when decomposed. If the valence of f_0 is n , then the 1-ring (the immediate neighbors) of f_0 consists of f_1, f_2, \dots, f_n . After decomposition, these vertices are replaced by details d_1, d_2, \dots, d_n .

For Catmull-Clark surfaces, there are three types of vertices in a semi-regular mesh (that is, a mesh that results from Catmull-Clark subdivision, or has the same

connectivity): vertex-, edge, and face-vertices, of which vertex-vertices are considered even, and edge- and face-vertices are considered odd. The representative fine vertex is $v^{k+1} \in \mathbf{C}^{k+1}$ of valence n , which shares an edge with edge vertices $e_1^{k+1}, e_2^{k+1}, \dots, e_n^{k+1}$, and shares a face with face vertices $f_1^{k+1}, f_2^{k+1}, \dots, f_n^{k+1}$. After decomposition, v^{k+1} is replaced by v^k , and the neighboring edge and face vertices are replaced by details, $d_1^e, d_2^e, \dots, d_n^e$ and $d_1^f, d_2^f, \dots, d_n^f$ respectively.

List of Variables

- A** The decomposition filter for coarsening: $\mathbf{C} = \mathbf{AF}$. The trial version is denoted $\widetilde{\mathbf{A}}$.
- $\widetilde{\mathbf{A}}$ The trial decomposition filter, such that $\widetilde{\mathbf{C}} = \widetilde{\mathbf{A}}\mathbf{F}$.
- B** The decomposition filter for computing details, or wavelet coefficients: $\mathbf{D} = \mathbf{BF}$. The trial version is denoted $\widetilde{\mathbf{B}}$.
- c A coarse vertex, $c \in \mathbf{C}$. A subscript 0 is often used to denote a representative vertex c_0 . A coarse vertex produced by the trial decomposition filter $\widetilde{\mathbf{A}}$ is denoted \tilde{c} .
- C** A coarse (low-resolution) object: \mathbf{C} represents a vector of the object's vertices.
- d A detail vector. When an object \mathbf{F} is decomposed, some vertices f_{odd} are replaced with detail vectors. In Catmull-Clark subdivision, there are vertex-, face- and edge-vertices; the corresponding details are denoted d^v, d^f and d^e respectively.
- D** A vector of wavelet coefficients produces by decomposition: $\mathbf{D} = \mathbf{BF}$.

- e In Catmull-Clark subdivision, refers to an edge-vertex.
- E Denotes the error in an object representation \mathbf{C}^{k-j} , relative to some original data \mathbf{C}^k . The least-squares error metric is used, meaning that $E(\mathbf{C}^{k-j}) = \sqrt{\|\mathbf{C}^k - \mathbf{P}^j \mathbf{C}^{k-j}\|^2}$.
- f A fine vertex, $f \in \mathbf{F}$. For Catmull-Clark surfaces, f refers to a face-vertex.
- \tilde{f} A fine vertex that results from subdividing a coarse approximation of f . If there is any error in the decomposition, then $f \neq \tilde{f}$, and $d = f - \tilde{f}$.
- \mathbf{F} A fine (high-resolution) object: \mathbf{F} represents a vector of the object's vertices.
- k Denotes the level of the multiresolution hierarchy at which an object exists, or at which a filter acts. For instance, $\mathbf{C} = \mathbf{A}\mathbf{F}$ could be alternatively expressed as $\mathbf{C}^{k-1} = \mathbf{A}^k \mathbf{C}^k$.
- \mathbf{L} A matrix encapsulating the computation of the refinement vectors from the details: $\Delta = \mathbf{L}\mathbf{D}$.
- n Vertex valence, or the number of edges incident to a vertex.
- n^f Face valence, or the number of vertices (equivalently edges) in a face.
- \mathbf{P} The subdivision matrix, $\mathbf{F} = \mathbf{P}\mathbf{C}$.
- \mathbf{Q} The reconstruction matrix for interpreting the detail terms: $F = \mathbf{P}\mathbf{C} + \mathbf{Q}\mathbf{D}$. The trial version is denoted $\tilde{\mathbf{Q}}$.
- r A coefficient used in Catmull-Clark refinement.
- s A coefficient used in Catmull-Clark refinement.
- t A coefficient used in Catmull-Clark refinement.

- v In Catmull-Clark subdivision, refers to a vertex-vertex.
- V^k A function space in which an object \mathbf{C} or \mathbf{F} exists.
- W^k A wavelet space, in which the details D exist. W^k is the complement of V^k in V^{k+1} .
- α The free variable in the wavelet constraint; when α is determined, the trial filters can be found. In Catmull-Clark subdivision, different α are required for face- and edge-vertices; they are denoted α_f and α_e respectively.
- β A parameter of Loop subdivision, β is a function of vertex valence n .
- δ A refinement vector, used to displace a coarse vertex \tilde{c} such that after subdivision, $c = \tilde{c} + \delta$ has lower error than \tilde{c} .
- Δ A collection of refinement vectors $\Delta = [\delta_0, \delta_1, \dots]$. An object $\tilde{\mathbf{C}}$ is refined as $\mathbf{C} = \tilde{\mathbf{C}} + \Delta$.
- κ The coefficient applied to the 1-ring of details to compute the Loop refinement vector δ .
- μ The scaling factor applied to a refinement vector δ under a partial refinement strategy.
- $\phi_i^k(x)$ Wavelets, which define the basis functions of W^k .
- $\Phi^k(x)$ The set of all wavelets of W^k .
- $\psi_i^k(x)$ Scaling functions, which define the basis functions of function space V^k .
- $\Psi^k(x)$ The set of all scaling functions of V^k .

Appendix B

System Interface

B.1 Cubic B-Spline Curves & Patches

A curve editing application was written to generate results for the cubic B-spline multiresolution system described in Chapter 4. Figure B.1 shows the main window of this application.

The editing area of the application is where users can manipulate vertices of the curve. Vertices are selected by clicking with the left mouse button, and moved by dragging them. Vertices can be deleted by right-clicking on them.

To increase the resolution of the curve by via subdivision, the user can click the Subdivide button. If there are any details available, they will be used to reconstruct high-frequency information.

The resolution of a curve can be reduced at any time by decomposing; this is accomplished by clicking the Decompose button. When decomposing, the application will also compute and store details in place of vertices that are removed during decomposition. The details allow for full reconstruction of the high-resolution features after any editing is done at the coarse level.

The filter selection box allows users to easily toggle between several filters: the trial filter (Eqn. 4.8), the W7 filter (Eqn. 4.26), the W11 filter (Eqn. 4.27), and the comparison filters of Samavati & Bartels [35].

Curves created in this application can be saved to disk in a proprietary curve

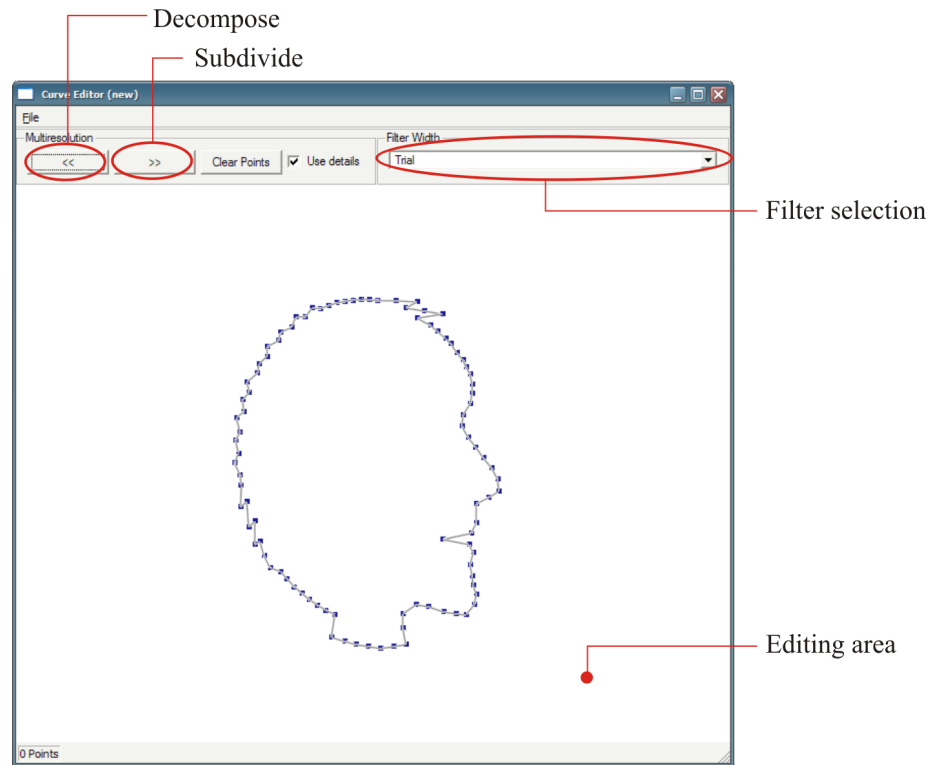


Figure B.1: The main window of the cubic B-spline curve editor.

format, and later be loaded back into the application for decomposition or further editing.

An extension of this application was also written to allow the creation and editing of 2D subdivision patches. The interface for the 2D program is similar.

B.2 Loop & Catmull-Clark Surfaces

To test the Loop and Catmull-Clark multiresolution systems described in Chapters 5 and 6, a mesh editing application was written. Mesh editing is an intensely complex task, so the editing options and capabilities of our system are fairly limited.

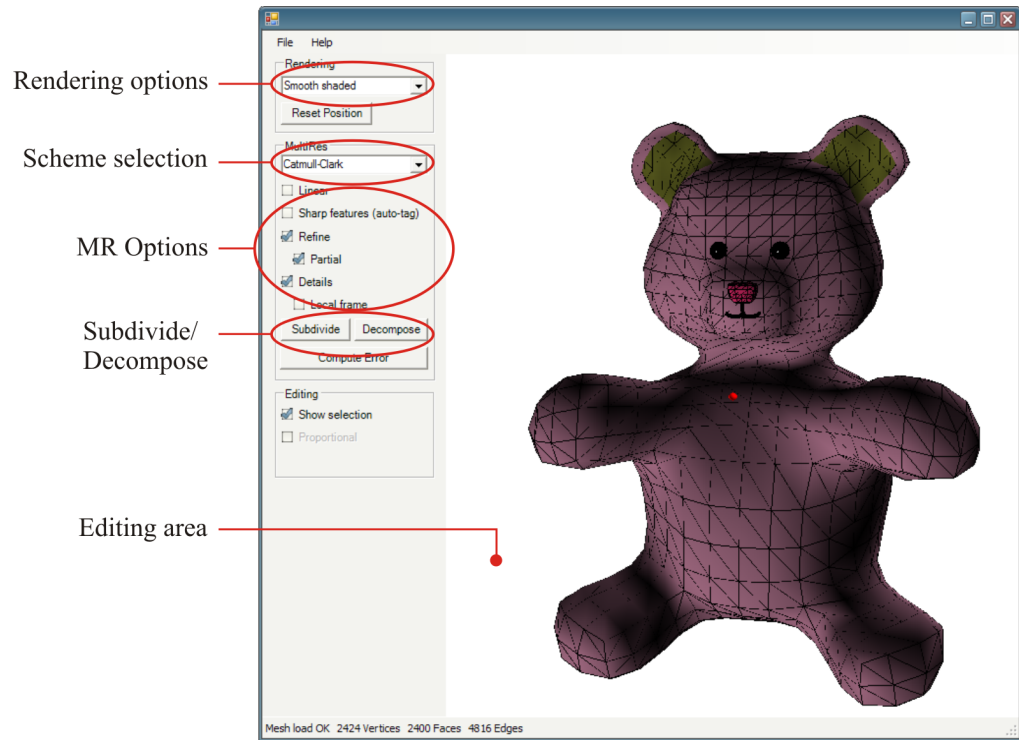


Figure B.2: System window.

The main application window is shown in Fig. B.2. On the left pane, there is a set of interface items for controlling the behavior of the underlying MR systems, while the right side of the window is dedicated to rendering the mesh and the editing interface.

There are four rendering options available to the user. The traditional rendering mode is *wireframe*, in which only the edge connections between vertices are rendered; the faces are transparent. For larger models, a wireframe rendering is indecipherable because of the sheer number of edges being rendered. The *outline* mode alleviates this issue by rendering only lines that would be visible with opaque faces. *Flat*

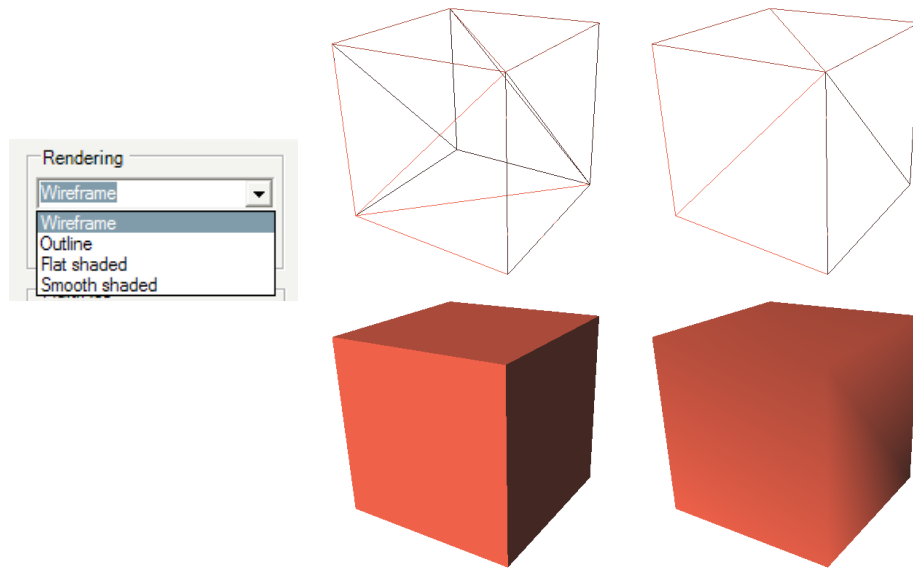


Figure B.3: Four different rendering modes are available: wireframe, outline, flat shaded, and smooth shaded.

shading is a traditional mode for visualizing the face structure of a mesh while at the same time conveying some color and material properties. Finally, *smooth shading* is a visually pleasing output that smoothly interpolates the material across each face. Figure ?? illustrates each of these modes.

The user can dynamically switch between each MR system with the scheme selection tool (Fig B.4). The default scheme is Catmull-Clark, but Loop and LQS Loop (an implementation of Li et al. [31] are also available. When a new scheme is selected, the currently loaded mesh is tested for compatibility by the scheme and adjusted accordingly. For example, Loop subdivision only operates on triangle meshes, so the Loop schemes will convert any non-triangle faces in the active mesh to triangles.

Regardless of which subdivision scheme is active, there are some common options available (Fig B.5). If the *Linear* checkbox is enabled, then subdivision will only split

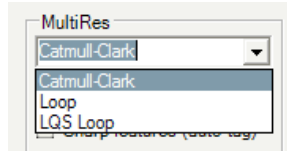


Figure B.4: Three multiresolution schemes were implemented: Catmull-Clark, (Chapter 6), Loop (Chapter 5), and LQS Loop (from Li et al. [31]).

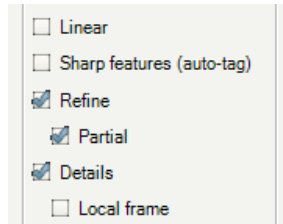


Figure B.5: Several options are available to control the behavior of each multiresolution system.

the faces; even vertices will not be displaced, and edge/face vertices will be simple linear averages. When the *Sharp features* option is enabled, the application will automatically identify edges that lie between faces that belong to different groups, and tag the edge and corresponding vertices as being sharp features; this means that they will be subdivided with boundary masks rather than regular masks. The *Refine* option, and the *Partial* sub-option, determines whether the refinement step will be applied during decomposition. Finally, the *Details* option dictates whether the application should use computed details during subdivision to reconstruct the high-frequency information; the sub-option *Local frame* controls whether the detail terms are represented in a global or local coordinate frame (local frames generally provide more intuitive editing results).

The editing area of the application allows a mesh to be interactively manipulated in some basic ways. Using the mouse, a user can select multiple vertices and

then apply affine transformations to them. There are three basic editing operations available: *translation*, *scaling*, and *rotation*. Figure B.6 illustrates the translation interface. When a vertex is selected, the vertex can be translated by holding the G key and dragging the mouse; the selected vertices will be displaced within a plane that is perpendicular to the view direction, as determined by the mesh's rotation quaternion.

The scaling operation is performed similarly. By holding the S key and dragging the mouse, the selected vertices are uniformly scaled about the user-defined origin. A helpful interface is displayed to indicate the amount of scaling that is being applied, as shown in Fig. B.7. The black box shows the initial position (scaling by 1.0), and the yellow box indicates the amount of scaling being applied. The results are also previewed while the mouse button is held down, and then applied when the button is released.

A vertex selection can be rotated about an arbitrary axis and origin by holding the R key and dragging the mouse. The axis of rotation is defined to be pointing into the screen, and is computed by rotating the z axis by the inverse of the mesh's quaternion. The origin of rotation is defined by the user. Figure B.8 illustrates the rotation operation, including the user interface that is displayed while the user is rotating.

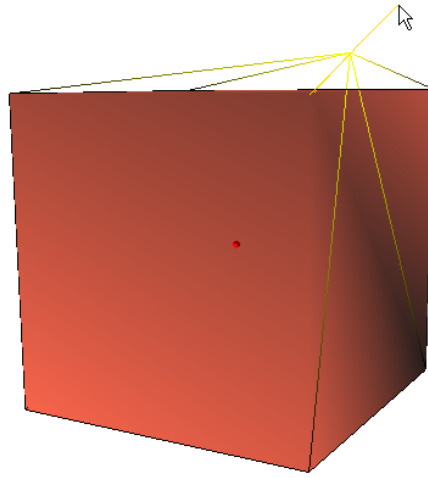


Figure B.6: Editing by translation: the selected vertices are moved in a plane perpendicular to the camera direction.

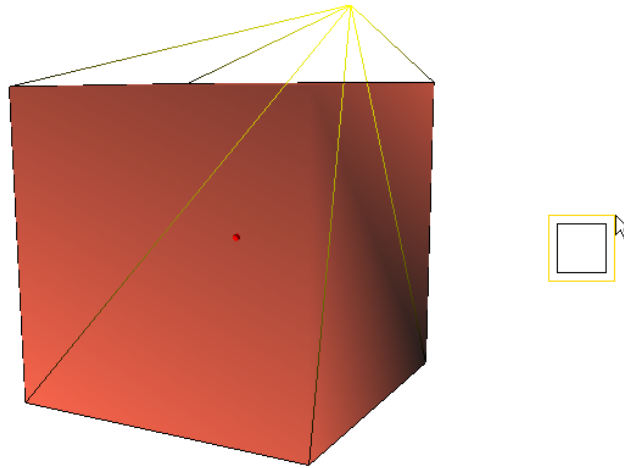


Figure B.7: Scaling: the selected vertices are scaled uniformly about a user-defined origin point. A visual guide is shown to indicate the amount of scaling being applied.

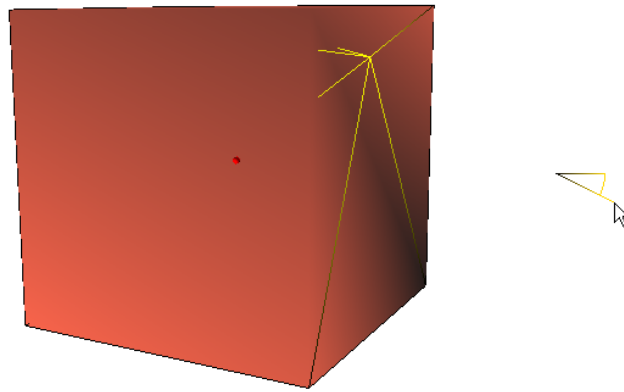


Figure B.8: When rotating, the selected vertices are rotated about an axis pointing out of the screen, centered at the user-defined origin point. A helpful user interface helps the user to control the amount of rotation.

Appendix C

Implementation Details

C.1 Cubic B-Spline Curves & Patches

The application described in Sec. B.1 was implemented in C++, using OpenGL for rendering duties.

Because of the simple nature of subdivision curves, built-in types were used as the data structures in the application. A standard template library (STL) *vector* was used to model both the curve and the details. The only difficulty in this approach is ensuring that the indexing is correct.

A subdivision patch editor was created later on to test the filters on more complex objects. This application was written in the Python programming language, with OpenGL again providing the rendering functionality.

C.2 Loop & Catmull-Clark Surfaces

The application described in Sec. B.2 was implemented in the C# programming language, with rendering duties handled by Microsoft DirectX. Version 2.0 of the .NET programming environment was used, mainly because the introduction of *generics* (equivalent to *templates* in C++) allowed some data structures to be implemented more naturally.

The class structure of the application is discussed in Sec. C.2.1, while Sec. C.2.2

describes the mesh data structure in depth. Finally, Sec. C.2.3 considers the important problem of mesh tagging.

C.2.1 Class Structure

The application was designed with extensibility in mind. Whenever possible, functionality was abstracted into a base class and then implemented explicitly in a child class. Figure C.1 shows the main classes and the relationships between them.

The top-level class that the user interface interacts with is the **MRObj** class. An **MRObj** exposes a simple interface, **IMeshInterface**, which defines a set of fundamental operations: Open, Save, Render, Refine (subdivide or reconstruct), and Coarsen (decompose).

An **MRObj** instance contains many other classes to facilitate all of this functionality. Of primary importance to the material in this thesis is the **MRMesh** class. An **MRObj** is comprised of zero or more **MRMesh**s (the *pieces*). This encapsulation was imposed to gracefully handle objects that have independent sets of faces and vertices. For instance, a car model might model the wheels as toruses that are not connected in any way to the body of the vehicle. If each independent set of faces is treated as a complete mesh, then it can be subdivided and decomposed independent of the rest of the object.

The **MRMesh** class is where the actual mesh data – vertices, faces, edges, materials, and so on – is stored. Each **MRMesh** is capable of rendering itself, so the parent **MRObj**’s render method simply calls the render method of each **MRMesh** piece. The geometric information stored in a half-edge data structure (Sec. C.2.2). In brief, each **MRMeshFace** and **MRMeshVertex** has a reference to a **MRMesh**-

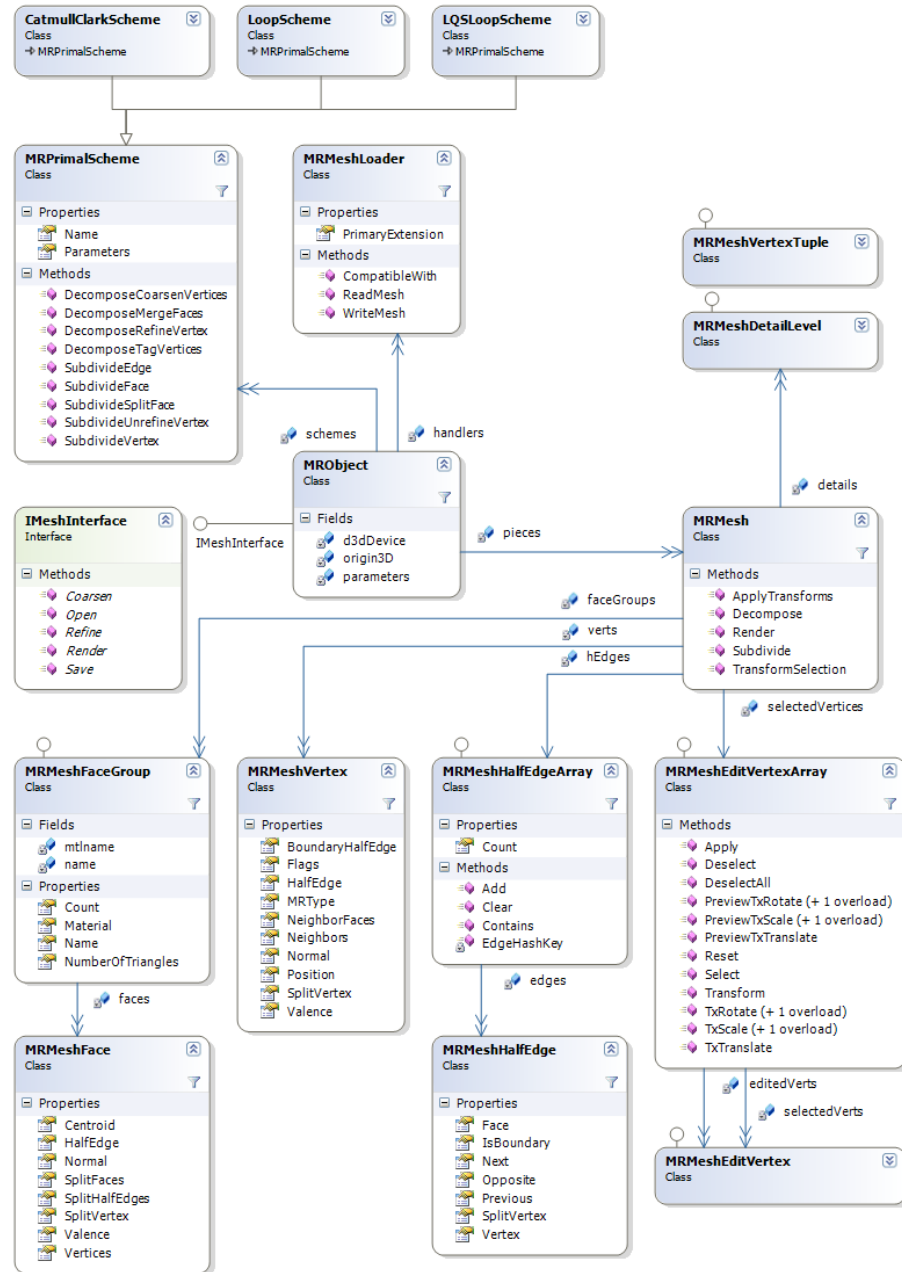


Figure C.1: Class diagram.

HalfEdge, from which the vertices in the face and the neighborhood of a vertex can be easily traversed. The faces are separated into **MRMeshFaceGroups**; usually, faces are grouped according to material type. The vertices are stored in a simple List data structure, while the half-edges are stored in a **MRMeshHalfEdgeArray** that encapsulates some hashing functionality.

To facilitate editing operations, each MRMesh contains an **MRMeshEditVertexArray**. Each **MRMeshEditVertex** is initialized to the position of its associated vertex, and when a user applies an editing operation an intermediate position is created. The MRMeshEditVertexArray class wraps up a collection of MRMeshEditVertexes and then exposes some methods for applying editing operations, such as translation, scaling, and rotation.

Finally, an MRMesh contains an array of **MRMeshDetailLevels**. As the name implies, each MRMeshDetailLevel represents the details from a particular level of the mesh hierarchy. Details are stored and looked up via a hashing algorithm based on the surrounding coarse vertices, which are represented by an instance of **MRMeshVertexTuple**. More specifically, an edge detail for edge $\langle v_1, v_2 \rangle$ is hashed by creating a consistent hash value from the sorted tuple (v_1, v_2) ; for a face detail, a hash value is created from a tuple made up of vertices in the face.

An MRMesh can be subdivided/reconstructed or decomposed by passing it an instance of the **MRPrimalScheme** class; an MRObject contains a list of available subdivision schemes from which the user can choose. **CatmullClarkScheme**, **LoopScheme**, and **LQSLoopScheme** are all instance of MRPrimalScheme.

To have a consistent MR interface and allow for easy swapping between different schemes, each instance of MRPrimalScheme must override the virtual methods

defined in `MRPrimalScheme`. For subdivision, there are the `SubdivideVertex`, `SubdivideFace`, `SubdivideEdge`, and `SubdivideSplitFace` methods. The former three methods apply the subdivision mask to each vertex, edge, and face (if applicable), while the latter method is responsible for creating the new face structure. As well, there is a `SubdivideUnrefineVertex`; as the name indicates, this method should compute and subtract a vertex's refinement vector.

For decomposition, `MRPrimalScheme` defines four virtual functions. `DecomposeTagVertices` is responsible for partitioning a mesh into *even* and *odd* vertices, if possible. `DecomposeCoarsenVertices` replaces even vertices with coarse approximations, and odd vertices with details. `DecomposeMergeFaces` computes the coarse face structure, while `DecomposeRefineVertex` computes and applies the per-vertex refinement.

To facilitate these subdivision operations, each face, vertex, and edge has some additional data elements for storing intermediate data; because the subdivision masks rely on the original positions of vertices, subdivision cannot be done in place. Each vertex, edge, and face has a *SplitVertex* that represents the location of the associated vertex after subdivision. Additionally, each face has a set of *SplitFaces*, which represents the new faces that result from splitting the original face.

The final noteworthy class is the **MRMeshLoader** class. Each instance of this class must implement `Read` and `Write` methods, and also state which file types it is able to process. For example, a loader for the Wavefront OBJ file type would list “obj” as its *PrimaryExtension*, and could possibly claim to be compatible with “mtl” files (OBJ material files).

The `MRObj` class contains a list of `MRMeshLoader` instances. When a user

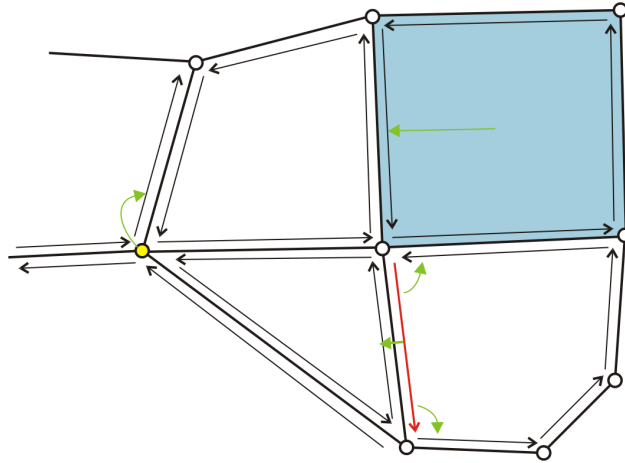


Figure C.2: The half-edge data structure. Each edge is split into two half-edges. A half-edge (red) stores pointers (green) to the next half-edge in the face, the opposite half-edge, and optionally to the previous half-edge. Each vertex (yellow) contains a pointer to some half-edge rooted at it. A face (blue) also requires only a single pointer to a half-edge in the face.

requests a particular file to be loaded, the `MRObj` will check each registered `MRMeshLoader` for compatibility with the file's extension; if a compatible loader is found, it is given the opportunity to load the file.

C.2.2 Half-Edge Data Structure

The main challenge in implementing a mesh-based MR system is in choosing a robust data structure. A common mesh data structure is the *half-edge* structure [24], illustrated in Fig. C.2.

In the half-edge structure, most of the connectivity information in a mesh is represented by edges. Each edge in a mesh is split into two directed edges, each of which contains a reference to the other as its *opposite* half-edge. Edges are related to the mesh's face structure: each edge (except boundary edges) is shared by two faces. Therefore, each half-edge is associated with exactly one face (the *adjacent face*), and

contains a reference to it. Each half-edge is also “rooted” at exactly one vertex (the *root vertex*) in the mesh and contains a reference to it as well. Finally, for rendering and subdividing a mesh there needs to be some way to traverse from vertex to vertex or around a face. To facilitate this, each half-edge contains a pointer to the *next* and *previous* half-edge in the associated face.

The geometric information is represented by a vertex structure, where a vertex v contains a position p , a normal n , and a pointer to *one of* the half-edges, h , rooted at it. For subdivision and multiresolution applications, a vertex’s 1-ring must often be traversed. This is easily accomplished in a half-edge structure. As Fig. C.3 illustrates, the opposite and next pointers of a half-edge can be used to walk around a vertex neighborhood. If h is rooted at v , then the neighbor along that edge is given by $h \rightarrow \text{next} \rightarrow \text{vertex}$. To traverse the vertex’s entire neighborhood, the half-edge can be updated as $h \leftarrow h \rightarrow \text{opposite} \rightarrow \text{next}$.

Traversing a face is even simpler than a vertex neighborhood traversal. Starting from the face’s half-edge, h , a face can be traversed by following the *next* pointers until the h is reached again. Figure C.4 illustrates the face traversal process.

C.2.3 Semi-Regular Mesh Tagging

Before a mesh can be decomposed, it must be partitioned into sets of even and odd vertices. The reasons are twofold. First, if the partitioning process fails, then the mesh must not have the proper semi-regular connectivity. Second, if the partitioning succeeds, then the MR system must know which vertices to replace with coarse approximations, and which to replace with details.

This process is called *mesh tagging*, and is a process applied before decomposition.

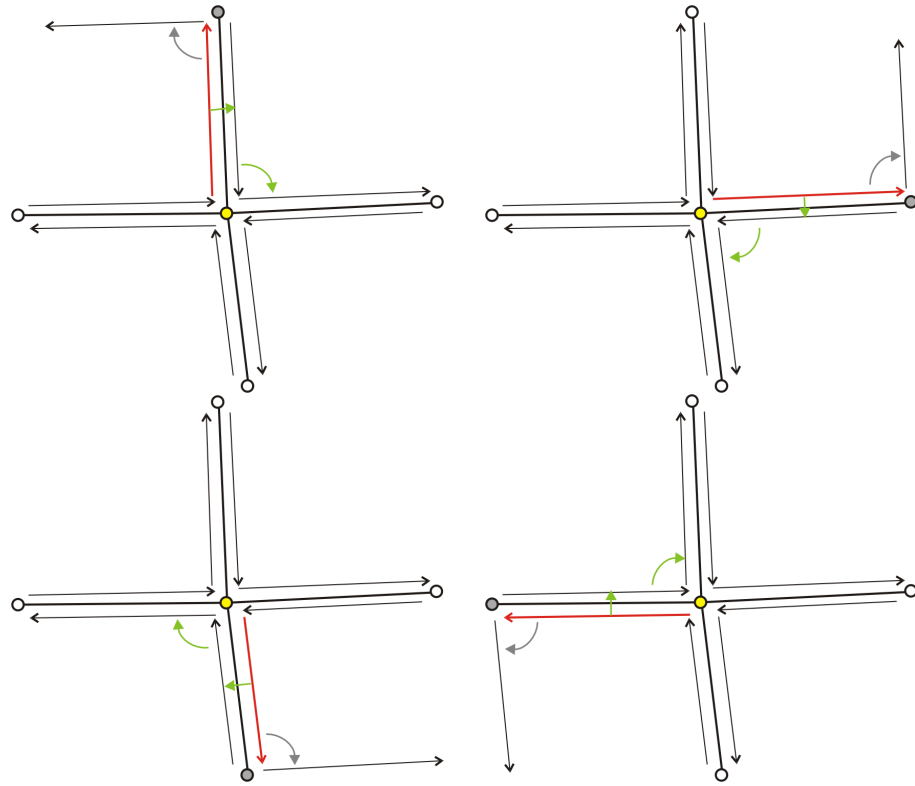


Figure C.3: Vertex traversal in a half-edge structure: starting from any half-edge h rooted at v , the neighbor vertex is given by $h \rightarrow \text{next} \rightarrow \text{vertex}$. To get to the next neighbor, h can be replaced by $h \rightarrow \text{opposite} \rightarrow \text{next}$.

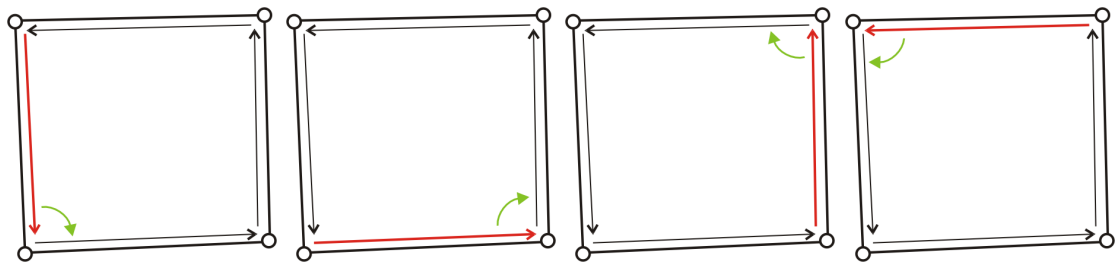


Figure C.4: A face can be traversed starting from h by following the next pointers until h is reached again.

Mesh tagging is essentially a graph traversal process that is initialized with a known vertex type. Then, the structure of the subdivision scheme in question determines what the neighborhood of the active vertex should consist of.

The necessity of mesh tagging underscores the need to represent an object as a set of independent meshes: if any vertex is unreachable from the initial vertex by a traversal procedure, then that vertex cannot be tagged and will not be known to the decomposition procedure.

If a mesh does not have subdivision connectivity, a remeshing process such as those discussed in Sec. 2.3.1 could be employed first.

Loop Meshes

In a Loop surface, there are two types of vertices: even vertices v correspond to vertices in the coarse mesh, and odd vertices e correspond to edges at the coarse level. By the face-splitting structure of Loop subdivision, every neighbor of an even vertex is an odd vertex. Every odd vertex has exactly six neighbors: the original endpoints of the edge, which are even, and two odd neighbors between them, forming a sequence of neighbors $v-e-e-v-e-e$.

To tag the vertices in a Loop mesh, the algorithm should be initialized with an even vertex. Fortunately such a vertex is easy to find except in pathological cases, because all odd vertices have a valence of six. Thus, any vertex with valence $n \neq 6$ must be even.

Figure C.5 illustrate the traversal algorithm. Starting at an even vertex (v), all neighbors can be tagged as odd (e). Then, in a breadth- or depth-first traversal, one of the tagged but unvisited vertices can be selected. If the selected vertex is

odd, then it must have been tagged by an even vertex, in which case the $v-e-e-v-e-e$ sequence is easily determined. If the selected vertex is even, then all neighbors can be tagged as odd. The algorithm proceeds until all vertices have been tagged.

Catmull-Clark Meshes

To decompose a mesh that has Catmull-Clark connectivity, we must be able to distinguish the v -vertices from the f - and e -vertices. Starting from a known v -vertex, we can use a breadth- or depth-first search to classify all remaining vertices by considering the vertex-edge structure of a mesh as a graph.

Starting from the known (or assumed) v -vertex v_{root} , all neighbor vertices can be tagged as e -vertices and added to the graph as children of v_{root} . Then, selecting the next active vertex v_{active} by depth or breadth, the algorithm proceeds as follows:

1. If v_{active} is an e -vertex:

Starting from the parent node – a v -vertex – tag the remaining neighbors as $f - v - f$ (if not previously tagged) and add the v -vertex neighbor as a child.

2. If v_{active} is a v -vertex:

Tag all untagged neighbors as e -vertices and add as children.

This process is illustrated in Figure C.6. In (a), a portion of a mesh is shown, along with the starting vertex v_{root} . From v_{root} , all immediate neighbors are tagged as e -vertices and added to the graph, as in (b); the process continues in (c), where an e -vertex is chosen as v_{active} , and its neighbors are tagged as $f - v - f$. After the process completes, we get a complete tagging of all vertices, as shown in (d).

The tagging process should be seeded with a v -vertex. In most cases it is reasonable to assume that vertex 0 is a v -vertex and spread from there. Supposing that this assumption does not hold – i.e. that we attempt to start from a non- v -vertex – we will either encounter an inconsistency or find an equivalent tagging.

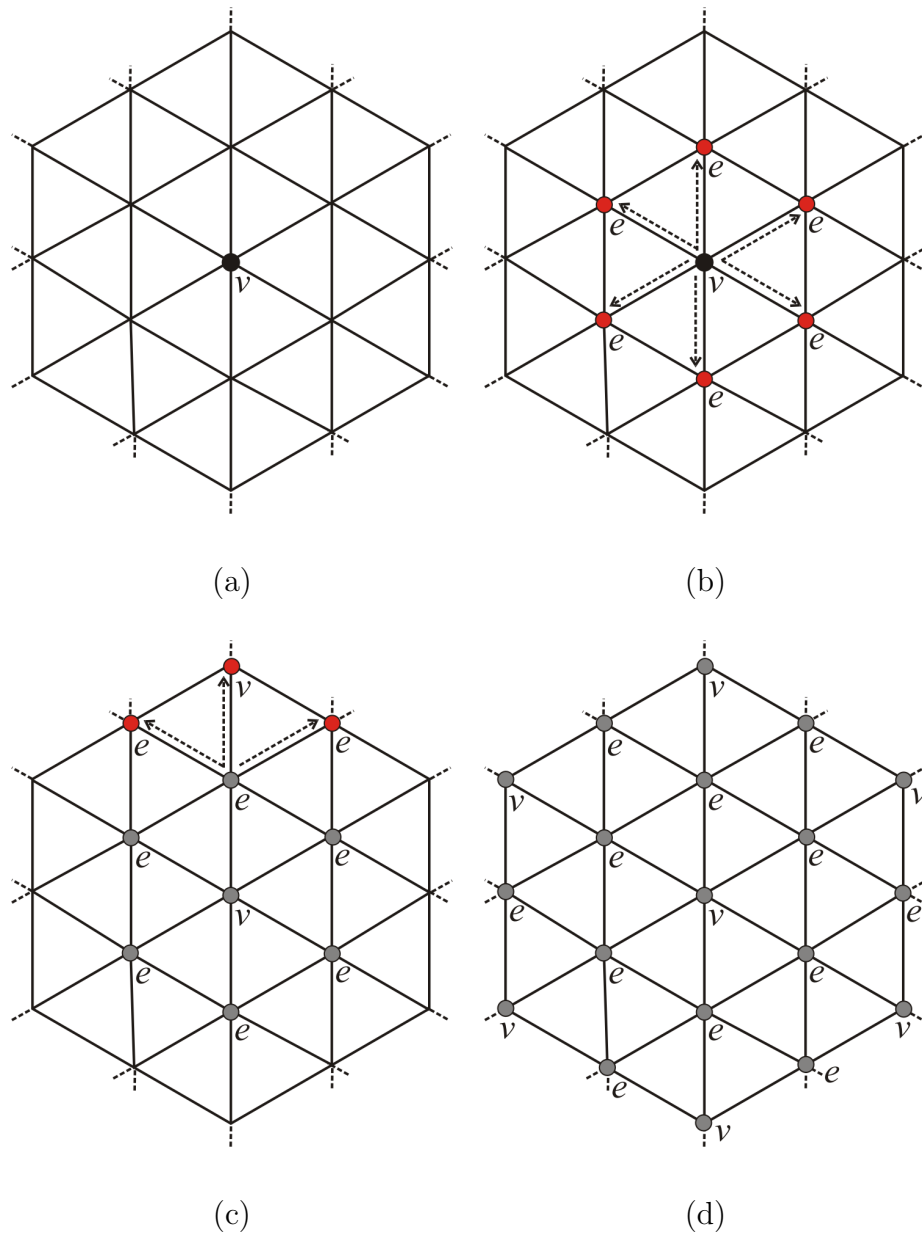


Figure C.5: Vertex tagging for Loop surfaces: (a) The algorithm is initialized with a known v -vertex (or a guess); (b) each neighbor of a v -vertex is an edge vertex; (c) each edge vertex has six neighbors, in the sequence v - e - e - v - e - e ; (d) a correct tagging is reached after visiting all vertices.

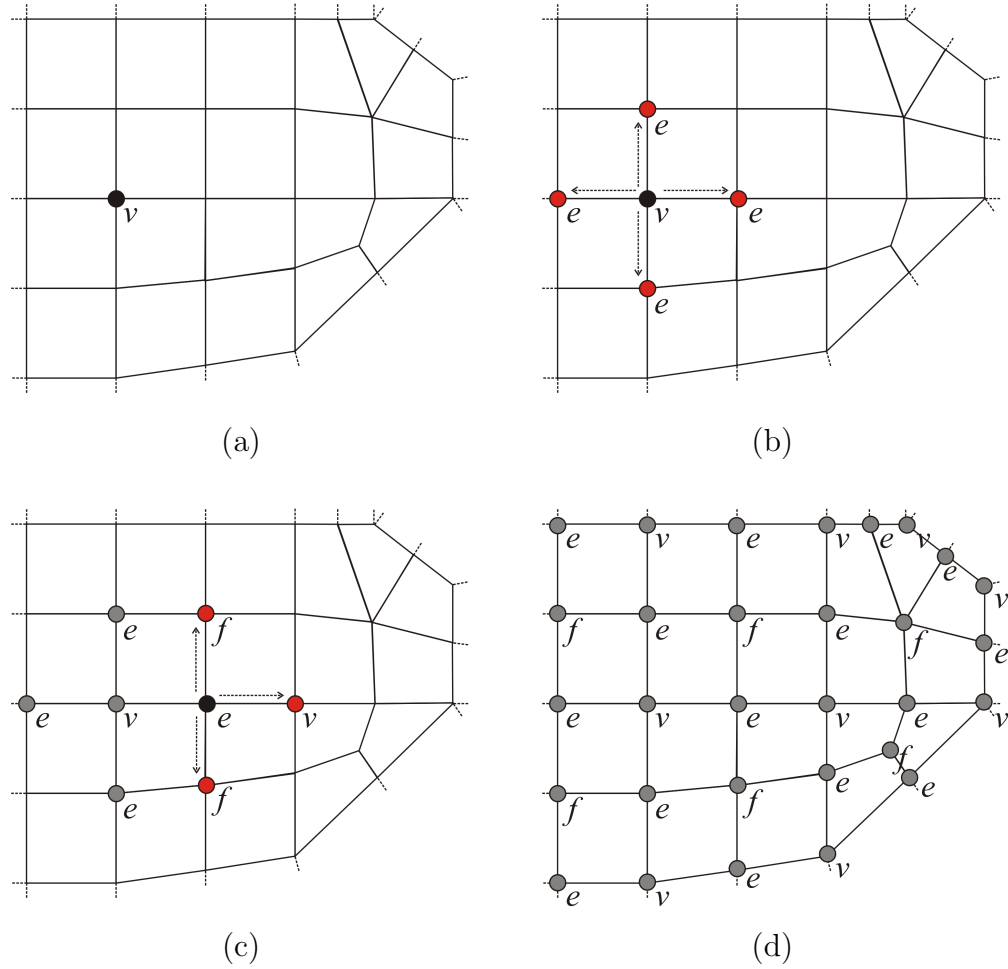


Figure C.6: Vertex tagging precedes decomposition: (a) The algorithm is initialized with a known v -vertex (or a guess); (b) each neighbor of a v -vertex is an edge vertex; (c) each edge vertex has four neighbors, in the sequence v - f - v - f ; (d) after visiting all vertices, we have a correct tagging.