Technical Report

UCAM-CL-TR-490
ISSN 1476-2986

Number 490

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Selective mesh refinement for rendering

Peter John Cameron Brown

April 2000

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted February 1998 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Emmanuel College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986
DOI *https://doi.org/10.48456/tr-490*

Abstract

A key task in computer graphics is the rendering of complex models. As a result, there exist a large number of schemes for improving the speed of the rendering process, many of which involve displaying only a simplified version of a model. When such a simplification is generated selectively, i.e. detail is only removed in specific regions of a model, we term this *selective mesh refinement.*

Selective mesh refinement can potentially produce a model approximation which can be displayed at greatly reduced cost while remaining perceptually equivalent to a rendering of the original. For this reason, the field of selective mesh refinement has been the subject of dramatically increased interest recently. The resulting selective refinement methods, though, are restricted in both the types of model which they can handle and the form of output meshes which they can generate.

Our primary thesis is that a selectively refined mesh can be produced by combining fragments of approximations to a model without regard to the underlying approximation method. Thus we can utilise existing approximation techniques to produce selectively refined meshes in n-dimensions. This means that the capabilities and characteristics of standard approximation methods can be retained in our selectively refined models.

We also show that a selectively refined approximation produced in this manner can be smoothly geometrically morphed into another selective refinement in order to satisfy modified refinement criteria. This geometric morphing is necessary to ensure that detail can be added and removed from models which are selectively refined with respect to their impact on the current view frustum. For example, if a model is selectively refined in this manner and the viewer approaches the model then more detail may have to be introduced to the displayed mesh in order to ensure that it satisfies the new refinement criteria. By geometrically morphing this introduction of detail we can ensure that the viewer is not distracted by "popping" artifacts.

We have developed a novel framework within which these proposals have been verified. This framework consists of a generalised resolution-based model representation, a means of specifying refinement criteria and algorithms which can perform the selective refinement and geometric morphing tasks. The framework has allowed us to demonstrate that these twin tasks can be performed both on the output of existing approximation techniques and with respect to a variety of refinement criteria.

Preface

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration.

I hereby declare that this dissertation is not substantially the same as any that I have submitted for a degree or diploma or other qualification at any other University. I further state that no part of this dissertation has already been or is being concurrently submitted for any such degree, diploma or other qualification.

This dissertation does not exceed sixty thousand words, including tables, footnotes and bibliography.

Publications

Sections of this work have been published previously [Bro96, Bro97a, Bro97b].

Trademarks

All trademarks contained in this dissertation are hereby acknowledged.

Acknowledgements

This work has been supported by a studentship from EPSRC together with contributions from Emmanuel College, Data Connection Ltd and Cambridge Philosophical Society. The machine on which the results were obtained was kindly donated by the Olivetti and Oracle Research Laboratory (ORL). The VRML Cessna and Easter Island statue models which are used in this dissertation are courtesy of Viewpoint Datalabs.

I would like to thank my three supervisors for their support and advice during my time in the Computer Laboratory. I am very grateful to Mike Gordon for giving me the initial support and freedom to select my own course of research. I was then welcomed into the Rainbow group by Neil Wiseman and, for an all too brief period, I experienced the wisdom of his gentle guidance before his untimely death. Peter Robinson kindly stepped into the breach and the majority of this work has been completed under his watchful eye.

My colleagues in the Rainbow group and Austin 4 have been excellent company for the past three years. The group has provided a stimulating working environment because of, and despite, all the "displacement activities" that PhD students can invent. There are too many people to mention personally but special thanks are due to Malcolm and Stefan for reviewing this dissertation and also to Margaret for never being too busy running the department to answer my questions.

The other Emmanuel residents of 43 Tenison Road during my third year also contributed to the enjoyable atmosphere which assisted the completion of this work. Thanks especially to Mark and Jo who put up with me for the entire three years.

Glossary

Area Of Interest, AOI : an area of a model which is required to be displayed at a higher resolution than its surrounding region.

Continuous Resolution Model Representation, CRMR : the form in which we represent a computer graphics model prior to producing a selectively refined version.

CRMR Hypermesh : the component of a CRMR which stores fragments of approximations to the original model.

CRMR DAG : the Directed Acyclic Graph component of a CRMR which represents the overlapping nature of the fragments contained in the associated Hypermesh.

Geometric morphing, Geomorphing : the process of smoothly interpolating between two different representations of a model.

Level Of Detail, LOD : a single-resolution approximation to a model.

MultiTriangulation, MT : Puppo and De Floriani's structure which can represent models in a resolution-based manner [Pup96, DPM96].

Progressive Mesh, PM : Hoppe's resolution-based model representation and associated approximation method [Hop96, Hop97b].

Refinement fragment : the term which we use to denote a fragment of an approximation to a model. Specifically, a refinement fragment can be used to replace a particular region of a lower resolution approximation to produce a higher resolution approximation.

Refinement operation : the process of refining a model approximation by inserting a refinement fragment.

Resolution Control Function, RCF : our composition of the criteria specifying the resolution which is required in a selectively refined version of a model.

Selective Mesh Refinement, SMR : the process of producing a selectively refined version of a model.

Virtual Reality Modelling Language, VRML : a 3D file interchange format used to provide 3D environments on the World Wide Web.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Motivation . . . 6
- 1.3 Thesis . . . 7
- 1.4 The Selective Mesh Refinement framework . . . 7
- 1.5 Contribution . . . 8
- 1.6 Structure of the dissertation . . . 9

2 Mesh Simplification **11**
- 2.1 Preliminaries . . . 11
  - 2.1.1 Surface definitions . . . 11
  - 2.1.2 Simplicial meshes . . . 13
- 2.2 Height field approximation . . . 14
  - 2.2.1 Regular grids . . . 16
  - 2.2.2 TINs . . . 20
  - 2.2.3 Feature-based . . . 26
- 2.3 Manifold surface approximation . . . 28
  - 2.3.1 Refinement . . . 29
  - 2.3.2 Decimation . . . 29
  - 2.3.3 Re-tiling . . . 31
  - 2.3.4 Energy optimisation . . . 33
  - 2.3.5 Wavelets . . . 34
- 2.4 Non-manifold approximation . . . 35
  - 2.4.1 Vertex clustering . . . 36
  - 2.4.2 Edge collapsing . . . 37
  - 2.4.3 Progressive Simplicial Complexes . . . 37
- 2.5 Summary . . . 38

3 Variable Resolution Schemes **39**
- 3.1 Other multiresolution methods . . . 39
  - 3.1.1 Multiple LODs . . . 40
  - 3.1.2 Selective refinement by remeshing . . . 42
- 3.2 Constrained approximation . . . 44

3.2.1 Variable resolution height fields 45
3.2.2 Progressive Meshes and extensions 50
3.3 HyperTriangulation 53
3.4 Related work 54
3.5 Review of approximation schemes 55
3.6 Resolution criteria 56
3.7 Summary 61

4 The SMR Framework **63**
4.1 SMR framework features 63
4.2 The SMR context 66
4.3 Terminology 67
4.3.1 Dimensions 67
4.3.2 Simplicial mesh notation 68
4.3.3 Edge notation 69
4.4 Continuous Resolution Model Representation 69
4.4.1 Refinement operations 72
4.4.2 Resolution attributes 74
4.4.3 CRMR construction 75
4.5 CRMR structures 78
4.5.1 CRMR Hypermesh component 79
4.5.2 CRMR DAG component 83
4.6 Resolution Control Function 85
4.6.1 RCF components 85
4.6.2 Object-space resolution criterion 87
4.6.3 Screen-space resolution criterion 91
4.7 Summary 92

5 Selective Mesh Refinement **93**
5.1 Minimal surface SMR 93
5.1.1 Refinement operation extraction 94
5.1.2 Surface expansion 95
5.1.3 Algorithmic complexity 104
5.1.4 Proofs of expanded mesh properties 105
5.2 Reduced extraction SMR 109
5.2.1 Algorithm 110
5.2.2 Proofs of expanded mesh properties 114
5.3 Summary 119

6 Geomorphing **121**
6.1 Introduction 121
6.2 Requirements 124
6.3 Algorithm 126

6.4 Proofs of geomorphed mesh properties 130
6.5 Summary 132

7 Results **133**
7.1 CRMR generation 133
7.1.1 Delaunay CRMR generation 133
7.1.2 Progressive Mesh CRMR generation 136
7.1.3 CRMR space requirements 138
7.2 Selective Mesh Refinement results 139
7.2.1 SMR of Delaunay CRMRs 139
7.2.2 SMR of Progressive Mesh CRMRs 150
7.3 Geomorphing results 157
7.4 Comparison of results 160
7.5 Summary 162

8 Conclusions **163**
8.1 Specific achievements 164
8.2 Future work 167

Chapter 1

Introduction

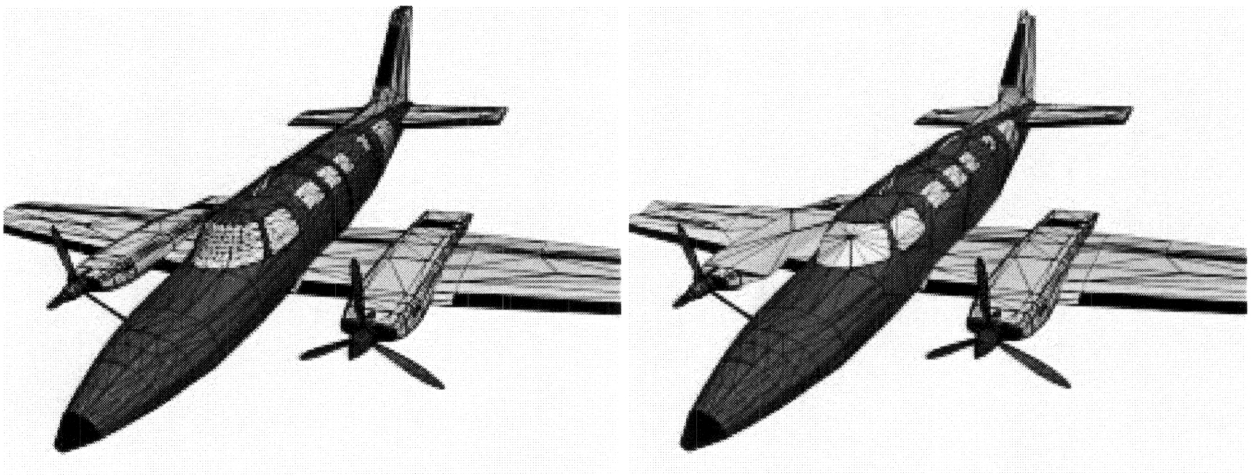
> *Think of the screen as a window into a virtual world. The task of computer graphics research is to make the picture in the window look real, sound real, interact real, feel real. [Sut65]*

The "task" to which Ivan Sutherland referred in 1965 seems no closer to completion now than at the time of his visionary statement – improving the quality of such "virtual worlds" is an open-ended research problem. This dissertation tackles the problem by proposing a framework which provides a flexible approach to reducing the geometric detail contained in graphical models selectively. In addition, the framework offers a mechanism for interpolating between such selectively refined models. We demonstrate that the resulting reduction in polygonal complexity can permit faster rendering and animation without significantly affecting the viewer's perception of a scene. An example of a selectively refined model is given in Figure 1.1.

1.1 Background

One way to reduce the cost of rendering a model is to display only an approximation of that model. There is a large (and rapidly expanding) set of schemes for producing such model approximations. Each of these approximation schemes can handle a particular class of model and produces approximations which have certain characteristics, e.g. triangular or polygonal, topologically equivalent or modified, etc. These approximation techniques are generally computationally expensive and, if they are to improve the speed with which a model can be manipulated, they must be performed as an off-line pre-processing step.

If only a single approximation of a model is prepared and stored as a representation of that model then this approximation is likely to reduce any storage and transmission costs relative to the original. This single approximation, though, can improve the rendering ability of an application only



(a) Viewpoint Datalabs' VRML Cessna model.

(b) Selectively refined model with reduced resolution apparent on the cockpit windscreen, starboard wing and tail.

Figure 1.1: Example of a selectively refined model.

when the distance from the viewer to the model is within a certain range. Figure 1.2 presents an example of a single approximation to a model, specifically a triangulation which represents the terrain surface of Mt St Helens, Washington. This approximation cannot be distinguished from the original data when its distance from the viewer is in the illustrated range. When the viewer is closer to this approximation, though, the fact that data has been removed is apparent and it would have been preferable to display a higher resolution representation; on the other hand, when the approximation is further away, its resolution could have been decreased without reducing the quality of the displayed image.

It is possible to realise the benefits of using an approximated model and also to render an image which is perceptually equivalent to the original model at any viewing distance. The simplest approach to this objective is to prepare and store more than one approximation of a model. This is the form of detail management which is invoked by both Open Inventor and VRML [Wer94, CB97]. A model can be stored as a set of approximations of which one is selected according to a simple criterion which determines the approximation which is appropriate for the requirements of the current scene. Each approximation is generally termed a *Level Of Detail (LOD)*. Figure 1.3 illustrates the selection of different representations according to their distance from the viewer.

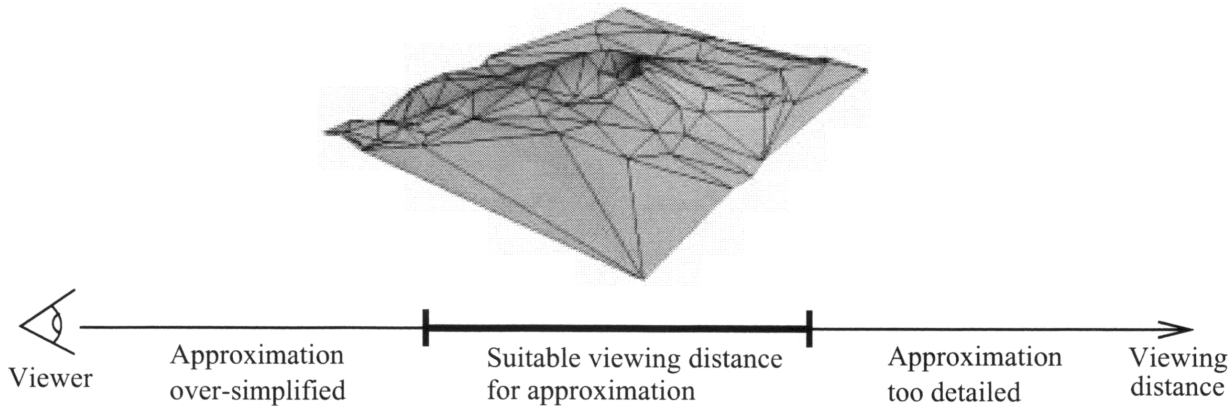


Figure 1.2: One approximation of Mt St Helens and the range of viewing distances for which it may be a suitable representation.

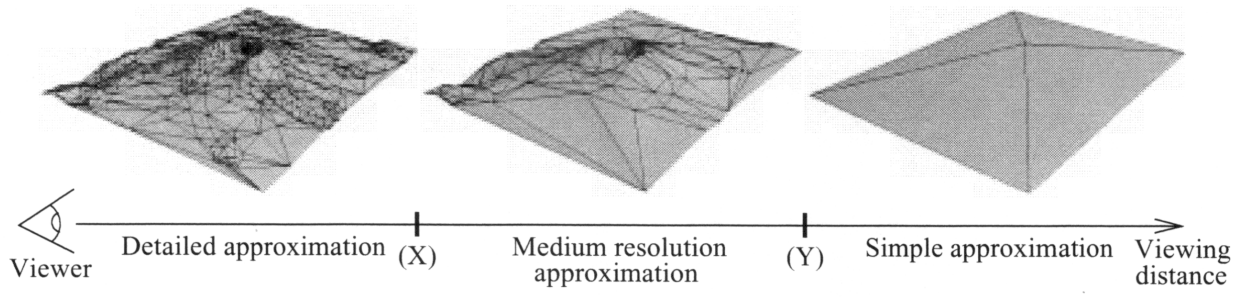


Figure 1.3: A set of three approximations of Mt St Helens from which one would be selected when the distance to the viewer lay in the ranges indicated. LOD switching would occur at points (X) and (Y).

There are a number of drawbacks to this *multiple LOD* approach to using model approximations:

- the individual LODs must be pre-calculated and, while there are now tools available to assist with this process (such as SGI's Cosmo Worlds[1], InnovMetric's IMCompress[2] and HP's DirectModel[3]), there is usually a degree of manual input required. Puppo [PS97] examines these tools in detail;

- as the viewer moves through a scene, the resulting changes in LODs will be apparent to the viewer as "popping" between different model approximations. There is no requirement for coherency between the LODs and hence no way in which one LOD can be smoothly transformed into another;

- there can be no selective refinement within the surface of a large object and so, even if only a small portion of a large object is being viewed in one frame, a high-resolution approximation of the entire object may be required;

- the time which is required to retrieve and render a particular LOD is not taken into consideration by the LOD selection criterion. Thus the user may have to wait for a high-resolution representation to be retrieved when a coarser LOD would have been sufficient;

- even if one LOD of an object has already been retrieved, the information contained in this representation is not used when another LOD of the same object is required later.

An extension of this multiple LOD approach was suggested in 1976 by Clark [Cla76] and this was the first paper to advocate a *multiresolution* approach to modelling. Clark suggested that an object can be represented by a hierarchy of components. The root of this hierarchy is a crude approximation of the object and every other node contains an improved approximation of a component which is represented in its parent node. A selection of these nodes can be used to represent the object for a particular scene such that the amount of detail in each object component is proportional to its projected area in screen-space. Also, by checking whether the bounding box of each node is on-screen, visibility culling can be performed. Thus the number of polygons which has to be rendered in each frame can be reduced significantly.

[1] http://cosmo.sgi.com/products/studio/worlds/datasheet.html
[2] http://www.innovmetric.com/anglais/page_ed.html
[3] http://hpcc920.external.hp.com/wsg/products/grfx/dmodel

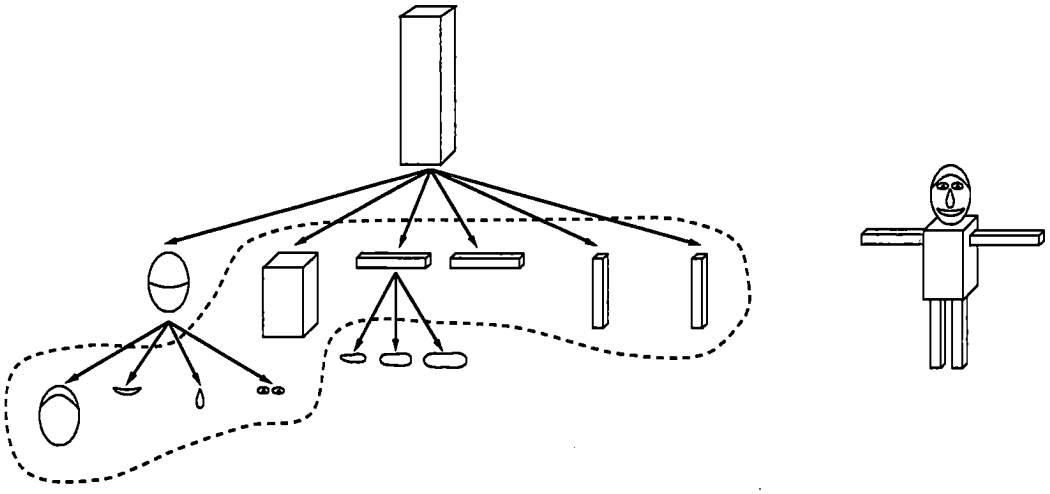


Figure 1.4: A hierarchy of object components representing a simple human body. The head and an arm are refined to two levels. The body on the right has been constructed from the set of components contained in the dashed outline.

The example which Clark gave, of a hierarchy representing a human body, is represented diagrammatically in Figure 1.4. If a model of a body is sufficiently far away from the viewer then it may cover only a few pixels when rendered and so the top level (coarsest) approximation in the hierarchy can instead be rendered at a much reduced cost and with minimal impact on the viewer's perception of the scene. If the viewer moves closer to the body then this approximation may no longer be sufficiently representative of the body and so the next level of the hierarchy may be used as a set of approximations to the body's components, e.g. its arms, legs, head and torso. This form of recursive descent can be used to determine the component approximations which are the minimum necessary to render a particular scene.

Sewell detailed a number of techniques to handle the tasks associated with this form of model hierarchy, such as the identification of model components [Sew96]. He also demonstrated that this approach reduces the granularity of the problems associated with merely selecting between complete model LODs. The remaining disadvantages are that a large surface such as a terrain cannot easily be resolved into separate components and also "popping" artifacts still arise when component LODs are switched.

As a means of resolving the first issue, papers in the area of terrain modelling (including [DP95, CPS95, dD95]) described how approximation techniques can be invoked locally such that a representative surface is produced which has a varying degree of detail across its surface. This is particularly

relevant to terrain scenes because the disparity between distances in object-space from the viewpoint to the closest and furthest points of the rendered surface is often large. Hence this *selective refinement* can usefully be applied to increase the resolution of a rendered surface close to the viewer while reducing the resolution of the surface in more distant regions. In this way, the polygon count of a displayed terrain can be reduced without affecting the perceived quality of the scene.

The task of selectively refining terrain surfaces has been the subject of dramatically increased interest recently but this selective refinement approach to scene optimisation can also usefully be applied to rendering more general forms of three-dimensional models. Given the rendering requirements of a particular scene, we could reduce the rendered polygon count by completely replacing those objects, or groups of objects, which project onto a small area of the screen (as Sewell advocated [Sew96]) and, in addition, selectively refine the more significant objects before rendering. The surfaces of these latter objects could be selectively refined according to criteria such as their proximity to the viewer, the prominence of their features, and also the resolution required in certain areas of the screen.

Authors such as Hoppe [Hop97b] and Xia and Varshney [XV96] have recently presented methods which can perform this selective refinement of three-dimensional models. One advantage of Hoppe's method is that geometric morphing (or *geomorphing*) can be performed between selectively refined meshes and hence this can resolve the final problem associated with multiple LOD selection – the "popping" artifacts.

1.2 Motivation

The previous section has highlighted how approximations of computer graphics models can be generated and used in rendering. The most recent of the available approximation methods can produce an approximation whose quality of representation can vary within its extent, i.e. they can produce a selectively refined approximation. This implies that we can specify the resolution requirements of a particular scene and then generate approximations to models in the scene which closely match these requirements. Thus we can produce a representation of a scene which is the most efficient in terms of the number of rendered polygons.

The polygonal reductions which can be achieved by this approach to rendering have been demonstrated by Xia, Varshney and Hoppe [XV96, Hop97b]. The selective refinement methods which these authors proposed are limited by being inextricably linked to a particular method of approximating a model. Thus their selectively refined representations reflect the characteristics of the underlying approximation method and these character-

istics are not necessarily suitable for every application.

This constraint on the production of selectively refined representations seems unnecessary when we consider Cignoni's approach to generating selectively refined terrain surfaces [CPS95]. This author demonstrated how restricted forms of selectively refined surfaces can be produced by combining fragments of single-resolution approximations to a terrain surface.

We expand the scope of this fragment-based approach beyond its limited domain of a specific approximation method operating on terrain surfaces and producing restricted selectively refined versions. We wish to produce unconstrained selectively refined representations of n-dimensional models as the result of applying any of a wide range of approximation methods. This would permit an existing approximation method to be applied to a model when the characteristics of that method were appropriate to either the model or the intended application. The resulting selectively refined version could then maintain those characteristics.

The fragment-based nature of this potential selective refinement process also appears to lend itself to geomorphing between selectively refined meshes without using the specific approximation technique of Hoppe [Hop97b]. If we can localise the necessary geometric morphing to the regions of these approximation fragments, it should also be possible to decouple the geomorphing process from the underlying approximation method. Thus we could ensure that if, for example, the resolution criteria of a scene changed during an animation of a selectively refined representation, the selective refinement could be adapted to meet the new resolution criteria without any "popping" artifacts.

1.3 Thesis

Our primary thesis is that an n-dimensional selectively refined mesh can be constructed from a series of single resolution Levels Of Detail without regard to the approximating process which constructed these LODs. Further, that geomorphing can also be performed independently of the original approximating process. As a result, a wide range of existing single-resolution approximation methods can be combined with the twin processes of selective refinement and geomorphing.

1.4 The Selective Mesh Refinement framework

In the process of exploring our thesis, we have developed a framework for performing selective refinement and geomorphing. This framework fulfills our objective of permitting these processes to be performed on the output

of existing single-resolution approximation methods. The key concept of our *Selective Mesh Refinement (SMR)* framework is that it separates the process of producing a resolution-based representation of a model from the associated tasks of:

1. extracting a selectively refined mesh which represents the model, and;

2. geomorphing between selectively refined meshes.

To facilitate this separation, we have developed a novel form of representing computer graphics models. This stores the information which can be generated by applying an existing approximation method to a model at a range of resolutions. We can extract from this information the portions of pre-generated approximations which can be combined into a complete representation of the original model. This selectively refined version of the model can be guaranteed to satisfy the current resolution criteria.

In addition, if the resolution criteria change, we can adapt an existing selective refinement to meet these new requirements. This process of adaptation can be performed as a sequence of localised modifications which can be geometrically morphed. Hence we can geomorph between selectively refined models without regard to the approximation method from which they originated.

1.5 Contribution

This section outlines the contribution of this dissertation to the field of computer graphics.

We introduce a generalised, lossless, continuous resolution format for representing models. This representation extends the existing concept of storing a set of fragments of approximations to a terrain [CPS95] by permitting the fragments to be generated by a range of approximation methods which operate on n-dimensional models. Furthermore, we permit certain topological operations to be represented by the fragments which we store and this permits us to handle a wider range of both approximation methods and approximated models than previously.

This model representation format can be implemented using a novel extension of a standard computer graphics data structure. This standard basis permits existing algorithms for essential operations such as point location to be easily adapted for use on our model representations.

Representing models in this resolution-oriented fashion is necessary to provide the input to our selective refinement algorithms. We present two algorithms which can selectively refine n-dimensional models with respect to some resolution specification. A selectively refined version of a model can

be generated by combining a selection of the fragments of approximations to that model which have been stored in our resolution-based representation. Thus, uniquely, we can demonstrate that the approximating process which generated the original single-resolution model approximations is not restricted to a specific method, as existing selective refinement techniques demand, but can be drawn from a wide range of existing methods. Hence we can permit the use of standard approximation methods which have been developed to operate on particular classes of model, such as terrain surfaces, or which generate approximations with desirable characteristics, e.g. smooth meshes. Such characteristics are retained in our selectively refined models.

We can prove that the selective refinements which our algorithms generate are guaranteed to satisfy the resolution criteria which have been specified as essential for a particular scene. We uniquely combine resolution criteria which have been specified in terms of both the space in which the model has been specified and that in which it is to be displayed. Thus we can ensure that, for example, the geometric error across the displayed portion of a selectively refined terrain surface is within a prescribed tolerance, while detail in the region outside the current view frustum can be culled. Also, we suggest how complex resolution requirements for terrain surfaces can be specified and visualised. Of particular interest here is how we can specify the regions of a terrain which are critical for all viewpoints, such as the ridges and peaks of mountains.

We have already noted the requirement for smoothly modifying a selectively refined model when resolution requirements change. Thus the process of geomorphing is inherently linked to the process of selective refinement. We present a geomorphing algorithm which can operate on our selectively refined models and hence is novel in being independent of the underlying approximation method.

This dissertation also presents the first quantitative comparison of selectively refined models produced using alternative approximation methods and with respect to a range of resolution criteria.

The specific details of this dissertation's contribution are presented in depth in Section 4.1.

1.6 Structure of the dissertation

The remainder of this dissertation is arranged in the following order.

Chapters 2 and 3 consider related background material. Chapter 2 introduces terminology associated with the field of model approximation and surveys a wide range of approximation methods. Chapter 3 considers existing methods which can generate selectively refined models from the output of these approximation methods and also how the resolution criteria which

these models are intended to meet can be specified.

Our Selective Mesh Refinement framework is detailed in Chapters 4 through 6. The first of these introduces definitions associated with our framework, our resolution-based model representation and the means by which we specify resolution requirements. Chapters 5 and 6 present our selective refinement and geomorphing algorithms respectively.

The results of applying our algorithms to various models and resolution criteria are presented in Chapter 7.

Finally, our conclusions and suggestions for areas of future exploration are detailed in Chapter 8.

Chapter 2

Mesh Simplification

The range of application areas in which approximating techniques for geometric meshes are required is reflected in the wide variety of these techniques. Each of these techniques has unique characteristics which determine its applicability and usefulness with respect to the requirements of specific applications.

In this chapter we identify approximation techniques which are relevant to our Selective Mesh Refinement framework and the field in which it lies. This is not intended as an exhaustive survey of mesh approximation methods; no such survey exists, although Heckbert and Garland [HG95, Hec97] and Puppo and Scopigno [PS97] have examined similar subsets of the field.

The purpose of this chapter is to examine how these approximation methods can produce one or more meshes which represent a given model. The potential to link these meshes in some manner, and extract a selectively refined representation due to this structuring, is examined in the next chapter. By separating the approximation methods from any mesh linkage which they may introduce, we attempt to treat these existing methods in the manner in which they can be incorporated into our framework.

We first introduce some of the standard terminology which is used to describe the types of meshes which can be handled by the methods in this chapter. This is followed by brief descriptions and discussion of each of the approximation methods which we have identified as being relevant, categorised according to the classes of models which they can handle.

2.1 Preliminaries

2.1.1 Surface definitions

In this section we present standard formal definitions of the categories of surfaces which can be handled by the approximating methods discussed in

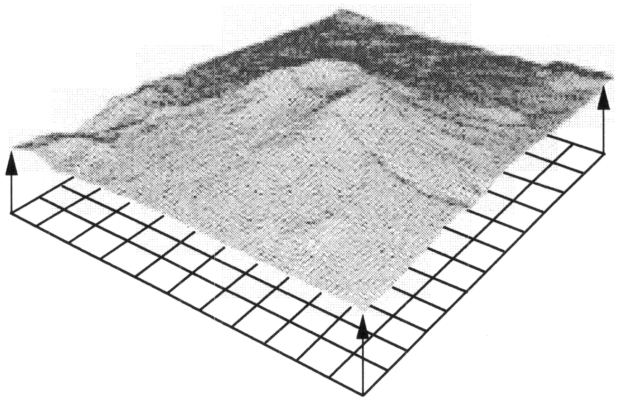


Figure 2.1: Mt St Helens datapoints relative to their planar domain.

this chapter [PS97].

2.1.1.1 Scalar fields

A *scalar field* is a continuous function $\Phi : \Omega \rightarrow \mathbb{R}$, where Ω is a connected domain in $\mathbb{R}^k$, $k \geq 1$. The image of Φ embedded in $\mathbb{R}^{k+1}$ space, i.e.

$$\mathcal{F} = \{(X, \Phi(X)) | X \in \Omega\} \subset \mathbb{R}^{k+1}$$

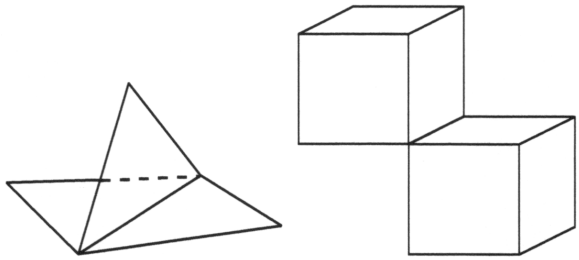
is called a *hypersurface.*

When $k = 2$, $\mathcal{F}$ is called a *height field.* A common example of a height field is a terrain surface, where the domain Ω is $\mathbb{R}^2$ and Φ is specified by elevation values at discrete sample points. Figure 2.1 shows the US Geological Survey's terrain data for Mt St Helens (subsampled by a factor of 4) displayed as a surface relative to a planar domain.

2.1.1.2 Manifold surfaces

A *manifold surface* $\mathcal{S}$ is a subset of Euclidean space $\mathbb{R}^k$, for some $k \geq 3$, such that the neighbourhood of each point of $\mathcal{S}$ is homeomorphic to the open disc in $\mathbb{R}^2$. A simple example of this kind of surface is the surface of a sphere or a closed polyhedron.

A *manifold surface with boundary* $\mathcal{S}$ is a subset of Euclidean space $\mathbb{R}^k$, for some $k \geq 2$, such that the neighbourhood of each point of $\mathcal{S}$ is homeomorphic to either the open disc, or to the half-disc (which is obtained by intersecting the open disc with the closed half-plane of the positive x coordinates). The *boundary* $\delta\mathcal{S}$ of such a surface is the set of all points of $\mathcal{S}$ which do not have a neighbourhood homeomorphic to the open disc. The classic example of a



(a) One edge adjacent to three facets.

(b) Two cubes joined along an edge.

Figure 2.2: Examples of non-manifold surfaces.

manifold surface with boundary is the Utah teapot, in which the boundary of the otherwise closed surface is the rim of the spout.

To clarify these definitions using counter-examples, two non-manifold surfaces are presented in Figure 2.2.

2.1.2 Simplicial meshes

All of the outputs of the approximation methods which this chapter discusses, apart from the wavelet-based methods, can be represented as *simplicial meshes*, i.e. meshes of triangles or tetrahedra. We introduce these meshes by following the definitions of Popović, Puppo and Moore [PH97, PS97, Moo92] which themselves are extensions of the standard topological definitions of Spanier [Spa66].

A *simplex* is the most elementary geometric figure of a given dimension (Figure 2.3). Formally, a *k-simplex* is a subset s of $\mathbb{R}^n$, $0 \leq k \leq n$, which is the locus of points that are convex combinations of the $k+1$ *vertices* of s. The *faces* of a simplex s are the non-empty subsets of s (Figure 2.4).

A finite set S of simplices in $\mathbb{R}^n$ is a *simplicial mesh* when the following conditions hold:

1. for each simplex $s \in S$, all the faces of s belong to S;

2. for each pair of simplices $s_0, s_1 \in S$, either $s_0 \cap s_1 = \emptyset$ or $s_0 \cap s_1$ is a simplex of S;

3. each simplex s is a face of some simplex s' (possibly coincident with s) having maximum dimension among all of the simplices of S.

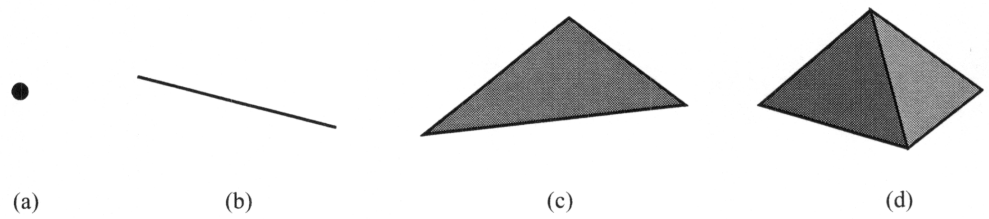


Figure 2.3: 0,1,2- and 3-dimensional simplices: point, line, triangle and tetrahedron.

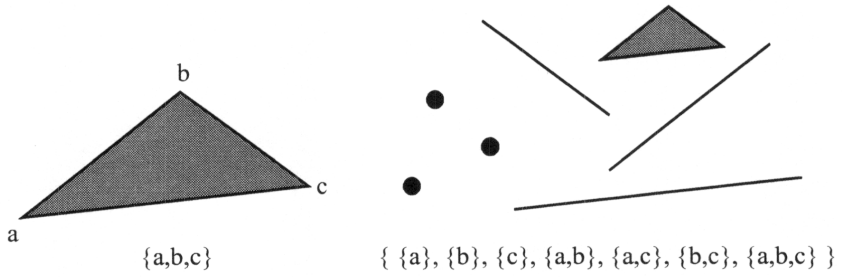


Figure 2.4: A 2-simplex (a triangle) and its faces.

A simplicial mesh S is called a *d-simplicial mesh* if d is the maximum among the dimensions of the simplices belonging to S. We refer to a d-simplicial mesh simply as a mesh, when no ambiguity will result. An example of a set of simplices which is not a simplicial mesh is given in Figure 2.5.

2.2 Height field approximation

The task of approximating height fields is a significant one, notably because the demand for real-time performance in flight simulators involves the use of approximated terrain surfaces. Indeed, this demand has led to a wide range of terrain approximators, of which the most significant are considered here. As we shall see in Sections 2.3 and 2.4, the key concepts of these approximation methods are reflected in a number of the methods which can handle manifold and non-manifold meshes.

Height field representation schemes can be classified as either *regular* or *irregular* according to whether they use datapoints which lie in a regular grid. The original flight simulators [Sch83, CMR90] used a grid-based system and this is also the case with most of their current descendants [Mue95]. The use of irregular representations is a relatively recent innovation and is almost

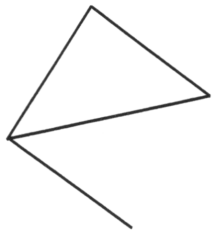
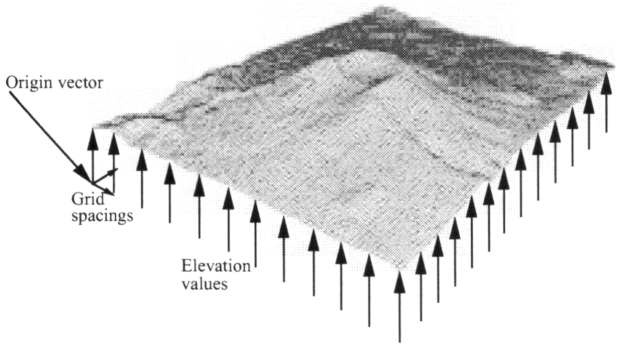
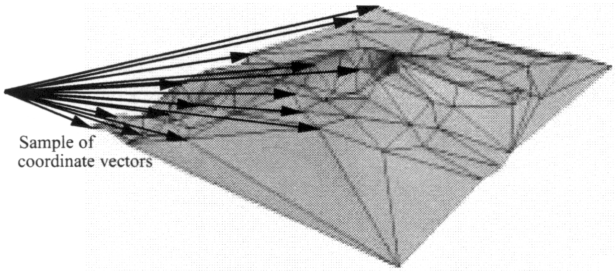


Figure 2.5: Two simplices of mixed dimensionality do not satisfy the third condition which is required of a simplicial mesh.



(a) Storage required for regular grid data. This is one of the standard forms in which the US Geological Survey distribute their data, known as the Digital Elevation Model format [EC84].



(b) A sample of the vector triples required to store an irregular terrain mesh.

Figure 2.6: Grid versus TIN storage.

entirely confined to triangle-based systems, whose underlying structure is termed a *Triangulated Irregular Network (TIN).*

Representating a terrain as a grid of elevation values has the advantage of simplicity – only the grid's origin, the grid spacing and the elevation values themselves must be stored (Figure 2.6a). In contrast, a TIN requires each datapoint to be stored as a coordinate triple (Figure 2.6b).

An initial examination of these alternative schemes may therefore suggest that the grid-based system is inherently more compact. The advantage of a TIN, though, is that it can be *adaptive* with respect to the detail in a surface. Thus highly-detailed areas such as mountains can be represented by a higher density of datapoints and areas of low detail, such as East Anglia, can be represented using proportionally less data (Figure 2.7). Whether

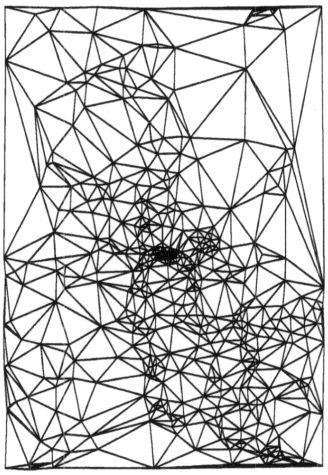


Figure 2.7: A TIN representing the Mt St Helens dataset with its adaptive nature apparent in the highly-detailed crater area in the centre of the dataset.

the adaptive nature of a TIN sufficiently counteracts its increased storage per data element is a moot point – see [Kum94] and [Hec97] for conflicting views.

We examine a set of grid-based terrain approximation schemes in Section 2.2.1 and then their TIN-based counterparts in Section 2.2.2. One other method which is of interest due to its consideration of the *critical lines* (ridges, channels, etc) in a terrain is mentioned in Section 2.2.3.

2.2.1 Regular grids

In this section we discuss how the inherent redundancy of grid-based systems has been partially eliminated in the implementations of Evans and Sutherland [CMR90] and the Naval Postgraduate School's NPSNET system [FZPM93]. We also examine a more recent grid-based method [LKR+96] which can guarantee the screen-space accuracy of a displayed terrain and a wavelet-based technique [GGS95] which provides local control over the output surface detail.

The simplicity and compact nature of grid-based representation were the factors which motivated its use in Evans and Sutherland's commercial flight simulator systems after their earlier experimentation with TIN representations [Sch83, Mue95]. Cosman [CMR90] described an E&S visual system whose database contains regular grids of elevation values. This data is stored as layers of Level Of Detail grids, where each LOD is subsampled to a quarter

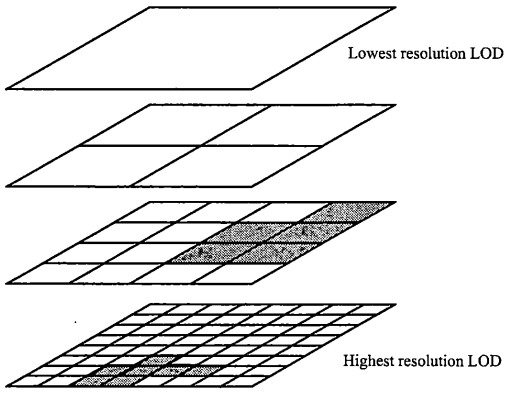


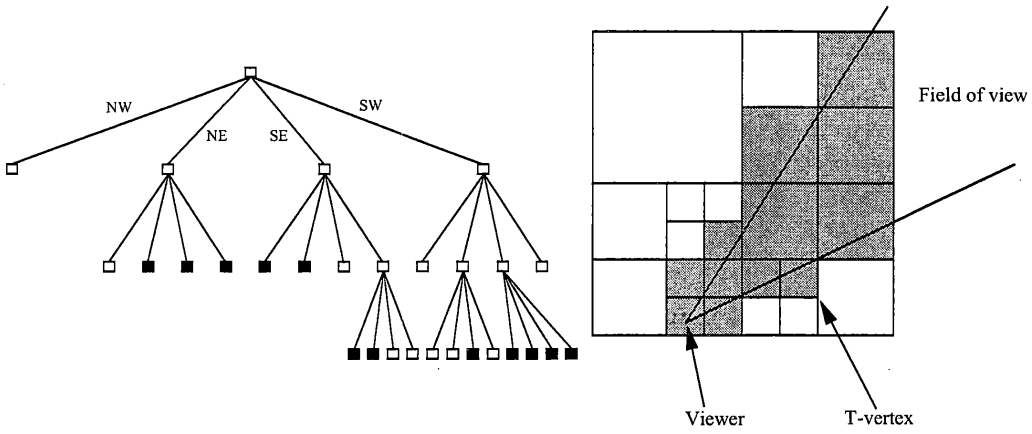
Figure 2.8: Cosman's layers of regular grid LODs.

of the number of points in its parent LOD, as in Figure 2.8.

Cosman did not describe the linkage between these LODs explicitly but their hierarchical nature is similar to the quadtree-based [Sam84] system of Falby *et al*'s NPSNET visual simulation environment [FZPM93]. At the highest level of detail, polygons which are the cells between the finest resolution grid data points are held in the leaves of a quadtree. These are then combined in fours to produce successively lower resolution representations until a 1km square of terrain is represented by the node at the root of a quadtree. These 1km squares are paged in and out of memory as required by the viewer's movement within a scene.

Figure 2.9a contains a subset of the quadtree which could be used to link the terrain polygons of Figure 2.8. Specifically, this quadtree subset could generate the plan view of Figure 2.9b. In this view, we can see that the depth of the nodes used to generate the surface is proportional to the nodes' significance in the scene, i.e a *view-dependent* approximation has been produced. As the plan view illustrates, this has resulted in higher resolution nodes being used in the area around the viewer. The nodes which are visible in the plan view have been shaded in Figures 2.8 and 2.9.

Although such grid-based LODs can be generated and stored in a hierarchy relatively easily, a further polygonalisation step is required to combine multiple layers of these LODs into a seamless surface. The plan view of Figure 2.9 illustrates the *T-vertex* problem which results when polygons from LODs in different levels of the hierarchy are adjacent in an extracted surface. These T-vertices would be visible as gaps if this surface was displayed. DeHaemer and Zyda [DZ91] described a number of ways in which a surface extraction algorithm could eliminate T-vertices from an approximating surface; the standard approach is to triangulate the lower-resolution polygon adjacent to a T-vertex (Figure 2.10). The 16 possible triangulations which



(a) Subset of quadtree used to link layers of regular grid LODs.

(b) Plan view of terrain nodes extracted from quadtree according to the field of view illustrated.

Figure 2.9: NPSNET hierarchy and plan view.

may be required between mis-matched LOD nodes which vary by only 1 level in a hierarchy were illustrated by Gross *et al* [GGS95].

Another disadvantage of using grid-based systems was highlighted by Scarlatos [Sca90]. She demonstrated that the use of subsampling to produce LODs in a grid-based system can cause the representations of significant features to appear to move between Levels Of Detail and hence concluded that transitions between these LODs could not be smooth.

This conclusion was invalidated by Lindstrom *et al*'s "continuous level-of-detail" grid-based system [LKR+96] which guarantees an upper bound on the screen-space error resulting from the display of an approximated portion of terrain. In addition, this method avoids the T-vertex problem associated with grid-based LODs by representing the surface as a hierarchy of right triangles. This use of right triangles, combined with a restriction that triangles are combined pair-wise at each level in the hierarchy, results in a simple dependency graph (Figure 2.11). The decision as to whether a particular triangle is extracted for the current scene is made using an approximation of the screen-space error which that triangle would introduce. The compact nature of the triangle dependency graph and the efficiency with which this screen-space error measure can be implemented was demonstrated by the real-time performance which Lindstrom presented.

The final grid-based system which we shall briefly examine is Gross *et al*'s wavelet-based technique [GGS95, GSG96]. This method uses a wavelet

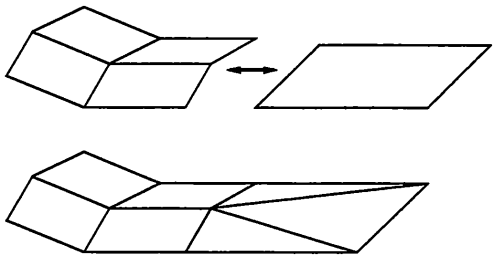


Figure 2.10: Two LOD nodes from a grid-based hierarchy are joined by a simple split of the lower-resolution LOD.

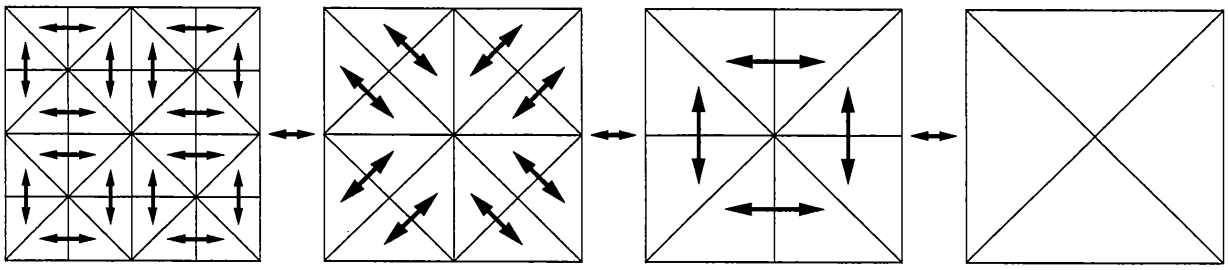


Figure 2.11: The basis of Lindstrom's grid-based hierarchy: pairs of right triangles which can be combined to form other right triangles.

transform to identify the significant areas of a terrain and then applies the inverse wavelet transform to identify the nodes of a quadtree which are required for a particular scene. Although this kind of wavelet-based technique lies outside the scope of our SMR framework, the significant aspect of this technique is how the output surface's resolution can be controlled. An Area Of Interest (AOI) can be specified by a Gaussian weighting function applied to the coefficients of the wavelet transform. We shall see later that the method by which AOIs are specified in our framework can be viewed as an adaptation of this method.

2.2.2 TINs

Schemes to approximate a height field using a Triangulated Irregular Network can be classified according to whether or not the topology of their TINs is determined by elevation information. Triangulations which take this into account are called *data dependent.* In this section we review typical examples of both data dependent and data independent triangulation methods. A feature common to all of these methods is that they can be viewed as making repeated local modifications to an approximating surface. We shall see later that this attribute makes them suitable candidates for inclusion in our SMR framework.

2.2.2.1 Data independent triangulations

The most common data independent triangulation method, i.e. a method which only considers the datapoints' (x, y) domain positions, is the *Delaunay criterion.* A Delaunay triangulation $\mathcal{T}$ is such that for each triangle t in $\mathcal{T}$, there is no vertex of $\mathcal{T}$ in the interior of t's circumcircle [Law77, GS77, PS85]. As Figure 2.12 illustrates, this triangulation criterion ensures that triangles have good aspect ratios and hence *sliver* triangles are avoided. Sliver triangles are undesirable for applications such as Finite Element Modelling and graphical rendering since their shape can cause numerical inaccuracies in FEM calculations and visual discontinuities in smoothly shaded surfaces.

Fowler and Little [FL79] are often cited as the originators of Delaunay approximating TINs. These authors construct a TIN in two stages. First, the structure of a terrain is characterised as a graph in terms of its significant points by passing a 2×2 filter over the data (see Section 2.2.3 for details). Secondly, a set of points with the highest approximation error is added to this skeleton so that the model meets a specified error tolerance. The retriangulation during this stage is performed with respect to the Delaunay criterion.

The second step in Fowler and Little's method – *greedy point insertion* [Hec97] – is a common feature of many triangulation schemes. Points are

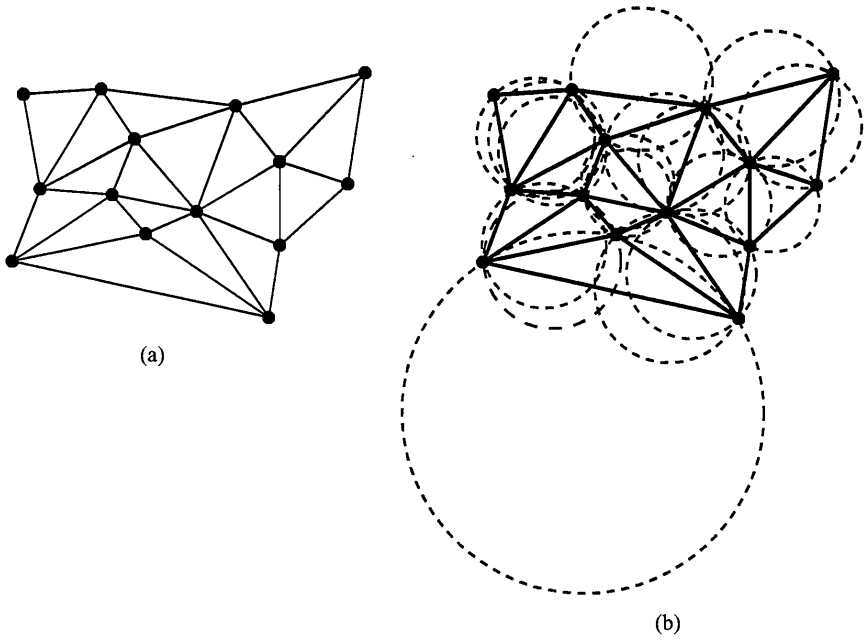


Figure 2.12: A Delaunay triangulation (a) overlaid with the circumcircles of its triangles (b) to demonstrate that none encloses another vertex of the triangulation.

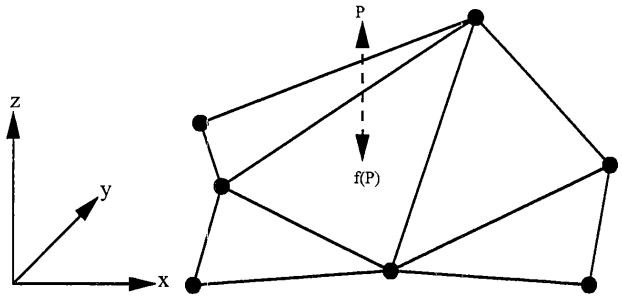


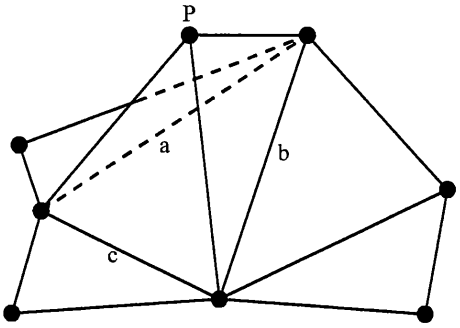
Figure 2.13: The Delaunay selector criterion. The next candidate point for insertion is the one which is furthest removed, vertically, from the current approximating surface. This error term, at a point P, can be written as $e(P) = |z - f(P)|$ where $P = (x, y, z)$ and $f(P)$ is the surface elevation at P computed as a linear interpolation of the z values at the vertices of the triangle whose projection encloses the projection of P.

inserted, either singly or in parallel, to an existing mesh in order to improve the quality of its approximation to a surface. The insertion step is termed "greedy" because the decision as to which points to insert is irrevocable. The points inserted are typically the ones which are at the maximum absolute distance from the current approximating surface (Figure 2.13). When this criterion is used to insert points in a Delaunay triangulation, it is known as the *Delaunay selector* criterion [DFP85].

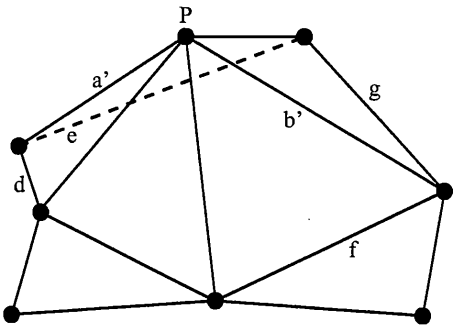
The retriangulation step which is invoked by Fowler and Little's method requires the Delaunay quality of a triangulation to be maintained during sequential point insertions. When a point is inserted in a triangle, the original edges of that triangle may have to be swapped if the newly-created subtriangles do not satisfy the Delaunay criterion (Figure 2.14a). In turn, this may force other edges to be swapped (Figures 2.14b,c). The effect of a point insertion is localised, though, to a *star polygon* of triangles which are coincident with the inserted point (Figure 2.14d).

De Floriani *et al* [DFP83, DFP85] demonstrated that this Delaunay retriangulation step can be used in isolation, i.e. without Fowler and Little's initial feature detection, to approximate a height field. The base mesh to which De Floriani's method sequentially adds points according to the Delaunay selector criterion is simply a planar TIN covering the boundary of the input dataset's domain. Points are added until the model matches the original data to within a certain tolerance. Optimised algorithms for this scheme were presented by Heckbert and Garland [HG95, GH96].

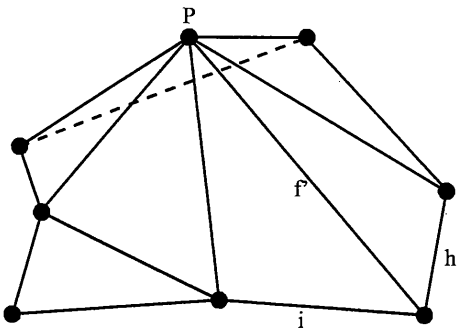
De Floriani and her co-authors have used Delaunay triangulations ex-



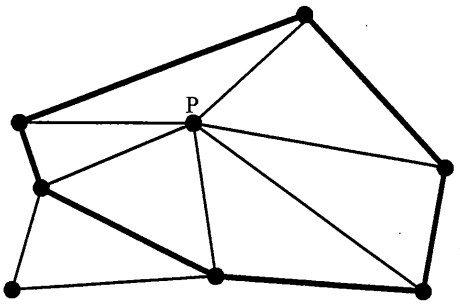
(a) Point P from Figure 2.13 is inserted into its surrounding triangle; edges a, b and c are checked against the Delaunay criterion.



(b) Edges a and b have been swapped with the opposite diagonals in their containing quadrilaterals. As a result, edges d, e, f and g must be revalidated (assuming that this is a portion of a larger triangulation).



(c) Edge f has been similarly swapped; a further two edges must be checked against the Delaunay criterion.



(d) In a plan view of the scene, the star polygon of modified triangles around P is highlighted.

Figure 2.14: A point insertion in a Delaunay triangulation.

tensively as height field approximators. Extensions presented by these authors include Delaunay triangulation over arbitrarily-shaped (rather than strictly convex) domains [DFP85]; a constrained Delaunay triangulation algorithm [DP92b] which produces a triangulation containing given straight-line segments; and hierarchical adaptations [De 87, DMP93, DP92a, De 89]. This hierarchical work will be covered in Chapter 3.

Schröder and Roßbach [SR94] took the opposite approach to De Floriani's technique of repeatedly making local refinements to an approximating

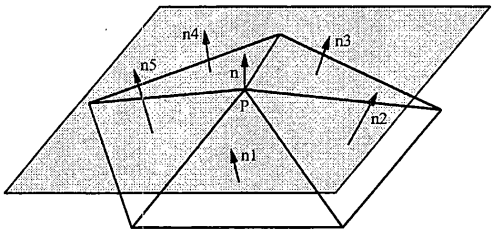


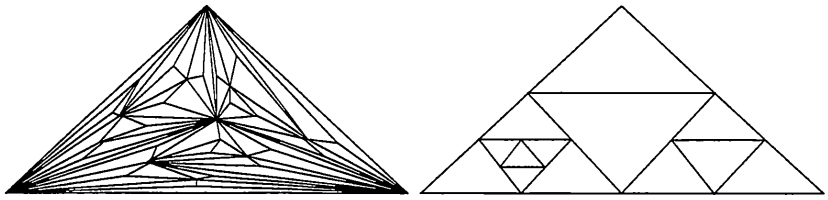
Figure 2.15: Schröder's measure of roughness at point P is the maximum angle between the averaged normal n and its adjacent triangles' normals, $n_1 \ldots n_5$.

surface: they argued that it is easier to find insignificant points on the surface than it is to detect the most important ones. Therefore Schröder and Roßbach's method initially generates a high resolution TIN representing the given surface and then repeatedly reduces the number of points in this mesh. The criterion which is used to assess the significance of a point calculates a measure of the "roughness" of the terrain at that point (Figure 2.15). When the roughness value of a point indicates that it can be removed from the mesh without affecting the overall representation significantly, the area around the removed point is retriangulated.

Two of the original hierarchical triangulation methods which De Floriani addressed are mentioned here to complete our examination of data independent triangulation techniques.

- A *nested ternary triangulation* [DFNP82] can be constructed by iteratively inserting the data point with maximum error and simply connecting that point to its surrounding triangle's vertices. This inevitably leads to an approximating surface which contains long thin triangles (Figure 2.16a) whose lack of suitability for representing a surface visually was apparent in the contours which were extracted from such a representation in [DFNP82].

- A *nested quarternary triangulation* [De 87] can be created by repeatedly subdividing each triangle into four subtriangles. These subtriangles are formed by joining three points, each lying on a different side of the original triangle (Figure 2.16b). This structure avoids the sliver triangles of the ternary version, but introduces the T-vertex problem which was discussed for grid-based solutions in Section 2.2.1.



(a) Ternary triangulation. (b) Quarternary triangulation.

Figure 2.16: Simple triangulations.

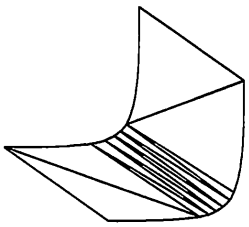


Figure 2.17: Sliver triangles are required to give a good approximation of this curved surface.

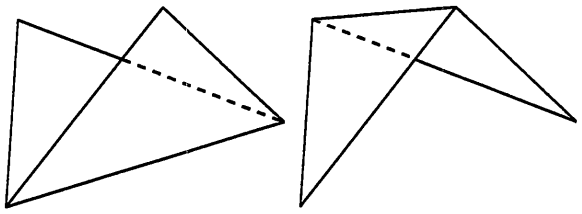
2.2.2.2 Data dependent triangulations

Although the Delaunay criterion produces a "good quality" triangulation in terms of its triangles' aspect ratios, it does not necessarily produce the triangulation which is the best approximation to a given height field for two reasons:

1. sliver triangles are necessary to give a good approximation to some surfaces (Figure 2.17) [Rip92b];

2. the swapping of edges which the Delaunay criterion invokes can cause artificial break lines where none exist in the original terrain (Figure 2.18) [Sca90, SP92].

Both of these reasons were cited by Scarlatos as justifications for her two non-Delaunay triangulation methods. The coarsening process described in [Sca90] repeatedly removes the least significant points in a high resolution triangulation. The "significance" of a point is determined by a) the difference between its elevation and the weighted average of its adjacent vertices, and b) the number of its adjacencies. The retriangulation which is invoked after



(a) Channel line between two triangles.

(b) Ridge line version.

Figure 2.18: The effect in three dimensions of swapping an edge within a quadrilateral. The accuracy with which this approximation resembles a height field will be affected by such an edge swap.

each point removal attempts to retain the terrain's critical lines which are present in the resulting star polygon.

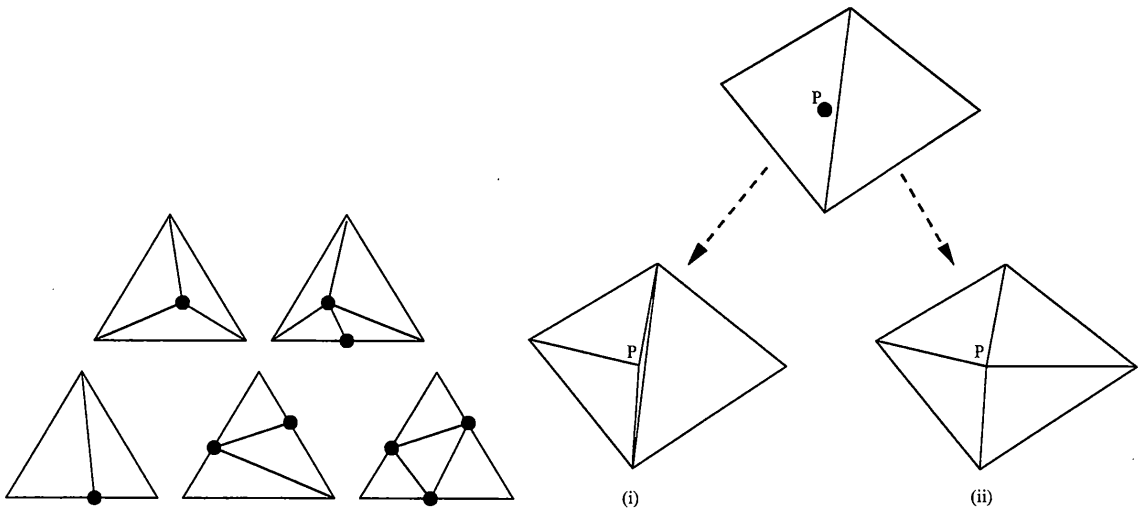
This critical line maintenance also featured in [SP92]. The method in this paper reverts to greedy point insertion but, unlike De Floriani *et al*'s approach, when a triangle is split due to the introduction of new points the splitting process is governed by the critical lines which pass through that triangle. The five possible resulting retriangulations are depicted in Figure 2.19a. The algorithm can also split an existing line in order to accommodate a new point which lies close to it rather than force the creation of a sliver triangle (Figure 2.19b).

Data-dependent triangulations were also explored by Rippa [Rip92a] and Garland [GH96] using error terms involving either the normals of pairs of triangles or the L_2 norm (the sum of squares of the absolute vertical errors of all the datapoints). These authors concluded that a more accurate triangulation than a Delaunay one could be produced at greater computational expense. In one example, a 12% improvement in the error of a Delaunay greedy insertion algorithm was obtained at a cost 3-4 times greater [Hec97].

2.2.3 Feature-based

There are other triangulation methods which follow the example of Fowler and Little (Section 2.2.2.1) by triangulating the points and lines in a terrain dataset which have been flagged as being "significant". These methods are regarded as producing inferior quality approximations compared to other techniques [Hec97] but we shall examine one method whose feature detection criteria is used later.

Douglas [Dou86] presented an entirely different terrain representation



(a) Five possible split strategies for preserving cartographic coherence. The points which have been detected as lying on critical lines are highlighted.

(b) Insertion of point P normally leads to the introduction of a long, thin triangle as in (i). Scarlatos modifies an original line to accommodate P as in (ii).

Figure 2.19: Scarlatos' data dependent triangulation (from [SP92]).

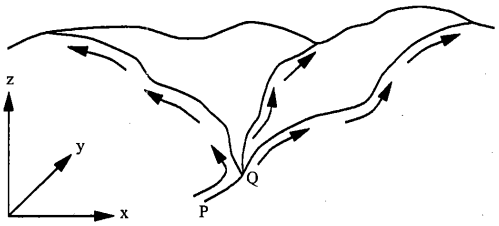


Figure 2.20: Candidate points would be detected on the three break lines indicated (two ridges and a channel) by the initial convolution of Peucker and Douglas' method. These could be connected by traversing through the candidate points in the directions shown, starting first from candidate point P and then from Q.

model to those described previously, which he called a *richline* model. This representation stored the "information rich" lines which were extracted from a terrain surface; these lines were defined to be the ridges, channels and other breaks in slope.

Several methods for detecting these critical lines were described by Douglas and we elaborate on one of the most simple, which was applied by Fowler and Little. This method, originated by Peucker and Douglas [PD75], is a local method whereby a 2×2 matrix is passed over the dataset, marking the two elements which are not its highest and lowest components in every position. Once this convolution is complete, the remaining unmarked points are potential ridge and channel candidate points. Fowler and Little use a hill-climbing algorithm to connect these candidates into lines. Starting from a point which is a candidate and is lower than all its neighbouring candidates, the algorithm repeatedly climbs to the highest neighbouring ridge candidate until a peak or an existing ridge is encountered. The nature of this algorithm is indicated in Figure 2.20.

2.3 Manifold surface approximation

Although many manifold-handling approximation techniques work in the same manner as height field approximators, i.e. they repeatedly make local approximations to an approximating surface, a manifold has no analogue of a height field's planar domain. This introduces complexities to both the calculation of the quality of an approximation and also to any retriangulation step. Note that an approximating technique which can handle a manifold surface with boundary can also, of course, approximate a height field.

In this section we discuss a method which approximates a manifold surface

using refinement in Section 2.3.1 and then its converse, decimating methods which repeatedly remove points from a high resolution representation of a manifold surface, in Section 2.3.2. Turk's retiling method and other methods which also use optimisation techniques are discussed in Sections 2.3.3 and 2.3.4. Finally, as an aside we mention wavelet-based manifold approximators in Section 2.3.5. All of the methods described in this section, except the refinement technique of Section 2.3.1, can handle manifold surfaces with or without boundaries.

2.3.1 Refinement

Faugeras *et al* [FHMB84] extended De Floriani's nested ternary triangulation (which applies only to a height field; see Section 2.2.2.1) to any input mesh which can be represented by a planar graph embedded on the surface of an object of genus 0, i.e. an object homeomorphic to a sphere.

Faugeras' method creates this graph by repeated refinement of an initial "pancake-like" [Hec97] two-triangle approximation bounded by three vertices from the input mesh. Each point in the mesh is associated with its closest triangle in the current approximation. The refinement step adds the point with maximum error to its containing triangle and splits that triangle accordingly. Edges can also be split to avoid the sliver triangle problem of the ternary version but in general the resulting graph retains the poor quality of approximation associated with that height field method.

2.3.2 Decimation

The term *decimation* was introduced by Schroeder *et al* [SZL92] to describe the repeated removal of points from a TIN to produce a mesh of lower complexity which corresponds to the original to within a given tolerance. We discuss Schroeder's method and similar approaches which use alternative error measures and retriangulation steps. In the same vein, we examine two techniques which can guarantee that an output decimated surface lies within a certain tolerance from the original model. To conclude this section, we mention some methods which identify those facets of a model which are sufficiently planar to permit merging of these facets and thus reduce the complexity of a model.

Schroeder *et al* [SZL92], adopted a simpler local error measure than that of Schröder and Roßbach which was described in Section 2.2.2.1. Schroeder's measure simply ascertains the distance from a vertex P to the average plane obtained from its surrounding triangles (Figure 2.21). If this distance is less than a specified parameter then P can be removed and the resulting star polygon retriangulated. An additional feature of this method is that sharp edges and corners are identified and retained during the decimation process.

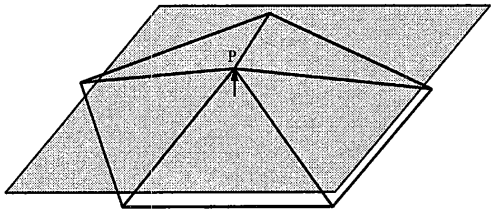


Figure 2.21: Schroeder's error measure: the distance between a vertex P and the plane formed as the average of P's surrounding triangles.

Klein *et al* [KLS96] and Bajaj and Schikore [BS96] are among the authors who have proposed variants of Schroeder's basic decimation step. Both of their methods concentrate on ensuring that there is a global upper bound on the error of the decimated meshes. The former method guarantees that the *Hausdorff distance* between a decimated mesh and the original is less than a user-specified tolerance. The Hausdorff distance $\mathcal{H}(S_1, S_2)$ between two surfaces (S_1, S_2) is the maximum of the max-min distances between any point on S_1 and a point on S_2 and vice versa. Formally, the Hausdorff distance is defined as:

$$\mathcal{H}(S_1, S_2) = \max(\max_{y \in S_2} \min_{x \in S_1} \|x - y\|, \max_{y \in S_1} \min_{x \in S_2} \|x - y\|)$$

This is the standard measure of consistency between two surfaces; it has the property that $S_1 \equiv S_2 \iff \mathcal{H}(S_1, S_2) = 0$.

The task of providing global error control is central to Cohen *et al*'s *simplification envelopes* method [CVM+96, Var94]. This method initially establishes an inner and an outer simplification envelope around the surface to be approximated. These polygonal envelopes are approximations to offset surfaces of the original model, the major difference being that the envelopes are guaranteed to be non-self-intersecting. Then a form of mesh decimation can be performed locally or globally on the original surface while ensuring that the retriangulated modifications lie between the inner and outer simplification envelopes. Thus the approximated surface can be guaranteed to lie within a given error tolerance from the original surface.

Cohen suggested that the simplification envelopes method could be made locally adaptive by making the "width" between the inner and outer envelopes inversely proportional to the distance from the viewer (Figure 2.22), but this would not be computationally feasible for a dynamic scene.

A method which can be viewed as the dual of the simplification envelopes approach is Guéziec's simplification inside a tolerance volume [Gué96]. Instead of predefining the volume within which simplification can proceed,

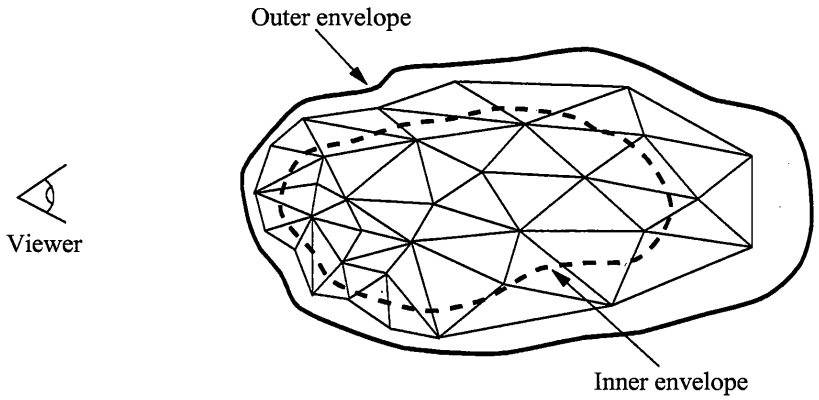


Figure 2.22: Cohen's simplification envelopes bounding the volume within which simplification can take place (the inner envelope lies inside the object). Here, the displacement between these envelopes is inversely proportional to the distance from the viewer to give a form of static view-dependency.

Guéziec's method maintains, during a set of local decimation steps, a volume which is defined as the convex hull containing projections of the errors which have been introduced at each vertex (Figure 2.23). This volume grows as the simplification proceeds and the process terminates when the volume reaches the size of a given tolerance volume.

An alternative scheme for determining which vertices in a surface can be decimated was proposed by Hinker and Hanson [HH93]. Their method permits nearly coplanar facets to be identified and merged. To be fully efficient, this requires adjacent near-coplanar sets to be identified in order for their boundaries to be similarly simplified. Figure 2.24 shows how the number of facets in a planar area can be drastically reduced if the boundary of this area is also simplified. Similar approaches were suggested by DeHaemer and Zyda [DZ91] and (for height fields) Taylor and Barrett [TB94].

2.3.3 Re-tiling

Turk [Tur92] advocated introducing new points onto a polygonal mesh by a method of point repulsion which adds more vertices in regions of higher curvature. The old points can then be discarded, with repeated local retriangulation (or *re-tiling*) as necessary, until a new mesh consisting of the new points is obtained. This approximated surface is guaranteed to be topologically equivalent to the original.

This approximation technique works best for smooth surfaces with no

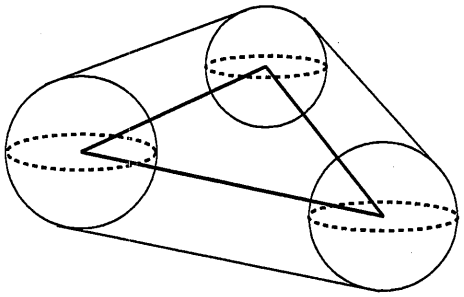


Figure 2.23: The volume within which simplification can proceed for a single triangle in Guéziec's method. The volume is formed as the convex union of a set of spheres whose radii are the errors which previous simplification has introduced across the surface of the triangle.

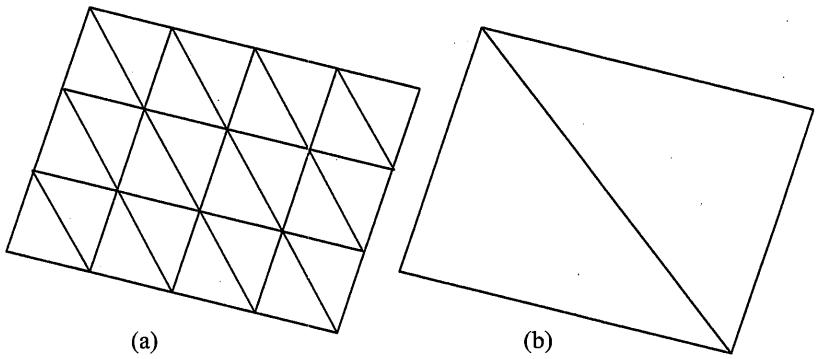


Figure 2.24: A coplanar (or nearly coplanar) set of facets (a) can be reduced to a simpler set (b) if the boundary of the region can be simplified correspondingly.

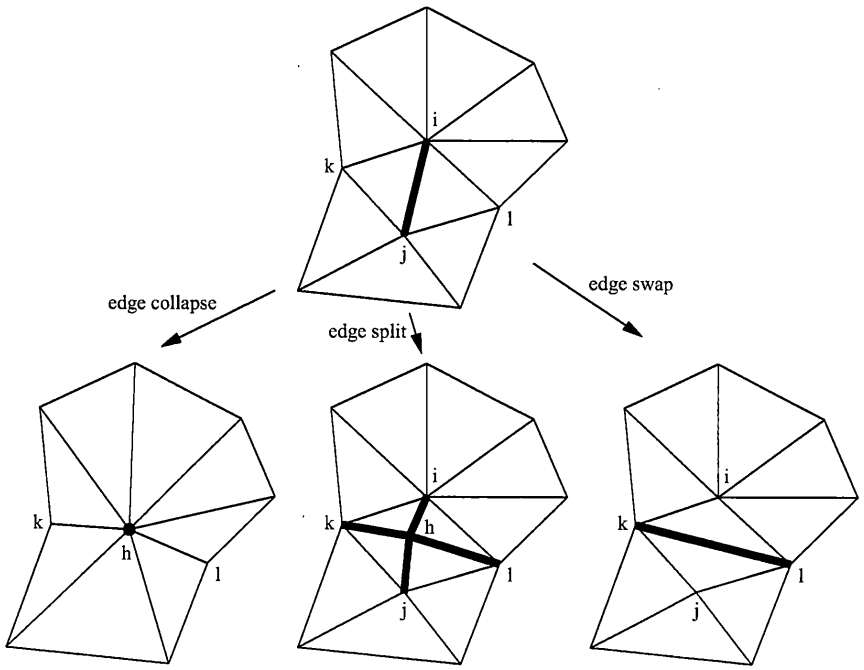


Figure 2.25: The three possible local modifications invoked by Hoppe (from [HDD+93]).

discontinuities.

2.3.4 Energy optimisation

Hoppe *et al* [HDD+93, Hop94] posed the mesh simplification task as an energy minimisation problem. Their energy function combines a measure of the geometric similarity between an approximating mesh and the original data with a cost proportional to the number of datapoints in the approximating mesh. An additional term, which Hoppe called "spring energy", assists the optimisation process in finding a consistent local minimum.

Their method repeatedly coarsens an approximating simplicial mesh during the minimisation process using local modifications (Figure 2.25). Only modifications which maintain the topological type of the mesh are permitted. The new vertices which can be introduced by an edge collapse or an edge split are positioned according to a local optimisation strategy.

Hoppe later refined this method as the *Progressive Meshes (PM)* approach [Hop96]. This approach uses only the edge collapse operation to produce simplified meshes and specifies an efficient variable resolution representation for polygonal meshes which takes advantage of the invertible nature

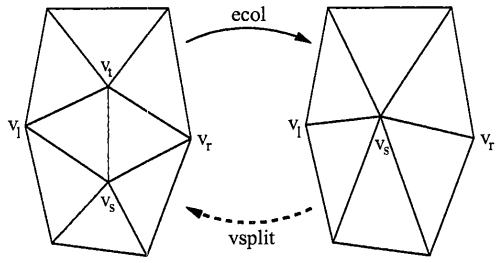


Figure 2.26: Hoppe's edge collapse and vertex split operations (from [Hop96]).

of this operation. Also, the spring energy term was removed from the energy function and replaced with a measure of the significance of each edge to the discontinuity curves in the mesh. This ensures that discontinuities such as colour changes and sharp edges can be retained in simplified meshes.

In the PM approach, the edge collapse transformation is viewed as in Figure 2.26. An edge collapse transformation (*ecol*) unifies two vertices into one and removes two triangles in the process. Its inverse operation is termed a vertex split (*vsplit*).

The PM representation of a model is a base coarse mesh together with a sequence of vertex splits which can be used to recreate the original mesh (Figure 2.27). Hoppe demonstrated that a PM representation can permit:

- smooth geometric morphing (or *geomorphing*) between approximations extracted at different points along the sequence of vertex splits. Note that these meshes are still single-resolution approximations, i.e. the resolution is fixed across an entire mesh;
- a model to be progressively transmitted by initially sending the base mesh and then successively adding detail (by geomorphing) when the vertex splits are sent;
- a space-efficient form of representation;
- a degree of selective refinement. The limitations of Hoppe's original selective refinement scheme, together with his later extensions, are discussed in Section 3.2.2.

2.3.5 Wavelets

Multiresolution analysis of surfaces for graphical rendering was pioneered by Lounsbery and Eck *et al* [LDW94, EDD+95]. These authors demonstrated that a surface can be represented by a base mesh together with a set of

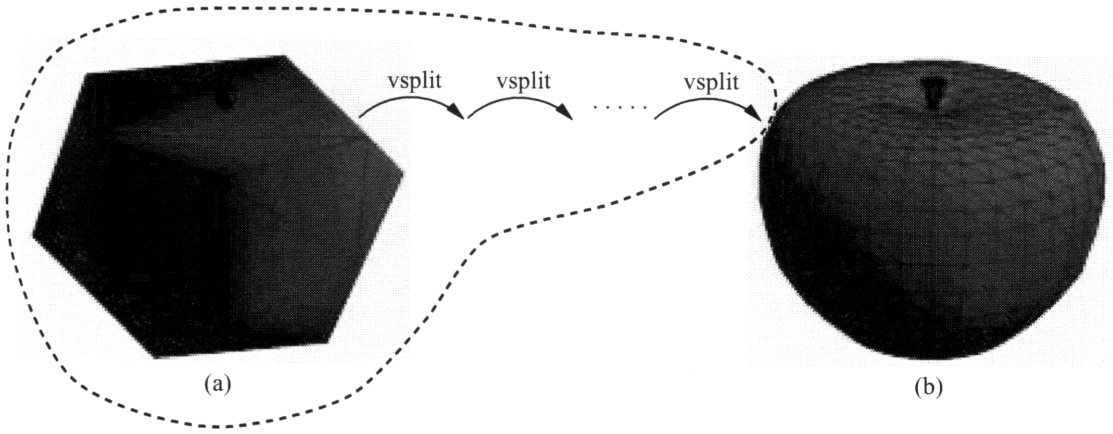


Figure 2.27: A base mesh (a) together with a sequence of *vsplits* are an object's PM representation (outlined). This can be used to reconstruct the original mesh (b).

local modifications in the form of wavelet coefficients. This wavelet-based approach can lead to a more compact representation and naturally permits the extraction of multiple single-resolution approximations. In addition, geomorphing can be performed between these approximations.

Lounsbery noted that multiresolution editing of a wavelet-represented model is possible, i.e. that changes can be made at various levels of detail and these changes will be reflected throughout the range of resolutions at which approximations can be extracted. A wavelet representation cannot, however, directly permit a selectively refined approximation to be extracted in the manner of our SMR framework. As we discussed in Section 2.2.1, Gross combined a wavelet technique with a quadtree structure to permit this kind of variable resolution extraction.

2.4 Non-manifold approximation

Although the SMR framework which this thesis describes cannot handle non-manifold models, this section reviews a number of non-manifold-handling approximation methods since one of the most recent [GH97] could be adapted to work within the framework. The natural ancestors of this method are reviewed primarily for completeness, but also as a basis for the discussion of how the framework could be extended in the future.

The significance of these non-manifold approximation methods, apart from the extension to the range of models which can be approximated, is their ability to modify the topology of an input model during the approxi-

mating step. This topology-modifying characteristic was originally advocated by Rossignac [RB93] but was denigrated by contemporary authors [Sew96, HG94]. The view that a topology-preserving method (such as any of the ones previously described) was superior to a topology-modifying one was due to the belief that all of the topological features of a model were significant and that their elimination detracted from an approximation [Sca90]. This argument has been overshadowed recently, though, by the demand for approximators to handle non-manifold models, and also for *full-range approximation* [RR96], i.e. the ability to reduce any model to a minimal representation such as a tetrahedron or sphere. This latter requirement ensures that any model viewed from afar can be represented by a suitably small number of facets; also, if progressive transmission of a model is desired (such as in the Progressive Mesh approach reviewed in Section 2.3.4) then the base mesh can be transmitted with a guaranteed cost.

We first examine Rossignac's vertex clustering approach and its descendants (Section 2.4.1). This is followed by a discussion of non-manifold-handling edge collapsing methods (Section 2.4.2). We conclude with a discussion of Popović and Hoppe's *Progressive Simplicial Complexes* (Section 2.4.3) which can be viewed as a non-manifold extension of Hoppe's PM approach.

2.4.1 Vertex clustering

The approximation methods described in this section satisfy the non-manifold and full-range requirements by their use of a relatively crude approximating step. This *vertex clustering* step can be viewed as an extension of localised vertex decimation (Section 2.3.2) in that a group of vertices are identified for elimination and then they are replaced by a single representative vertex. The simplicity of this step permits its application to vertices and faces for which no topology has been defined.

Rossignac [RB93] originated this form of approximation process and his method can be partitioned into the following phases.

Grading Each vertex is assigned a weight according to its estimated perceptual importance. This is determined by the likelihood that a vertex can appear on a silhouette edge and also whether it lies on the boundary of a large face.

Clustering The space within which the vertices lie is partitioned into voxels of uniform size and vertices are clustered into their containing voxel.

Synthesis A representative vertex is identified for each cluster. This vertex is one which is geometrically close to the average of all the vertices in its cluster, weighted by their grading. If links between the original

vertices and their representatives are maintained, smooth transitions between levels of detail can be performed.

This linkage was combined with an octree spatial subdivision hierarchy by Luebke [LE97] to permit the production of selectively refined representations from vertex clustering approaches. Such representations suffer from the low quality of approximation inherent in these approaches.

Elimination The result of clustering vertices into representative examples is that triangles can collapse into points or lines and edges can collapse to points. These degenerate cases can be eliminated.

Low and Tan [LT97] offered a more accurate method of estimating the importance of each vertex in the grading phase and modified the clustering step to permit vertices to move between cells and hence removed the artificial uniformity of the voxelisation process.

2.4.2 Edge collapsing

The edge collapse operation, whose use on manifold surfaces was described above, was applied to non-manifold meshes in a topology-varying manner by Ronfard [RR96] and Garland [GH97]. The aim of these authors' methods is to retain a closer geometric similarity between approximated meshes than is possible with the vertex clustering approach.

Both of these methods perform local edge collapse operations and optimise the position of the replacement vertex (as depicted in Figure 2.26). The resulting approximation error can be measured as the sum or maximum of the distances from the new vertex to the planes of the original object's surfaces. While Ronfard accumulates at a vertex the planes of the original object whose surfaces have been approximated by that vertex, Garland maintains this error measure more efficiently in the form of quadric matrices.

Garland also permits *pair contractions* rather than just edge contractions: a pair of vertices can be aggregated even if they are not connected by an edge. This permits previously disconnected regions of a model to be combined and hence non-manifold surfaces can be approximated with greater accuracy and efficiency.

2.4.3 Progressive Simplicial Complexes

Popović and Hoppe's Progressive Simplicial Complexes (PSC) representation [PH97] takes both Hoppe's Progressive Mesh approach and Garland's pair contraction step to their logical conclusions. The PSC representation adheres to the PM concept of a model being represented by a base entity

together with a set of refining modifications. Where the PSC method generalises the PM approach is that it uses a vertex split modification which is potentially topology-modifying.

This generalised *vsplit* operation permits the PSC representation to handle non-manifold objects and also allows it to perform full-range approximation. Thus the PSC representation of an arbitrary triangulated model is a single vertex together with a set of these generalised *vsplit* operations.

Popović described how geomorphing these *vsplits* could be performed in a similar manner to the PM geomorphing, but no means of producing a selectively refined approximation was suggested.

2.5 Summary

This chapter has considered a range of existing approximation techniques which can handle a variety of model classes – height fields and manifold and non-manifold surfaces. Many of these techniques have a common mode of operation which is to make repeated local modifications to an initial approximation. We shall see later how such methods can be incorporated into our Selective Mesh Refinement framework.

Chapter 3

Variable Resolution Schemes

In this chapter we describe how the approximation methods which were detailed in Chapter 2 have been used in previous work to generate some form of selectively refined output, i.e. an approximation whose resolution varies within its extent. Also under consideration is whether it is possible to geomorph between the approximations output by these methods.

Chapter 1 introduced some approaches which fall within the general umbrella of "multiresolution" methods but which cannot produce selectively refined approximations. For completeness, we indicate how some of the approximation methods previously discussed can generate such non-variable resolution output in Section 3.1.

Sections 3.2 and 3.3 detail how some of the approximation methods previously described can produce selectively refined approximations. In the first of these sections, we consider techniques which explicitly constrain an approximating technique in order to permit variable resolution mesh extraction. In the second, we describe Cignoni's variable resolution terrain extraction scheme [CPS95] which was the inspiration for our SMR framework.

Section 3.4 describes the *MultiTriangulation* approach of Puppo and De Floriani which is contemporary with our work and which concentrates on the selective refinement of terrain surfaces.

Section 3.5 reviews the approximation methods which have been discussed in this chapter and the previous one.

Finally, Section 3.6 examines the resolution criteria which have been applied in previous work to generate selectively refined output.

3.1 Other multiresolution methods

The following two sections consider methods which add some degree of "multiresolution" ability to the output of specific single-resolution approximation techniques. Section 3.1.1 describes two simple forms of linkage between LODs

which can assist with certain mesh-related operations. Section 3.1.2 describes how variable resolution meshes can be obtained from the output of particular approximation methods by performing some remeshing to "splice" together several single-resolution meshes.

3.1.1 Multiple LODs

The original and simplest form of "multiresolution" modelling is the multiple single-resolution Levels Of Detail scheme which is utilised by Open Inventor and VRML. This method, together with its drawbacks, was introduced in Section 1.1. The major advantages of this method are its simplicity and universality: any approximation-producing algorithm can be incorporated into this scheme.

There are obvious benefits to be realised by establishing some form of correspondence between the multiple LODs in this scheme. We consider below two methods which retain connections between multiple LODs in order to tackle a specific task. In general, the benefits of these methods are offset by their requirement that the LODs must be generated by a specific approximation method.

Pyramidal linkage By linking the triangles which overlap in adjacent Levels Of Detail, De Floriani [De 89] and Scarlatos [Sca90] described how point location and other domain searches could be performed more efficiently than on a single approximation. In addition to storing each LOD as a standard triangulation, their methods explicitly link every triangle in an LOD to those triangles which it overlaps in the preceding and succeeding LODs.

Figure 3.1 shows the links which would result from two point insertions in a small triangulation. In practice, the differences between each LOD would be greater since space constraints would limit the number of LODs which could be stored in a pyramid.

Fragment tracking Turk and Rossignac suggested how their specific approximation techniques (of re-tiling and vertex clustering, respectively; Sections 2.3.3 and 2.4.1) could admit a form of geometric morphing between LODs generated by these methods. In each case this can be performed using links created during the algorithms' local modification steps.

During the re-tiling step of Turk's method (Section 2.3.3), each re-tiled region can be projected onto its lower resolution equivalent and the intersections between these two triangulations retained (Figure 3.2) so that the fragments of one can be smoothly interpolated into the triangles of the other when required.

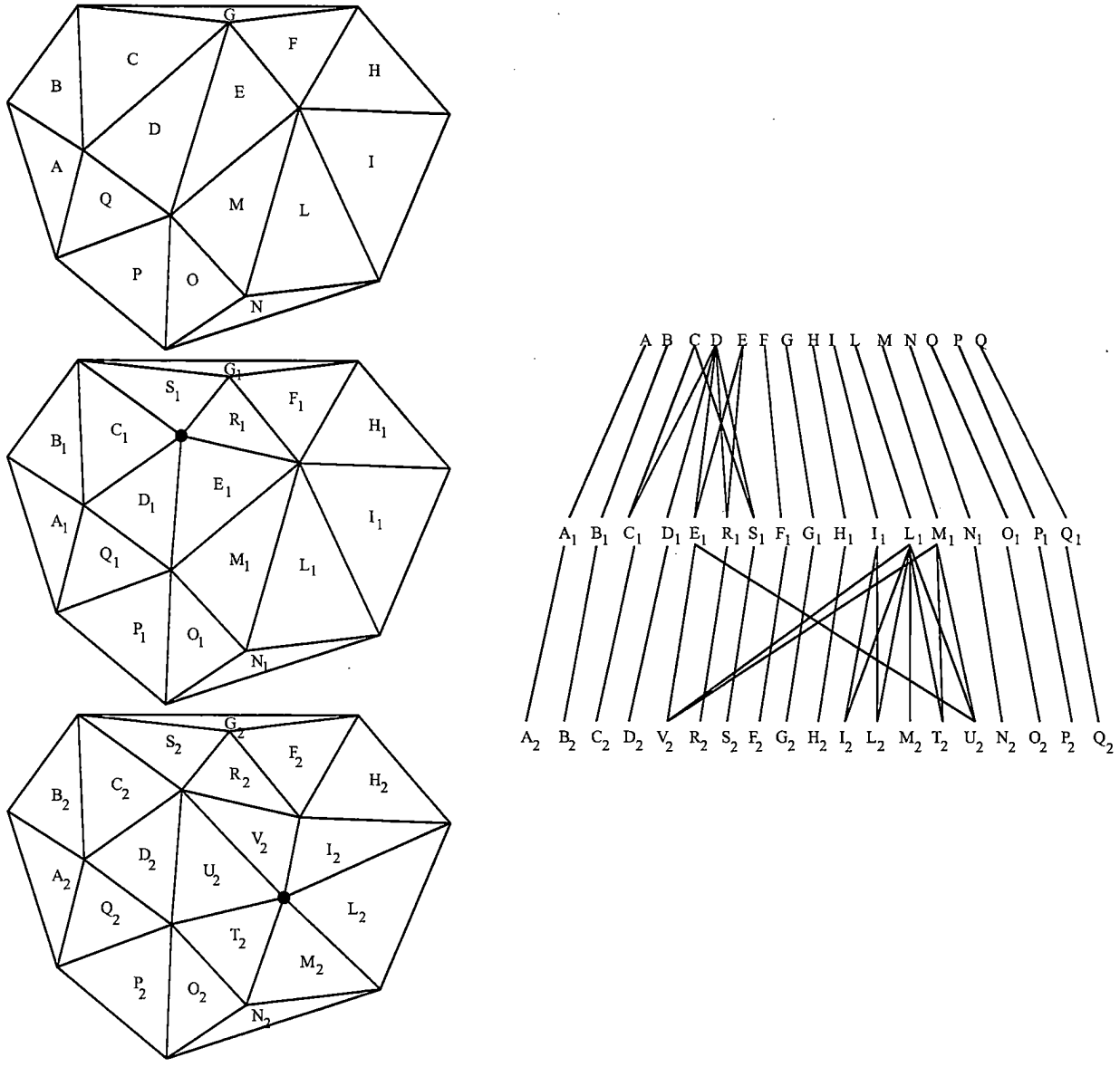
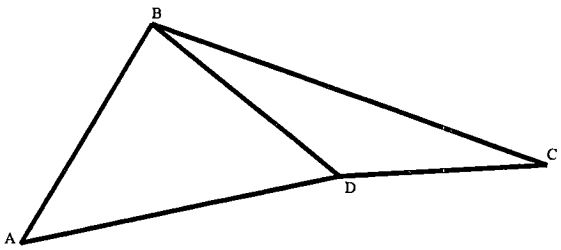
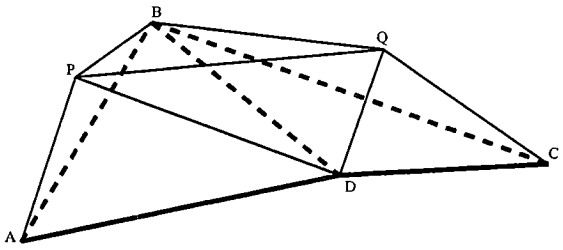


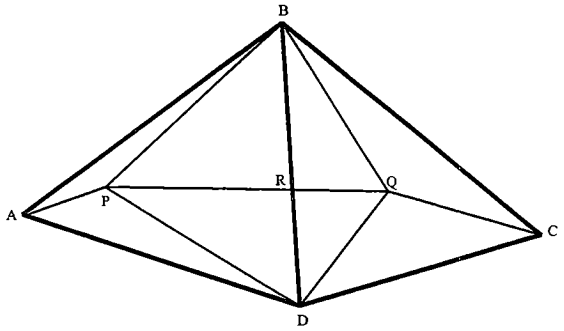
Figure 3.1: Pyramidal linkage between triangles in LODs (adapted from [De 89]). The impact of two point insertions is reflected in the links between overlapping triangles in adjacent LODs.



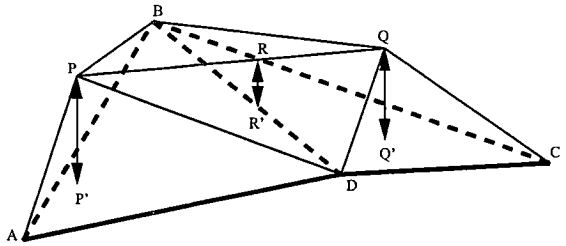
(a) A region $ABCD$ which is to be morphed into a higher resolution version.



(b) Points P and Q introduce a higher resolution triangulation of $ABCD$. We wish to interpolate between the triangles (ADB,CBD) and their higher resolution equivalents.



(c) A plan view of region $ABCD$. Note that the line PQ intersects the low resolution triangulation at R.



(d) The projections of P and Q onto the original triangulation are stored as well as R and its projection so that the line PQ can be linearly interpolated from the fragment which existed prior to the introduction of the higher resolution triangulation.

Figure 3.2: Information required for the smooth introduction of a higher resolution triangulation according to Turk's method.

3.1.2 Selective refinement by remeshing

A hierarchy of component LODs such as the human body example given in Section 1.1 can permit a scene to be displayed which contains distinct objects, or components of objects, at differing resolutions (Figure 3.3). Such a hierarchy cannot, though, provide a variable resolution solution for large objects if they do not lend themselves to being partitioned into components. Also, the use of an LOD hierarchy can only assist with geomorphing between LODs if the underlying approximation method provides an approach such as the fragment tracking described above to permit the necessary mor-

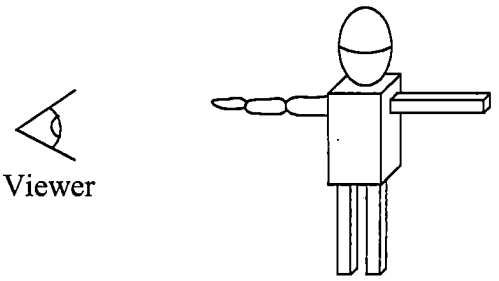


Figure 3.3: An object extracted in a view-dependent manner from the hierarchy of Figure 1.4, i.e. with higher resolution components in the viewer's foreground.

phing. As an aside, we note that hardware transparency blending can be used to circumvent both of these problems: LODs can be smoothly introduced/removed/merged using blending. This requires hardware which supports blending and also incurs the additional cost of rendering more than one LOD of a single object during an LOD transition.

This section discusses some advances on Clark, Funkhouser and Sewell's component-based LOD hierarchies [Cla76, FS93, Sew96] which permit LODs to be combined using a degree of remeshing to provide selective mesh refinement. We do not regard this as "true" selective mesh refinement because the remeshing step removes any guarantee on the quality of the output surface and also adds complexity to the geomorphing problem.

The pyramidal structures of De Floriani and Scarlatos [De 89, Sca90] which were described in the previous section do not immediately lend themselves to producing a variable resolution surface [DP92a]. Scarlatos stated that such a surface could be produced by "fusing" the LODs contained in her pyramidal structure and Figure 3.4 demonstrates this for the previous simple pyramid example. If we wish to merge the left portion of the lower resolution LOD from Figure 3.1 with the right portion of the higher resolution LOD in that diagram then the unshaded set of triangles in Figure 3.4 could be extracted by a single pass through the pyramid. The shaded triangle represents a hole in the extracted surface for which there is no existing triangulation in the pyramid. Filling this hole with one or more triangles invalidates any guarantee of the resolution of the output surface because the accuracy with which these triangles represent the original object is not known. The complexity of the retriangulation necessary for a real example would depend on the coherency of the pyramid's LODs.

An alternative hierarchical system which requires retriangulation to generate a variable resolution output is the quadtree structure which Cosman

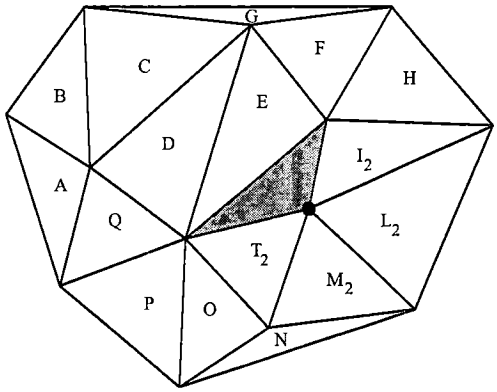


Figure 3.4: To satisfy some resolution criterion, the triangles created by the first point insertion in Figure 3.1 have not been included in this mesh, but those which were caused by the second point insertion have. This results in a hole in the triangulation (shaded) for which no compatible set of triangles exists in the pyramid of LODs.

and Zyda [CMR90, FZPM93] used in their simulators. The quadtree T-vertex problem which necessitates remeshing at the boundaries of these LOD nodes was covered in Section 2.2.1.

Ferguson's *continuous terrain LOD* system [FEKR90] presents a means of avoiding the T-vertex remeshing which would otherwise be required when selectively refining a quarternary triangulation. His surface extraction algorithm instead moves nodes' boundary vertices when necessary to extract a seamless surface from the quarternary hierarchy. Figure 3.5 illustrates one vertex movement which would be performed if two triangles from different levels in a hierarchy were adjacent in an extracted surface. A similar means of meshing triangulated nodes from a hierarchy was adopted by Scarlatos (Section 2.2.2.2) to combine fragments which could have non-matching boundary edges.

Ferguson suggested that this vertex movement could be performed using linear interpolation to smoothly introduce vertices when higher resolution quarternary subdivisions are used to replace lower resolution triangles. In this way, geomorphing could be performed between such remeshed variable resolution surfaces.

3.2 Constrained approximation

In this section we discuss more complex algorithms and structures which discard the multiple LOD system and permit true selective mesh refinement,

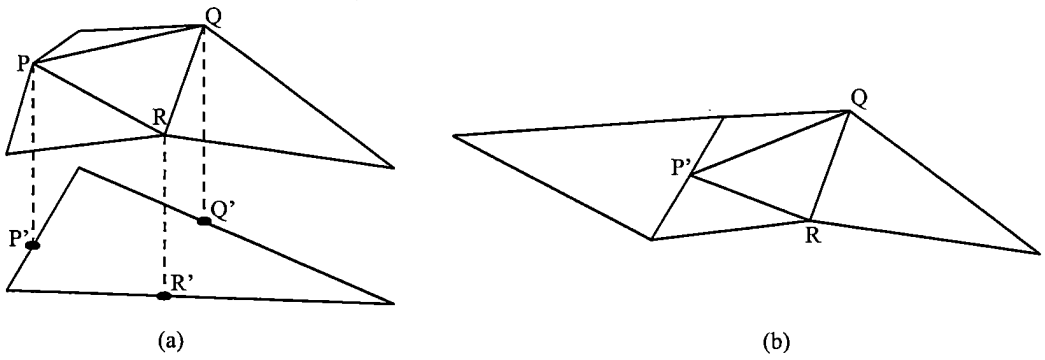


Figure 3.5: The edge vertices of a quarternary subdivision of a triangle are associated with their projections on the original boundary (a). When this subdivided node is adjacent to a non-subdivided triangle (b), edge vertex P is retained at its lower resolution position P'.

i.e. a variable resolution surface can be generated from the precomputed representations without further remeshing. A common characteristic of these methods is that such surface extraction is enabled by placing constraints on the approximation process.

We first examine how constraints placed on some of the approximation methods which we introduced in Section 2.2.2 can enable the generation of selectively refined surfaces representing height fields. This is followed by a detailed study of how a Progressive Mesh representation can produce selectively refined manifold surfaces.

3.2.1 Variable resolution height fields

The simplest structure from which a variable resolution representation of a height field can be produced is De Floriani's nested ternary triangulation [DFNP82]. For example, the set of point insertions which produced the subdivision of Figure 2.16a can be represented by the hierarchy of Figure 3.6. A variable resolution surface can be extracted from this structure by a recursive traversal of the tree, starting at the root node, which queries whether or not each node meets the resolution requirements of its domain. If a node does indeed meet these requirements then its triangles are added to the current triangulation, otherwise the traversal continues on its child nodes. Such a traversal produces a complete surface because each node is a direct replacement of its parent triangle, i.e. the boundary of each node retains the three edges of its parent triangle – each node is said to be *compatible* with its siblings. Unfortunately, the sliver triangles which a ternary triangulation contains are unacceptable for rendering purposes.

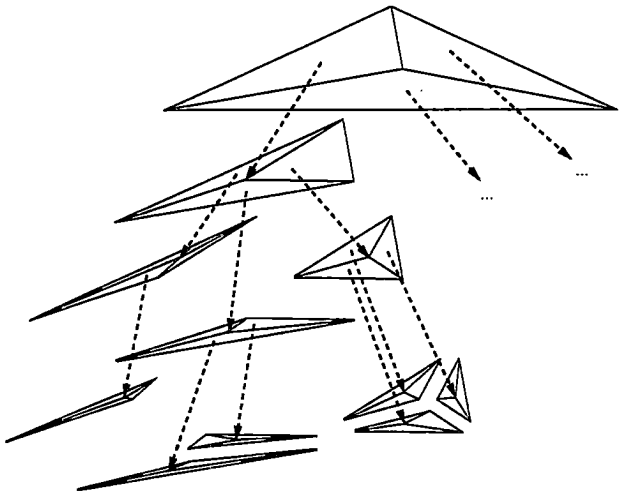


Figure 3.6: Part of the tree which could represent the point insertions used to produce Figure 2.16a.

Combining compatible triangulated fragments to provide a variable resolution surface, though, is an approach which has been adopted by many authors and it forms the basis for our SMR framework. The requirement to use a more suitable, and hence more complex, triangulation approach than the ternary one implies that a more complex structure is required to manage the relationships between triangulated fragments.

Lindstrom's vertex dependency tree which was discussed in Section 2.2.1 is one structure which can be used to perform very efficient variable resolution surface extraction. However, the highly-constrained approximation method which is the basis for this method's real-time performance means that it can only be applied to grid-based height fields.

The two alternative structures which we review in the following sections can handle irregular triangulations and are more relevant to our primary aim of a flexible scheme for variable resolution surface extraction.

3.2.1.1 De Floriani's HTIN

De Floriani initially presented her *Hierarchical TIN (HTIN)* [DP92a, DMP93] as an alternative scheme to the pyramidal linkage discussed above. That is, she described how surfaces of constant, but not variable, resolution could be extracted from an HTIN. In this section we concentrate on her later work [DP95] which presented two variable resolution extraction routines, one of which requires a remeshing step.

An HTIN is constructed with respect to an increasing set of resolution

values, $\{\varepsilon_0, \ldots \varepsilon_k\}$. The root level of an HTIN is a TIN which satisfies some precision ε_0. Each triangle of this TIN is then refined individually into a triangulation which satisfies ε_1. This operation is performed recursively on every new triangulation until each portion of the original domain has been refined to the highest required tolerance, ε_k.

The HTIN triangle-to-triangulation refinement step initially inserts vertices at the points of maximum error on the edge of a triangle and then retriangulates the interior using the Delaunay selector criterion to insert points as necessary to meet the given resolution tolerance for that triangle. This is illustrated in Figure 3.7a.

The edge point insertions are constrained to match those on adjacent triangles at the same level in the hierarchy (Figure 3.7b). Thus a single-resolution surface can be extracted from this hierarchy by a simple recursive procedure which traverses down to the necessary level of the hierarchy. The boundary matching rule guarantees that the resulting surface will be complete.

If a variable resolution surface is extracted using the same recursive routine, nodes from different levels in the hierarchy which do not match along their adjacent boundaries may be extracted and hence only a generalised triangulation (containing T-vertices) may be extracted [DP95]. This can be retriangulated to produce a complete surface but, as before, this step invalidates any guarantee of the resolution of the surface.

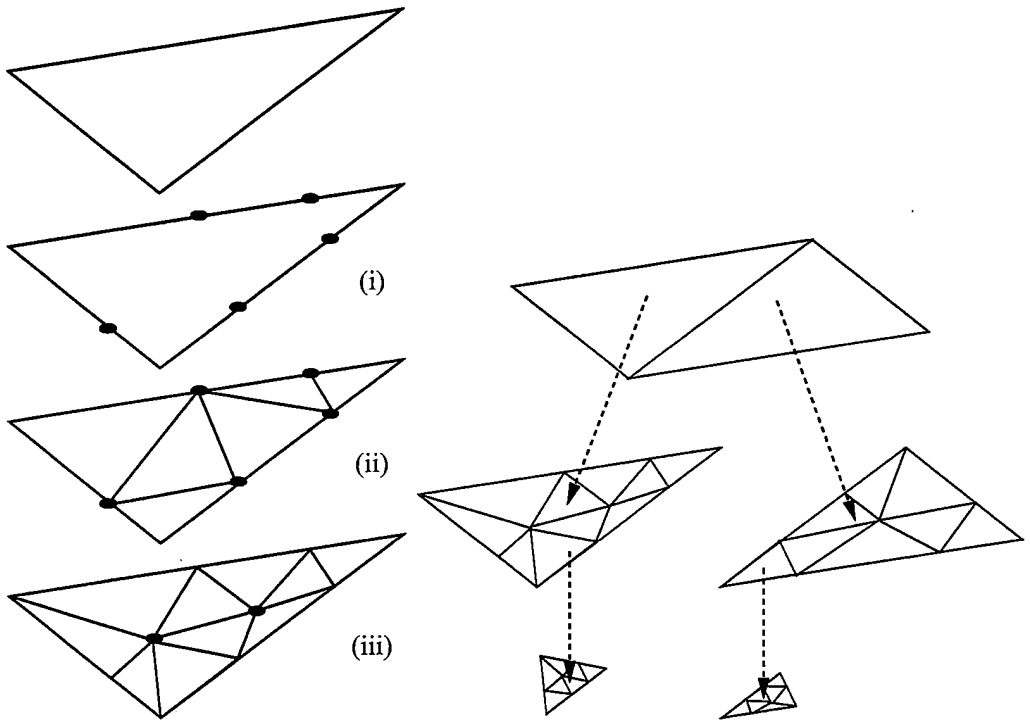
De Floriani also adapted Cignoni's variable resolution extraction algorithm [CPS95] to produce a true selectively refined surface from an HTIN. This algorithm is discussed in Section 3.3.

3.2.1.2 De Berg and Dobrindt's star polygon graph

De Berg and Dobrindt presented an alternative scheme for combining Delaunay triangulations such that both constant and variable resolution surfaces can be extracted [dD95]. This was the first scheme which ensured that its output variable resolution surfaces also satisfied the Delaunay criterion. Unfortunately, its extraction algorithm cannot guarantee that these output surfaces satisfy the input resolution criterion.

The layers of de Berg and Dobrindt's hierarchical structure correspond to Delaunay triangulations produced by repeated coarsening of a high resolution triangulation. This coarsening step is constrained by permitting only non-adjacent vertices to be removed. The resulting retriangulation is therefore restricted to the regions of a number of disjoint star polygons (Figure 3.8).

Rather than linking the pre- and post-coarsening triangles between LODs individually, as in De Floriani's pyramidal linkage, de Berg and Dobrindt's triangles are associated with the star polygon in which they reside (Figure 3.9). The complete set of star polygons which result from a sequence



(a) Triangle refinement in an HTIN. The points with significant error on the edges of a triangle are detected (i) and triangulated (ii). Further points are added into the triangulation to satisfy the resolution requirement for this level of the HTIN (iii).

(b) A portion of an HTIN hierarchy. Note the matching boundaries of triangulations at each level in the hierarchy.

Figure 3.7: HTIN refinement and hierarchy (from [DP92a]).

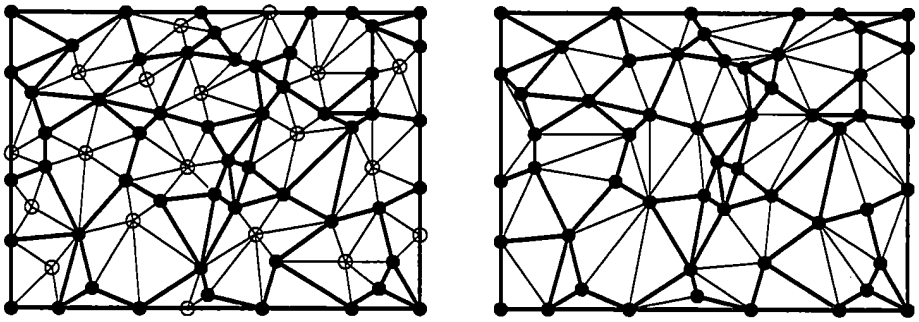


Figure 3.8: A set of pairwise non-adjacent vertices (circled on left) whose star polygons are retriangulated to form a coarser LOD (from [dD95]).

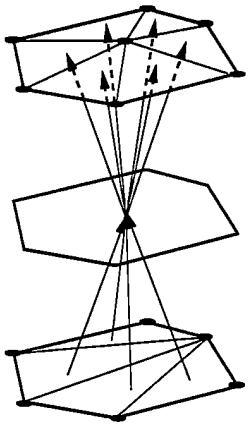


Figure 3.9: A star polygon linking two sets of compatible triangles.

of coarsenings of the original triangulation are then linked by arcs which represent their overlapping dependencies.

De Berg and Dobrindt's surface extraction algorithm which operates on this structure is essentially greedy in nature and, while it uses the star polygon graph to ensure that a complete surface is produced, it does not make full use of the dependencies between the star polygons. As a result, triangles which do not satisfy the resolution criterion may be added to the output surface.

Cohen-Or and Levanoni [CL96] simplified de Berg and Dobrindt's approach by omitting the pairwise non-adjacent constraint on the vertices removed during the coarsening phase. They also reduced the dependency structure between retriangulated regions to a tree (rather than a graph) and decreased the number of levels in the hierarchy. This simplified their extrac-

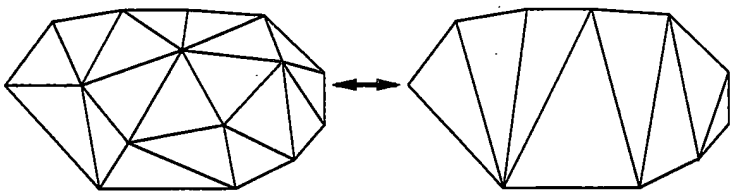


Figure 3.10: Two triangulations which could be smoothly interpolated by Cohen-Or's method (from [CL96]).

tion procedure at the cost of increasing the size of the fragments from which a variable resolution surface could be constructed, and hence reducing the amount of variability which is possible across such a surface.

These authors therefore concentrated on a means to geomorph between the pre- and post-coarsening sets of triangles in a fragment so that the visual effect of a transition between these large triangulations could be minimised. Rather than the naive fragment tracking approach to geomorphing which was presented in Section 3.1.1, they demonstrated a morphing technique which did not introduce any additional intermediary triangles. By sliding the interior vertices of a fragment to its boundary and then moving the resulting interior lines as necessary, smooth transitions between general polygonal fragments such as those shown in Figure 3.10 could be effected.

3.2.2 Progressive Meshes and extensions

The Progressive Mesh approach which was described in Section 2.3.4 is unsuited to producing variable resolution output. In this section we examine the reason for this and then describe how Hoppe and Xia independently added hierarchical structures to the PM representation to enable true selective refinement to be performed [Hop97b, XV96].

Recall that the PM representation of a model is a base mesh together with a sequence of vertex split operations. Consider the variable resolution mesh generation problem: we wish to iterate through the sequence of vertex splits, performing only those which are necessary for the production of a mesh which meets the input resolution criterion.

As Figure 2.26 shows, there are inherent dependencies between vertex splits which must be taken into consideration – a vertex split $vsplit_i$ cannot proceed unless the v_s, v_l and v_r vertices upon which it operates exist in the current mesh. If one of these vertices should have been introduced by a previous $vsplit_j$ but this $vsplit_j$ was not performed because it was not required by the resolution criterion then it would appear that the desired split, $vsplit_i$, cannot be performed. Hoppe originally circumvented this problem by defin-

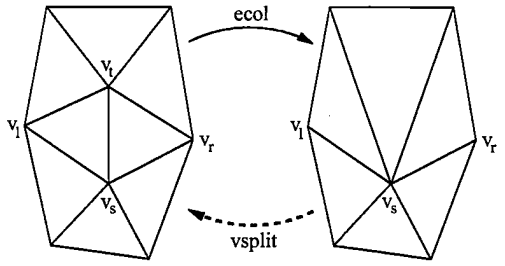


Figure 3.11: Xia's edge collapse and vertex split operations.

ing the *closest living ancestor* of a vertex to be a neighbouring one which does exist in the current mesh and which can be used as a substitute vertex for a *vsplit* [Hop96]. This permits a variable resolution mesh to be produced which is selectively refined according to some criterion with only a single pass through the *vsplit* records. However this construction method can provide no guarantee of the resolution of an output surface.

Xia and Varshney [XV96] demonstrated that such selection of *vsplits* from a sequence which has been designed for producing single-resolution meshes will not necessarily generate the simplest possible selectively refined mesh. They therefore suggested that a *merge tree* (which is actually a forest of binary trees) is required to represent the parent-child relationship between the original vertex which preceded a *vsplit* and the two vertices which are produced by that operation. During the generation of this structure, the *ecol* operation is restricted to merging one vertex with another, as in Figure 3.11, rather than producing an intermediate vertex like Hoppe's *vsplit*. Also, as many independent edge collapses as possible are performed at each level of the merge tree in the style of de Berg and Dobrindt's pairwise non-adjacent vertex removal (Figures 3.12 and 3.13) but without their Delaunay retriangulation. Both of these limitations reduce the quality of the approximating process and its resulting meshes.

Xia and Varshney were able, though, to perform selective refinement without resorting to the "closest living ancestor" fix which Hoppe described. These authors associate with each *ecol* or *vsplit* operation the vertices which surround the region in which that transform operates. These vertices are then regarded as prerequisites which must exist in the mesh before the corresponding operation can be performed. The merge tree permits forwards and backwards traversals (via *vsplits* and *ecols*) through these dependencies to ensure that a complete mesh which satisfies the current resolution criterion can always be generated.

A similar, though less constrained, hierarchy of dependencies was developed independently by Hoppe [Hop97b] as an extension to his PM representa-

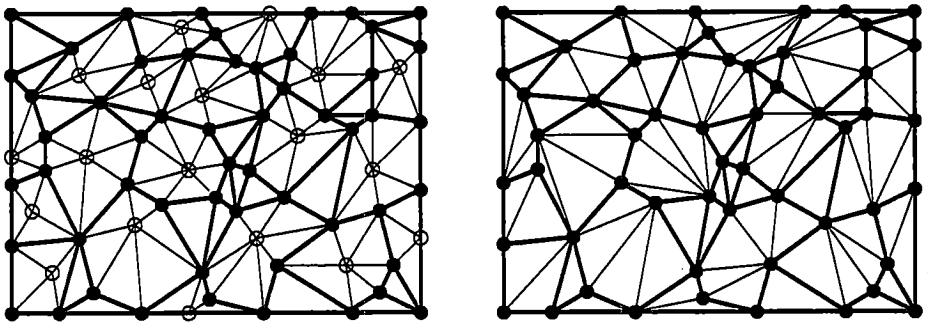


Figure 3.12: Child vertices (circled on left) are merged with their parents to form a coarser LOD (cf. the higher quality retriangulation in Figure 3.8).

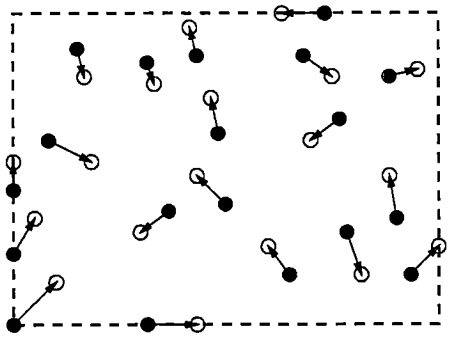


Figure 3.13: A level of the merge tree resulting from the merges of Figure 3.12. Each arrow indicates a parent vertex which could be split to create a child by a *vsplit* operation.

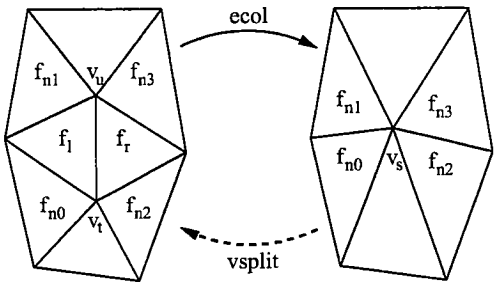


Figure 3.14: Hoppe's view-dependent PM updated *vsplit*/*ecol* notation. The named faces on the left and right are the dependencies of the *ecol* and *vsplit* operations respectively.

tion. Rather than limiting the PM construction process, Hoppe preprocesses a standard PM representation into a vertex hierarchy by linking them in a binary forest like Xia and Varshney's. He also associates the faces in the region of a *vsplit*/*ecol* with the corresponding vertex as dependencies of those operations. Figure 3.14 indicates the faces which Hoppe regards as prerequisites for a *vsplit* or *ecol* operation.

The combination of this vertex hierarchy and the transform prerequisites permitted Hoppe to describe an algorithm which can traverse through the current active set of vertices and use view-dependent resolution criteria to determine whether each vertex should be left as it is, split or collapsed. The resulting set of extracted faces remains a manifold surface which satisfies the current resolution critera.

3.3 HyperTriangulation

In this section we discuss Cignoni's *HyperTriangulation* structure and the associated algorithms which can permit a complete selectively refined terrain surface to be produced for a specific resolution criterion [CPS95].

A HyperTriangulation (HT) explicitly stores the fragments of a terrain surface which are affected by a sequence of point insertions. Each point insertion, which is made according to the Delaunay selector criterion, induces a local retriangulation. This retriangulation is "pasted" onto the current approximation along the edges of the fragment's boundary as a "bubble" above that approximation (Figure 3.15). Thus a HyperTriangulation can store a base mesh together with a history of the resolution changes made to that mesh by a sequence of point insertions.

Two error values are associated with each triangle in an HT to identify the point at which they were generated during the approximation process:

- *birth error*: the global error in the approximation just before the triangle was created;
- *death error*: the global error in the approximation just before the triangle was destroyed (which is zero if the triangle is part of the highest resolution triangulation).

Cignoni *et al* gave algorithms for extracting a surface at a constant or variable resolution. A restriction on the latter surface extraction routine is that the error function against which a variable resolution approximation can be constructed must be of the form:

$$E(p) = f_e(d(v_p, p))$$

where $f_e : \mathbb{R} \rightarrow \mathbb{R}$ is a monotonically non-decreasing function, d is the Euclidean distance in $\mathbb{R}^2$ and v_p is a fixed point (typically the viewpoint).

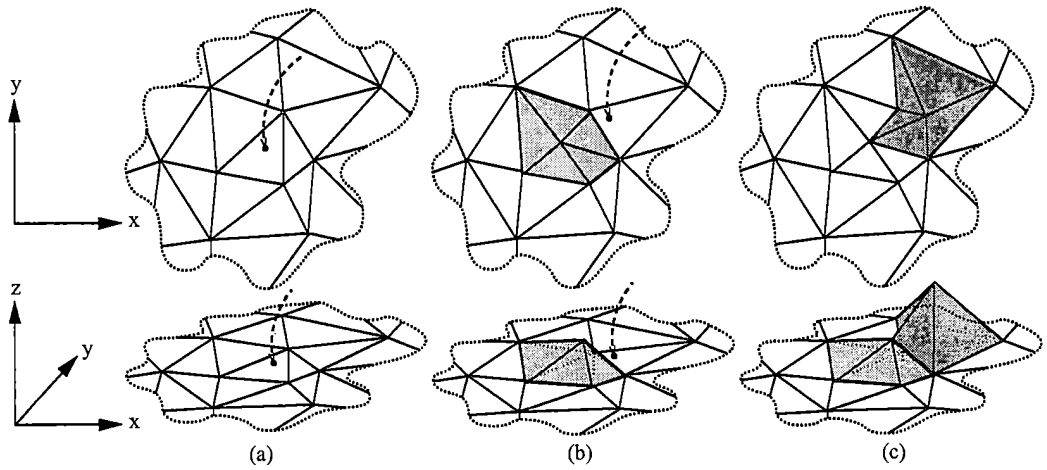


Figure 3.15: Plan (top) and perspective views of two point insertions. A point inserted in mesh (a) adds a "bubble" of facets on top of that mesh (b) and a second point insertion adds a further refinement region (c). From [CPS95].

A standard error function which meets this criterion is an inverted Gaussian distribution, centred at the viewpoint. Applying this error function would produce a selectively refined surface with increased resolution close to the viewer. This kind of variable resolution extraction can be useful for flight simulator applications but it cannot take into account the importance of other regions of a landscape.

3.4 Related work

The recent increased interest in the field of selectively refined meshes which Chapter 1 noted has centred around the work of Puppo and De Floriani. Their work concentrates on the generation of selectively refined terrain surfaces and especially on the theoretical aspects of the structures required to perform this generation. In this section we examine the work of these authors which was developed independently and in parallel with our own [Bro96, Bro97a, Bro97b].

Puppo and De Floriani introduced their *MultiTriangulation (MT)* approach to the problem of selectively refining a terrain surface (or higher dimensional scalar field) in [Pup96], [DPM96] and [DMPB96]. These papers formalised and generalised Cignoni's HyperTriangulation structure by defining a MultiTriangulation as a partially ordered set of fragments of approximations to a terrain. Thus a MultiTriangulation of fragments could, by

implication, be created from a number of approximating methods. A selectively refined surface could be extracted from this partially ordered set by a two-pass algorithm which first identified a superset of the triangles which would lie in the selectively refined surface and then removed the unnecessary triangles from this set. Cignoni's constraint on the output resolution function was eliminated by the selective refinement algorithm presented in [Pup96]. This set of papers dealt exclusively with the theoretical aspects of the partially ordered set from which a MultiTriangulation could be constructed as well as the data structures which could be used to represent an MT.

The first demonstrations of an implementation of a MultiTriangulation were presented in [DMP97b, DMP97c]. These showed that approximations produced according to the Delaunay criterion could be combined, using the algorithms associated with the MultiTriangulation, into a surface which satisfied a resolution function which was proportional to the distance to the viewer, in the same manner as Cignoni's HyperTriangulation algorithm.

De Floriani extended the MultiTriangulation's capabilities to handling fragments of manifold approximations in [DMP97b]. This paper provided an alternative definition of the fragments from which an MT was constructed which eliminated the original two-dimensional domain test. This permitted Puppo to use the MT as a basis for discussing the merits of a range of approximation methods in [PS97]. Furthermore, an image of a selectively refined three-dimensional model was presented by De Floriani [DMP97a] although no quantitative results regarding this mesh were given. The resolution function against which this mesh was generated was the same as that used for the terrain examples, i.e. the distance to the viewer determined the resolution of the mesh.

This last paper by De Floriani *et al* also presented an algorithm which permitted an existing mesh to be modified to satisfy new resolution criteria. This algorithm operated by refining an existing surface without considering the possibility that resolution requirements can decrease. Hence it will not necessarily produce the surface with the minimal set of facets which satisfies given resolution requirements. The potential for geomorphing between existing and modified meshes was not considered by De Floriani.

3.5 Review of approximation schemes

The following table presents a comparison of the approximation methods and variable resolution mesh generators which have been discussed in this chapter and the previous one. They have been classified according to the class of model which they can take as input (height field, manifold or non-manifold); the ordering within these categories is consistent with that in which they were described.

The columns of the table indicate:

- whether the corresponding approximation method operates by repeated local modifications of the input mesh;
- whether the approximation method can vary the topology of the input mesh;
- whether the approximation method can provide some form of variable resolution output. The elements in parenthesis in this column indicate: de Berg and Dobrindt's variable resolution extraction routine does not necessarily meet the input criterion; Cignoni's HyperTriangulation is limited to a specific form of input resolution criterion; and producing a variable resolution output from the simplification envelopes approach requires a complex resolution specifier.
- whether geomorphing between surfaces generated at different resolutions is possible;
- whether the output of the method can be treated as a set of simplices;
- whether the method can be incorporated into our SMR framework. The edge collapsing method of Garland and Heckbert is marked in parenthesis here since we shall indicate in Chapter 8 how it could be modified to work within the framework.

A complementary table which covers a similar set of approximation methods but which concentrates on detailing the output characteristics of single-resolution methods was provided by Puppo [PS97].

3.6 Resolution criteria

If a model is to be represented by a selectively refined approximation then the resolution criteria which this approximation is required to satisfy must be specified. These criteria can be written in terms of either the space in which the model is defined or the screen on which the approximation is to be rendered. We refer to these forms of resolution criteria as *object-space* and *screen-space* respectively. This section classifies the resolution criteria which have been used in previous work according to these categories.

Open Inventor and VRML [Wer94, CB97] use a simple criterion to select which of a given set of Levels Of Detail is appropriate for a particular scene. The criterion used by these environments has converged to an object-space measure which depends on the distance between the centre of an object and the viewer (see Sewell [Sew96] for details of the variations between Inventor and VRML).

Input	Method	Page reviewed	Local modifications	Topology-varying	Variable resolution	Geomorphing	Simplicial output	SMR-compatible
Height field	Quadtrees [CMR90, FZPM93, Sam84]	16			•	•	•	
	Right triangles [LKR+96]	18			•	•	•	•
	Wavelet/quadtree [GSG96]	18			•	•		
	Delaunay [FL79]	20	•				•	•
	HTIN [DP92a]	22,46	•		•		•	•
	Decimation [SR94]	23	•				•	•
	Ternary [DFNP82]	24,45	•		•		•	•
	Quarternary [De 87]	24	•				•	•
	Pyramidal [Sca90, De 89]	25,40					•	•
	Cartographic coherence [SP92]	26	•		•		•	•
	Richline [Dou86]	26						
	Continuous terrain LOD [FEKR90]	44	•		•	•	•	•
	Star polygon graph [dD95]	47	•		(•)		•	•
	Temporal continuity [CL96]	49	•		•	•	•	•
	HyperTriangulation [CPS95]	53	•		(•)		•	•
	MultiTriangulation (1996) [Pup96, DPM96]	54	•		•		•	•
Manifold	Refining [FHMB84]	29	•				•	•
	Decimation [SZL92, KLS96, BS96]	29	•				•	•
	Simplification envelopes [CVM+96]	30			(•)		•	
	Tolerance volume [Gué96]	30					•	
	Coplanar decimation [HH93, DZ91]	31	•				•	•
	Re-tiling [Tur92]	31				•	•	
	Energy optimisation [HDD+93]	33	•				•	•
	Progressive Meshes [Hop97b, XV96]	33,50	•		•	•	•	•
	Wavelets [LDW94, EDD+95]	34				•		
	MultiTriangulation (1997) [DMP97b, PS97]	55	•		•		•	•
Non-manifold	Vertex clustering [RB93, LT97]	36	•	•		•	•	
	Hierarchical Dynamic Simplification [LE97]	37	•	•	•	•	•	
	Edge collapsing [RR96, GH97]	37	•	•			•	(•)
	Progressive Simplicial Complexes [PH97]	37	•	•			•	

Table 3.1: Pertinent attributes of the methods reviewed in Chapters 2 and 3.

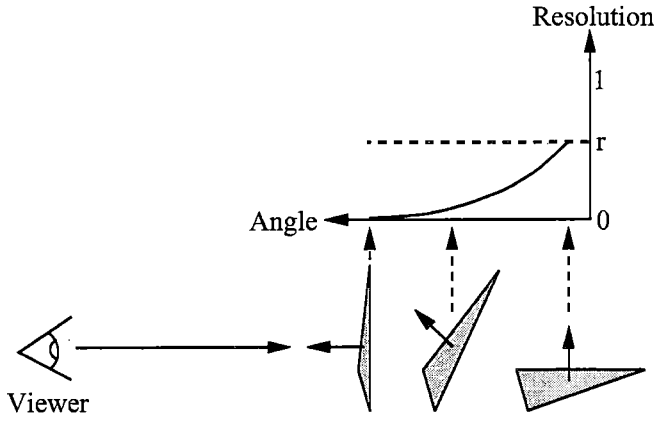


Figure 3.16: The graph at the top of the diagram depicts qualitatively how the minimum object-space resolution which we require in a particular triangle can be allowed to vary with respect to the angle between that triangle's normal and the viewing direction. Specifically, when the triangle is perpendicular to the viewer its resolution is not significant but when it is parallel to the view direction its associated object-space resolution (r) must be compared with the minimum acceptable screen-space resolution.

Cignoni, Puppo and De Floriani also use an object-space measure [CPS95, Pup96, DPM96]. The aim of their measure, though, is to identify the maximum permissible error in each facet of a triangulated mesh. These authors use a function which linearly decreases with distance from the current viewpoint and can therefore be specified as a tolerance for each point p of a mesh:

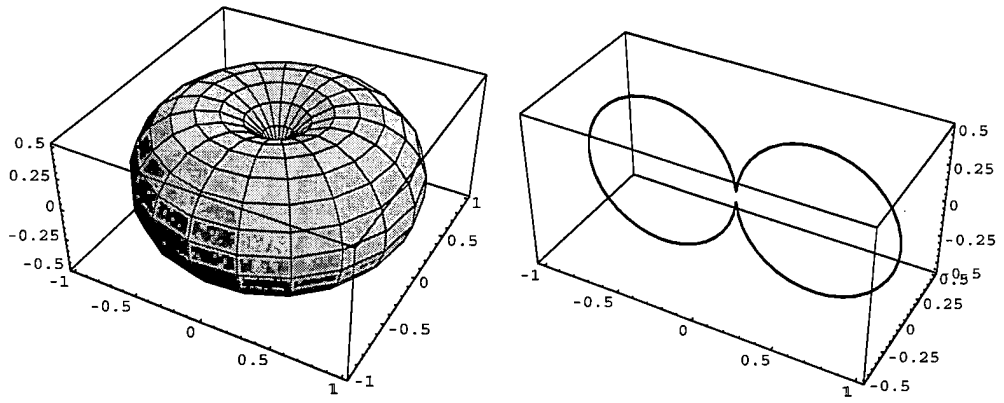
$$\tau(p) = K \times d(p, v)$$

where v is the viewpoint, K is a positive constant specified by the user, and d denotes the Euclidean distance [DMP97c].

Lindstrom *et al* and Hoppe [LKR+96, Hop97b] advocate an alternative, view-dependent, resolution criterion. These authors project their object-space approximation errors with respect to the current view direction to assess the impact of these errors in the viewing plane. Figure 3.16 illustrates such a criterion.

Lindstrom's error projection can be reduced to $\delta||(0,0,1) \times \vec{v}||$ where δ is the object-space error value at a mesh point and $\vec{v}$ is a unit vector along the viewing direction. Figure 3.17 shows the value of this projection as a function of view direction – the resulting graph is a "bialy" [LKR+96], or solid torus, of radius δ.

Hoppe supplements this projection formula with terms which capture



(a) Lindstrom's error projection as a function of viewing angle for $\delta = 1$.

(b) A contour through the graph in (a) at $y = 0$.

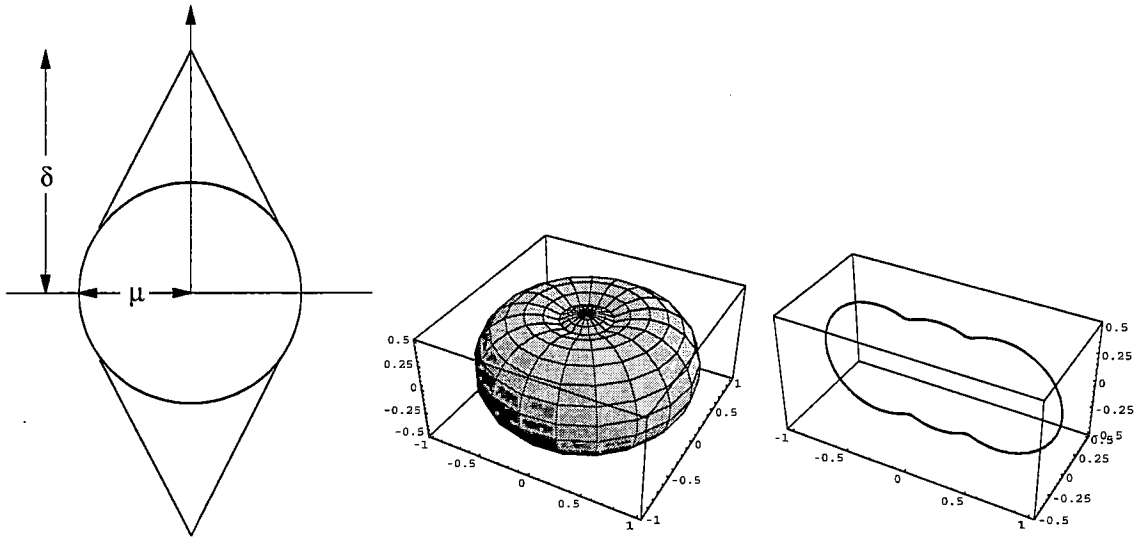
Figure 3.17: Lindstrom's error projection.

dependencies due to both surface normals and also edges which represent discontinuities in the mesh. He specifies a *deviation space* $D_{\vec{n}}(\mu, \delta)$ which is parameterised with respect to a normal $\vec{n}$ and whose projected radius along a direction $\vec{v}$ is $\max(\mu, \delta||\vec{n} \times \vec{v}||)$. The terms μ and δ are illustrated in Figure 3.18a. These parameters are calculated for each vertex in a PM representation such that each vertex's deviation space encloses the error vectors associated with sampled points on the facets in the neighbourhood of that vertex.

The graph of the projected radius of Hoppe's deviation space for a surface normal of $(0, 0, 1)$ is shown in Figures 3.18(b) and (c). As can be seen, this is the same graph as for Lindstrom's projection, but unioned with a sphere of radius μ. The result of using this as a screen-space resolution criterion is that the resolution of a region which is perpendicular to the viewer is guaranteed to be above a given minimum, whereas Lindstrom's method would permit this region to be displayed at the lowest possible resolution.

Hoppe augments this screen-space similarity check with frustum culling and back-face culling tests which are described below.

View frustum A bounding sphere is associated with each vertex in a PM hierarchy such that the sphere contains all of the faces in the neighbourhoods of that vertex and its descendants. If a vertex's bounding sphere lies outside the view frustum then its surrounding region does not need to be refined.



(a) Cross-section of the deviation space $D_{\vec{n}}(\mu, \delta)$.

(b) The extent of the screen-space projection of this deviation space as a function of viewing angle for $\mu = 0.5$ and $\delta = 1$.

(c) A contour through (b) at $y = 0$.

Figure 3.18: Hoppe's screen-space resolution criterion (from [Hop97b]).

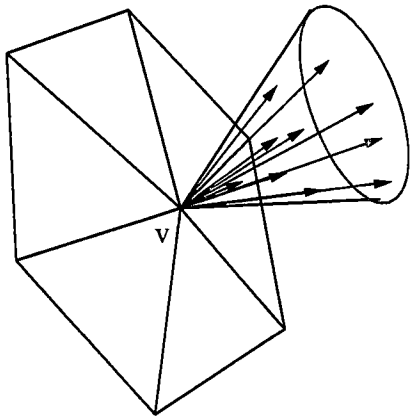


Figure 3.19: The normals of the facets in the original mesh which correspond to the facets around v can be bounded by a cone of normals.

Surface orientation The normals of the facets in the original model which are approximated by the region around a PM vertex and its descendants are contained in a *cone of normals*. This cone is associated with that vertex in the PM hierarchy (Figure 3.19). If the orientation of vertex v's cone of normals is such that we can guarantee that all of the faces in that region of the mesh are back-facing then v's region does not need to be refined.

3.7 Summary

This chapter and the previous one have established that there is a wide variety of approximation methods which take specific classes of models as input and produce single-resolution meshes with varying characteristics. We have also described how some of the approximation methods can produce variable resolution meshes directly and others can be combined with additional structures and algorithms to permit the generation of such meshes. These observations underlined the fact that, regardless of how deeply an existing approximation method appears embedded in its associated variable resolution mesh generator, it is possible to distinguish between these processes.

This chapter has also established that, prior to the recent work of Puppo and De Floriani, there was no method for incorporating existing approximation methods into a selective refinement technique. Even when the contemporary MultiTriangulation approach is considered, there remains a need for a selective refinement framework which can handle a range of approximation methods operating on both scalar fields and manifold meshes and which provides the capability to geomorph between meshes. This is the aim of our SMR framework which is introduced in the next chapter.

Chapter 4

The Selective Mesh Refinement Framework

The previous two chapters have introduced a wide range of scalar field and manifold approximating methods, many of which operate by performing repeated local modifications on an existing approximation. Of particular note is Cignoni's HyperTriangulation approach (Section 3.3) which retains the local modifications associated with a particular terrain approximation technique in order to permit a selectively refined surface to be generated. In this chapter we introduce our SMR framework which extends this work in a number of directions in order to provide a flexible approach to the generation and geometric morphing of selectively refined simplicial meshes in n-dimensions.

We first detail the specific achievements of our SMR framework with reference to previous work in Section 4.1. We also consider how the SMR framework relates to other scene optimisation techniques in Section 4.2. Section 4.3 covers terminology which we shall use in the remainder of this dissertation. This is followed by a description of our resolution-based model representation in Section 4.4 and the structures with which this representation can be implemented in Section 4.5. The manner in which we specify the resolution which is required of an output mesh is detailed in Section 4.6.

4.1 SMR framework features

The components of our SMR framework are visualised in Figure 4.1 and the novel features of the framework are introduced below.

- A resolution-based representation of a model is constructed during a pre-processing step. This means that some simplicial mesh-based approximation process is applied to the model and the local modifications which this process generates are stored as individual simplicial fragments in a *Continuous Resolution Model Representation (CRMR).*

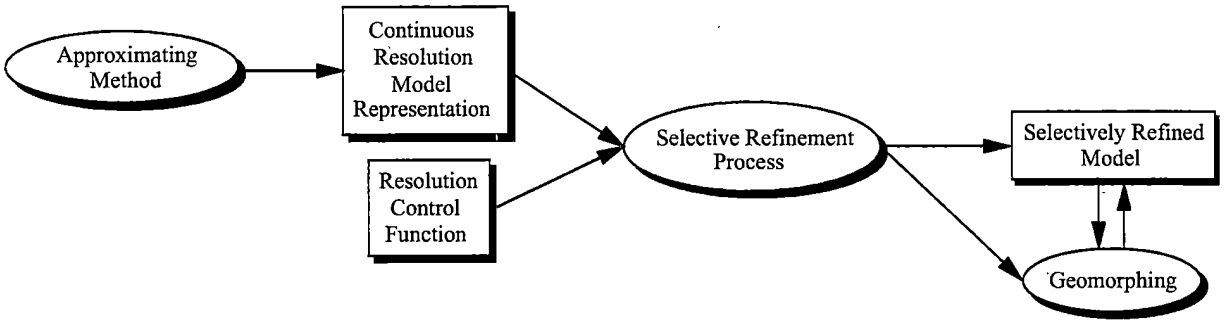


Figure 4.1: Selective Mesh Refinement framework

These fragments are ordered such that they are refinements of a base mesh. This explicit separation of the approximation process from the production of a selectively refined mesh is the basis for the flexibility of the framework.

Our CRMR takes the concept of Cignoni's HyperTriangulation, which was restricted to terrain surfaces and a Delaunay approximation process, to its logical conclusion by permitting the fragments contained in a CRMR to be generated by a range of scalar field and manifold approximating techniques. The concept of our CRMR is also similar to the most recent papers regarding Puppo and De Floriani's MultiTriangulation (Section 3.4; [DMP97b, PS97]) in which they define an MT such that it can handle fragments of approximations to three-dimensional manifold meshes.

Our CRMR is unique in permitting certain topological operations to be performed by the fragments generated by the underlying approximation process. Specifically, fragments can introduce either holes in a mesh or additional simplicial meshes. This latter option means that a CRMR can actually represent the resolution information associated with a set of simplicial meshes.

The two abstract structures, which we refer to as the *Hypermesh* and *DAG*, with which we represent a CRMR are also novel. The Hypermesh maintains the adjacency information of simplices in a CRMR's fragments in both the spatial and resolution senses. The DAG contains an abstraction of this information which relates to the overlapping nature of the fragments in the spatial dimension and also their ordering with respect to resolution.

Our Hypermesh is similar in concept to Cignoni's HyperTriangulation

but differs in retaining the boundary edges of each fragment as distinct versions of their predecessors (unlike the "bubbles" of Cignoni's fragments in Figure 3.15). The implementation which we present for our Hypermesh in $2\frac{1}{2}$- and 3-dimensions reflects this difference as a novel structure which extends Guibas and Stolfi's manifold-representing *quadedge* data structure [GS85].

- The other input to the selective refinement process which is shown in Figure 4.1 is a *Resolution Control Function (RCF).* This is our collective term for the resolution criteria which an output selectively refined mesh must satisfy. Section 3.6 discussed various existing resolution criteria and drew the distinction between object-space and screen-space criteria. Cignoni, Puppo and De Floriani concentrated exclusively on generating selectively refined meshes with respect to object-space resolution criteria and their results to date have used only a simple object-space criterion which depends on the distance to the viewpoint. In contrast, Hoppe advocated and demonstrated only screen-space resolution criteria.

  Our RCF provides a consistent interface which permits these two kinds of criteria to be combined in the specification of a particular scene.

- The selective refinement process which is central to Figure 4.1 is the task of producing a selectively refined mesh from the fragments contained in a CRMR such that the mesh satisfies the given Resolution Control Function. We refer to this task as the *SMR process.*

  We present two algorithms which can perform the SMR process – the *minimal surface* and *reduced extraction* algorithms. Both algorithms can generate a selectively refined mesh which satisfies any resolution requirement presented by the RCF and hence we have eliminated the non-decreasing constraint which was essential to Cignoni's output resolution function (Section 3.3). This resolution constraint was also eliminated by Puppo's MultiTriangulation algorithm [Pup96] and Hoppe's view-dependent Progressive Mesh paper [Hop97b].

  Our algorithms are novel in that they separate the task of identifying the fragments which are necessary for the generation of a selectively refined mesh from that of generating a mesh using these fragments. This distinction has two major advantages:

  1. the set of fragments from which an existing selectively refined mesh was constructed can potentially be used for future operations such as geomorphing;
  2. our traversal over a set of fragments which generates a selectively refined mesh can proceed in the nature of an advancing front algo-

rithm. This means that the adjacencies of the extracted simplices can be maintained and also that the regions of highest resolution, and therefore greatest significance, can be extracted first. Hence if the traversal process is being used to render a mesh and this rendering process has to be aborted by a frame-time constraint then the displayed mesh will not necessarily be complete but it will contain the most significant features of the underlying model.

In contrast, Puppo's algorithm combines traversals of fragment and triangle queues and Hoppe's is only suitable for generating a selectively refined Progressive Mesh.

The additional novelty of the reduced extraction algorithm which we present is that it attempts to reduce the cost of the first phase of our selective refinement process (that of identifying the fragments which are necessary for the output mesh).

- We also present a geomorphing algorithm which can smoothly morph between two selectively refined meshes and which may be invoked due to an alteration of the resolution criteria. This algorithm is independent of the approximation method which generated the given CRMR because it can reduce the regions which must be morphed to the locality of individual fragments in the CRMR. We assume that external routines exist which permit such localised morphing.

  This geomorphing ability is unique among fragment-based selective refinement systems (i.e. the HyperTriangulation and MultiTriangulation approaches) and is made possible by our selective refinement algorithms not only generating a mesh but also the set of fragments from which that mesh was constructed. Hoppe presented a geomorphing algorithm for his view-dependent Progressive Mesh representation [Hop97b] but this is specific to the PM approach.

- Finally, our results chapter (Chapter 7) provides the first quantitative examination of selective mesh refinement applied to various terrain and three-dimensional manifold datasets and also with respect to a range of resolution criteria.

4.2 The SMR context

The intended role of the SMR process in a rendering system is indicated in Figure 4.2. Every object in a virtual environment is held as a CRMR. The initial scene optimisation step, as in Sewell [Sew96], is a fast visibility check to determine which models lie inside the view frustum and hence require

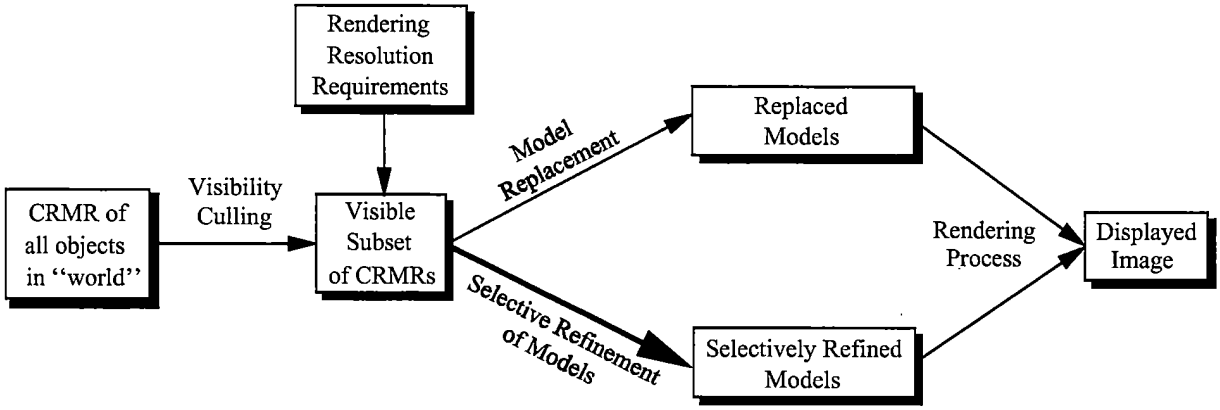


Figure 4.2: The SMR process (bold arrow) as a component of a rendering system.

further processing. Those which are insignificant in terms of their projected screen area can be replaced by rough approximations such as bounding volume representations, etc. The remaining CRMRs can be selectively refined according to the current resolution requirements before being passed to the rendering process.

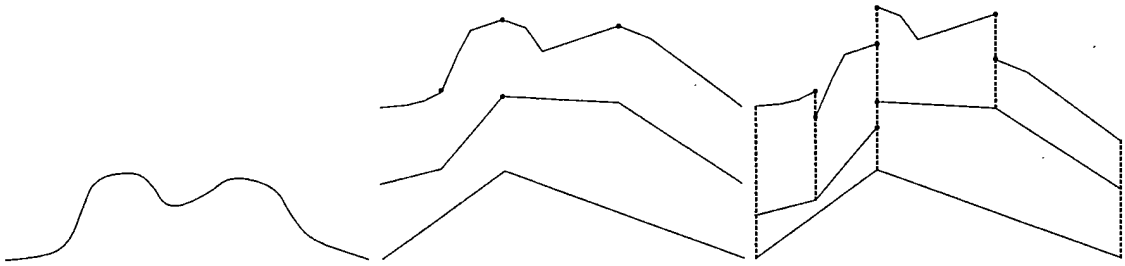
Note that when a selectively refined mesh is rendered in a scene, it is dependent on the current resolution requirements which themselves may depend on the current view frustum. Hence geomorphing may be required between any displayed selectively refined meshes and their updated versions. This feedback loop is not depicted in Figure 4.2.

4.3 Terminology

4.3.1 Dimensions

When the input to the SMR framework is from a terrain approximation method, we say that $2\frac{1}{2}$-*dimensional* selective refinement is being performed. If the input is from a three-dimensional manifold approximation process then this is denoted, naturally, as the *3-dimensional* case. Since these are the typical instances for which selective mesh refinement will be invoked in rendering applications, we concentrate on describing the $2\frac{1}{2}$- and 3-dimensional cases.

The SMR framework is extensible, though, to other dimensions. The $1\frac{1}{2}$-dimensional case, for example, would be where a function $y = f(x)$, $f : \mathbb{R} \rightarrow \mathbb{R}$ was approximated by a series of line segments and these segments were then input as a CRMR to the SMR framework (Figure 4.3). The $3\frac{1}{2}$-



(a) A function $y = f(x)$.

(b) Three single-resolution approximations to the function in (a).

(c) The fragments which can be extracted from these approximations, shown linked at their corresponding points. A selectively refined approximation to (a) could be extracted from this information.

Figure 4.3: Example of $1\frac{1}{2}$-dimensional input to the SMR framework.

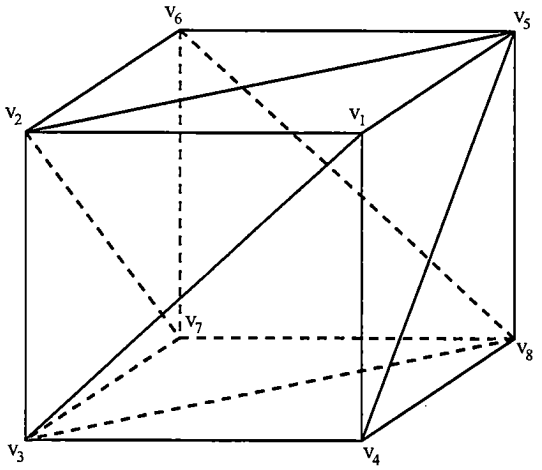
dimensional case would involve selectively refining a volumetric visualisation of a function $w = g(x, y, z)$, $g : \mathbb{R}^3 \rightarrow \mathbb{R}$.

4.3.2 Simplicial mesh notation

To assist with our description of SMR in three-dimensional Euclidean space, we require a means of describing 2-simplicial meshes, i.e. two-dimensional surfaces which satisfy the properties identified in Section 2.1.2. We specify a 2-simplicial mesh as a tuple $\mathcal{M} = (V, F, A)$ where

- $V = \{\vec{v}_1, \ldots, \vec{v}_n\}$ is the set of position vectors of the vertices of the mesh;
- F is a set of ordered triples of vertex indices corresponding to the faces in the mesh. These faces are of course the 2-simplices (triangles) of a 2-simplicial mesh and it is in this form that the notation can be generalised to higher dimensions – the set F could contain the n-simplices of an n-simplicial mesh;
- A is the attribute information associated with the mesh. A may include discrete and scalar attributes such as material identifiers, normals, etc, as Hoppe described in [Hop96]. If we are dealing with $2\frac{1}{2}$-dimensional SMR, one of the attributes contained in A will be the elevation values associated with the 2-dimensional position vectors in V.

Figure 4.4 shows possible V and F sets for a simplicial mesh representing the surface of a cube.



$$V = \{\ \vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4, \vec{v}_5, \vec{v}_6, \vec{v}_7, \vec{v}_8\ \}$$
$$F = \{\ \{1,2,3\}, \{1,3,4\}, \{5,8,6\}, \{6,8,7\}, \{5,1,4\}, \{5,4,8\}, \{6,2,5\}, \{2,1,5\}, \{2,6,7\}, \{3,2,7\}, \{3,7,8\}, \{3,8,4\}\ \}$$

Figure 4.4: The triangulated surface of a cube and the corresponding V and F sets in our notation of this simplicial mesh.

We define the *boundary* of a simplicial mesh, $B(\mathcal{M})$ to be the set of directed edges of triangles in $\mathcal{M}$ such that each edge $\vec{e} \in B(\mathcal{M})$ lies on only one triangle in the mesh and is directed such that this triangle lies to its left (where the orientation of the triangle is defined by the ordering of its vertices in F).

4.3.3 Edge notation

We use the $\vec{e}.Sym$ notation of Guibas and Stolfi's *quadedge* data structure [GS85] to indicate the mirror image of a directed edge $\vec{e}$ (i.e. an edge with the opposite direction to $\vec{e}$). We also use the quadedge notation to denote the previous and next edges around a triangle lying to the left of an edge $\vec{e}$ as $\vec{e}.Lprev$ and $\vec{e}.Lnext$ respectively (Figure 4.5).

4.4 Continuous Resolution Model Representation

The basis of our selective refinement process is that a selectively refined mesh can be generated by combining fragments of triangulated approximations to an object (if we assume that we are working in the $2\frac{1}{2}$- and 3-dimensional cases), where these approximations have been generated at a range of resolutions. We require a novel data structure to permit our SMR algorithms to traverse between fragments' facets which are adjacent in either the spatial or resolution sense. This is the purpose of a CRMR.

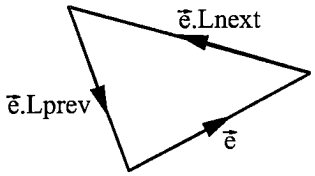


Figure 4.5: The *Lprev* and *Lnext* operators.

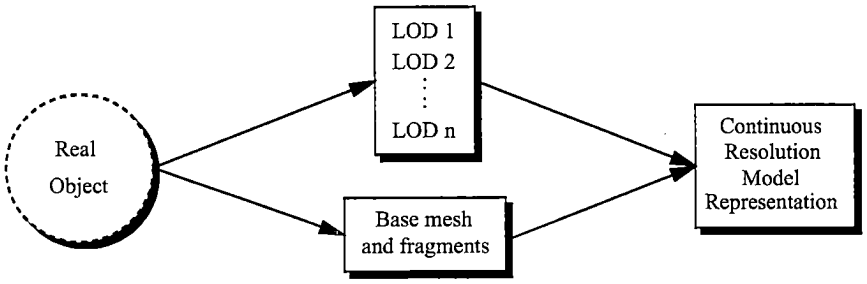


Figure 4.6: Pre-processing CRMR creation step

We follow the concept introduced by Cignoni of representing a model as a base, coarse, approximating mesh together with a history of the refinements which are required to improve the base mesh with respect to some resolution metric (Section 3.3). Our Continuous Resolution Model Representation is more general, though, than previous such fragment-based representations. This means that we can transform the output of many existing approximation methods into a CRMR (the table on page 57 indicates which of the reviewed methods are amenable to generating a CRMR). Specifically, we can preprocess the output of any height field or manifold approximating method which produces simplicial meshes and which operates by making repeated local modifications to a mesh. This latter requirement ensures that there is a degree of edge correspondence between the resulting fragments and this increases the potential for variability in the output selectively refined meshes.

The off-line pre-processing step in which a CRMR of an object can be produced is depicted in Figure 4.6. This shows that the kind of single-resolution approximation method with which we are concerned can produce either a set of single-resolution approximations (LODs), or a base mesh together with a set of incremental changes to that mesh. Examples of the former include pyramidal methods (Section 3.1.1), and notable examples of the latter (which we shall demonstrate within the SMR framework later) include the Delaunay triangulation and Progressive Mesh approaches (Sections 2.2.2.1 and 3.2.2).

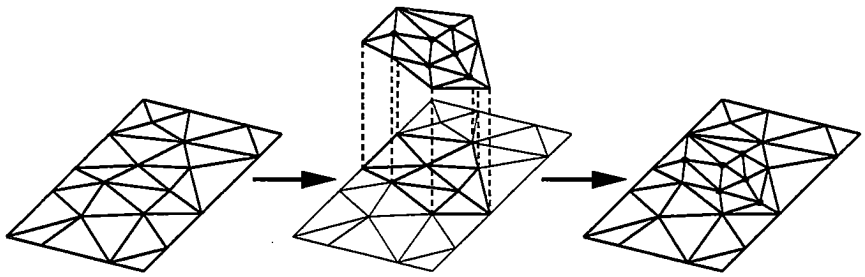


Figure 4.7: A refinement operation transforming one 2-simplicial mesh into another. In the process, two of the original vertices are removed and six others are added.

From all of these methods we can generate the information which is stored in a CRMR, i.e. a base mesh and a set of modifications of that mesh.

We term these modifications of a base mesh *refinement operations*. Each refinement operation is characterised explicitly by the set of triangles which it replaces, together with the simplicial mesh or meshes which it introduces. The diagram in Figure 4.7 shows a simple refinement operation transforming one simplicial mesh into another.

We can therefore define a CRMR as a base mesh, $\mathcal{M}^0 = (V^0, F^0, A^0)$ together with a sequence of refinement operations, $\{R^1, \ldots, R^m\}$. The refinement operations are a set of transformations which can produce a sequence of single-resolution meshes $\{\mathcal{M}^1, \ldots, \mathcal{M}^m\}$ from the base mesh $\mathcal{M}^0$, such that

$$\mathcal{M}^i = R^i(\mathcal{M}^{i-1}), 1 \leq i \leq m$$

We require that the ordering of refinement operations $\{R^1, \ldots, R^m\}$ is such that the resolution of the meshes $\mathcal{M}^0, \ldots, \mathcal{M}^m$ is increasing with respect to some resolution metric, r, which operates on these meshes. We adopt the convention that the range of r is $[0, 1]$ and is such that $r(\mathcal{M}^0) = 0$ and $r(\mathcal{M}^m) = 1$.

As we have seen, the standard resolution metric for $2\frac{1}{2}$-dimensional surfaces is the maximum vertical displacement between an approximating surface and the original dataset. Also, the Hausdorff distance is one measure of resolution for three-dimensional approximations (Section 2.3.2). Our resolution metric, r, can be generated trivially by these standard measures. For example, the Hausdorff distance could be converted using:

$$r(\mathcal{M}^i) = 1 - \frac{\mathcal{H}(\mathcal{M}^i, \mathcal{M}^m)}{\mathcal{H}(\mathcal{M}^0, \mathcal{M}^m)} : 0 \leq i \leq m$$

Refinement operations are defined formally in Section 4.4.1 and the resolution attributes which we associate with each operation are discussed in

Section 4.4.2. The conversions which may be required to produce a CRMR from the output of existing approximation methods are presented in Section 4.4.3.

4.4.1 Refinement operations

We proceed to formalise our previous description of a refinement operation as a transformation of one mesh into another which is of a higher resolution.

A refinement operation $R^i(\mathcal{M}^{i-1}) = \mathcal{M}^i$ is characterised in the most generic form possible, which is to explicitly store the portion of the simplicial mesh in $\mathcal{M}^{i-1}$ which R^i affects, together with the simplicial mesh with which that area of the mesh is replaced. Thus we associate with R^i the following attributes (for $1 \leq i \leq m$):

- $PreF(R^i) = F^{i-1} \backslash F^i$ is the set of triangles which existed in $\mathcal{M}^{i-1}$ and which were replaced by R^i. $PreF(R^i)$ is referred to as the *pre-operation* set.

- $V(R^i) = V^i \backslash V^{i-1}$ is the set of position vectors of the vertices which did not exist in $\mathcal{M}^{i-1}$ and which were introduced by R^i. Note that although our framework can handle refinement operations whose pre-operation set of triangles contains vertices which do not exist in $\mathcal{M}^i$, we simplify our notation by not representing this kind of operation.

- $F(R^i) = F^i \backslash F^{i-1}$ is the set of triangles which were introduced by R^i: $F(R^i)$ is referred to as the *post-operation* set. $F(R^i)$ is a set of triples of vertex indices and we assume that these indices are unique over the sets $V(R^0) \ldots V(R^m)$ in order to permit set comparisons between the $F(R^i)$'s;

- $A(R^i)$ is the set of attributes associated with $V(R^i)$ and $F(R^i)$.

For completeness, we let R^0 be a pseudo refinement operation representing $\mathcal{M}^0$, where $PreF(R^0) = \emptyset$, $V(R^0) = V^0$, $F(R^0) = F^0$ and $A(R^0) = A^0$. Hence the approximating meshes $\mathcal{M}^0 \ldots \mathcal{M}^m$ are completely represented by $R^0 \ldots R^m$.

Figure 4.8 shows a sequence of refinement operations in diagrammatic form. We describe the space in which the $2\frac{1}{2}$-dimensional fragments are viewed in this diagram as $2\frac{1}{2}$-dimensional *resolution space*. We also use this abstract form to represent 3-dimensional resolution spaces in which the base mesh may be a simplicial mesh without a boundary even though we still visualise it as a planar bounded region.

To conclude the specification of refinement operations, we also require that they are both *complete* and *minimal*, where we define:

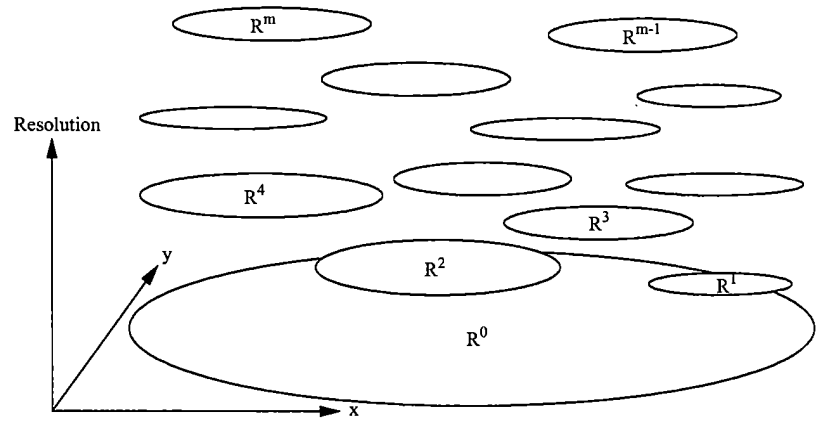


Figure 4.8: A sequence of $2\frac{1}{2}$-dimensional refinement operations represented by an abstract form of the fragments which they introduce.

- R^i is *complete* iff the boundary of the set of triangles which this operation affects is equal to, or is contained within, the boundary of its replacement triangles. This holds for all pre-operation edges except those which lie on the boundary of $\mathcal{M}^{i-1}$ – each of these boundary edges may be replaced by a set of edges to permit refinement along manifold boundaries.

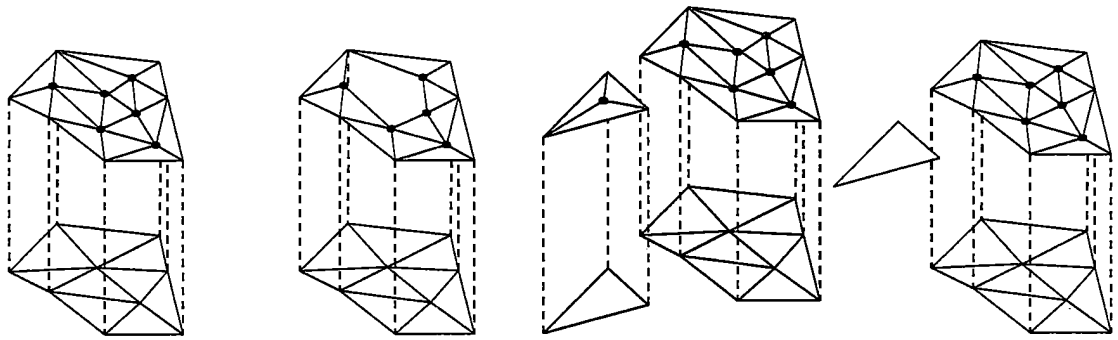
  Figures 4.9 and 4.10 demonstrate (for non-boundary refinement operations) that this definition permits a range of modifications to be specified, notably the topology-varying ones of Figures 4.9(b) and (d).

  The refinement operation depicted in Figure 4.9d is particularly significant because this implies that an operation can "spawn" a separate simplicial mesh which itself may be the subject of further refinement operations. Thus our CRMR and SMR framework can handle models which are constructed from one or more simplicial meshes, although we continue to refer to the output of our SMR process simply as a "mesh".

  The ability to handle a set of simplicial meshes is necessary if the model represented by a CRMR is a set of unconnected manifolds, which is the case for many VRML models. Chapter 7 contains an example of a model which utilises this ability. Such unconnected manifolds may also be generated by approximation methods such as Garland and Heckbert's topology-modifying method (Section 2.4.2).

  Formally, this definition can be phrased as:

$$R^i \text{ is complete } \iff \forall \vec{e} \in B(PreF(R^i)) \backslash B(\mathcal{M}^{i-1}), \vec{e} \in B(F(R^i))$$



(a) Contiguous set of triangles replaced by another.

(b) Refinement operation which introduces a hole.

(c) Disjoint simplicial meshes which are part of one refinement operation.

(d) Refinement operation which refines an existing region and also introduces a new simplicial mesh.

Figure 4.9: Complete refinement operations

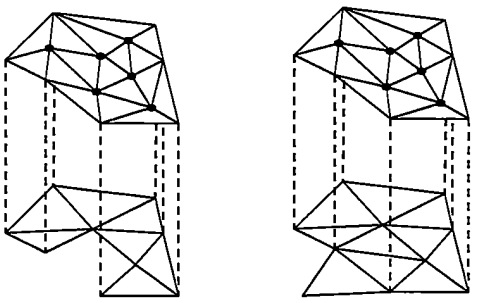
- R^i is *minimal* $\iff \nexists S_1 \subset PreF(R^i), S_2 \subset F(R^i)$ s.t. $B(S_1) = B(S_2)$, i.e. there is no chain of edges in $PreF(R^i)$ which also occurs in $F(R^i)$ and hence there is no subset of triangles in $PreF(R^i)$ which can be completely replaced by a subset of $F(R^i)$. Thus we ensure that refinement operations are specified as locally as possible which increases the potential variation in the selectively refined meshes which can be produced by these refinement operations. Figure 4.9c represents a non-minimal refinement operation.

Successive refinement operations will re-use some of the post-operation facets of other refinement operations in their pre-operation set. To avoid duplication, therefore, we need only store the post-operation facets of each refinement operation. Each post-operation facet group, $F(R^i)$, is what we refer to as a *refinement fragment*, or simply a *fragment*.

4.4.2 Resolution attributes

One of the tasks of the attribute set, $A(R^i)$, is to associate resolution information with a refinement operation. We adapt the *birth error* and *death error* terms of Cignoni *et al* (Section 3.3) to conform to our resolution-based notation.

We say that a refinement operation R^i has a *birth resolution*, $birthres^i$, which is the maximum resolution in the mesh $\mathcal{M}^{i-1}$ just before R^i was per-



(a) Incomplete pre-refinement set of triangles.

(b) Incomplete post-refinement set of triangles.

Figure 4.10: Incomplete refinement operations.

formed (we assume $birthres^0 = 0$); and each triangle t in $F(R^i)$ has a *death resolution*, $deathres(t)$, which is the maximum resolution in the overall mesh just before t was replaced by another refinement operation (or 1 if t is part of $\mathcal{M}^m$).

The previous requirement for the resolution of the refinement operations to be monotonically increasing implies that $birthres^i > birthres^{i-1}$ for $0 < i \leq m$.

The diagrams such as Figure 4.8 which contain an abstract representation of refinement operation's fragments can be regarded as visualising these fragments at their birth resolutions.

4.4.3 CRMR construction

In this section we consider how refinement operation information can be extracted from the output of the simplicial mesh-producing approximation methods described in the previous chapters.

These methods can be categorised as:

- those which produce multiple LODs which may have to be split into fragments before they can be incorporated into a CRMR;
- those which are directly compatible with a CRMR (i.e. they operate by producing fragments corresponding to local modifications);
- those which may require some reordering of the fragments which they produce, and;

- those whose fragments require adaptation to comply with our specification of refinement operations.

These classes of methods are considered in turn below.

4.4.3.1 Multiple LOD generators

Methods such as De Floriani's and Scarlatos' pyramidal schemes (Section 3.1.1) generate Levels Of Detail which do not necessarily have any edge coherency, i.e. there may be no edges which persist between LODs. Such pyramidal methods can be converted to a refinement operation representation trivially by holding each LOD as the post-operation simplicial mesh which a refinement operation produces. If the LODs can be split into more localised fragments, though, then this will permit the production of variable resolution meshes. The potential for localisation can be checked using the refinement operation minimality test described above. If multiple minimal refinement operations can be identified within one LOD then a unique birth resolution must be associated with each of these operations.

4.4.3.2 Directly compatible methods

The majority of the reviewed approximation methods are refining methods. That is, they can be regarded as producing a coarse base mesh together with a sequence of refining fragments which can directly satisfy our refinement operation completeness and minimality conditions. These are Fowler and Little's Delaunay-generating method (page 20), the ternary triangulation (page 24), Faugeras' manifold refining method (page 29), De Berg and Dobrindt's method (page 47), Cignoni's HyperTriangulation (page 53) and Puppo's MultiTriangulation (page 54). Lindstrom's right triangles (page 18) can also be regarded in this category if the triangles are combined as refinement fragments in pairs or fours as indicated in Figure 4.11.

4.4.3.3 Fragment reordering

The output of a decimating approximation method, i.e. one which operates on a high resolution mesh and monotonically decreases the resolution of this mesh by making local modifications, such as Schroeder's method [SZL92], can be easily converted to a CRMR by reversing the order of the resulting meshes.

We can also create a CRMR from a refining or decimating process which is non-monotonic with respect to the resolution of its output approximations. This will be true of a Progressive Mesh representation, for example, if the fragments are created with respect to the standard monotonically increasing

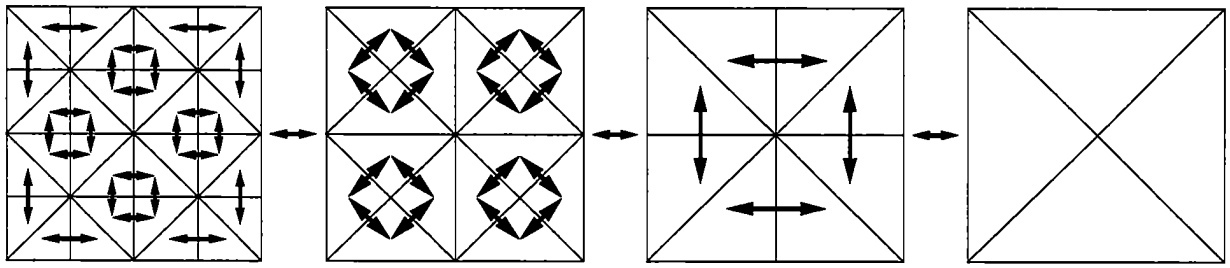


Figure 4.11: The fashion in which Lindstrom's right triangles could be combined in order to produce a CRMR (cf. Figure 2.11).

energy function but we desire a selective refinement to be produced with respect to a similarity-based metric such as the Hausdorff distance.

If we assume that an approximating process has produced a set of approximations by repeated local refinement and as a result we have a set $\{\mathcal{M}^0, \ldots, \mathcal{M}^m\}$ such that $\mathcal{M}^m$ is finer than $\mathcal{M}^0$, but $\mathcal{M}^{i+1}$ is not necessarily finer than $\mathcal{M}^i$ for $0 \leq i < m$. As before, we create the pseudo refinement operation R^0 from $\mathcal{M}^0$ and initially create the refinement operations $R^1 \ldots R^m$ by comparing $\mathcal{M}^i$ with $\mathcal{M}^{i-1}$ for $1 \leq i \leq m$.

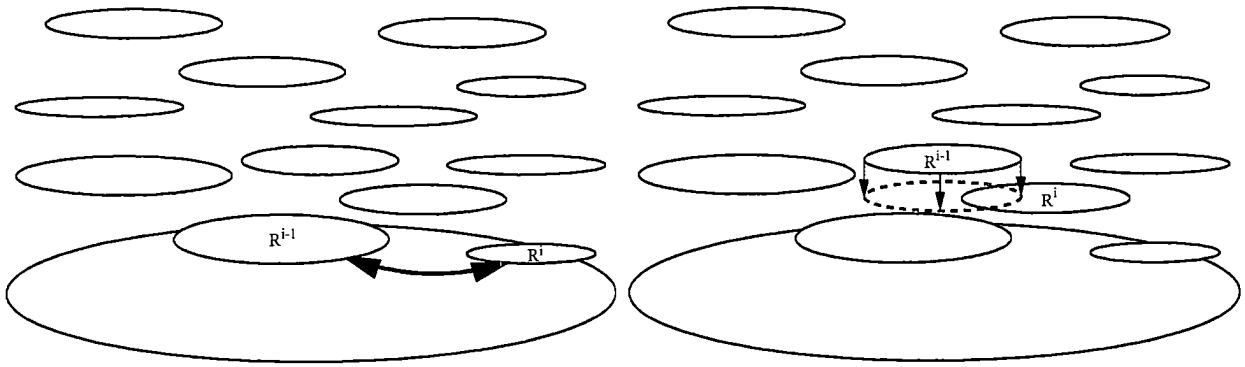
To ensure that the refinement operations are in birth resolution order, though, some reordering and coalescing of their fragments may now be necessary. We iterate through the set of refinement operations, comparing $birthres^i$ with $birthres^{i-1}$ for $1 \leq i \leq m$. If R^i is coarser than R^{i-1} and $F(R^i) \cap F(R^{i-1}) = \emptyset$ then there is no overlap between these fragments and so we can swap R^i and R^{i-1} and then repeat this test for R^{i-1} and R^{i-2}, etc (Figure 4.12a). If R^i is coarser than R^{i-1} but $F(R^i) \cap F(R^{i-1}) \neq \emptyset$ for some i in this iteration then we must coalesce these operations' fragments by setting:

$$\begin{aligned} PreF(R^{i-1}) &= PreF(R^i) \cup (PreF(R^{i-1}) \backslash F(R^i)) \\ V(R^{i-1}) &= V(R^{i-1}) \cup V(R^i) \\ F(R^{i-1}) &= F(R^{i-1}) \cup (F(R^i) \backslash PreF(R^{i-1})) \\ A(R^{i-1}) &= A(R^{i-1}) \cup A(R^i) \end{aligned}$$

and then removing R^i (Figure 4.12b).

4.4.3.4 Fragment adaptation

Finally, some approximation methods generate fragments which have points inserted on their boundaries, and these boundaries do not necessarily coin-



(a) Two refinement operations need to be reordered and they do not overlap so the reordering is trivial.

(b) Two overlapping refinement operations can be reordered by coalescing the finer one into the coarser.

Figure 4.12: Reordering refinement operations to ensure monotonicity in their birth resolutions.

cide with the boundary of one of the meshes $\mathcal{M}^0 \ldots \mathcal{M}^m$. This kind of fragment breaks the refinement operation completeness requirement. Examples of this category of approximation method are De Floriani's HTIN (page 22), the quarternary triangulation (page 24), Scarlatos' cartographic coherence (page 26), co-planar decimation (page 31) and Ferguson's continuous terrain LOD (page 44).

To handle this case, we can either treat the method as per the multiple LODs approach, which typically results in large fragments since there is not necessarily a high degree of edge coherency between the output LODs, or we can adapt the approximation method. This is the preferred approach since the above methods all produce fragments which can be easily constrained to satisfy the completeness and minimality conditions. For example, the co-planar decimation approach could produce valid refinement operations if we prevented the removal of vertices on the boundary of any patch which was identified as being nearly coplanar.

4.5 CRMR structures

We require two structures to maintain both the spatial and resolution topologies of the refinement fragments in a CRMR. The *Hypermesh* component stores the complete spatial topology of each fragment but only represents that fragment's relationships with other fragments by linking its boundary

edges with corresponding edges in these fragments (Section 4.5.1). The *DAG* component is a Directed Acyclic Graph which represents an abstraction of the Hypermesh structure; this is described in Section 4.5.2.

4.5.1 CRMR Hypermesh component

Section 3.3 described Cignoni's *HyperTriangulation (HT)* structure which maintains the spatial and resolution topologies of fragments of triangulated approximations to a terrain surface. These relationships are retained by associating the boundary edges of each successive fragment with their corresponding edges on the previous mesh (Figure 3.15). This edge correspondence is essential to our selective refinement approach and hence the Hypermesh component of a CRMR maintains these relationships for a sequence of refinement operations using an adaptation of Cignoni's HT.

The two conceptual differences between our Hypermesh and Cignoni's HyperTriangulation are: a) we invert the approximation metric, which is now based on "resolution" rather than "error"; and b) we distinguish between edges which are geometrically identical but which exist on different fragments.

The primary reason for these two modifications is to permit the Hypermesh structure to handle models in such a way that the Resolution Control Function can be easily specified. The first modification permits us to take a more consistent approach throughout the SMR framework to the metric against which a model is selectively refined. As well as being a useful syntactic alteration (we can refer to a finer fragment as "higher", for example), the benefits of being able to specify an RCF as the minimum resolution which is required throughout a model are demonstrated in Section 4.6.

The second modification means that if a single geometric edge lies on more than one fragment then there exists a version of this edge for each of the fragments on which it lies. Cignoni's "bubbles" of fragments can therefore be viewed as planar regions in the $2\frac{1}{2}$-dimensional case, as Figure 4.13 shows. Hence we say that an edge is "born" when a fragment of which it is part is "born". Similarly, an edge "dies" (perhaps to be replaced by a copy of itself over a new resolution range) when a triangle of which it is a boundary "dies".

To maintain the version relationships, an edge may be linked to finer and coarser versions of itself. If such versions exist then they will be "alive" for intervals of resolution which do not intersect with that of the original edge. The refinement operation completeness condition permits an edge which is on the boundary of a mesh to be refined into a set of edges and therefore along mesh boundaries the finer/coarser relationships may be one-to-many and many-to-one, respectively, rather than the normal one-to-one (see Figure 4.14). If a fragment boundary edge surrounds a hole in the interior of that fragment, or the edge is part of a separate simplicial mesh which has

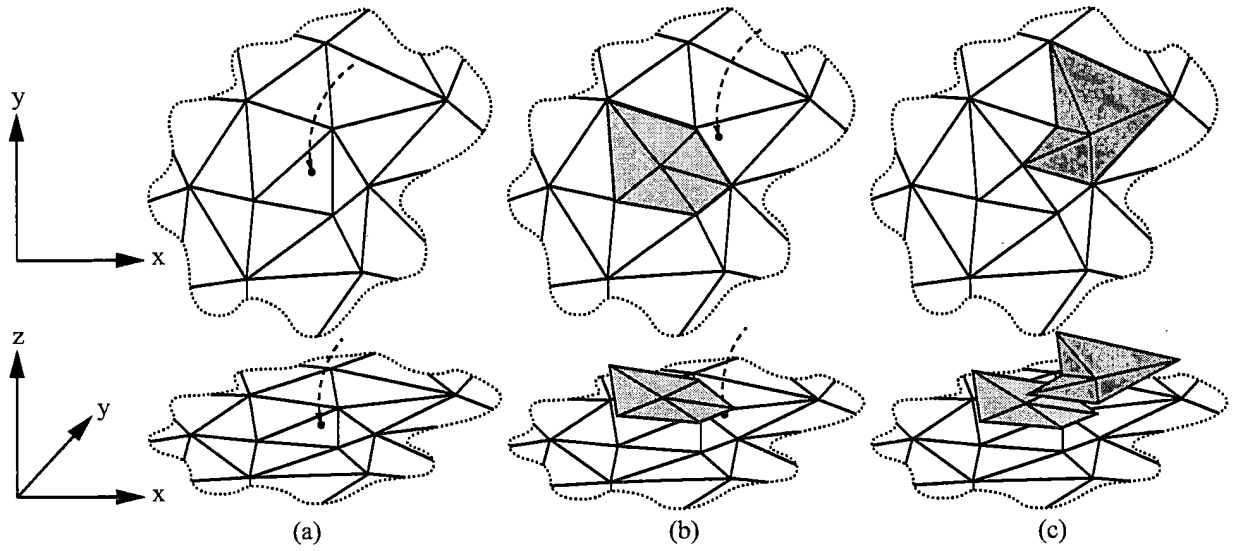


Figure 4.13: Planar regions corresponding to the fragments caused by two point insertions (cf. Figure 3.15).

been spawned by this fragment (as in Figures 4.9(b) and (d)), then the edge has no coarser versions.

By distinguishing between geometrically identical edges which have been created at different resolutions, we can simplify both the visualisation and implementation of the Hypermesh structure compared to the HyperTriangulation. Cignoni's visualisation of a fragment as a "bubble" of faces pasted on top of an existing mesh becomes confusing if a sequence of refinement operations are to be viewed. If we embed our planar fragments in three-dimensional space (for the $2\frac{1}{2}$-D case which Cignoni considered) then we can directly relate them to the Resolution Control Function and the significance of this will be discussed in Section 4.6.2. The novel data structure with which we implemented our Hypermesh for the $2\frac{1}{2}$- and 3-dimensional cases is described in the next section.

4.5.1.1 Hypermesh implementation

The data structure which Cignoni used to represent a HyperTriangulation is based on the *facet-edge* structure which was described by Dobkin and Laszlo [DL89] for the purpose of representing cell complexes in three dimensions. The basic primitive in this structure is a face and one of its edges – a *facet-edge.*

Cignoni explained this structure further using Figure 4.15a. A facet-

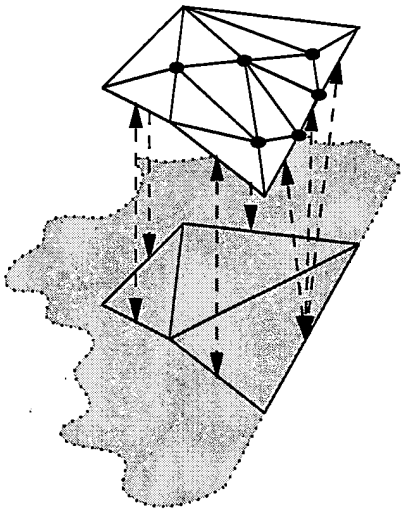


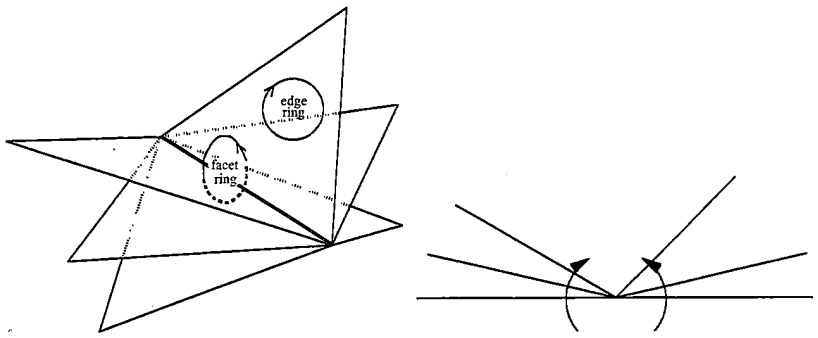
Figure 4.14: An edge may be refined into a set of edges along the boundary of a mesh. Otherwise, the finer/coarser relationship between edge versions is strictly one-to-one.

edge fe denotes two rings in a cell complex C: the *edge-ring* is formed by all the edges of the boundary of face f; the *facet-ring* is formed by all the faces incident at edge e. The traversal functions *enext* and *fnext* permit movement from one facet-edge to the next along the edge-ring and facet-ring respectively.

In order to apply the facet-edge structure to the task of terrain modelling, Cignoni augmented the structure. Each facet-ring was split into two separate bidirectional chains, according to which side of the corresponding edge its triangles lay (Figure 4.15b). A further link was maintained from each facet-edge to another facet-edge on the other side of the edge. Hence, for each facet-edge, a facet-ring was handled by the operators:

- *fnext:* next facet-edge (with lower error) in the facet-ring;
- *fprev:* previous facet-edge (with higher error) in the facet-ring;
- *fother:* facet-edge on the other side of the edge.

This modified facet-edge data structure enabled Cignoni to present the constant resolution and variable resolution extraction algorithms mentioned in Section 3.3. As a representation for a HyperTriangulation, though, this structure is unnecessarily complex since its ability to model a three-dimensional structure is only being applied to the problem of modelling links between two-dimensional meshes. This led us to propose our *fanedge* data structure as a simple means of handling a Hypermesh.



(a) A number of facet-edges are depicted around one (bold) edge; the facet-ring and edge-ring of one particular facet-edge are highlighted

(b) A cross-section of (a) showing the two chains of facets (*fnext* and *fprev*) into which the facet-ring would be divided.

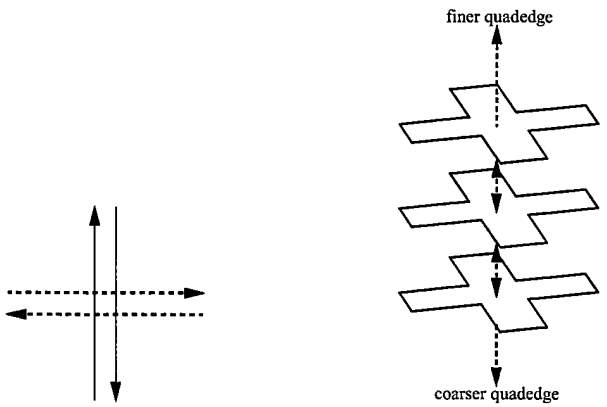
Figure 4.15: Facet-edges (from [CPS95]).

Our fanedge structure is based on Guibas and Stolfi's quadedge structure, which is a well documented structure for handling subdivisions of 2-manifold meshes. The primary reason for using the quadedge structure as the foundation of our Hypermesh representation is to enable the use of existing quadedge-based algorithms for tasks such as point location and Delaunay triangulation [GS85, Lis94].

Guibas and Stolfi's quadedge data structure is similar to Baumgart's winged-edge structure for modelling solids [Bau75]. A set of quadedges can be used to represent simultaneously a general subdivision of a two-dimensional manifold and its dual. Each edge in the subdivision is represented by one quadedge structure. Internally, a single undirected subdivision edge is represented by four directed versions in the quadedge. These versions are the two directed forms of that subdivision edge and the two directed forms of the dual of that edge. The four forms in which a subdivision edge is stored within a quadedge structure are shown in Figure 4.16a.

We define a fanedge as one or more quadedges which are ordered with respect to the resolutions at which they were created (their birth resolutions). Traversal operators *finer* and *coarser* are provided to permit movement to the next higher, or lower, resolution quadedge (Figure 4.16b). This can be extended trivially to permit the one-to-many relationships which are required at a mesh boundary edge whereas the corresponding extension to the facet-edge structure would be more complex.

The final item of information which we store in each quadedge is the birth



(a) The four directed versions of a single real edge which are stored in a quadedge.

(b) A fanedge consisting of three (abstract) quadedge layers.

Figure 4.16: A quadedge and a fanedge.

resolution of that layer. From this, we can infer both the birth and death resolutions of each quadedge in a fanedge since the birth resolution of one quadedge is the death resolution of its predecessor.

4.5.2 CRMR DAG component

The Hypermesh component alone could be used to perform simple selective refinement by making only minor modifications to Cignoni's Hypertriangulation variable resolution algorithm. As we discussed in Section 3.3, such selective refinement would be restricted to producing a mesh which satisfies a monotonically decreasing error function (Figure 4.17).

Cignoni's algorithm starts from the finest triangle which is required in the output surface (e.g. one of the unshaded set of triangles on the left of Figure 4.17) and iteratively adds triangles to the output surface in order of increasing error. Thus the output surface can be viewed as being "grown" around a starting triangle.

We wish to remove the non-increasing error constraint (or, in our terminology, non-decreasing resolution constraint) from the resolution specifier. This could obviously be handled by locating the maxima of a given RCF and hence determining the set of facets at which extraction should be initiated and around which the output mesh could be grown using a parallelised form of Cignoni's method. The disadvantage of this approach, though, is that it is not always possible to determine the maxima of a resolution specifier,

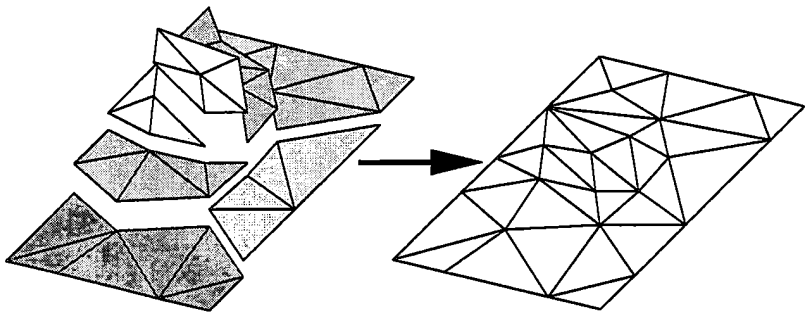


Figure 4.17: A set of triangles which may be extracted by Cignoni's variable resolution algorithm in order to satisfy some non-decreasing error function. The diagram on the left shows, in $2\frac{1}{2}$-dimensional resolution space, the set of fragments which have been used to construct the mesh on the right.

particularly if it is view-dependent.

The mesh extraction algorithms which are described in the following chapter can take advantage of the Hypermesh structure to avoid determining an RCF's maxima. To do this, though, they require an abstraction of the information contained in a Hypermesh, which we call the *Directed Acyclic Graph (DAG)* component of a CRMR.

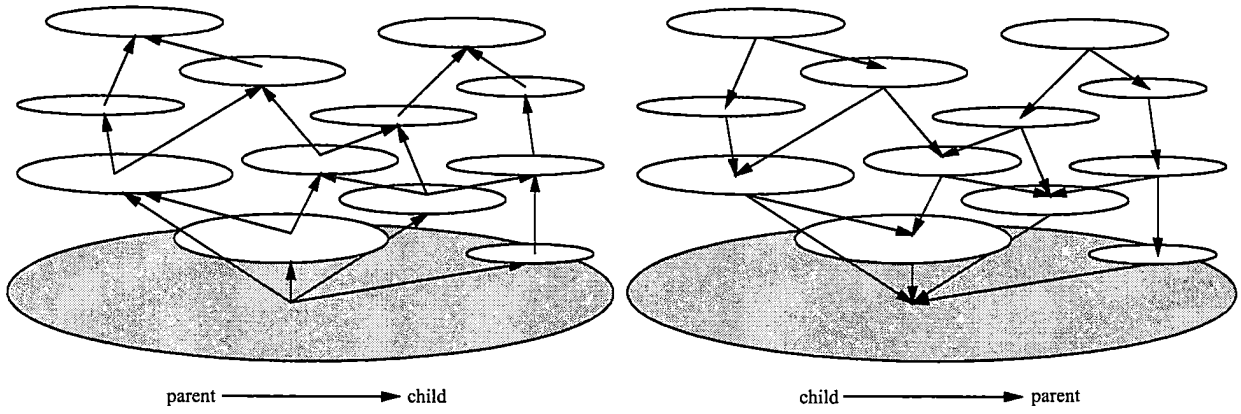
A CRMR's DAG is a graph whose nodes represent the refinement operations contained in the Hypermesh and whose arcs represent the overlapping nature of these operations' fragments.

We define the sets of *children* and *parents* of a refinement operation R^i as

$$\begin{aligned} \mathit{children}(R^i) &= \{\ R^k : F(R^i) \cap \mathit{PreF}(R^k) \neq \emptyset\ \} \\ \mathit{parents}(R^i) &= \{\ R^k : \mathit{PreF}(R^i) \cap F(R^k) \neq \emptyset\ \} \end{aligned}$$

The sets of *ancestors* and *descendants* of a particular operation R^i can be defined as extensions of these sets in the usual way.

We define a CRMR's DAG as the set of nodes $\{R^0, \ldots, R^m\}$ together with the set of arcs $\{(R^i, R^j) : R^j \in \mathit{children}(R^i)\}$. The root of a DAG is R^0. The arcs in Figure 4.18 indicate the inverse children and parent relationships which a DAG represents as an abstraction of its corresponding Hypermesh. Note that although this diagram indicates the spatial extent of each of its nodes' refinement fragments, this information is only stored in the Hypermesh.



(a) Child arcs.

(b) Parent arcs.

Figure 4.18: The DAG corresponding to the Hypermesh of Figure 4.8. The fragment associated with R^0 is shaded in both diagrams.

4.6 Resolution Control Function

Resolution Control Function (RCF) is the term we use to encompass all of the resolution constraints which must be met by a mesh produced from the SMR process. An RCF can be regarded as an "oracle" to which fragments are passed to determine whether they satisfy the current resolution criteria.

The following sections first outline the components from which an RCF may be constructed and the interface which we require of an RCF (Section 4.6.1). We then examine the potential object-space and screen-space resolution criteria which an RCF may contain (Sections 4.6.2 and 4.6.3).

4.6.1 RCF components

A Resolution Control Function is typically composed of a number of components, each dealing with an aspect of the resolution which is desired in a particular output scene. If we assume that each component takes a refinement fragment, R^i, as input and returns whether it satisfies a particular resolution criterion then the following queries will typically be encapsulated in an RCF's components:

- does any part of $F(R^i)$ lies inside the view frustum?
- is $F(R^i)$ completely back-facing, i.e. does none of its facets face towards the viewer?

- is $F(R^i)$ sufficiently similar to the portion of the original model which that fragment represents? Such a similarity criterion can be phrased in terms of *object-space* or *screen-space* dependencies, i.e. in terms of the approximation error between a fragment and its corresponding model region or in terms of the projected size of that error.

These queries are the ones which Hoppe identified as suitable refinement criteria and which we discussed in terms of his Progressive Meshes approach in Section 3.6. Indeed, we use the bounding sphere and "cone of normals" tests which Hoppe advocated as our view frustum and back-face culling checks. We translate his vertex-based refinement criteria into our fragment-based RCF approach in Figure 4.19. This specifies the order of the tests which we use to determine whether a refinement operation satisfies the minimum resolution criterion which the RCF specifies. If this true then we say that the refinement operation *is finer than* the RCF. Note that if a fragment is not visible in the current frame (i.e. it is outside the view frustum or is back-facing) then we automatically return that its refinement operation is finer than the RCF with the aim of reducing the resolution in this area of the mesh.

Function: IsFiner
  In: R^i: refinement operation to be tested against the RCF
  Out: True or False
begin
  if $F(R^i)$ is outside the view frustum **then** return True
  if $F(R^i)$ is back-facing **then** return True
  if $F(R^i)$ satisfies the similarity criterion **then** return True
  return False
end

Figure 4.19: Pseudo-code outlining an RCF's resolution tests.

The similarity criterion which is the third test in Figure 4.19 can be specified in terms of either object- or screen-space characteristics. This test can also be performed on the pre-operation facets of a refinement operation in which case the test becomes:

if $PreF(R^i)$ satisfies the similarity criterion **then** return True

We can use this line to replace the third test in the pseudo-code of Figure 4.19 and if a fragment satisfies this new RCF check then it is deemed to be *completely finer than* the RCF. For completeness, we specify that R^0 is never completely finer than the RCF since $PreF(R^0)$ is defined to be null.

An implementation of this form of RCF can cache the view frustum and back-face culling tests' results within the "IsFiner" and "IsCompletelyFiner" checks for each refinement operation since these results are independent of the similarity criterion.

4.6.2 Object-space resolution criterion

An example of an application for which a guarantee on the object-space resolution of a selectively refined mesh could be important is a flight simulator. For example, if a simulated flight is close to the surface of a terrain (termed *nap-of-the-earth* flying [Sch83]) then it is important that the geometric difference between the displayed surface and the terrain data should be minimal in certain areas, notably the region around the aircraft and along silhouette edges. This requirement is more complex than the simple object-space resolution criteria which have been used previously (Section 3.6). This section concentrates on how we can specify a complex $2\frac{1}{2}$-dimensional object-space resolution criterion and also discusses how a 3-dimensional criterion can be implemented.

We can specify the minimum object-space resolution which is required over the domain, Ω, of a "half-dimensional" surface as a function which has the same dimensionality as the surface. For example, an object-space resolution function for the Mt St Helens terrain data of Figure 2.1 could be as depicted in Figure 4.20. This $2\frac{1}{2}$-dimensional function has a domain which is equal to the domain of the terrain dataset and a range of $[0, 1]$. The higher regions of the function indicate areas in which the surface's object-space resolution must be correspondingly higher – hence both the Mt St Helens crater and an off-centre region are of particular interest according to this function.

This $2\frac{1}{2}$-dimensional resolution space visualisation of the object-space resolution criterion can be combined with an embedding of the Hypermesh to obtain a visual comparison of a CRMR's fragments and the resolution criterion. Rather than Cignoni's qualitative embedding of his HyperTriangulation's fragments in three dimensions, we can position each triangle of a CRMR with an elevation which is equal to its death resolution. Figure 4.21a shows how the pre- and post-operation triangles of one refinement operation can be visualised in resolution space; the shaded triangles represent the refinement fragment. Figure 4.21b shows all the refinement fragments from a CRMR of the Mt St Helens dataset in resolution space.

The object-space RCF similarity test can then be applied by comparing each triangle in a fragment with the value of the object-space resolution function in the triangle's domain: if the triangle is completely higher than the function then that triangle satisfies the criterion.

More formally, the minimum resolution required of each triangle in a $2\frac{1}{2}$-

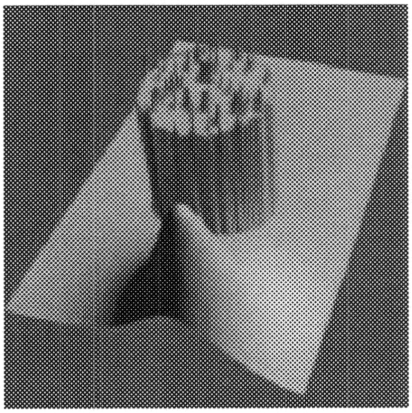
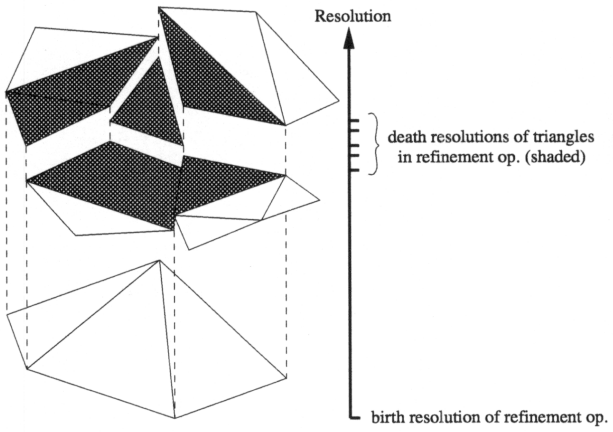
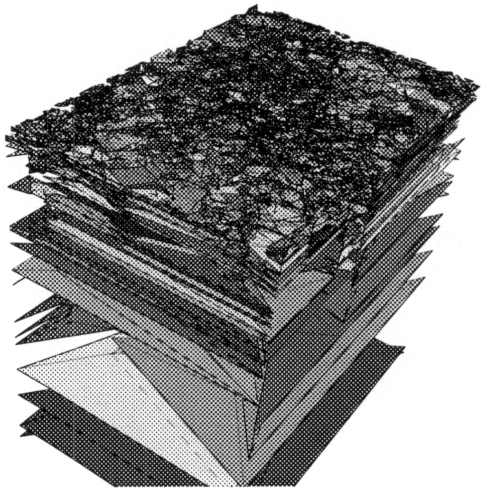


Figure 4.20: An object-space resolution specifier for the Mt St Helens dataset. The viewpoint is close to the edge of the domain; an additional Area Of Interest is centred on the mountain's crater. A critical line detector has also identified important lines in this area.



(a) The pre- and post-operation triangles of one refinement operation in $2\frac{1}{2}$-dimensional resolution space. The additional triangles attached to the refinement fragment's triangles are the pre-operation sets of other operations.



(b) The pre-operation triangle sets for all of the refinement operations in a CRMR of the Mt St Helens dataset viewed in resolution space.

Figure 4.21: Refinement operations in resolution space.

dimensional selectively refined surface is specified as a single-valued bivariate function $g : \Omega \rightarrow [0, 1]$. A triangle t in the extracted surface is said to satisfy this function if $\forall p \in D(t), \mathit{deathres}(t) \geq g(p)$, where $D(t)$ is the domain of the projection of t on Ω.

The object-space RCF test then becomes, for the IsFiner and IsCompletelyFiner tests, respectively:

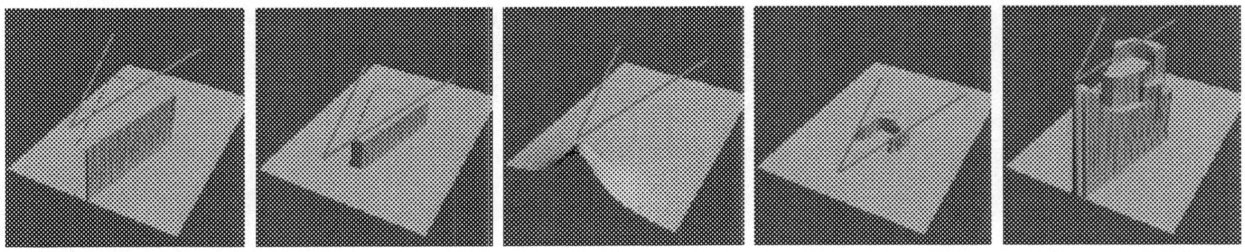
- R^i satisfies the similarity criterion if $\forall t \in F(R^i), \forall p \in D(t), \mathit{deathres}(t) \geq g(p)$, i.e. if the post-refinement set of triangles in operation R^i satisfies g.
- R^i satisfies the similarity criterion if $\forall p \in D(R^i), \mathit{birthres}^i \geq g(p)$, i.e. if the pre-refinement set satisfies the RCF.

The object-space resolution function, g, can be specified by combining bell-shaped functions, such as that in the foreground of Figure 4.20 which specifies an Area Of Interest, and the output of terrain feature detectors. The former are similar in concept to the "magnifying glass" Gaussian weighting functions which were applied by Gross (Section 2.2.1) to wavelet transforms in order to highlight certain regions. An example of thresholded output from our terrain feature detector, which applies Peucker and Douglas' critical line detector (Section 2.2.3), is given in Figure 4.22d. This could be used as an object-space resolution criterion to ensure that the critical lines in a terrain are retained in every frame of a fly-through.

Figure 4.22 illustrates how other resolution requirements, including a view frustum test, can be incorporated into an object-space RCF. The minimum object-space resolution of all the triangles in the view frustum denoted by the orange lines is specified to be 0.5 by component (a); an area of enhanced resolution in the centre of the screen is simulated by requiring the triangles which lie in the corresponding object-space cone (b) to be at a higher resolution; and the viewpoint is highlighted by a Gaussian function centred on that point in (c); component (d) is the result of applying our critical line detection technique to the Mt St Helens data. Components (b), (c) and (d) were clipped by the view frustum and then added to (a) to produce the object-space function of Figure 4.22e.

The general form of this object-space resolution specifier is applicable to SMR in $1\frac{1}{2}$- and $2\frac{1}{2}$-dimensions and is extensible to the other "half-dimension" cases. It is less applicable in cases which are analogous to three dimensions.

In these cases, we may still wish to specify significant regions of a model which should be retained at a higher level of resolution than their surroundings. Our strategy for meeting this requirement is rather "ad hoc", but our primary aim is to permit a simple means by which these significant regions can be specified and we are not aware of an existing comparable method.



(a) View frustum component.

(b) Enhanced screen area component

(c) Area Of Interest centred on the viewpoint.

(d) Critical lines' enhancement component.

(e) Function obtained by combining these components.

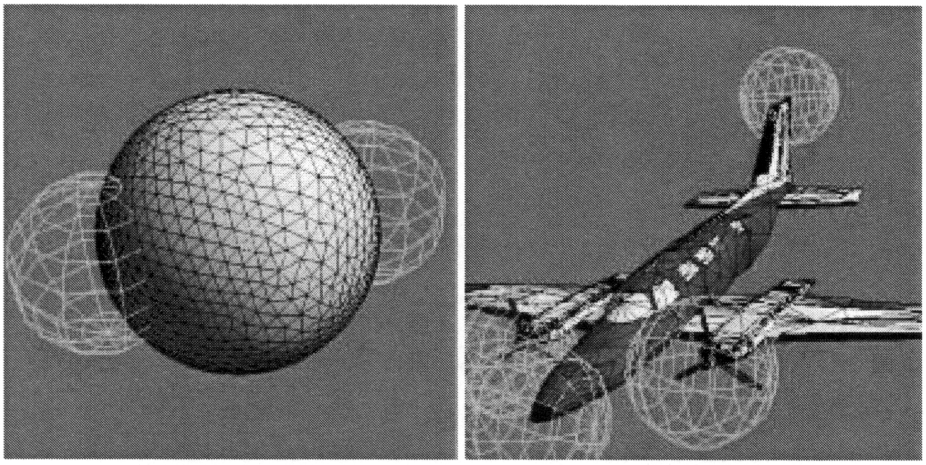
Figure 4.22: Components of a complex object-space function for the Mt St Helens model. The orange lines denote the extent of the view frustum in the xy-plane.

Taking the 3-dimensional case as an example, we first embed the given CRMR's refinement fragments in a three-dimensional "resolution space". This embedding is performed by projecting each vertex of the fragments' triangles along its "current" normal, where the current normal of a vertex v in $\mathrm{F}(R^i)$ is defined as the average of the normals of the faces adjacent to v in $\mathcal{M}^i$ (i.e. the mesh after R^i has been invoked). The distance along its current normal that each vertex is projected is equal to the death resolution of the fragment within which the vertex lies, scaled by the approximation error in the base mesh of the CRMR. This scaling is necessary to relate our $[0, 1]$ range of resolution values to the space in which the model was defined.

The positioning of fragments in 3-dimensional resolution space can be compared to the production of an "exploded diagram" of the original model.

We can then specify the significant regions of a model as volumes, typically spheres, centred at the point of significance. These volumes are termed *volumes of significance*. For fine control over the object-space resolution of a mesh, the radius of a volume of significance (if it is a sphere) can be set equal to the minimum resolution at which we wish the surrounding region of the model to be displayed, multiplied by the maximum error in the base mesh. Figure 4.23a shows the mesh of a sphere which has been selectively refined according to the two spherical volumes of significance illustrated. These are centred at two points on the surface of the original model and have radii equal to $0.997 \times$ maximum error in the sphere CRMR's base mesh. The regions around these spheres have been refined accordingly.

Alternatively, a coarse level of object-space resolution control can be gained by positioning relatively large volumes of significance around regions of the model which we wish to refine completely. Figure 4.23b shows four



(a) Sphere selectively refined with respect to the two object-space volumes of significance shown.

(b) Cessna selectively refined with respect to the four volumes shown.

Figure 4.23: Object-space RCF tests applied to two CRMRs.

volumes of significance placed around a model of a Cessna which have radii far in excess of the maximum error in the CRMR's base mesh. These regions of the mesh have been refined to the level of the original model.

This formulation of the object-space RCF test means that it can be treated as a geometric test between the pre- or post-operation triangles of a refinement operation and the specified volumes of significance. A triangle satisfies the object-space criterion if it lies outside all of the volumes of significance in resolution space (or if its fragment has no children). Such a test can be regarded as an inverted form of Cohen's simplification envelopes (Section 2.3.2) since it specifies that triangles are valid only if they lie outside given volumes.

The two disadvantages of this approach are that a highly-specific region of significance may require a complex volume definition and also projected fragments can intersect at corners of a model which may result in a volume of significance causing more refinement than was desired. We feel that these drawbacks are offset by the simplicity of the approach.

4.6.3 Screen-space resolution criterion

Our RCF screen-space resolution test is an adaptation of Hoppe's view-dependent formula which was described in Section 3.6. We remove the μ component from this equation because our birth and death resolution terms are scalar terms from which we cannot derive a deviation space (and hence we revert in part to the screen-space test of Lindstrom which was depicted

in Figure 3.17). We also have to adapt our resolution-based terms to match Hoppe's error-based scheme. The screen-space similarity tests are carried out, for the IsFiner and IsCompletelyFiner tests, respectively, as:

- R^i satisfies the similarity criterion if

$$\forall t \in F(R^i), (1 - \mathit{deathres}(t))\varepsilon \left\|\vec{n}_t \times \vec{v}\right\| \leq 2\tau(\vec{p}_t) \cot \frac{\varphi}{2}$$

  where $\vec{v}$ is a unit vector in the direction from the viewer to the centre of the fragment's bounding sphere, $\vec{n}_t$ is the normal of triangle t, $\vec{p}_t$ is the projected centre of t, τ is the screen-space tolerance as a fraction of viewport size (parameterised with respect to screen position), ε is the approximation error in the CRMR's base mesh and φ is the field-of-view angle. The right-hand side of this equation converts the screen-space tolerance (which we allow to vary within the extent of the viewport) to a pixel-based measure, as per Hoppe [Hop97b].

- R^i satisfies the similarity criterion if

$$(1 - \mathit{birthres}^i)\varepsilon \left\|\vec{n} \times \vec{v}\right\| \leq 2\tau(\vec{p}) \cot \frac{\varphi}{2}$$

  where $\vec{n}$ is the average normal of the facets in $F(R^i)$ and $\vec{p}$ is the projected centre of $F(R^i)$. In our implementation, we approximate $\vec{p}$ by the projected centre of $F(R^i)$'s bounding sphere.

$\tau : \mathbb{R}^2 \rightarrow [0, 1]$ is the screen-space tolerance function. This may be a constant value if we wish to guarantee that the pixel-based difference between the selectively refined mesh and the original is below this given minimum everywhere. Alternatively, we can use τ to specify screen-based Areas Of Interest. For example, we can specify that the centre of a scene should be displayed at an enhanced resolution to permit degradation of detail in the periphery of a head-mounted display, as advocated by Watson [WWHR97].

4.7 Summary

This chapter has introduced our Selective Mesh Refinement framework. The components of this framework which are passed as inputs to the SMR process, the Continuous Resolution Model Representation and the Resolution Control Function, have been described in detail. The former represents a model in a resolution-based manner as a base approximation together with a sequence of refinements. Our RCF provides a consistent fragment-based interface for a variety of existing object-space and screen-space refinement criteria.

Chapter 5

Selective Mesh Refinement

This chapter presents two alternative algorithms which perform the SMR process, i.e. the production of a selectively refined mesh from the CRMR and RCF inputs detailed in the previous chapter. Our *minimal surface* method generates the selectively refined mesh containing the smallest set of facets which represents the original model and which satisfies the given RCF. A modification of this, the *reduced extraction* technique, also produces a representative selectively refined mesh which satisfies the given resolution requirements, but this method improves the speed of the SMR process at the cost of producing a non-minimal mesh.

These algorithms are described in Sections 5.1 and 5.2 together with proofs of the properties which we assert for the output of these algorithms. To simplify our terminology and notation, this chapter concentrates on describing the generation of selectively refined meshes in the $2\frac{1}{2}$- and 3-dimensional SMR cases, although the algorithms and proofs are extensible to other dimensions, e.g. if we replace "triangle" by "tetrahedron" and "edge" by "face".

5.1 Minimal surface SMR

The *minimal surface* SMR task is to produce a selectively refined mesh which meets the following conditions:

- the mesh represents the original model, i.e. it is constructed from the refinement fragments contained in a CRMR in such a way that each portion of the mesh is part of one of the single-resolution approximations from which the CRMR was produced;
- all of the facets in the mesh satisfy the given RCF;
- the mesh is the smallest set of facets which meet the above two conditions;

- the mesh satisfies the Delaunay criterion if the input CRMR's refinement fragments satisfy this criterion (Section 2.2.2.1).

The minimal surface selective refinement algorithm meets these objectives by first selecting an appropriate set of refinement operations from a CRMR and then combining them to produce the desired mesh. This first stage – *refinement operation extraction* – uses the CRMR's DAG to infer the spatial dependencies between refinement fragments in order to ensure they can be combined by the second stage. This latter stage, called *surface expansion*, then uses the CRMR's Hypermesh to perform a single pass through the extracted operations' fragments, expanding those triangles which are necessary and sufficient to produce a representative mesh which satisfies the given RCF.

These two stages are detailed in Sections 5.1.1 and 5.1.2. The algorithmic complexity of the minimal surface SMR process is noted in Section 5.1.3 and proofs of the resulting mesh's properties are presented in Section 5.1.4.

5.1.1 Refinement operation extraction

We construct a list, X, of refinement operations ordered by increasing birth resolution by performing a partial traversal of the given CRMR's DAG. X must contain sufficient operations to permit a complete mesh to be extracted from the fragments of these operations such that the mesh satisfies the given RCF. The traversal proceeds by placing R^0 on X and continues until:

$$
\begin{aligned}
R^i \in X \iff\ & i = 0 \text{ or} \\
& (\exists R^j \in X : R^i \in \mathit{children}(R^j), R^j \text{ is not finer than} \\
& \text{the RCF and } R^i \text{ is not completely finer than the RCF) or} \\
& \exists R^k \in X : R^i \in \mathit{ancestors}(R^k)
\end{aligned}
$$

Informally, this means that a) the children of every operation in X which does not satisfy the RCF are also in X if they themselves are not completely finer than the RCF and b) the ancestor operations of every operation in X are also in X. As Figure 5.1 shows, this equates to making a "cut" through the CRMR's DAG and placing all of the refinement operations which fall below that cut on X.

Figure 5.2 presents the refinement operation extraction process in algorithmic form. This routine is initiated by a call to $\mathit{AddToList}(R^0, X)$ where X has been initialised to $\emptyset$. This initially places R^0 on the list X and then adds elements as necessary to ensure that the above condition is satisfied.

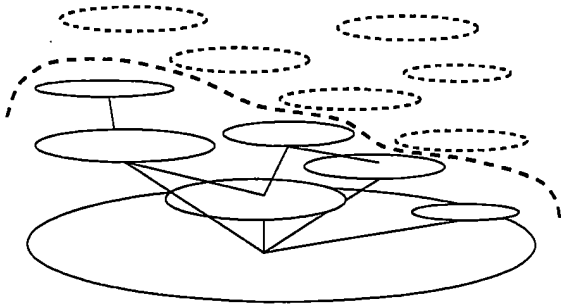


Figure 5.1: The set of refinement operations (drawn as fragments with solid borders) which would be placed on the list X according to the definition on page 94 to satisfy an RCF which specified that operations above the dashed line were completely finer than the RCF. Note that our definition of X in the minimal surface method does not assume that every RCF can be represented by such a simple partitioning of a CRMR's DAG.

```
Function: AddToList
  In:      R^i: operation which may be added to list
  In/Out:X: list of extracted operations
begin
  if R^i ∉ X then
    X := X ∪ R^i
    if R^i is not finer than the RCF then
      for each child, R^j, of R^i do
        if R^j is not completely finer than the RCF then
          AddToList(R^j,X)
        endif
      endfor
    endif
    for each parent, R^k, of R^i do
      AddToList(R^k,X)
    endfor
  endif
end
```

Figure 5.2: The *AddToList* routine.

5.1.2 Surface expansion

The purpose of the surface expansion step is to obtain a mesh which satisfies the RCF from the refinement fragments of operations on the extracted list X.

Our surface expansion algorithm can perform this task in a single traversal of X.

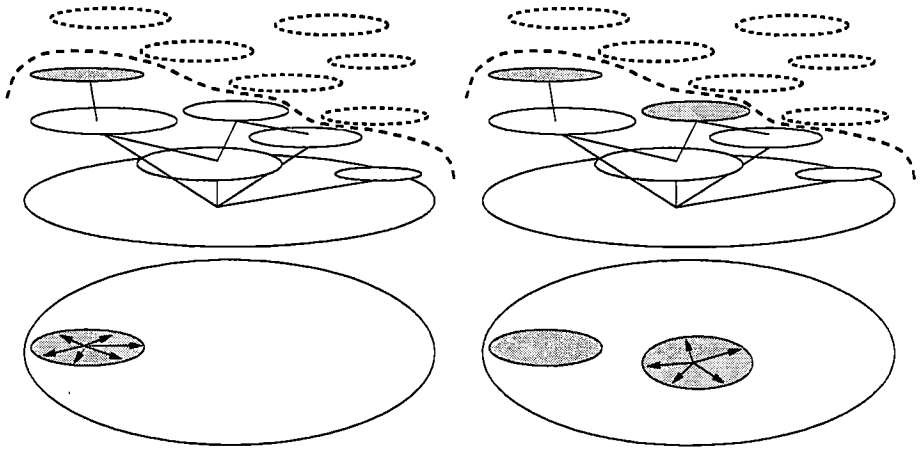
Figure 5.3 gives an indication of how our algorithm would operate on the set of refinement operations which were indicated in the previous example. As this diagram illustrates, we traverse the members of X in order of decreasing resolution and therefore, with respect to the CRMR's DAG, we perform a bottom-up breadth-first search on these fragments. Triangles are expanded from these fragments such that they can be visualised as "cascading" from certain completely-expanded fragments in the CRMR's Hypermesh. Alternatively, as Figure 5.3 shows, the resulting mesh can be viewed as being "grown" around particular fragments – such fragments are regarded as "starting points" for the surface expansion procedure.

The surface expansion operation can therefore be regarded as an instance of the *advancing front* class of algorithms [Lo85], although our front may be more than one contiguous set of edges. A useful side-effect of this mode of operation is that triangles are expanded in an order which is related to their resolution, i.e. highest resolution first. Thus, for example, if the SMR process is being used to generate a mesh for a rendering application which requires a guaranteed frame rate then the surface expansion process can be aborted on each frame refresh with the expectation that the mesh already expanded will contain the significant features of the model.

As is the nature of an advancing front algorithm, we add entities to an *active front* in such a way that the front is extended by these additions. The active front in our method is a set of directed edges. This set is a subset of the edges bounding the currently-expanded mesh because we omit edges which correspond to holes in refinement fragments (i.e. they are "internal" fragment boundaries). We denote the current set of expanded simplices as mesh $\widehat{\mathcal{M}}$ and the set of directed edges which is the current active front as $\widehat{E}$. Figure 5.4 highlights $\widehat{\mathcal{M}}$ and $\widehat{E}$ during the expansion of a mesh.

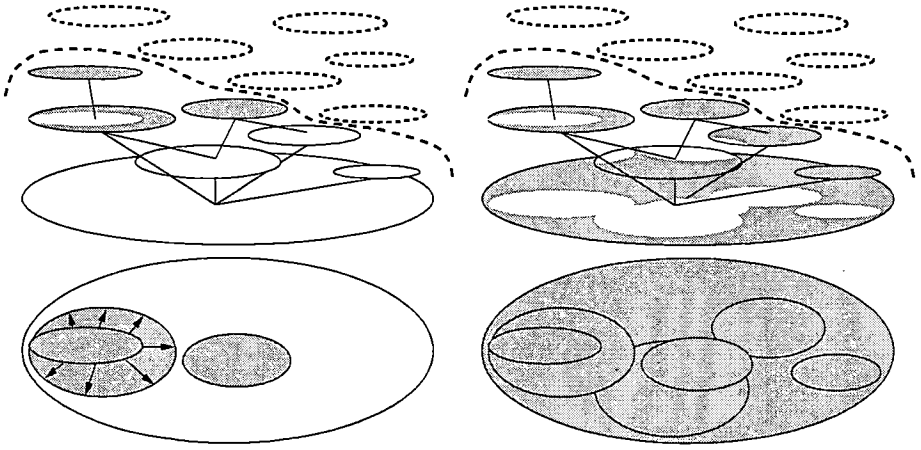
We define an operation *Infill* which performs the two actions of adding triangles to a currently-expanded mesh and extending the active front. As Figure 5.3 shows, we must avoid adding triangles which overlap with ones which have already been expanded. Specifically, *Infill* acts on a refinement operation from X, R^i, and adds those triangles in R^i's fragment which are not elements of the pre-operation sets of fragments which have already been added to $\widehat{\mathcal{M}}$.

This definition suggests that we must retain a list of the fragments which have been added to $\widehat{\mathcal{M}}$, but this is obviated by our use of the active front. We define the direction of the edges in $\widehat{E}$ to be such that the current set of expanded triangles is on the left of these edges (where "left" is defined locally by the orientation of the triangles adjacent to each edge in the active front). Therefore *Infill* acting on R^i must add the triangles in $F(R^i)$ which



(a) One refinement fragment is completely expanded, i.e. all of its triangles are added to the output mesh (shaded in the lower figure).

(b) A second, disjoint, fragment is completely expanded.



(c) The uncovered region of another fragment is added to the existing mesh.

(d) This process continues until the traversal through X is complete.

Figure 5.3: The progress of our surface expansion routine in abstract views of both the CRMR's DAG and the output mesh.

lie to the right of $\widehat{E}$ to $\widehat{\mathcal{M}}$. Simultaneously, of course, the active front must be updated to reflect the new additions to $\widehat{\mathcal{M}}$.

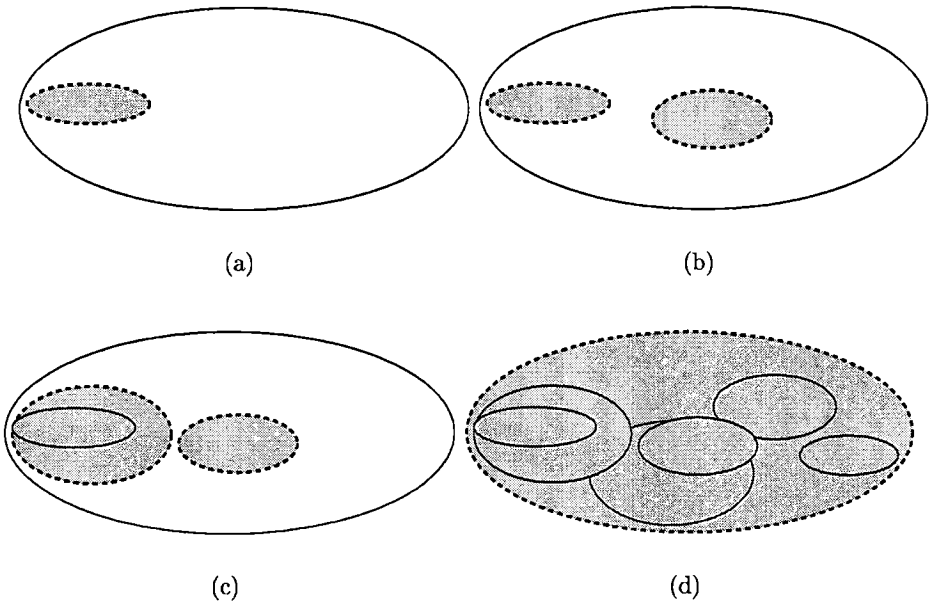


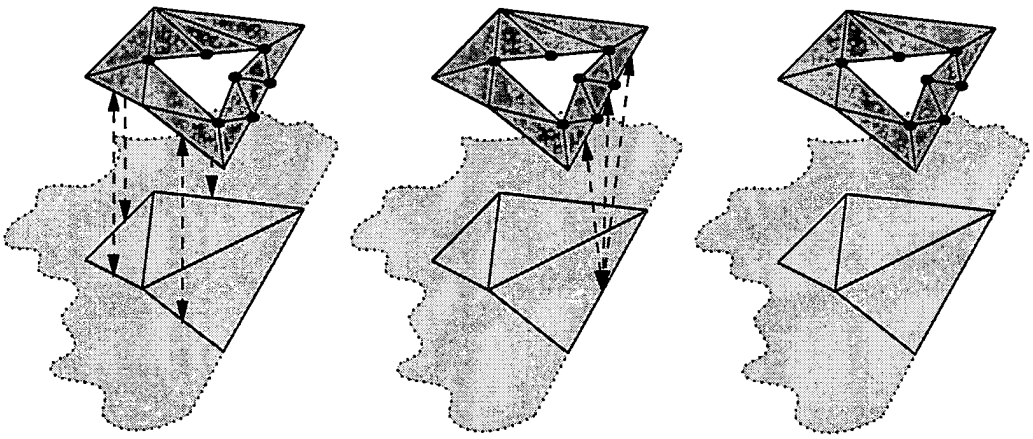
Figure 5.4: The advancing front (dashed lines) viewed on top of the output mesh corresponding to the surface expansion of Figure 5.3.

We specify the exact operation of *Infill* on R^i by a set of rules which are detailed below. In this set of rules, and the remainder of this chapter, we refer to an additional attribute, $E(R^i)$, of refinement operation R^i which is the set of directed edges corresponding to the edges in fragment $F(R^i)$. Thus the edges of each triangle in $F(R^i)$ are represented twice in $E(R^i)$, i.e. once for each of their two possible directions.

1. We first determine whether the refinement fragment of R^i is a "starting point" around which the output mesh will be grown, i.e. it corresponds to a local resolution maximum among the refinement operations in the list X. As Figures 5.3(a) and (b) illustrate, such fragments are expanded completely.

   A refinement operation R^i is an expansion starting point if none of its fragment's edges are on the active front and none of the children of R^i have previously been encountered in this traversal, i.e. $\widehat{E} \cap E(R^i) = \emptyset$ and either $children(R^i) = \emptyset$ or no member of $children(R^i)$ has previously been encountered. When these conditions are met, $F(R^i)$ is completely added to $\widehat{\mathcal{M}}$ and the boundary of $F(R^i)$ is added to $\widehat{E}$.

   Note that the conditions which determine whether an operation is a



(a) The normal linkage between the boundary edges of a fragment (represented by the upper plane) and their coarser versions.

(b) The potential many-to-one relationship between a fragment's boundary edges which lie on the boundary of the original model and their coarser version.

(c) Internal boundary edges of a fragment have no coarser versions.

Figure 5.5: The three possible linkages from the boundary edges of a fragment.

surface expansion starting point cannot be reduced to checking only whether none of its fragment's edges lie on the active front. This may be true of starting point fragments as well as fragments which are completely "covered" by higher resolution refinement operations which are also on X.

2. If some of R^i's fragment edges lie on the active front, i.e. $\widehat{E} \cap E(R^i) \neq \emptyset$ then the action of *Infill* is to add to $\widehat{\mathcal{M}}$ the triangles in $F(R^i)$ which lie to the right of $\widehat{E}$, updating the active front to the resulting new boundary of the expanded region. Any edges added to $\widehat{E}$ will lie on the boundary of $F(R^i)$. This is the case illustrated by Figure 5.3(c).

3. The above two rules cause edges on the boundary of a refinement fragment to be added to $\widehat{E}$. The nature of the CRMR, though, requires the active front to progress in both the spatial and resolution directions.

   Recall that Section 4.5.1 described that the boundary edges of each fragment may be linked to one or more coarser versions in the corresponding Hypermesh. A boundary edge normally has one coarser version, but will not have a such a version if it is on the internal boundary

of a fragment, i.e. it bounds a hole in a fragment. Alternatively, a boundary edge can have more than one coarser version if it represents the boundary of the original model. These three cases are illustrated in Figure 5.5.

When a fragment boundary edge, $\vec{e}$, is encountered by the above two rules and $\vec{e}$ has at least one coarser version then these coarser version(s) are used to replace $\vec{e}$ in $\widehat{E}$. Note that we determine the coarser versions of $\vec{e}$ here as the closest coarser versions of $\vec{e}$ which lie on a fragment which is in X. This definition will be modified in our description of the reduced extraction SMR algorithm.

The above explanation completely defines the surface expansion step of the SMR process. For completeness, pseudo-code of the *Infill* routine and its subroutine, *PerformExpansion*, are presented in the sections below.

5.1.2.1 The *Infill* routine

Infill(R^i) expands the triangles of $F(R^i)$ which are in the area bounded by the fragment's perimeter and the active front, $\widehat{E}$. In addition, if an expanded triangle contains a boundary edge, $\vec{e}$, then that edge is replaced in the active front by its appropriate coarser versions. As in rule (3) above, this usually means that $\vec{e}$ is replaced by another edge which is the closest coarser version of $\vec{e}$ which is on a fragment on X. If $\vec{e}$ (or one of its coarser versions) is a boundary edge then that edge may have more than one replacement, in which case the search for coarser edges on X will continue for each replacement. Alternatively, if $\vec{e}$ is an internal boundary edge of $F(R^i)$ then it will have no coarser version and so $\vec{e}$ can be removed from $\widehat{E}$ without replacement.

In the pseudo-code in Figure 5.6, the first **if** statement relates to the first rule which was specified in the definition of *Infill* above, i.e. whether $F(R^i)$ is a "starting point" fragment which should be completely expanded. If this condition is true then the complete expansion of $F(R^i)$ is initiated by calling *PerformExpansion* to expand a single triangle to the left of some directed edge, $\vec{e} \in E(R^i)$, and then placing that edge on the active front to ensure that there is a complete chain of directed edges in $\widehat{E}$. This is depicted in Figure 5.7a.

For simplicity, the *Infill* pseudo-code assumes that fragments are connected regions, although the definition on page 72 permits them to be disjoint groups of triangles. Our implementation handles this by associating with every fragment a list which contains one triangle from each unconnected component of that fragment. The "starting point" expansion then proceeds by calling *PerformExpansion* for each of these unconnected components.

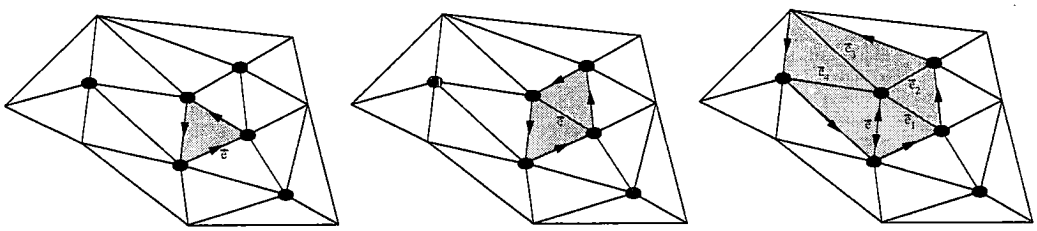
The **while...endwhile** loop which completes the *Infill* routine expands all of the triangles in $F(R^i)$ which lie to the right of the active front. When

Function: Infill
  In: R^i: refinement operation whose fragment is to be "infilled"
  X: extracted operations list
  In/Out: $\widehat{\mathcal{M}}$: current expanded mesh
  $\widehat{E}$: current active front
begin
  // check whether this fragment should be expanded completely
  if $\widehat{E} \cap E(R^i) = \emptyset$ and $children(R^i) \cap X = \emptyset$ **then**
    // initiate the complete expansion of this fragment by expanding one
    // triangle
    let $\vec{e}$ be some directed edge of $E(R^i)$
    PerformExpansion($\vec{e}$, $F(R^i)$, X, $\widehat{\mathcal{M}}$, $\widehat{E}$)
    // complete the chain of active front edges on this fragment
    $\widehat{E} := \widehat{E} \cup \{\vec{e}\}$
  endif
  // expand all triangles outside the active front in this fragment
  while $\exists \vec{e} \in \widehat{E}$ s.t. $\vec{e} \in E(R^i)$ **do**
    let $\vec{e}$ be some directed edge s.t. $\vec{e} \in \widehat{E}$ and $\vec{e} \in E(R^i)$
    $\widehat{E} := \widehat{E} \backslash \{\vec{e}\}$
    if $\vec{e}.Sym \in \widehat{E}$ **then**
      // we have already expanded to the right of $\vec{e}$, so merge the active front
      // along this edge
      $\widehat{E} := \widehat{E} \backslash \{\vec{e}.Sym\}$
    else
      // extract the triangle to the right of $\vec{e}$
      PerformExpansion($\vec{e}.Sym$, $F(R^i)$, X, $\widehat{\mathcal{M}}$, $\widehat{E}$)
    endif
  endwhile
end

Figure 5.6: The *Infill* routine.

an edge $\vec{e}$ is removed from the active front and its symmetric equivalent, $\vec{e}.Sym$, does not exist in the active front then the triangle to the right of $\vec{e}$ is extracted, as in Figure 5.7b. Alternatively, if both of these edges existed in the active front then we simply remove them as Figure 5.7c demonstrates.

This approach lends itself to an efficient implementation using the OpenGL graphics library. The shaded set of triangles in Figure 5.7c indicates how our implementation can make use of the OpenGL *triangle fan* construct. A triangle fan in this sense is a set of triangles which share a common point, as in Figure 5.8. If we can send the triangle data to the OpenGL pipeline as a set of fans then the amount of vertex data which has to be processed can be



(a) Starting the complete expansion of a fragment. The triangle to the left of some edge $\vec{e} \in E(R^i)$ is added to $\widehat{\mathcal{M}}$ and its boundary edges (including $\vec{e}$) are placed on $\widehat{E}$.

(b) One edge of the active front resulting from (a) is now taken as edge $\vec{e}$. This edge is removed from $\widehat{E}$ and the triangle to its right is added to $\widehat{\mathcal{M}}$.

(c) If the process in (b) continues for edges $\vec{e}_1 \ldots \vec{e}_4$ then a complete ring of triangles would be extracted. If we now operate on edge $\vec{e}$, its symmetric equivalent will also be found on the active front and so both will be removed from $\widehat{E}$.

Figure 5.7: The operation of the *Infill* routine.

reduced. This can be realised by modifying the *Infill* and *PerformExtraction* routines presented here such that each edge which is taken from the active front continues the current triangle fan if at all possible (we do this by always adding $\vec{e}.Lnext$ to $\widehat{E}$ in *PerformExtraction* and treating $\widehat{E}$ as a stack). In this way, all triangle data can be sent in the form of triangle fans and a fan need only be terminated when one is completed (as in Figure 5.7c), a fragment boundary edge is encountered, or there is a discrete attribute change across a triangle boundary (such as a colour change).

5.1.2.2 The *PerformExpansion* routine

If there is a triangle to the left of the given edge, $\vec{e}$, in $F(R^i)$ then *PerformExpansion* expands that triangle by adding it to $\widehat{\mathcal{M}}$ and advancing the active front to the triangle's other edges, as Figure 5.7b illustrates. If there is no triangle to the left of $\vec{e}$, i.e. $\vec{e}$ is on the clockwise perimeter of the fragment $F(R^i)$, then we replace $\vec{e}$ by its appropriate coarser versions as detailed in the description of the *Infill* routine above and demonstrated in Figure 5.9. For simplicity, the *PerformExpansion* pseudo-code in Figure 5.10 assumes that every edge in the Hypermesh has only one coarser version.

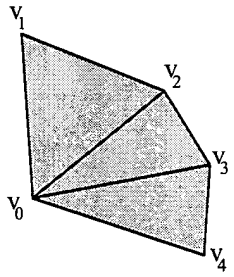


Figure 5.8: A fan of triangles which can be sent to the OpenGL pipeline by referencing only vertices $v_0 \ldots v_4$.

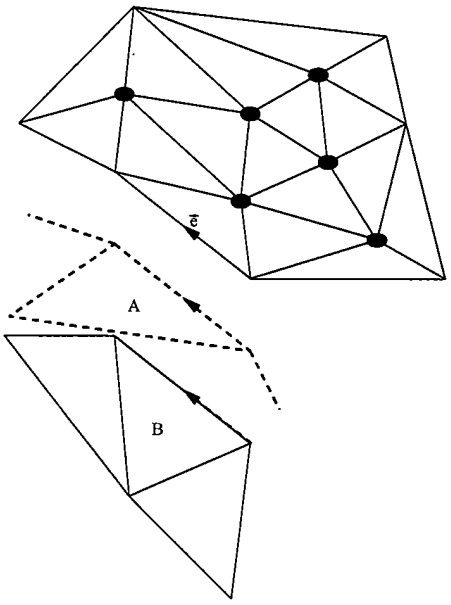


Figure 5.9: The coarser version of an edge, $\vec{e}$, which lies on the boundary of a fragment may be selected as the one on fragment B rather than the version on fragment A because the operation whose fragment is A is not a member of the extracted operations list, X.

Function: PerformExpansion
In: $\vec{e}$: directed edge. If there is a triangle to the left of $\vec{e}$ in $F(R^i)$ then it is expanded
$F(R^i)$: fragment which contains $\vec{e}$
X: extracted operations list
In/Out: $\widehat{\mathcal{M}}$: current expanded mesh
$\widehat{E}$: current active front
begin
if there is no triangle to the left of $\vec{e}$ in $F(R^i)$ **then**
// find the closest coarser version of $\vec{e}$ which is on a fragment in X
$\vec{c} := coarser(\vec{e})$
while fragment on which $\vec{c}$ lies $\notin X$ **do**
$\vec{c} := coarser(\vec{c})$
endwhile
// add this replacement edge to the active front
$\widehat{E} := \widehat{E} \cup \{\vec{c}.Sym\}$
else
// expand the triangle to the left of $\vec{e}$
$\widehat{\mathcal{M}} := \widehat{\mathcal{M}} \cup \{\text{triangle to the left of } \vec{e} \text{ in } F(R^i)\}$
// add the other edges of the expanded triangle to the active front
$\widehat{E} := \widehat{E} \cup \{\vec{e}.Lnext, \vec{e}.Lprev\}$
endif
end

Figure 5.10: The *PerformExpansion* routine.

5.1.3 Algorithmic complexity

The complexity of the above algorithm is a combination of the costs of both the refinement operation extraction and surface expansion steps.

A trivial upper bound on the cost of the former step is $O(m)$, where m is the number of refinement operations embedded in the CRMR and hence is a factor which is introduced by the approximation process from which the CRMR of a model was produced. The Delaunay approximation method which is used by our implementation, for example, generates refinement operations such that $m \approx \frac{1}{2}|V^m|$, where $|V^m|$ is the number of vertices in the highest resolution triangulation of the input CRMR.

The complexity of the refinement operation extraction step can also be viewed as $\Theta(|X|)$ since $|X|$ is the number of refinement operations extracted.

The cost of the other step – the surface expansion step – is $\Theta(|\widehat{\mathcal{M}}|)$ since it expands exactly $|\widehat{\mathcal{M}}|$ triangles. Alternatively, this cost can also be regarded as $\Theta(|X|)$ since the triangulation $\widehat{\mathcal{M}}$ is generated by a traversal through $|X|$

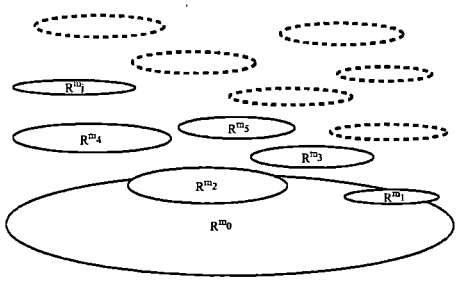


Figure 5.11: An example X list, showing a set of refinement fragments $R^{m_0} \ldots R^{m_j}$ (as fragments with solid outlines) which have been extracted from a CRMR's DAG.

extracted refinement operations and we can assume that there is a constant upper bound on the number of triangles in each refinement fragment since, by the nature of the CRMR generating process, each refinement operation should be localised.

Hence the overall cost of the minimal surface SMR process is $\Theta(|X|)$. Figures for $|X|$ are detailed in the results chapter.

5.1.4 Proofs of expanded mesh properties

This section present proofs of the following theorems.

- An expanded selectively refined mesh satisfies the RCF.
- An expanded mesh represents the original model from which the given CRMR was constructed.
- An expanded mesh is the smallest set of facets which can be expanded from the given CRMR which both satisfies the RCF and represents the original model.
- If the refinement fragments in a $2\frac{1}{2}$-dimensional CRMR satisfy the Delaunay criterion then meshes expanded from that CRMR will also satisfy this criterion.

Lemma 5.1.1 An expanded selectively refined mesh $\widehat{\mathcal{M}}$ which has been constructed from a list X is equivalent to a combination of the refinement operations in this list.

> **Proof:** We assume that we are given a list X which has been produced according to the definition on page 94 and therefore that X contains the refinement operations $\{R^{m_0}, R^{m_1}, \ldots, R^{m_j}\}$

for some j s.t. $0 \leq j \leq m$ and $m_0 = 0$. Figure 5.11 depicts an example of such an X.

The *Infill* operation, according to our original definition on page 96 is such that it "acts on a refinement operation from X, R^i, and adds those triangles in R^i's fragment which are not elements of the pre-operation sets of fragments which have already been added to $\widehat{\mathcal{M}}$".

Hence, remembering that the *Infill* operation is applied to a set of refinement operations in finer to coarser order, we can state the result of the *Infill* operation applied to the set X as:

$$\begin{aligned} \mathit{Infill}(\{R^{m_j},\ldots,R^{m_1},R^{m_0}\}) &= \mathit{Infill}(\{R^{m_j},\ldots,R^{m_1}\}) \cup \\ &\quad [F(R^{m_0}) \setminus \bigcup_{1\leq k\leq j} \mathit{PreF}(R^{m_k})] \\ &= F(R^{m_j}) \cup \\ &\quad [F(R^{m_{j-1}}) \setminus \mathit{PreF}(R^{m_j})] \cup \\ &\quad [F(R^{m_{j-2}}) \setminus \bigcup_{k=\{m_{j-1},m_j\}} \mathit{PreF}(R^{m_k})] \cup \\ &\quad \vdots \\ &\quad [F(R^{m_0}) \setminus \bigcup_{1\leq k\leq j} \mathit{PreF}(R^{m_k})] \end{aligned} \tag{5.1}$$

Also, each refinement operation R^i acting on $\mathcal{M}^{i-1}$ is defined to be the removal of the set of facets $\mathit{PreF}(R^i)$ and the insertion of the replacement set $F(R^i)$. Therefore the set of refinement operations in X can be written in the form:

$$\begin{aligned} \widehat{\mathcal{M}} &= R^j(R^{j-1}(\ldots R^{m_2}(R^{m_1}(F(R^{m_0})))\ldots)) \\ &= R^j(R^{j-1}(\ldots R^{m_2}(F(R^{m_1}) \cup [F(R^{m_0}) \setminus \mathit{PreF}(R^{m_1})])\ldots)) \\ &= F(R^{m_j}) \cup \\ &\quad [[F(R^{m_{j-1}}) \cup \\ &\quad\quad [\ldots \\ &\quad\quad\quad [[F(R^{m_2}) \cup \\ &\quad\quad\quad\quad [[F(R^{m_1}) \cup \\ &\quad\quad\quad\quad\quad [F(R^{m_0}) \setminus \mathit{PreF}(R^{m_1})] \\ &\quad\quad\quad\quad] \setminus \mathit{PreF}(R^{m_2})] \\ &\quad\quad\quad] \setminus \mathit{PreF}(R^{m_3})] \\ &\quad\quad \ldots] \\ &\quad] \setminus \mathit{PreF}(R^{m_j})] \end{aligned} \tag{5.2}$$

By inspection, Equations 5.1 and 5.2 are equivalent. Therefore we have verified that a mesh created by the surface expansion step is identical to that which would be produced by a combination of refinement operations.

Theorem 5.1.1 All of the triangles in $\widehat{\mathcal{M}}$ satisfy the RCF, i.e. $\forall t \in \widehat{\mathcal{M}}$

$$\begin{aligned} &\exists R^i \text{ s.t. } t \in F(R^i) \text{ and } R^i \text{ is finer than the RCF} \\ &\text{or } \exists R^j \text{ s.t. } t \in PreF(R^j) \text{ and } R^j \text{ is completely finer than the RCF} \end{aligned} \tag{5.3}$$

Also, every triangle added to $\widehat{\mathcal{M}}$ is compatible with the existing expanded mesh. Specifically, $\forall t \in \widehat{\mathcal{M}}$

$$\begin{aligned} &\nexists R^i \text{ s.t. } t \in PreF(R^i) \\ &\text{or } \exists R^j \text{ s.t. } t \in PreF(R^j) \text{ and } R^j \notin X \text{ and } descendants(R^j) \cap X = \emptyset \end{aligned} \tag{5.4}$$

Proof: Assume we have a set $X = \{R^{m_0}, \ldots R^{m_j}\}$ which meets the minimal surface definition of X and that we wish to demonstrate the above statements for $\widehat{\mathcal{M}} = Infill(X)$, as in the previous proof.

Statements (5.3) and (5.4) are both true after $Infill(R^{m_j})$ since the definition of X ensures that $descendants(R^{m_j}) \cap X = \emptyset$ and that R^{m_j} is finer than the RCF.

We assume that both of these statements hold for the mesh produced by $Infill(\{R^{m_j}, R^{m_{j-1}}, \ldots, R^{m_k}\})$ for some $k : 0 < k \leq j$.

We must now prove that $Infill(\{R^{m_j}, \ldots, R^{m_{k-1}}\})$ satisfies statements (5.3) and (5.4).

If no triangles were added to the currently-expanded mesh by the addition of $R^{m_{k-1}}$ to the set upon which *Infill* performs, then the statements obviously remain true.

Alternatively, if a triangle t was added to the mesh during the inductive step then the specification of the result of the *Infill* operation in Lemma 5.1.1 above implies that

$$t \notin PreF(R^{m_j}) \cup \ldots \cup PreF(R^{m_k})$$

and hence either $children(R^{m_{k-1}}) = \emptyset$ or $t \in PreF(R^x)$ for some $R^x \notin X$. In the former case, $R^{m_{k-1}}$ must be finer than the RCF and t must be compatible with the triangles curren

$R^x \notin X \Rightarrow descendants(R^x) \cap X = \emptyset$ and
$\nexists R^y \in X : R^x \in children(R^y), R^y$ is not finer than the RCF and R^x is not completely finer than the RCF
$\Rightarrow R^{m_{k-1}}$ is finer than the RCF or
R^x is completely finer than the RCF

Hence t satisfies the RCF and $descendants(R^x) \cap X = \emptyset$. Therefore t has no spatial dependencies with any of the triangles already in the mesh and we can say that t is compatible with the currently-expanded mesh.

Theorem 5.1.2 A selectively refined mesh $\widehat{\mathcal{M}}$ is a representation of the original model $\mathcal{M}$ from which the given CRMR was constructed.

Proof: Lemma 5.1.1 has shown that $\widehat{\mathcal{M}}$ is equivalent to a combination of the refinement operations in a given list X and therefore $\widehat{\mathcal{M}}$ is a combination of $\mathcal{M}^0, \ldots, \mathcal{M}^m$, where these are the approximations to $\mathcal{M}$ which are contained in the given CRMR.

Theorem 5.1.1 has, in addition, shown that this combination of refinement operations is such that each addition of a triangle to the current expanded mesh is compatible with that mesh.

Therefore, the iteration of *Infill* on list X from R^{m_j} to R^0 ensures that a mesh which is representative of $\mathcal{M}$ is generated.

Theorem 5.1.3 $\widehat{\mathcal{M}}$ is the smallest triangle set which can be generated from a combination of the refinement operations in a CRMR such that $\widehat{\mathcal{M}}$ is a representation of the original model which satisfies the given RCF.

This theorem assumes that the fragments of $R^0, \ldots, R^m$ are such that $|PreF(R^i)| < |F(R^i)|$ for $0 \leq i \leq m$, i.e. that each refinement operation introduces more points than it removes and its post-operation fragment is a larger set of triangles than its pre-refinement set. Puppo [Pup96] terms this the *monotonicity* condition.

Proof: Assume, as before, that we have constructed mesh $\widehat{\mathcal{M}}$ as $Infill(X)$ from a list $X = \{R^{m_j}, \ldots, R^{m_0}\}$ which meets the previous definition of X.

Now assume that we can construct an alternative selectively refined mesh, $\widehat{\mathcal{M}}'$, which contains less triangles than $\widehat{\mathcal{M}}$ and still satisfies the RCF. This implies that $\widehat{\mathcal{M}}'$ is constructed from a list X' which omits one or more of the elements of X and replaces

them with one or more alternative refinement operations from the CRMR.

If X' omits a refinement operation R^i from X where R^i is a "starting point" for surface expansion then R^i must be replaced by at least one of its children in order to ensure that $\widehat{\mathcal{M}}'$ satisfies the RCF and the monotonicity condition implies that $\widehat{\mathcal{M}}'$ cannot therefore be smaller than $\widehat{\mathcal{M}}$.

Alternatively, X' may remove a refinement operation R^j from X where R^j is not a "starting point" for expansion but is such that one or more triangles of $F(R^j)$ are included in $\widehat{\mathcal{M}}$. The minimality condition of refinement operations (Section 4.4.1), though, implies that we cannot replace R^j by other operations which will be compatible with the child operations of R^j and therefore a complete mesh cannot be constructed.

Theorem 5.1.4 If SMR is performed for the $2\frac{1}{2}$-dimensional case and the fragments of the refinement operations, $R^0, \ldots, R^m$ satisfy the Delaunay criterion (Section 2.2.2.1) then the output mesh, $\widehat{\mathcal{M}}$, also satisfies this criterion.

Proof: Following de Berg and Dobrint's Delaunay-satisfiability proof [dD95], we note that the vertices in each refinement fragment $R^i, 1 \leq i \leq m$, cannot have an influence, in terms of the Delaunay criterion, outside that fragment. Hence when we add triangles to the currently-expanded mesh during iterations of *Infill*, these triangles will not affect the Delaunay nature of the mesh.

5.2 Reduced extraction SMR

This section describes a modified version of the above algorithm in which the factor which governs the running-time of the process – the cost of the refinement operation expansion step – is decreased. This modified process, termed the *reduced extraction* method, is restricted to performing SMR in the "half-dimensional" cases introduced in Section 4.3.1, i.e. $1\frac{1}{2}$- and $2\frac{1}{2}$-dimensions, etc. Also, the given RCF must be able to be represented (in the $2\frac{1}{2}$-D case) by a bivariate piecewise-continuous function which bisects the input CRMR's Hypermesh into those refinement operations which are, and those which are not, completely finer than the RCF. Thus the object-space resolution functions which were presented in Section 4.6.2 are generally applicable to this method.

The key operation in the above algorithm is the detection of "starting point" fragments around which an expanded mesh is "grown". We can identify such refinement fragments without requiring that the ancestors of all of

the corresponding refinement operations in X are also present in that list, but only if we can make the above assumption regarding the RCF bisecting the CRMR's Hypermesh.

Removing this requirement from the refinement operation extraction procedure also means that the surface expansion step may be required to add further operations to X in order to permit the generation of a complete mesh. Another implication of this modification is that the test for whether a given refinement fragment is a "starting point" for surface expansion must verify that the fragment does not overlap with the currently expanded mesh and hence we must perform a geometric domain intersection test – this is the factor which limits this method to operating on "half-dimensional" cases.

The results presented in Chapter 7 demonstrate that this approach reduces the running-time of the refinement operation extraction step, which is the major cost of the minimal surface algorithm, and that the increased complexity in the surface expansion step does not impact on the overall improvement in the speed of the SMR process. It should be noted, though, that the mesh output by this algorithm has all of the attributes of one generated by the previous method bar the minimality property. The asymptotic complexity of the mesh generation process is not improved by this method and therefore if the cost of rendering the output mesh is more significant than the complexity of the SMR process then the minimal surface algorithm may be, overall, a more efficient approach to the selective refinement problem.

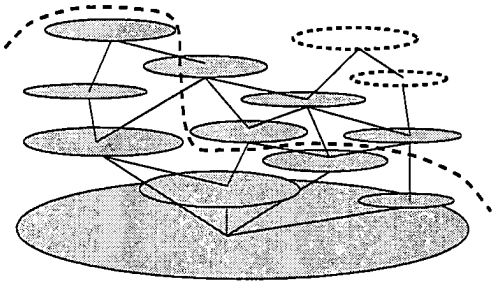
5.2.1 Algorithm

The reduced extraction selective refinement algorithm proceeds in two stages, as before, with the following modifications to the minimal surface algorithm.

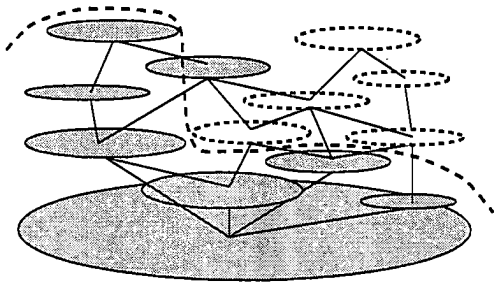
- The modified refinement operation extraction process does not require that all of the ancestors of each refinement operation in the extracted operations' list, X, are also in that list. We again construct X by performing a partial traversal of the CRMR DAG, but now the traversal terminates when:

$$R^i \in X \iff i = 0 \text{ or } (\exists R^j \in X : R^i \in \mathit{children}(R^j), R^j \text{ is not finer than the RCF and } R^i \text{ is not completely finer than the RCF}) \tag{5.5}$$

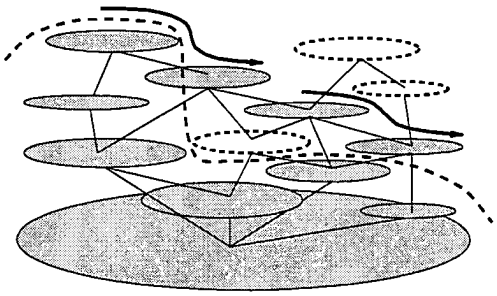
  Less formally, this means that an operation $R^i \in X$ iff R^i is the base mesh or R^i is not completely finer than the RCF and also R^i has a parent which is in X and which is not finer than the RCF. Figures 5.12(a) and (b) illustrate the effect of this modification.



(a) The minimal surface algorithm's operation extraction step would add the shaded fragments to list X to satisfy the definition on page 94 for the given RCF (dashed line).



(b) The reduced extraction algorithm's definition of X would only require these shaded fragments to be added to the list.



(c) During the reduced extraction algorithm's surface expansion step, other operations may have to be added to the X shown in (b) in order to produce a complete mesh. The thick arrows indicate the "cascading" flow of the surface expansion operation.

Figure 5.12: A comparison of the operation of the minimal surface and reduced extraction algorithms.

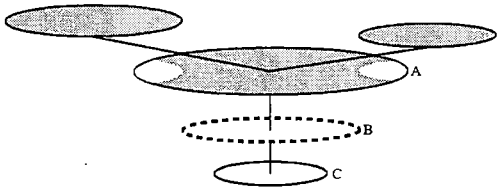


Figure 5.13: A subgraph of a CRMR's DAG, illustrating a case for which the reduced expansion algorithm's domain intersection test is necessary. We assume that fragment B was not placed on list X by the refinement operation extraction step and that fragment C was. The surface expansion process then added the refinement operation whose fragment is A onto list X; this fragment completely covers fragments B and C. If we merely checked whether fragments' children had been encountered, fragment C could be expanded even though its domain had already been covered by the triangles expanded from A.

- This reduced extraction of operations in X means that we must add an extra condition to the step which identifies whether a fragment is a "starting point" for surface expansion (step (1) in the surface expansion process). In addition to checking that no edges of the fragment are on the active front, we must verify that none of the descendants of R^i have been encountered (i.e. not just that none of its children have been encountered). Figure 5.13 illustrates a case which requires this extra test to ensure that a valid mesh is extracted.

  Performing this test for descendants (i.e. $\mathit{descendants}(R^i) \cap X = \emptyset$) on every element of X could be a costly recursive procedure. Instead, we implement the condition in two clauses: firstly, as before, a check that none of the children of R^i have been encountered; and, secondly, a geometric test to ensure that the domain of this fragment does not intersect with the currently expanded mesh. The "domain", $D(F(R^i))$, of a fragment in the $1\frac{1}{2}$- and $2\frac{1}{2}$-dimensional cases is the domain of its projection on the 1- or 2-dimensional basis for the space in which SMR is being performed (e.g. the domain of a terrain fragment is its projection onto the xy-plane). This test cannot easily be extended to non-"half-dimensional" cases.

  Thus we phrase the descendants' test as:

$$\mathit{children}(R^i) \cap X = \emptyset \text{ and } D(F(R^i)) \cap D(\widehat{\mathcal{M}}) = \emptyset$$

  The first of these two clauses involves information which is immediately available from a CRMR's DAG and our experiments have shown that

the second, more costly, clause is rarely invoked. This means that most fragments which are potential "starting points" for surface expansion, because none of the edges in the active front lie on these fragments, can be dismissed immediately because one or more of their children have already been encountered.

The domain intersection test can be performed in the $2\frac{1}{2}$-dimensional case by checking whether all of the boundary vertices of $F(R^i)$ lie inside the current domain of $\widehat{\mathcal{M}}$ using a simple method such as Surany's point-in-polygon test [Sur85].

- The new refinement operation extraction step produces a list which is a subset of that produced by the minimal surface operation extraction process. Let us call the list produced by the new extraction step X^+. This list is sufficient for step (1) of the surface expansion procedure to identify the "starting points" for surface expansion but it will not necessarily contain sufficient operations to permit a complete mesh to be generated from their corresponding fragments.

  We must therefore amend step (3) of the surface expansion process. When a fragment boundary edge, $\vec{e}$, is encountered and that edge has at least one coarser version then these version(s) are added to the active front and the operation R^j whose fragment contains those edges is added to X^+ if R^j is not already on the list (Figure 5.12c).

The impact of these modifications on the pseudo-code presented above for the minimal surface algorithm is discussed in the next two sections.

5.2.1.1 Refinement operation extraction

The key difference between this algorithm and the minimal surface one is that the ancestors of every operation added to the list X are not necessarily also on that list. Therefore the new *AddToList* routine differs from the one in Figure 5.2 in that we remove the loop which adds any parents of R^i to X.

5.2.1.2 Surface expansion

The new *Infill* and *PerformExpansion* routines both differ from those presented previously in that the list of extracted operations, X, is an "In/Out" parameter rather than only an input (and we now term this parameter "X^+").

The only other change to the *Infill* routine given in Figure 5.6 is a modification to the condition which identifies the surface expansion "starting point" fragments. We add the geometric fragment domain test described above to give the line

if $\widehat{E} \cap E(R^i) = \emptyset$ and $children(R^i) \cap X^+ = \emptyset$ and $D(F(R^i)) \cap D(\widehat{\mathcal{M}}) = \emptyset$ **then**

The operation of *PerformExpansion* diverges from the code given in Figure 5.10 in its selection of the coarser version(s) of a fragment's boundary edge $\vec{e}$ and also because it may have to add more operations to X^+. The resulting code is sufficiently different from the previous routine to warrant its inclusion as Figure 5.14. Again note that we simplify the pseudo-code by assuming that if $\vec{e}$ is a fragment boundary edge then it has one, and only one, immediately coarser version.

```
Function: PerformExpansion
  In:      ē: directed edge. If there is a triangle to the left of ē in F(R^i)
             then it is expanded
           F(R^i): fragment which contains ē
  In/Out: X+: extracted operations list
           M̂: current expanded mesh
           Ê: current active front
begin
  if there is no triangle to the left of ē in F(R^i) then
    // obtain the closest coarser version of ē
    c̄ := coarser(ē)
    // place this replacement edge on the active front
    Ê := Ê ∪ {c̄.Sym}
    // add a refinement operation to X+ if necessary
    if the refinement operation whose fragment contains c̄ is not in X+ then
      X+ := X+ ∪ { refinement operation whose fragment contains c̄ }
    endif
  else
    // expand the triangle to the left of ē
    M̂ := M̂ ∪ {triangle to the left of ē in F(R^i)}
    // add the other edges of the expanded triangle to the active front
    Ê := Ê ∪ {ē.Lnext, ē.Lprev}
  endif
end
```

Figure 5.14: The *PerformExpansion* routine for the reduced extraction SMR process.

5.2.2 Proofs of expanded mesh properties

This section presents proofs that a mesh generated by the reduced extraction method is complete (Theorem 5.2.1), satisfies the RCF (Theorem 5.2.2) and can satisfy the Delaunay criteria (Theorem 5.2.3).

To simplify our terminology, we prove these mesh properties for the $2\frac{1}{2}$-dimensional case but the proofs also hold in other "half-dimensions". We use Ω to denote the 2-dimensional planar domain of the $2\frac{1}{2}$-dimensional case.

Lemma 5.2.1 We can reduce the reduced extraction definition of X (Eq. 5.5) to $R^i \in X \iff i = 0$ or R^i is not completely finer than the RCF.

Proof: One implication of the above statement, $R^i \in X \Rightarrow i = 0$ or R^i is not completely finer than the RCF, follows automatically from the definition (Eq. 5.5). We prove the reverse implication, R^i is not completely finer than the RCF or $i = 0 \Rightarrow R^i \in X$, by induction:

- Firstly, $i = 0 \Rightarrow R^i \in X$ is true from the definition of X trivially.
- Assume true for R^j, for some $j, 1 \leq j < m$
- Require to prove that R^{j+1} is not completely finer than the RCF $\Rightarrow R^{j+1} \in X$.

  R^{j+1} is not completely finer than the RCF
  $\Rightarrow$ $\exists R^k \in \mathit{parents}(R^{j+1})$ s.t. R^k is not finer than the RCF
  $\Rightarrow$ $\exists R^k \in X : R^{j+1} \in \mathit{children}(R^k)$ and
  R^k is not completely finer than the RCF, by assumption
  $\Rightarrow$ $R^{j+1} \in X$ from the definition of X

Lemma 5.2.2 $\forall p \in \Omega$, $\exists t \in F(R^i)$ s.t. $p \in D(t)$ and t is finer than the RCF, where $D(t)$ is the projection of t onto Ω and triangle t is finer than the RCF implies (as in Theorem 5.1.1):

R^i is finer than the RCF
or $\exists R^j$ s.t. $t \in \mathit{PreF}(R^j)$ and R^j is completely finer than the RCF

Thus the new specification of X ensures that for every point in the domain there is a triangle which contains that point, is finer than the RCF and is in the fragments of the members of X.

Proof: By Lemma 5.2.1, for every triangle t in a fragment $F(R^i)$, where $R^i \in X$, such that t is not finer than the RCF, we can find another refinement operation in X with a higher birth resolution which contains a refinement of the domain of t. The result follows by induction and our assumption regarding the RCF bisecting the Hypermesh.

Lemma 5.2.3 Let K be the set of directed edges which are projections on Ω of the directed edges in the active front $\widehat{E}$. (Hence K represents the projection on Ω of the boundary of the currently expanded mesh.) A loop invariant of the surface expansion routine is:

- the directed edges in K form a set of one or more closed loops in Ω such that the domains of these closed loops are disjoint, and;
- for all of the edges in $\widehat{E}$, there is some $\varepsilon \in [0,1]$ for which the edges are ε-*alive*. By this, we mean that each edge e of triangle t on fragment $F(R^i)$ is *alive* for the resolution interval $[birthres^i, deathres(e)]$, where $deathres(e) = deathres(t)$, and that all of the resolution intervals of the edges in $\widehat{E}$ overlap at a particular ε. This diverges from Cignoni's terminology [CPS95] due to our different scheme for handling edges in a CRMR's Hypermesh and hence we regard edges as being alive for only a subset of the intervals in which their adjacent triangles are alive.

Proof: In the first call which we make to $Infill(R^i)$, R^i is the finest node in X^+. Hence $\widehat{E} \cap E(R^i) = \emptyset$, $children(R^i) \cap X^+ = \emptyset$ and $D(F(R^i)) \cap D(\widehat{\mathcal{M}}) = \emptyset$, so $F(R^i)$ will be expanded completely. Once that has occurred, the parents of the boundary edges of $F(R^i)$ will be the only elements in K and hence $D(K) = D(F(R^i))$. Thus K will be a set of one or more closed loops in Ω. Also, $\forall \vec{e} \in \widehat{E}$, $deathres(\vec{e}) = birthres^i$, and so we can say that the edges in $\widehat{E}$ are ε-alive for $\varepsilon = birthres^i$.

Assume the loop invariant in the above Lemma is true at some stage during the expansion process – after performing $Infill(R^j)$, say, for $0 < j \leq m$.

Now we must prove that the loop invariant holds after the next call to *Infill*, which will be made for the region R^r where $R^r \in X^+$ and $\nexists l : r < l < j$ s.t. $R^l \in X^+$. Note:

$$\widehat{E} \cap E(R^s) = \emptyset \text{ for } j \leq s \leq m \tag{5.6}$$

since either $R^s \notin X^+$ or the set $\{\vec{e} : e \in \widehat{E} \cap E(R^s)\}$ will have been emptied by a previous call to $Infill(R^s)$.

We first examine the case where $\widehat{E} \cap E(R^r) = \emptyset$, $children(R^r) \cap X^+ = \emptyset$ and $D(F(R^r)) \cap D(\widehat{\mathcal{M}}) = \emptyset$. Before $Infill(R^r)$, we know that:

$$\max_{\vec{e} \in \widehat{E}} birthres(\vec{e}) < birthres^r$$
$$\min_{\vec{e} \in \widehat{E}} deathres(\vec{e}) > birthres^r$$

due to Eq. 5.6, the fact that $\widehat{E} \cap E(R^r) = \emptyset$ and the expansion of refinement operations being in finest to coarsest order. As in the first call to *Infill*, $F(R^r)$ will be expanded completely and each edge, $\vec{e}$, which is added to $\widehat{E}$ will be such that $deathres(\vec{e}) = birthres^r$. Hence we can say that $\forall \vec{e} \in \widehat{E}$, $\vec{e}$ is ε-alive for $\varepsilon = birthres^r$ before the $Infill(R^r)$ call and that this remains true after the call. Also, we have added to the mesh a fragment which was previously disjoint from the region bounded by $\widehat{E}$ and hence K will remain a set of closed loops.

Now consider the case where $\widehat{E} \cap E(R^r) \neq \emptyset$ on entry to $Infill(R^r)$ and R^r is a starting point for expansion. At this point, we know that:

$$\max_{\vec{e} \in \widehat{E}} birthres(\vec{e}) \leq birthres^r$$
$$\min_{\vec{e} \in \widehat{E}} deathres(\vec{e}) > birthres^r$$

Once the set $\{\vec{e} : e \in \widehat{E} \cap E(R^r)\}$ has been emptied by the *while* loop, the edges, $\vec{e}$, which were added to $\widehat{E}$ will have been such that $deathres(\vec{e}) = birthres^r$ and hence the edges of $\widehat{E}$ will still be ε-alive. Also, the action of "infilling" this region will have added the edges on the boundary of a set of disjoint triangles in $F(R^r)$ to K and hence K will remain a set of closed loops of directed edges.

Corollary 5.2.1 After $Infill(R^i)$ has been performed for $i = 0$, $D(K)$ will cover Ω.

Theorem 5.2.1 The triangles expanded by iterating *Infill* over X^+ form a complete triangulation, $\widehat{\mathcal{M}}$, of the domain, Ω.

Proof: The *PerformExpansion* routine expands triangles to the left of the directed edges in $\widehat{E}$. Lemma 5.2.3 indicates that these edges are ε-alive throughout the expansion process and hence triangles expanded adjacent to $\widehat{E}$ will be compatible with (i.e. will not overlap) the currently-expanded triangulation, $\widehat{\mathcal{M}}$. Corollary 5.2.1 implies that the triangulation will cover Ω on completion of the iteration of *Infill*.

Corollary 5.2.2 $t \in F(R^i)$ is expanded by $Infill(R^i)$ iff $D(t) \cap D(\widehat{\mathcal{M}}) = \emptyset$ i.e. a triangle is expanded iff no other triangles which overlap its domain have already been expanded.

Theorem 5.2.2 The expanded mesh satisfies the RCF throughout the domain, i.e.

$$\forall p \in \Omega, \exists t \in \widehat{\mathcal{M}} \text{ s.t. } p \in D(t) \text{ and } t \in F(R^i) \text{ and either:} \quad (5.7)$$
$$R^i \text{ is finer than the RCF}$$
$$\text{or } \exists R^j \text{ s.t. } t \in PreF(R^j) \text{ and } R^j \text{ is completely finer than the RCF}$$

Proof: We again use induction to prove this theorem, examining the first call to *Infill* and then a later call to the same function, and hence show that Eq. 5.7 is a loop invariant of the iteration of *Infill* over X^+.

On the first call to *Infill*, the finest refinement fragment in X^+ is expanded completely. At this stage, $X^+ = X$ and hence Lemma 5.2.1 proves that Eq. 5.7 is true.

Assume that Eq. 5.7 is true at some stage during the expansion process – after performing *Infill*(R^j), say, for $0 < j \leq m$.

Now we must prove that Eq. 5.7 holds after the next call to *Infill*, which will be made for the refinement operation R^r where $R^r \in X^+$ and $\nexists l : r < l < j$ s.t. $R^l \in X^+$

If $R^r \notin X$ then this is trivially true since, from Lemma 5.2.1, such an R^r must be completely finer than the RCF and hence the addition of any of the triangles in $F(R^i)$ to $\widehat{\mathcal{M}}$ will not affect Eq. 5.7.

If we assume $R^r \in X$ and that *Infill*(R^r) expands a triangle t from $F(R^r)$ which invalidates Eq. 5.7, i.e. either R^r is not finer than the RCF or $t \in PreF(R^s)$ and R^s is not completely finer than the RCF, then by Lemma 5.2.2, for all $p \in D(t)$,

$$\exists \bar{t} \in F(R^u) : R^u \in X, r < u < m, \text{ s.t. } p \in D(\bar{t})$$
$$\text{and } t \text{ satisfies the RCF (as in the statement of this theorem)}$$

and hence, by Corollary 5.2.2, triangle $\bar{t}$ or a refinement of $\bar{t}$ would already have been expanded and this contradicts our assumption that t is expanded by *Infill*(R^r).

Theorem 5.2.3 The minimal surface method's theorem and proof regarding the Delaunay criterion (Theorem 5.1.4) also holds for the reduced extraction method.

5.3 Summary

This chapter has presented two algorithms for producing a selectively refined mesh from a given CRMR such that the output is guaranteed to satisfy the resolution criteria specified in an RCF. The minimal surface algorithm generates the selectively refined mesh which contains the smallest set of facets which can represent the original model while satisfying the given RCF. The alternative reduced extraction algorithm takes a more direct approach to generating a selectively refined mesh and will not necessarily produce the minimal mesh.

Chapter 6

Geomorphing

The process of geometric morphing, or *geomorphing*, was introduced by Chapter 4 as a subsidiary task within the SMR framework. In this chapter we view the geomorphing process as central to the framework if a selectively refined mesh is to be animated without "popping" artifacts (Figure 6.1). We describe an algorithm which performs geomorphing on a mesh generated by the minimal surface selective refinement algorithm by smoothly adapting the mesh on a fragment-by-fragment basis. In this way we can perform geometric morphing on selectively refined meshes generated from any of the approximation methods identified previously if we are given a means to morph locally between the pre- and post-refinement facets of a refinement operation.

We introduce our geomorphing technique and its particular prerequisites in Sections 6.1 and 6.2. The geomorphing algorithm itself and proofs of the result of this process are given in Sections 6.3 and 6.4.

6.1 Introduction

The resolution criteria in an RCF may change during an animated sequence which contains a rendering of a selectively refined mesh produced by one of the SMR algorithms from the previous chapter. If the RCF contains view-dependent components, such as those described in Section 4.6.3, then movement of the viewpoint will induce a change in the RCF. This may result in different selectively refined meshes being produced for successive frames and if these were displayed without modification then the resulting temporal discontinuities would be distracting to the viewer. We therefore require a method of smoothly geometrically morphing between two such meshes.

Any correspondence between the facets of meshes generated according to two different RCFs is not immediately obvious. For example, Figure 6.2 depicts the triangles in a simple selectively refined terrain mesh which would be affected by the movement of an Area Of Interest. The AOI peak in the RCF

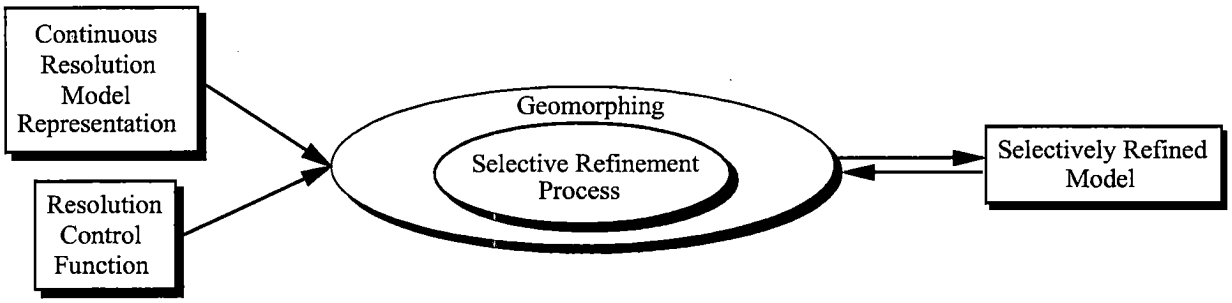


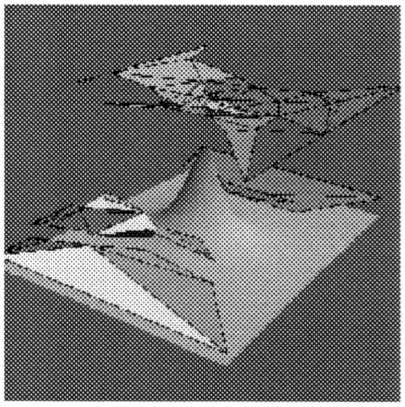
Figure 6.1: The geomorphing process can be regarded as central to our SMR framework (cf. Figure 4.1).

has been moved towards the viewer between Figures 6.2(a) and (d) (in which the expanded surface's triangles are coloured according to the refinement fragments of which they are part), with the result that the low-resolution triangles in the foreground cannot satisfy the new RCF and hence some of these have been replaced by higher resolution triangles. Figures 6.2(c) and (f) highlight the triangles which have been geomorphed, and the retriangulation into which they were morphed, due to the modification of the RCF.

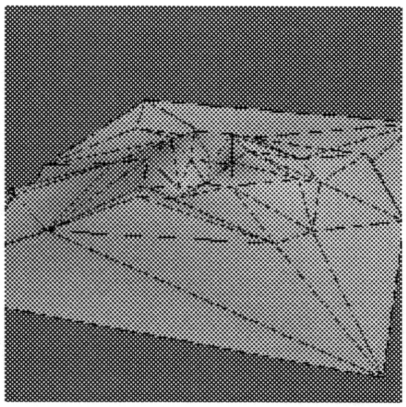
We identify the correspondences between such meshes by considering the differences between the lists of refinement operations from which the pre- and post-morph meshes can be produced, i.e. the X lists of Chapter 5. Our geomorphing process can operate on the X (or X^+) lists generated by the minimal surface and reduced extraction algorithms of the previous chapter, although the optimality of the minimal surface algorithm makes it more desirable for use in geomorphing.

The geomorphing algorithm operates by first coarsening an existing, pre-morph mesh, and then refining the resulting mesh until it meets the new resolution criteria. Thus its operation can be visualised as in Figure 6.3 where a list of refinement operations is viewed in the context of its CRMR's DAG. We visualise the coarsening phase as reducing a mesh down to one which is generated by a set of refinement operations such that this set is the common denominator of those in the pre- and post-morph X lists.

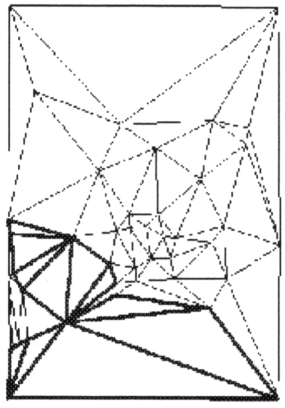
Note that our geomorphing process therefore coarsens a mesh during the transition phase in such a way that it will not necessarily meet either the pre- or post-morph RCFs. It would be trivial to reorder the coarsening and refining phases so that these criteria were satisfied throughout the transition and hence a superset of refinement operations would be the intermediary stage. This is the approach which is taken by Hoppe's geomorphing method [Hop97b], which is specific to the Progressive Mesh representation.



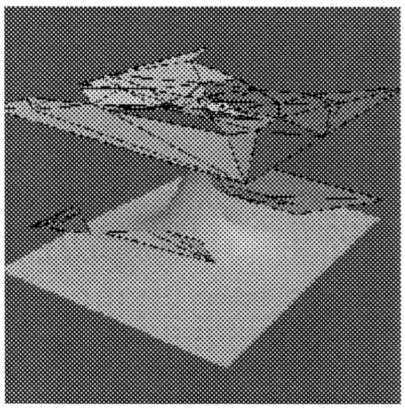
(a) Pre-morph RCF together with the expanded triangles at their death resolutions in resolution space.



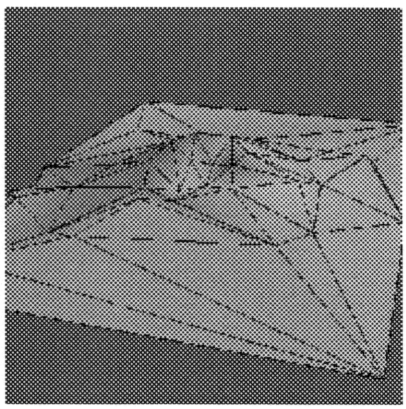
(b) Pre-morph perspective view of the expanded surface.



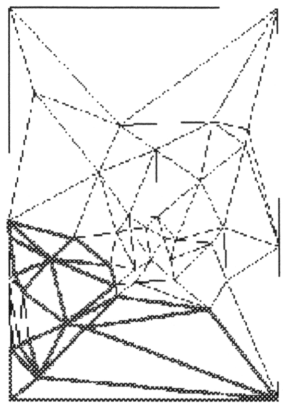
(c) Pre-morph plan view; the triangles which were replaced during the geomorph are highlighted.



(d) Post-morph RCF together with the new set of expanded triangles at their death resolutions.



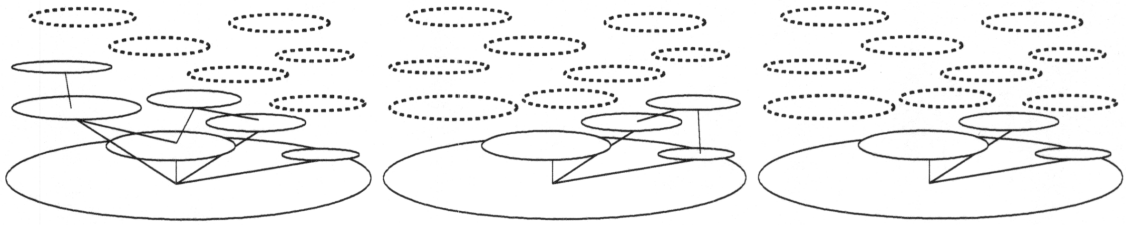
(e) Post-morph perspective view of the new selectively refined surface.



(f) Post-morph plan view; the triangles which were introduced during the geomorph are highlighted.

Figure 6.2: Simple demonstration of the operation of a geomorph on a low-resolution representation of the Mt St Helens dataset. The geomorph was necessitated by the movement of an object-space Area Of Interest in the RCF.

We take the alternative approach because additional facets may have to be introduced during the geomorphing phase to ensure a smooth traversal between two meshes and the computational savings associated with selective refinement would be lost if an excessively complex mesh was displayed during the geomorphing phase. Also, geomorphing takes place, by definition, when



(a) The pre-morph list of extracted operations, X, is shown as solid refinement fragments in the CRMR's DAG.

(b) The desired post-morph X list which corresponds to the list of refinement operations which would have been extracted if one of the SMR algorithms had been applied to the new RCF.

(c) During a geomorphing transition, the original X list is reduced to those refinement operations (drawn as solid fragments) which are common to both the pre- and post-morph lists.

Figure 6.3: The modifications made to an extracted list of refinement operations during a geomorphing transition.

either an object or the viewer is moving and hence the viewer's perception of detail will be naturally reduced during the transition phase. Thus a lower resolution mesh should be acceptable during this phase.

6.2 Requirements

This section details the minor modification which our geomorphing procedure requires of the previous selective mesh generation algorithms and also the local morphing routines which are prerequisites of the procedure. The local morphing routine which we have developed for transitions between Delaunay triangulations is described in detail.

The modification which we make to our selective mesh refinement algorithms is that we maintain a list, $C \subset X$, of refinement operations whose fragments have been completely extracted in the current selectively refined mesh. C is initialized to an empty list at the start of the surface expansion process and "starting point" refinement operations are added to it by the *Infill* routine.

The two morphing routines which our geomorphing procedures requires are denoted *MorphCoarsen*(R^i) and *MorphRefine*(R^i). These are defined as performing a smooth geometric transition between the post-refinement facets of operation R^i and the pre-refinement facets of R^i, and the reverse, respectively. We can only invoke these routines if all of the facets upon which they operate exist in the current selectively refined mesh. This constraint

governs our overall geomorphing strategy.

We associate *MorphCoarsen* and *MorphRefine* routines with a model's CRMR since their transitions between pre- and post-operation facets are dependent on the approximation method used to generate such a CRMR. The style of vertex introduction and removal which was utilised in the Progressive Mesh approximation method (shown in Figure 2.26) permits simple local transitions between pre- and post-refinement fragments. In general, though, approximation methods provide no such correspondence between their pre- and post-refinement fragments and hence a method such as Cohen-Or's for morphing between polygonal fragments could provide a non-specific basis for *MorphCoarsen/Refine*. This technique was discussed in Section 3.2.1.2.

When implementing our geomorphing algorithm for a Delaunay-generated CRMR, though, we found that Cohen-Or's fragment morphing method was not suitable because:

1. it requires every interior vertex of a fragment to be connected by an edge to the exterior of that fragment;

2. the method cannot easily be extended to handle pre-refinement triangle sets which contain interior vertices;

3. the morphing process proceeds in two phases and the latter of these is itself a number of steps, where the number of steps is unbounded and dependent on the complexity of the fragment being morphed.

As a result, we developed our own method of morphing between Delaunay triangulations which proceeds in two (single-step) phases. The information which this morphing method requires is described below and can be gathered during the incremental generation of a Delaunay triangulation.

- One phase of our local geomorphing transition involves interpolating between the location of a point and its *originator*. An originator point is associated with each inserted point, where a point's originator is defined as one of the vertices of the triangle in which the point was inserted.

- Information about any edge swapping which the Delaunay criterion necessitates is also retained. Specifically, we note the projected point where an edge and its swapped version intersect, taking into account the fact that one of the endpoints of the swapped version may be moved to the location of its originator when the swap is effected during geomorphing. We clarify this by demonstrating how we would perform a simple coarsening geomorph in Figure 6.4. Here, edge e has swapping information associated with it and this is used during the second phase

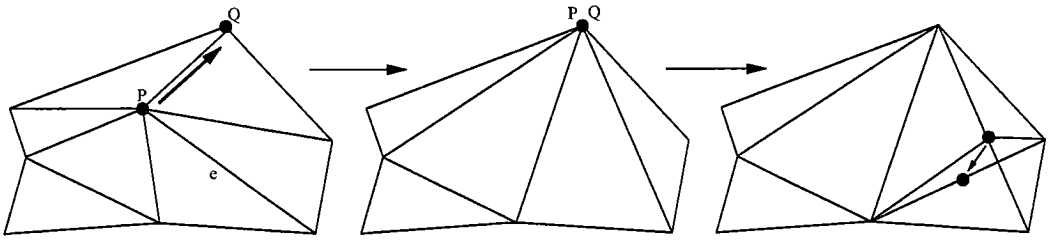


Figure 6.4: A coarsening geomorph of the finest fragment in Figure 2.14 is shown in an aerial view. Point P is initially moved to its originator, Q, and then the effect of swapping one edge within its quadrilateral is performed by splitting that quadrilaterial into four triangles and morphing them into the coarser form of the region.

of the coarsening. The reverse of this operation, refining, can be performed by invoking any swapping initially and then sliding points from their originators to their intended positions.

This example also demonstrates that some of the edge swaps which were performed during the creation of the Delaunay triangulation are not necessarily performed during geomorphing. This is due to the fact that the selection of the originators can reduce the amount of swapping which has to be effected during geomorphing. For this reason, we choose the originator of a point to be the vertex on its original surrounding triangle which is adjacent to the greatest number of edges which were swapped due to the insertion of that point.

A more complex refining geomorph is demonstrated in Figure 6.6. This uses information which was extracted from the series of Delaunay point insertions in Figure 6.5. The arrows in Figure 6.6c indicate that the geomorphing transition introduces one point by linear interpolation from an originator which is itself interpolated from its own originator. This is how we remove Cohen-Or's requirement to have interior points associated with ones on the boundary of a fragment – our linear interpolation process can be recursive.

To conclude this section, we note that the discrete and scalar attributes associated with the facets of a refinement fragment can be introduced during geomorphing by interpolating them as Hoppe described for the Progressive Mesh representation [Hop96].

6.3 Algorithm

Our geomorphing algorithm proceeds in the following two phases. Note that both the list X of refinement operations extracted for the current mesh and

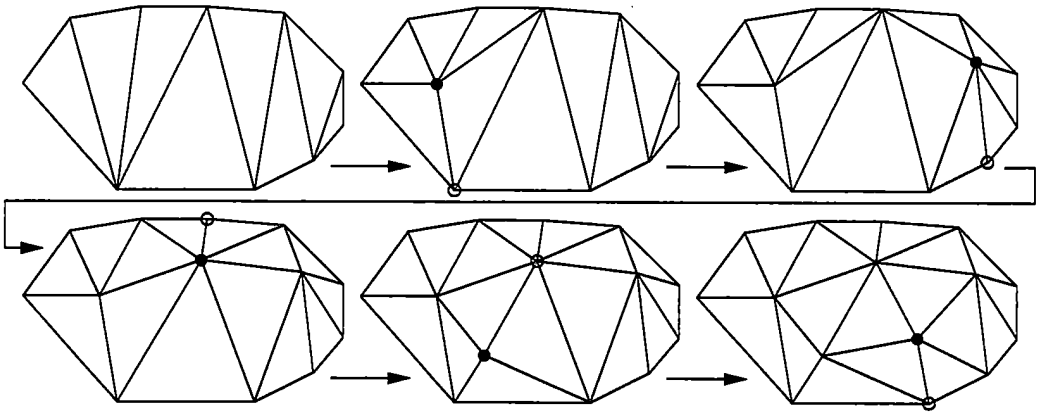


Figure 6.5: The finer fragment in Figure 3.10 as generated by a series of point insertions which satisfy the Delaunay criterion. Each new point is denoted by a filled circle and its originator by an empty circle.

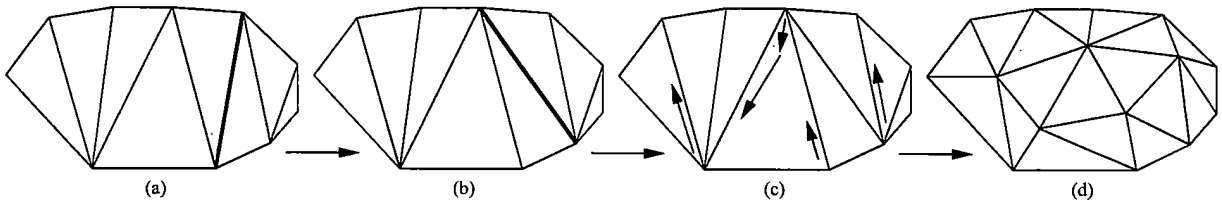


Figure 6.6: The process by which the information from Figure 6.5 can be used to smoothly refine the fragment. A single edge swap is effected between (a) and (b) by treating the quadrilateral in which the thick lines lie as four triangles and morphing appropriately in that region. The interior points can then introduced smoothly as indicated by the arrows in (c).

the list C of operations which could potentially be coarsened must have been created by a selective refinement algorithm or a previous geomorph.

1. **Coarsening**
   We repeatedly identify refinement operations in X whose fragments have been completely expanded in the current mesh and which are unnecessary for the new RCF.

   On each iteration, we traverse list C in order of decreasing resolution until we find an operation $R^i \in C$ which is completely finer than the RCF. We then render the mesh by performing the surface expansion part of the selective refinement algorithm and call *MorphCoarsen*(R^i) when R^i is encountered during the expansion process. Once the visual morph of this region has been completed, we can delete R^i from X so that X now reflects the set of refinement operations from which the displayed mesh originates. Note that the action of expanding a mesh will have modified the list C in preparation for the next iteration of this coarsening step.

   When no operations in C are completely finer than the new RCF, X contains the set of refinement operations which are common to both the pre- and post-geomorph meshes. This is the intermediary point indicated by Figure 6.3c.

2. **Refining**
   We traverse the list of extracted operations, X, from its coarsest element to its finest in order to further extract some of the operations from the CRMR and hence ensure the fragments of the operations in X are sufficient to permit the expansion of a mesh which will satisfy the new RCF.

   If an operation $R^i \in X$ is not finer than the RCF then we perform a set of refining morphs which are compatible with the current mesh and which result in all of the members of $children(R^i)$ which are not completely finer than the RCF being members of X. Specifically, for such a region R^i, we create a set containing the elements of $children(R^i)$ which are not completely finer than the RCF as well as the ancestors of these operations which are not already in X. The fragments of this set are then morphed into the current mesh by performing *MorphRefine* on the elements of the set in order of increasing resolution.

   The simplicity of this refining step is demonstrated by the following pseudo-code. First, we determine the set, Y, of operations which must be added to X in order to permit the expansion of a mesh which satisfies the new RCF:

if R^i is not finer than the RCF **then**
  for each $R^j \in children(R^i)$ **do**
    if R^j is not completely finer than the RCF and $R^j \notin X$ **then**
      AddToList(R^j, Y)
    endif
  endfor
endif

Now we must add the fragments from the operations in Y to the existing mesh in such a way that the additions are compatible with the current mesh at all times:

while $Y \neq \emptyset$ **do**
  remove R^k from Y where R^k is the element of Y with lowest birth resolution
  $X := X \cup R^k$
  expand mesh, calling *MorphRefine*(R^k) when R^k is encountered
endwhile

A degree of parallelisation can be introduced into the two phases of the geomorphing process described above. Where these coarsening and refining steps remove a single operation from a list (C in the former and Y in the latter), we can permit a group of operations to be removed if there are no dependencies between these operations. For the coarsening step, suitable operations can be removed from C until one is encountered which is a parent of those which are already in the removed group. At this point, a mesh can be generated and the appropriate local geomorphs can be performed in parallel. Similarly, we can remove a group of elements from Y in the refining step if each element of this group is such that none is a child of another in the group.

If there is a requirement to perform a geomorph within a certain time limit and parallelisation is not sufficient to meet this requirement then we can enforce a maximum *geomorphing period*. This is defined to be the period between the start of successive coarsening phases. To implement such a constraint, we start a timer when a coarsening phase is entered and only perform each necessary coarsening or refining if there is sufficient time remaining in the current geomorphing period. In addition, the refining phase is only entered if the maximum geomorphing period has not been exceeded at that point. This means that coarsenings are more likely to be performed during a time-limited geomorph which naturally assists with reducing the current geomorphing period since the complexity of the displayed approximation will be progressively reduced if necessary.

Our implementation enforces a maximum geomorphing period while the user is rotating a selectively refined mesh in our interactive application. When movement of the model ceases, the maximum geomorphing period is removed so that any pending coarsenings or refinings can be performed.

Thus the detail in the selective refinement can "catch up" with the demands imposed by the user.

6.4 Proofs of geomorphed mesh properties

We consider the effect of a geomorph on a list of extracted operations X which has been generated by the minimal surface selective refinement algorithm. We show that the coarsening step ensures that the list X satisfies some "common denominator" RCF during the transition.

We assume that the refinement operation extraction step of the minimal surface algorithm has been performed for an RCF, RCF_{old}, and that the geomorphing step is intended to transform the set of operations in X into one from which a mesh can be expanded which will satisfy a modified RCF, RCF_{new}.

Corollary 6.4.1 The above assumption implies that the operations which are initially in X are such that they meet the definition given on page 94 with respect to RCF_{old}, i.e.

$$\begin{aligned} R^i \in X \iff\ & i = 0 \text{ or} \\ & (\exists R^j \in X : R^i \in \mathit{children}(R^j), R^j \text{ is not finer than} \\ & RCF_{old} \text{ and } R^i \text{ is not completely finer than } RCF_{old}) \\ & \text{or } \exists R^k \in X : R^i \in \mathit{ancestors}(R^k) \end{aligned}$$

Theorem 6.4.1 The operations in X after the coarsening step are those which could have been placed in X if the operation extraction step of the selective refinement algorithm had been performed for a "common denominator" RCF, RCF_{cd}, where this is defined implicitly by:

- R^i is finer than $RCF_{cd} \iff R^i$ is finer than both RCF_{old} and RCF_{new};
- R^i is completely finer than $RCF_{cd} \iff R^i$ is completely finer than both RCF_{old} and RCF_{new}.

Proof: We are setting out to verify that after the coarsening step the operations in X satisfy:

$$\begin{aligned} R^i \in X \iff\ & i = 0 \text{ or} \\ & (\exists R^j \in X : R^i \in \mathit{children}(R^j), R^j \text{ is not finer than} \\ & RCF_{cd} \text{ and } R^i \text{ is not completely finer than } RCF_{cd}) \\ & \text{or } \exists R^k \in X : R^i \in \mathit{ancestors}(R^k) \end{aligned} \tag{6.1}$$

We first prove that:

$$\begin{aligned} &i = 0 \text{ or} \\ &(\exists R^j \in X : R^i \in children(R^j), R^j \text{ is not finer than} \\ &RCF_{cd} \text{ and } R^i \text{ is not completely finer than } RCF_{cd}) \\ &\text{or } \exists R^k \in X : R^i \in ancestors(R^k) \quad \Rightarrow \quad R^i \in X \end{aligned}$$

The coarsening step removes operations from X only if they have been placed on list C and are completely finer than RCF_{new}. Hence only operations which are completely finer than RCF_{cd} will be removed from X and thus both

$$i = 0 \Rightarrow R^i \in X$$

and

$$\begin{aligned} &\exists R^j \in X : R^i \in children(R^j), R^j \text{ is not finer than } RCF_{cd} \text{ and} \\ &R^i \text{ is not completely finer than } RCF_{cd} \Rightarrow R^i \in X \end{aligned}$$

will remain true.

Also, the definition of the surface expansion "starting point" fragments whose operations are placed on C implies that

$$\forall R^i, children(R^i) \cap X \neq \emptyset \Rightarrow R^i \notin C$$

and therefore the coarsening step will never remove an operation which is an ancestor of another on X. Hence the statement $\exists R^k \in X : R^i \in ancestors(R^k) \Rightarrow R^i \in X$ will remain true.

The reverse implication of statement (6.1) can be proved by contradiction. Assume that after the coarsening step an operation R^i exists in X which is not R^0 but is completely finer than RCF_{cd} and is such that $\nexists R^k \in X : R^i \in ancestors(R^k)$. This last point means that R^i would have been placed on list C before the coarsening step was invoked and, since our assumption implies that R^i is completely finer than RCF_{new}, it would have been removed from X by the coarsening step. This contradicts our assumption.

Therefore the refining step, which is effectively the minimal surface method's refinement operation extraction step, acts on an existing list of operations which satisfy the properties identified in Section 5.1.4. Hence the result of the geomorphing process is a list X which is identical to the one which would have been produced if the post-morph RCF had been applied to the selective refinement algorithm directly. It follows that the geomorphed mesh satisfies the minimal surface properties. (Corresponding properties can be demonstrated for a geomorph applied to a mesh produced by the reduced extraction algorithm.)

6.5 Summary

The task of geomorphing is closely linked to that of selective mesh refinement. Geomorphing is necessary, for example, to ensure that a selectively refined surface matches view-dependent resolution criteria during an animated sequence which contains that surface. We have presented a geomorphing algorithm which can smoothly modify an existing selectively refined mesh into another which matches new resolution criterion. This algorithm maintains our desire for independence from the underlying approximation method by invoking simple local morphing routines.

Chapter 7

Results

This chapter presents the results of applying the selective mesh refinement and geomorphing algorithms to a selection of $2\frac{1}{2}$- and 3- dimensional models. Section 7.1 describes the pre-processing which was performed to generate CRMRs from these datasets. Section 7.2 then presents the selectively refined meshes which were obtained by applying various RCFs to these CRMRs. Section 7.3 presents the results of geomorphing these meshes due to changes in the resolution criteria caused by traversing over the meshes. Finally, Section 7.4 compares our results with those of contemporary papers.

7.1 CRMR generation

The following sections describe how both terrain datasets and three-dimensional models were converted into Continuous Resolution Model Representations. A Delaunay triangulation approximating method was applied to the terrain surfaces and Hoppe's Progressive Mesh approximation technique was applied to both classes of model. For comparison with other representations, Section 7.1.3 contains details of the space requirements of the resulting CRMRs.

7.1.1 Delaunay CRMR generation

The Delaunay selector criterion which was described in Section 2.2.2.1 was used to produce a CRMR for several terrain datasets obtained from the US Geological Survey. Starting with a planar triangulation of each terrain's domain, points were inserted in sequence in accordance with the selector criterion. Each local retriangulation invoked by these point insertions was treated as a refinement fragment and inserted into the corresponding CRMR's Hypermesh. Points were inserted in parallel when necessary to ensure that these fragments were produced with monotonically increasing birth resolutions and

hence that the approximation method could be treated as a directly compatible method (as defined by Section 4.4.3.2). Thus we ensured that each fragment added to the CRMR was in non-decreasing order of resolution, as required by Section 4.4.

The replacement of triangles by higher resolution versions during each point insertion in this process was monitored to obtain the CRMR's DAG which reflected the overlapping nature of the fragments in the corresponding Hypermesh.

Statistics obtained during the production of our example CRMRs are presented in Table 7.1. This indicates that approximately 4% of the original datasets' points were inserted into each approximation. Note also that the resulting CRMRs had approximately half this number of refinement fragments which implies that an average of two points were inserted in each step of the approximation process.

Dataset	Points in dataset	Highest resolution mesh, $\mathcal{M}^m$			Fragments in CRMR
		Points	Triangles	Render time(s)	
Mt St Helens	145360	6000	11863	0.24	3045
Emory Peak	160801	6000	11875	0.24	3303
Crater Lake	153765	6000	11845	0.20	3102
Honolulu	196024	6000	11893	0.26	3150

Table 7.1: Terrain datasets and generated CRMRs.

The CRMRs generated by producing approximations of the US Geological Survey's Mt St Helens, Emory Peak, Crater Lake and Honolulu datasets in the manner described above are illustrated in Figure 7.1. This figure shows the highest resolution meshes which could be obtained by applying the surface expansion algorithm to the complete set of fragments in each CRMR. As is the convention in this chapter, the boundaries of the view frustum of the perspective scenes in this figure are drawn as yellow lines in their corresponding plan views.

The time to render each of these high resolution terrain meshes is given in Table 7.1. All of the timing results in this chapter were obtained on an SGI Indigo^2 175MHz R10000 Solid Impact with 64Mb RAM. Note that all timings refer to the time required to display the filled triangles of a particular mesh without rendering their wireframe outlines.

The CRMR Hypermesh and DAG which were produced during the approximation of the Mt St Helens dataset are shown explicitly in Figure 7.2. The fragments of the Hypermesh have been coloured randomly and are positioned in resolution space at their birth resolutions in Figure 7.2a. Each node of the DAG in Figure 7.2b is represented by a red sphere drawn at the

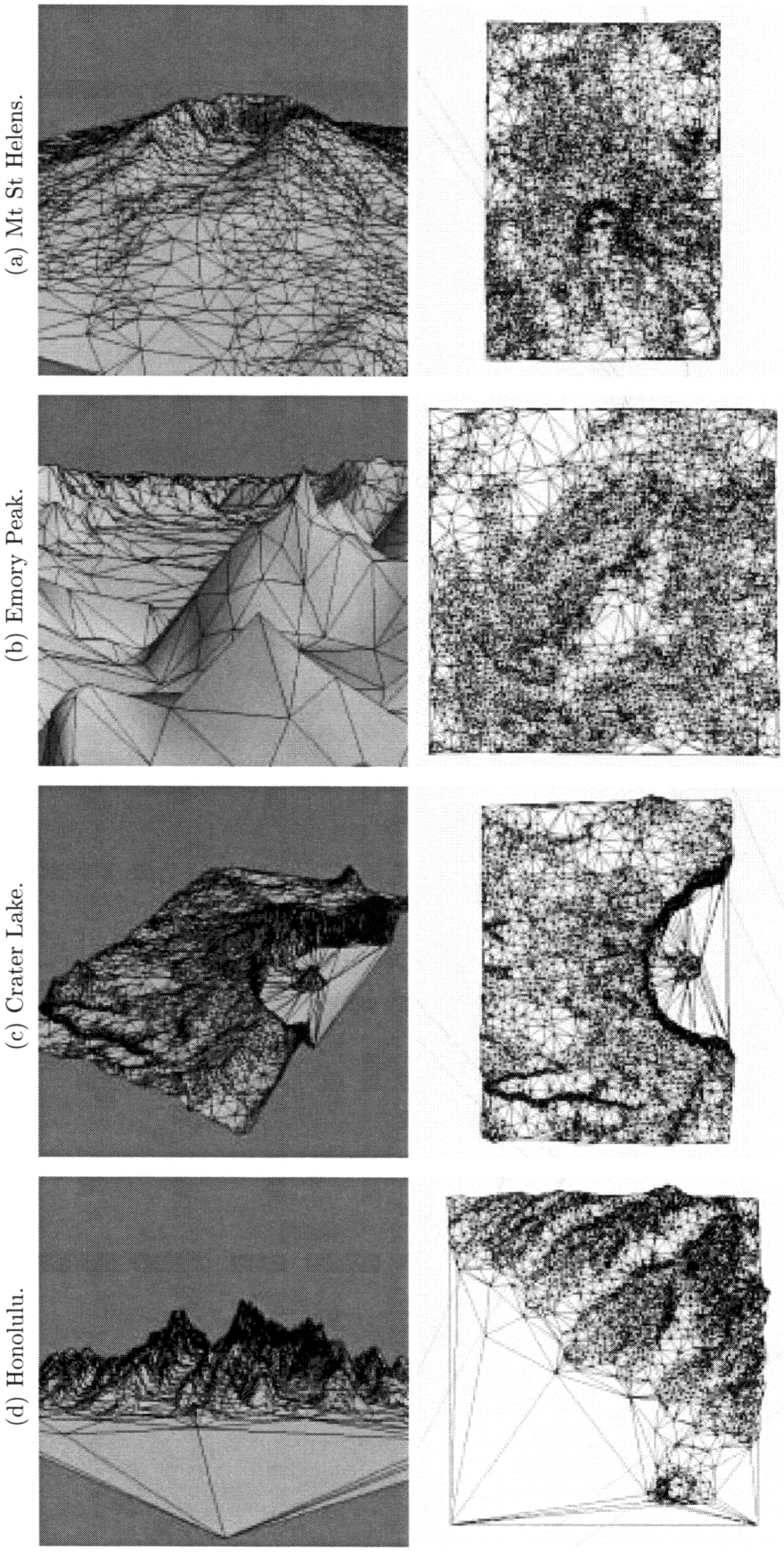
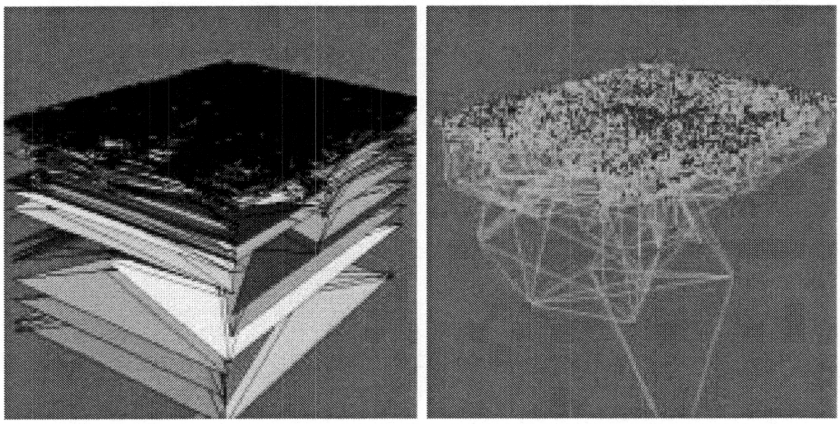


Figure 7.1: The highest resolution meshes obtainable from our terrain CRMRs.

centre of a triangle in the corresponding refinement fragment, positioned at its birth resolution in resolution space, and the arcs of the DAG are drawn as green lines. This demonstrates that the DAG is highly interconnected.



(a) Fragments of the Mt St Helens CRMR in resolution space.

(b) Mt St Helens CRMR's DAG drawn in resolution space.

Figure 7.2: The Mt St Helens Delaunay-generated CRMR and DAG.

7.1.2 Progressive Mesh CRMR generation

The Progressive Mesh approximation technique (Section 2.3.4) was used to produce CRMRs of several VRML models. This was performed primarily by modifying Hoppe's publicly-available code[1], which implemented his original surface simplification and reconstruction techniques [HDD+93], to comply with his later Progressive Mesh paper [Hop96, Hop97a].

A further conversion routine was required to produce CRMRs from the Progressive Mesh representations which this code generated. This conversion was necessary because the PM energy optimisation function is dependent on terms other than geometric similarity and we wished to produce selective refinements with respect to geometric resolution requirements. We therefore reordered and/or coalesced the PM fragments as described in Section 4.4.3.3 to ensure that they were in monotonic order with respect to the Hausdorff distance resolution metric.

The statistics of the base and highest resolution meshes obtainable from the resulting CRMRs of several VRML models are detailed in Table 7.2. These meshes are illustrated in Figure 7.3.

Table 7.2 also includes an entry for a CRMR of Mt St Helens which was obtained via the Progressive Mesh approach. This CRMR was generated by applying the Progressive Mesh approximation technique to the 6000 point

[1] http://www.cs.washington.edu/research/projects/grail2/www/software-data.html

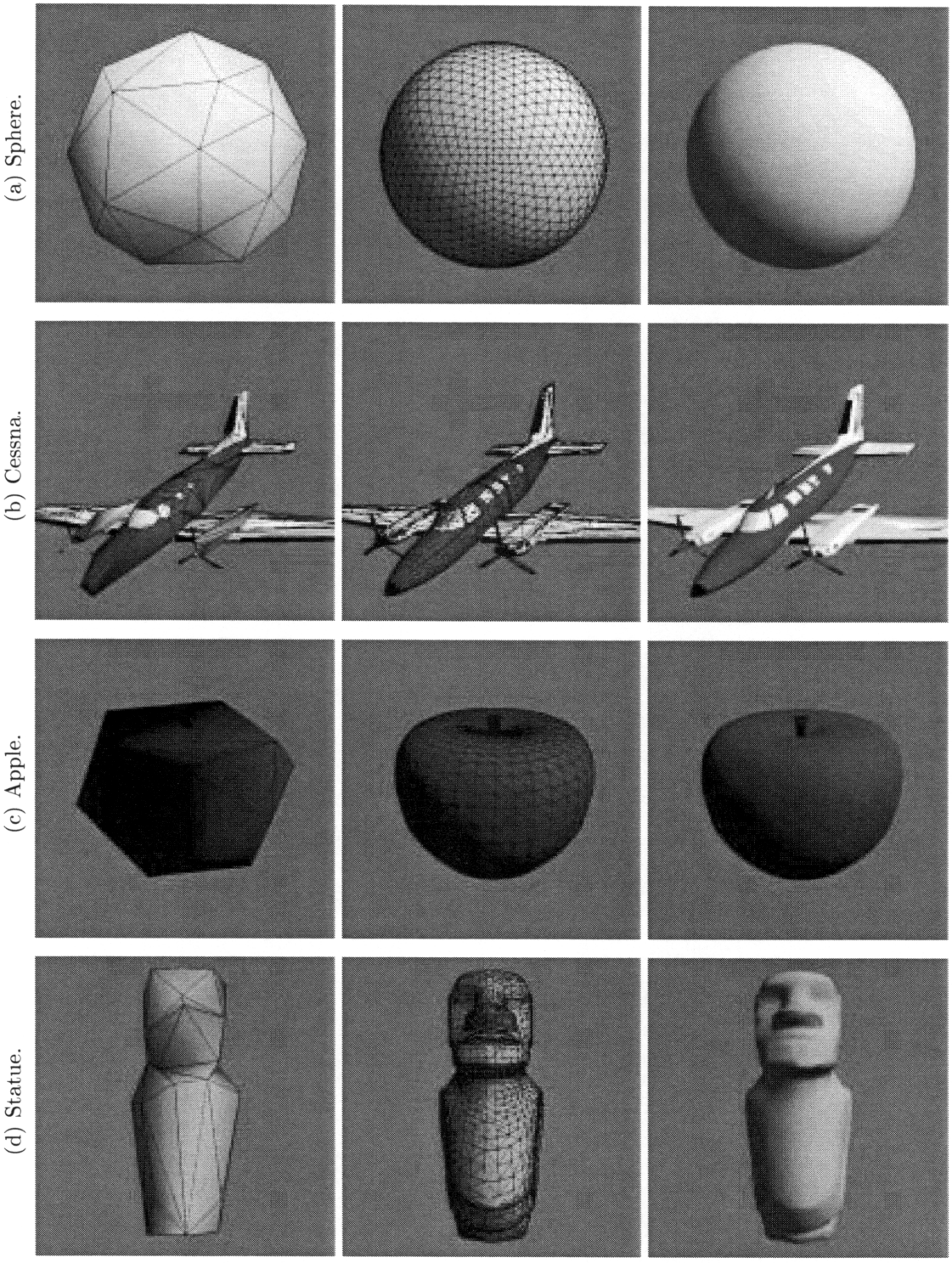


Figure 7.3: The base and highest resolution meshes obtainable from the CRMRs of 3D models which were used in this chapter. The base meshes are shown on the left; the highest resolution meshes ($\mathcal{M}^m$) are drawn in the middle column with wireframe outlines and on the right without.

Model	Base mesh, $\mathcal{M}^0$			Highest resolution mesh, $\mathcal{M}^m$			Fragments
	Pts	Tris	Render time(s)	Pts	Tris	Render time(s)	in CRMR
Sphere	26	49	0.01	1602	3200	0.10	1415
Cessna	314	586	0.02	3745	7446	0.28	2592
Apple	22	37	0.01	867	1704	0.06	710
Statue	21	40	0.01	2490	4976	0.16	2088
Mt St H.	109	115	0.01	6000	11863	0.24	4148

Table 7.2: 3D models approximated using Progressive Meshes and converted into CRMRs.

model of Mt St Helens depicted in Figure 7.1a. This CRMR contains significantly more fragments than the corresponding Delaunay version which reflects the fact that only one vertex is inserted by each standard PM fragment (before coalescing).

7.1.3 CRMR space requirements

Although our CRMR format was not primarily designed for space efficiency, the space required by the PM-generated CRMRs of our example models is presented in Table 7.3 for comparison with some alternative representations of these models.

This table gives the size of the VRML files from which these CRMRs were derived, for both the VRML ASCII-format file and its gzipped version (which is currently the only agreed standard for compressed VRML files). The gzipped size of our own CRMR files is contained in the next column of the table. Finally, an estimate of the size of the gzipped files which would be generated by Hoppe's implementation of his PM representation is given. These last numbers were obtained by multiplying the number of points in each dataset by 24.9 bytes, which is the average number of bytes per vertex in the PM statistics presented by Hoppe in [Hop97b].

Firstly, note that the figures for the VRML sphere files appear anomalous because the sphere is a basic primitive in VRML and hence it can be represented by a single VRML node. All of the other models had to be represented by a number of "IndexedFaceSet" nodes, i.e. sets of vertices and vertex indices.

The main conclusions which can be drawn from these statistics are that the gzipped VRML files are almost always smaller than the PM representations and that these in turn are significantly smaller than our CRMRs. The relatively small increase which a PM representation requires over its corresponding VRML version is offset by the advantages which can be derived

Dataset	VRML		CRMR	PM
	ASCII	Gzipped		
Sphere	140bytes	138bytes	104K	40K
Cessna	277K	71K	227K	93K
Apple	122K	30K	52K	22K
Statue	159K	55K	298K	62K
Mt St Helens	624K	121K	655K	149K

Table 7.3: Comparison of the space required by alternative representations of example datasets.

from the continuous resolution nature of a PM representation. The 2-4 times increase which our CRMR format represents over its PM counterpart can be accounted for by the fact that a CRMR makes no assumptions about the approximation technique which was used to create its fragments and hence has to represent each fragment completely.

7.2 Selective Mesh Refinement results

This section details the results of applying our selective mesh refinement algorithms to the CRMRs described above. Specifically, we present the outputs obtained by applying RCFs which were based on object-space and screen-space terms to these CRMRs. Selective refinement of the Delaunay-based CRMRs is considered in Section 7.2.1 and that of the Progressive Mesh CRMRs in Section 7.2.2.

7.2.1 SMR of Delaunay CRMRs

Our consideration of the results obtained from selectively refining the example Delaunay CRMRs is divided into the application of object-space RCF functions (Section 7.2.1.1) and screen-space RCFs (Section 7.2.1.2).

7.2.1.1 Object-space RCFs

Section 4.6.2 described how suitable object-space functions could be applied as RCFs to ensure that the critical features of terrain surfaces were retained in selectively refined versions of the surfaces. This section uses object-space RCF functions which were generated by a combination of manual and automatic techniques to produce selectively refined terrain meshes. The particular features of each of the meshes in Figure 7.1 are considered in turn, with

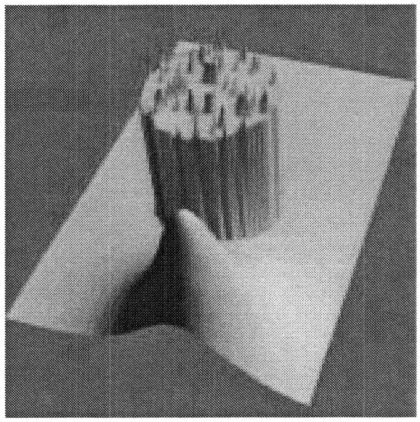


Figure 7.4: Mt St Helens object-space RCF function used to generate Figure 7.5.

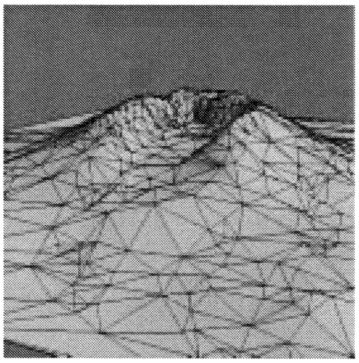
the aim of producing a selectively refined mesh which is appropriate to the perspective views in that figure.

Mt St Helens The object-space RCF function which we applied to the Mt St Helens CRMR was intended to highlight the crater in the centre of the dataset as well as an area in the foreground of the perspective view. Figure 7.4 views this function in resolution space. In resolution terms, this function can be described as having a base level of 70%, a circular thresholded region in the centre of the domain which rises to 95%, critical-line-detected elements which rise a further 3% above this threshold and a Gaussian element in the foreground which peaks at 99%.

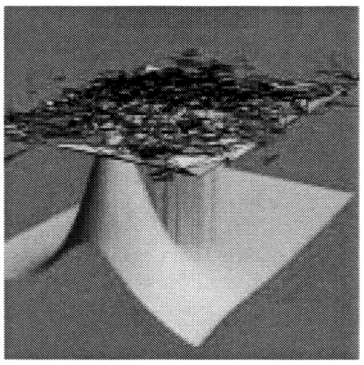
The semi-circular nature of the crater can be observed in the elements added by the critical line detection routine (Section 2.2.3). There is little connectivity between these elements because we found that this level of line-detection was sufficient for input to the SMR process without applying Peucker and Douglas' suggested methods for ensuring connectivity.

The output mesh obtained from the minimal surface SMR algorithm is given in Figure 7.5a. Comparison with Figure 7.1a shows that the high triangle density around the crater region has been maintained (in particular, the details of the ridge line of the crater have been retained) while the density outside this region has been drastically reduced. Figure 7.5b shows the triangles of the expanded surface in the same resolution space as the RCF object-space function and hence we can verify visually that the refined surface satisfies this function since all of the triangles lie above the RCF function.

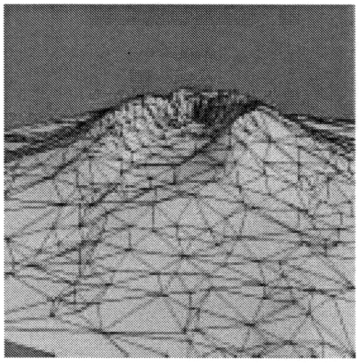
The reduced triangle count of this selectively refined mesh is reflected in the statistics of this mesh in the first two rows of Table 7.4. The columns of



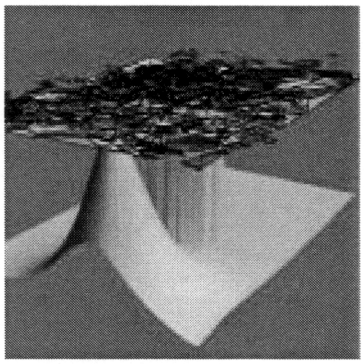
(a) Mesh generated by the minimal surface algorithm according to the object-space function of Figure 7.4 and with frustum and back-face culling enabled in the RCF.



(b) Triangles of (a) drawn at their death resolutions together with the object-space RCF function from Figure 7.4.



(c) Mesh generated by the reduced extraction algorithm according to the object-space function of Figure 7.4 and with frustum and back-face culling enabled in the RCF.



(d) Triangles of (c) drawn at their death resolutions together with the object-space RCF function.

Figure 7.5: The minimal surface algorithm applied to the Mt St Helens CRMR of Figure 7.2 and the object-space RCF function of Figure 7.4 generated the top row of images; the reduced extraction algorithm generated the corresponding bottom row.

this table are:

- the number of extracted refinement operations, $|X|$;
- the number of extracted refinement operations after surface expansion for the reduced extraction case, $|X^+|$;
- the time to extract the elements of list X;
- the number of triangles in the expanded surface, $|\hat{F}|$;
- the average number of triangles per triangle fan in the expanded surface;
- the time to traverse and render the expanded surface, and;
- the total time to extract and expand the surface.

The second row of the table indicates that no benefit was gained from applying the frustum and back-face culling RCF tests to the Mt St Helens scene since the resolution of that selectively refined mesh was already low outside the view frustum. The resulting minimal surfaces, though, could be rendered in 0.03s and the combined extraction and expansion time was less than that required to render the highest resolution surface in the CRMR.

The "triangles/fan" column in Table 7.4 shows that the surface expansion of a selectively refined mesh produced from a Delaunay-generated CRMR gives a consistent average of 1.3 triangles per triangle fan. This represents a significant saving over specifying each triangle individually, but the relatively low value reflects the fact that neither the refinement fragments generated by Delaunay triangulation nor the surface expansion procedure are particularly suited to the application of triangle fans.

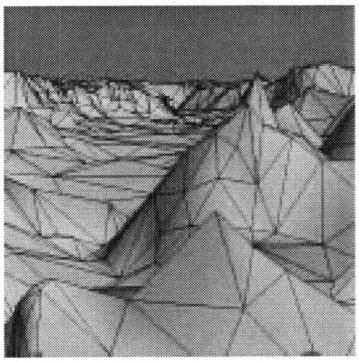
The reduced extraction algorithm produced a selectively refined mesh of Mt St Helens which was perceptually similar to that produced by the minimal surface algorithm for the same object-space RCF function (Figure 7.5c). The results in Table 7.4 show that the algorithm performed as expected, i.e. the number of refinement operations extracted was initially lower than for the minimal surface algorithm and this situation was reversed after the surface was expanded. As a result, the number of triangles in the expanded surface was higher than for the minimal surface algorithm but the extra cost of rendering these triangles had no impact on the overall time saving.

Emory Peak A similar object-space RCF function was applied to the Emory Peak CRMR, with a Gaussian Area Of Interest around the viewpoint and a circular thresholded Area Of Interest in the centre of the view frustum (Figure 7.6c). The top row of Figure 7.6 presents the output of the minimal surface algorithm applied to this object-space function with frustum and back-face

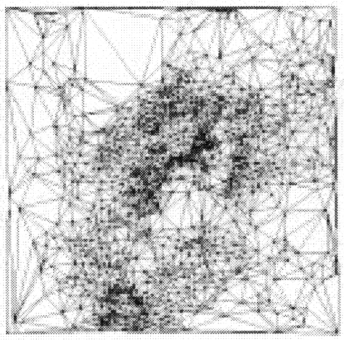
CRMR	Ref. op. extraction			Surface expansion			Total
Algorithm	$\|X\|$	$\|X^+\|$	Time(s)	$\|\hat{F}\|$	Triangles/fan	Time(s)	time(s)
Mt St Helens							
M.S., culling	462	-	0.20	1708	1.35	0.03	0.23
M.S., no culling	462	-	0.20	1708	1.35	0.03	0.23
R.E., culling	236	541	0.15	2235	1.36	0.04	0.19
Emory Peak							
M.S., culling	1052	-	0.22	3744	1.32	0.08	0.30
M.S., no culling	1184	-	0.28	4115	1.30	0.09	0.37
R.E., culling	801	1309	0.19	4781	1.32	0.09	0.28
Crater Lake							
M.S., culling	2256	-	0.54	7189	1.28	0.18	0.72
M.S., no culling	2265	-	0.54	7206	1.28	0.17	0.71
R.E., culling	2006	2335	0.42	7503	1.28	0.24	0.66
Honolulu							
M.S., culling	2336	-	0.43	8207	1.31	0.21	0.64
M.S., no culling	2396	-	0.43	8343	1.30	0.21	0.64
R.E., culling	2099	2434	0.39	8620	1.31	0.27	0.66

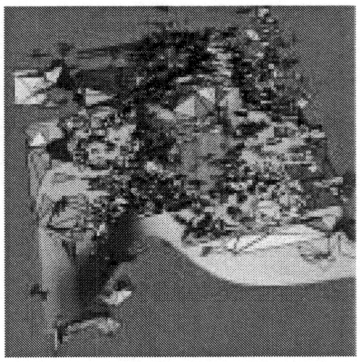
Table 7.4: Results of applying the minimal surface (M.S.) and reduced extraction (R.E.) algorithms to the Delaunay CRMRs with object-space RCF functions.

culling enabled. As Figure 7.6b demonstrates, this means that culled triangles may lie beneath the object-space RCF function in resolution space if they are not required to satisfy that function. Figures 7.6(d) and (e) present the mesh which was generated when culling was disabled.

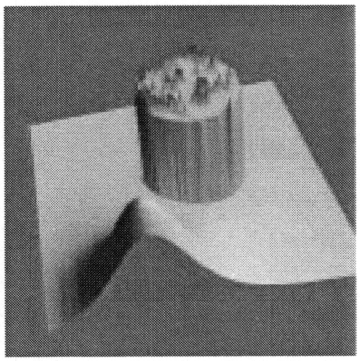


(a) Mesh generated with frustum and back-face culling enabled in the RCF.

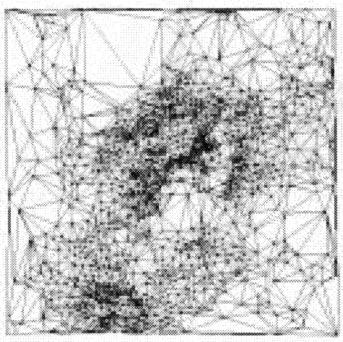




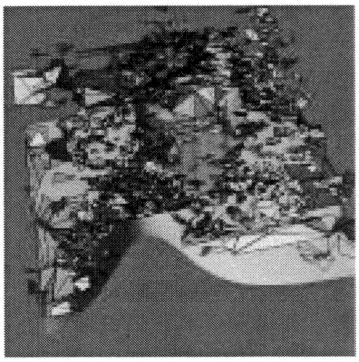
(b) Triangles of (a) drawn at their death resolutions together with the object-space RCF function of (c).



(c) Object-space RCF function.



(d) Mesh generated without frustum or back-face culling.

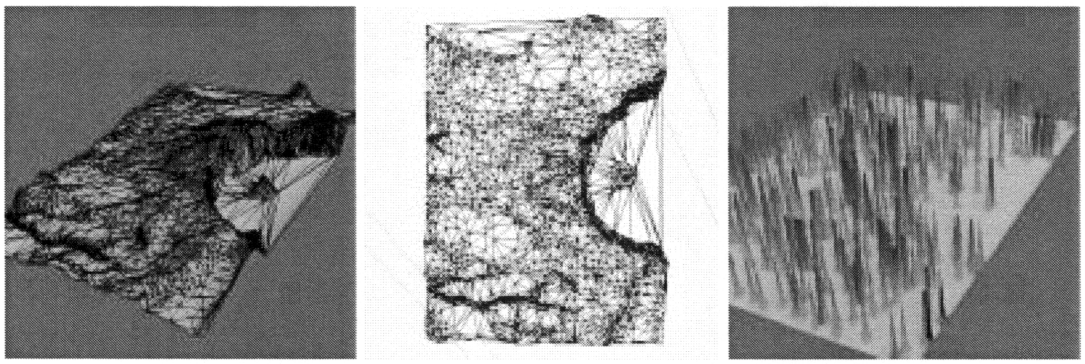


(e) Triangles of (d) drawn at their death resolutions together with the object-space RCF function.

Figure 7.6: Meshes output when the minimal surface algorithm was applied to the Emory Peak CRMR and an object-space RCF function. The top row shows the output generated when frustum and back-face culling was applied; this was disabled to generate (d) and (e).

The figures in Table 7.4 for the Emory Peak CRMR are similar to those of the Mt St Helens example, although the overall time for extraction and expansion was not lower than the time to render the CRMR's highest resolution mesh in this case.

Crater Lake An RCF object-space function which was generated solely by our critical line detection routine was applied to the Crater Lake CRMR. This function is shown in Figure 7.7b and represents a base resolution level of 90% together with critical-line-detected elements which rise 8% above that base. The planar "crater" area is clearly specified in this function as being of low importance.



(a) Mesh generated with frustum and back-face culling in the RCF.

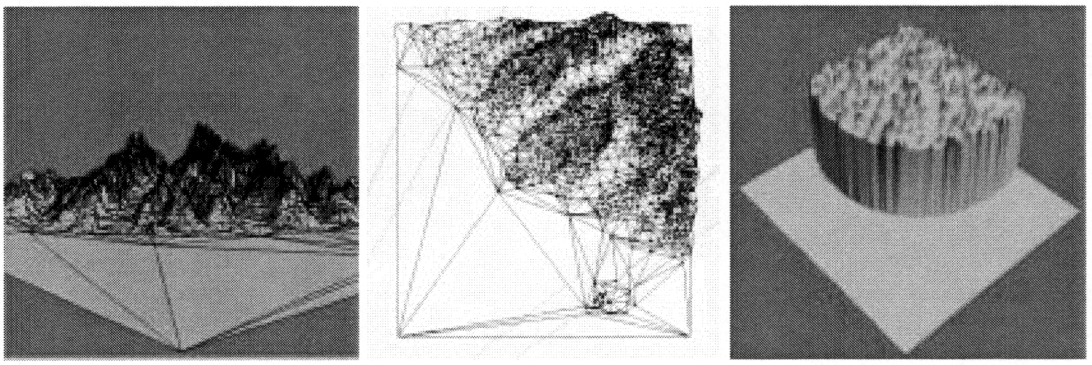
(b) Object-space RCF function.

Figure 7.7: Mesh output when the minimal surface algorithm was applied to the Crater Lake CRMR and the object-space RCF function in (b).

The output from the minimal surface algorithm applied to this RCF function with culling enabled is given in Figure 7.7a. This mesh is very similar to the high resolution version in Figure 7.1c, although areas in which large triangles are present can be observed to contain less detail. Table 7.4 also demonstrates that this selectively refined mesh contains a larger proportion of the high resolution version's triangles than in the previous examples. The time to render this expanded surface is correspondingly higher, although less than the time required to render the highest resolution mesh.

Honolulu A thresholded Area Of Interest augmented by critical-line-detected elements was used as the object-space RCF function for the Honolulu CRMR. Figure 7.8b illustrates that the valleys in this dataset are visible between the critical line elements in the object-space function. This RCF produced the surface illustrated in Figure 7.8a when applied via the minimal surface algorithm.

In this case, the time to perform surface expansion under the reduced extraction scheme was longer than the time to display the highest resolution mesh. This was caused by the severe "step" in the RCF object-space function leading to a large number of refinement operations being added to the list X^+ during the surface expansion phase.



(a) Mesh generated with frustum and back-face culling in the RCF.

(b) Object-space RCF function.

Figure 7.8: Mesh output when the minimal surface algorithm was applied to the Honolulu CRMR and the object-space RCF function in (b).

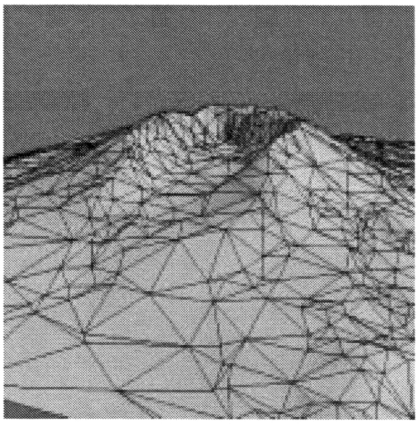
7.2.1.2 Screen-space RCFs

This section presents images and timings obtained from applying screen-space RCFs to the terrain CRMRs using the minimal surface algorithm.

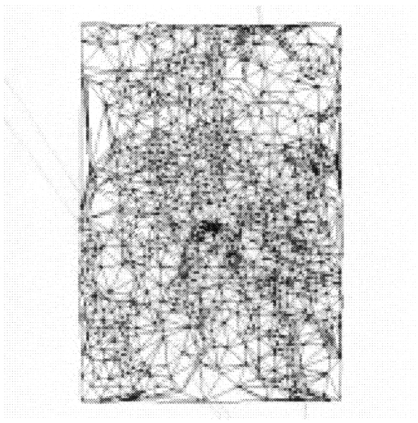
Figure 7.9 illustrates two meshes which were obtained from the Mt St Helens CRMR for a typical screen-space RCF which combined a particular screen-space tolerance with frustum and back-face culling. (Recall from Section 4.6.3 that the tolerances are expressed as percentages of the viewport's width.) The effect of this RCF is particularly noticeable in Figure 7.9d, where a low screen-space tolerance ensured that the triangle density inside the view frustum was high while the frustum-culling component reduced the density outside the viewspace. The statistics associated with these meshes are presented in the first two rows of Table 7.5. Note that this table (and its successors in the remainder of this chapter) differs from Table 7.4 in that the column labelled $|X^+|$ has been omitted since the reduced extraction algorithm is not compatible with a screen-space RCF (Section 5.2).

All of the entries in this table demonstrate that the triangle count and computational cost of a selectively refined mesh is inversely proportional to its screen-space tolerance. Thus the cost of displaying a scene is directly proportional to its quality. In all of the example cases, the selectively refined surfaces were rendered more quickly than their highest resolution versions from Figure 7.1 (although the combined extraction and expansion times were longer).

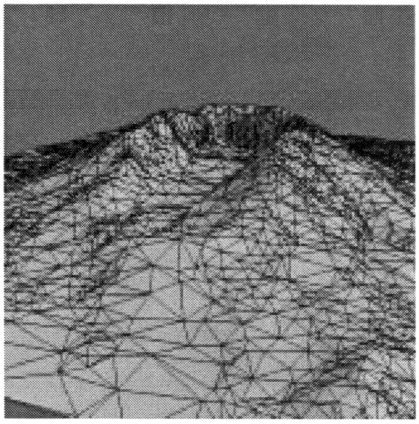
Section 4.6.3 discussed how a screen-space Area Of Interest could be implemented using a variable screen-space tolerance. Figure 7.10 presents an example of a screen-space AOI for the Mt St Helens CRMR. In this case, the tolerance used in determining whether a fragment satisfies the screen-space RCF is reduced to 0% when the fragment's bounding sphere intersects with



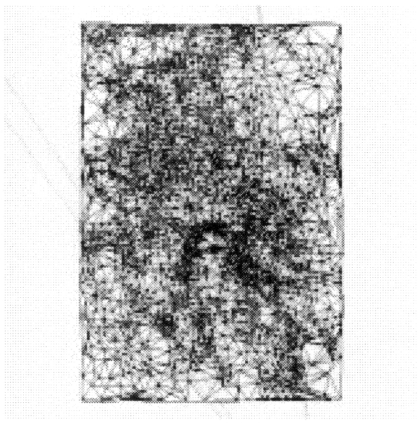
(a) 0.033% screen-space tolerance.



(b) Plan view of (a).

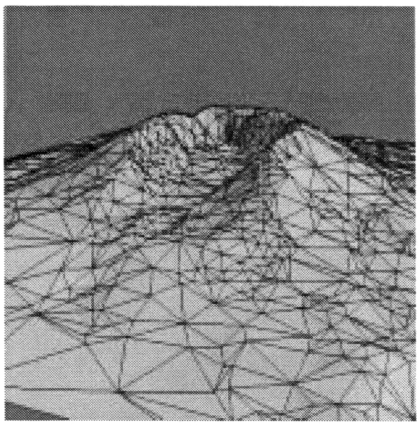


(c) 0.010% screen-space tolerance.

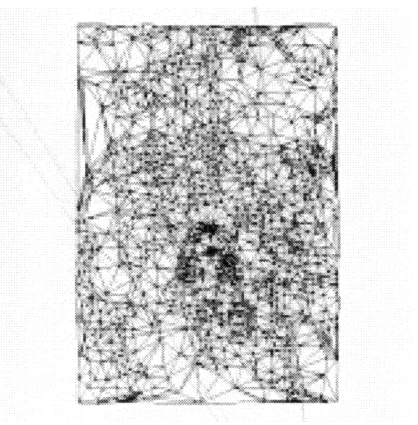


(d) Plan view of (c).

Figure 7.9: Mt St Helens selectively refined according to screen-space RCFs.



(a) 0.033% screen-space tolerance outside the yellow box and 0% tolerance inside (3537 triangles).



(b) Plan view of (a).

Figure 7.10: Refinement with respect to a screen-space Area Of Interest.

CRMR Tolerance	Ref. op. extraction $\|X\|$	Time(s)	Surface expansion $\|\hat{F}\|$	Triangles/fan	Time(s)	Total time(s)
Mt St Helens						
0.033%	899	0.23	3117	1.32	0.06	0.29
0.010%	2746	0.53	9218	1.29	0.21	0.74
Emory Peak						
0.033%	1154	0.24	3989	1.29	0.09	0.33
0.010%	1962	0.32	6648	1.30	0.18	0.50
Crater Lake						
0.033%	642	0.18	1952	1.26	0.04	0.22
0.010%	3145	0.65	9735	1.27	0.24	0.89
Honolulu						
0.033%	1516	0.33	5304	1.30	0.12	0.45
0.010%	3213	0.60	10947	1.29	0.26	0.86

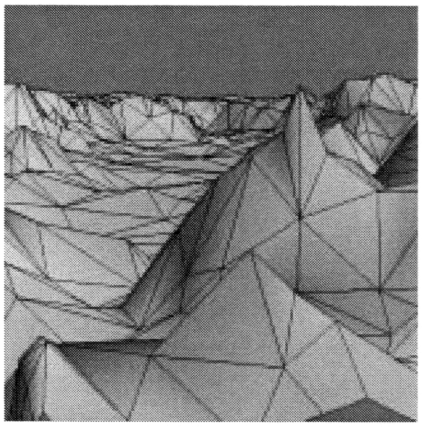
Table 7.5: Results of applying the minimal surface algorithm to the Delaunay CRMRs with screen-space RCFs.

the view-space projection of the AOI. The increased triangle density in this region is apparent in both the perspective and plan views (cf. Figure 7.9a,b).

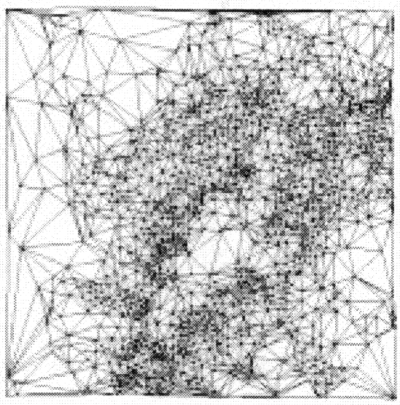
Figure 7.11 presents two further meshes produced using screen-space RCFs (with frustum and back-face culling), this time applied to the Emory Peak CRMR. The perspective views in particular demonstrate how varying the screen-space tolerance can affect the size of triangles throughout a scene.

To assess what contribution each of the individual components of a screen-space RCF made to the triangle-count of a selectively refined mesh, the components of the RCF which was used to generate Figures 7.11(a) and (b) were applied separately to the Emory Peak CRMR. The resulting selectively refined meshes are presented in Figure 7.12 together with the number of triangles in each mesh.

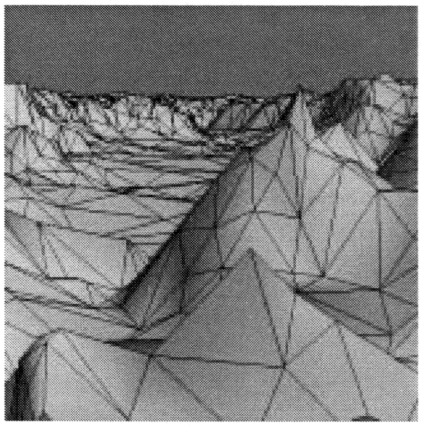
The first point which these plan views demonstrate is that the back-face culling RCF component did not make a significant impact on the highest resolution mesh's triangle count of 11875. This was not unexpected since the nature of this terrain surface, combined with the elevation of the viewpoint, implied that few polygons were eligible for back-face culling in this scene. Secondly, both the frustum culling and screen-space tolerance RCF components produced meshes which had markedly fewer triangles than the highest resolution mesh but it was only when all three components were combined that a major reduction in triangles was obtained.



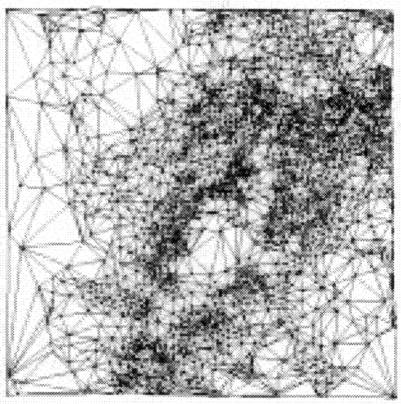
(a) 0.033% tolerance.



(b) Plan view of (a).

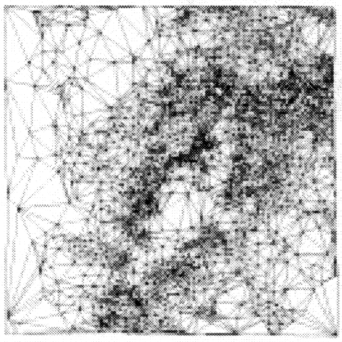


(c) 0.010% tolerance.

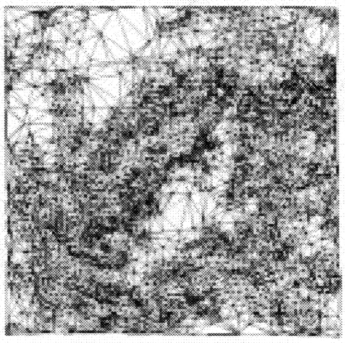


(d) Plan view of (c).

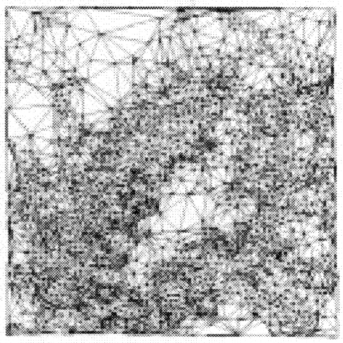
Figure 7.11: Emory Peak selectively refined according to two screen-space tolerances.



(a) Frustum culling only (6713 triangles).



(b) Back-face culling only (11845 triangles).



(c) 0.033% screen-space tolerance only (7027 triangles).

Figure 7.12: Emory Peak selectively refined according to the individual components of the screen-space RCF used in Figure 7.11(a).

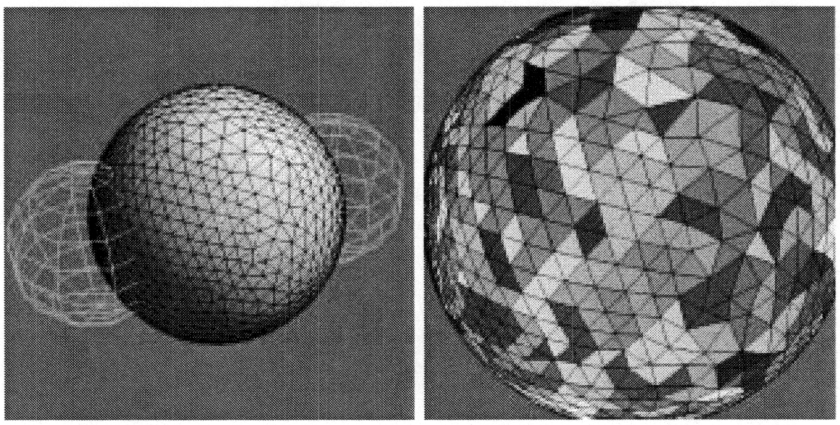
7.2.2 SMR of Progressive Mesh CRMRs

As for the Delaunay-generated CRMRs, we consider first the application of object-space RCF functions to our example Progressive Mesh CRMRs (Section 7.2.2.1) and then the results obtained by applying screen-space RCFs (Section 7.2.2.2). All of the results in this section were obtained using the minimal surface algorithm.

7.2.2.1 Object-space RCFs

This section presents two demonstrations of the "volumes of significance" which Section 4.6.2 suggested could be used to specify regions of interest for three-dimensional models.

The yellow wireframe spheres in Figure 7.13a indicate the extent of two volumes of significance. These are spheres which are centred at points diametrically opposed to each other on the sphere model's CRMR and which have radii equal to (0.997 × geometric error in the sphere CRMR's base mesh). The resulting selectively refined mesh shows that the two sides of the sphere have been highly refined while a band of triangles on the centreline is clearly at a lower resolution.



(a) Mesh selectively refined with respect to the two object-space volumes of significance shown.

(b) Fragments of (a) drawn at their death resolutions in resolution space.

Figure 7.13: An object-space RCF applied to the sphere CRMR.

Figure 7.13b shows the same selectively refined mesh with each fragment drawn at its death resolution in the three-dimensional resolution space described in Section 4.6.2. The viewpoint of this scene is the same as that of Figure 7.13a and hence we can see that the volumes of significance are completely enclosed by the selectively refined fragments and thus that the output mesh satisfies the object-space RCF. Although the smaller and larger trian-

gles in this image have different birth resolutions, the gaps between these triangles in their projected positions are too small to be visible.

The top row of Table 7.6 gives the statistics for this SMR example. While the resulting mesh has a suitably reduced rendering time compared to its highest resolution version, the refinement operation extraction process is computationally expensive due to the three-dimensional geometric object-space RCF test.

CRMR	Ref. op. extraction		Surface expansion			Total
	$\|X\|$	Time(s)	$\|\hat{F}\|$	Triangles/fan	Time(s)	time(s)
Sphere	1039	0.73	2312	1.38	0.08	0.81
Cessna	1897	1.21	5798	1.42	0.21	1.42

Table 7.6: Results of applying the minimal surface algorithm to two Progressive Mesh CRMRs with object-space RCFs.

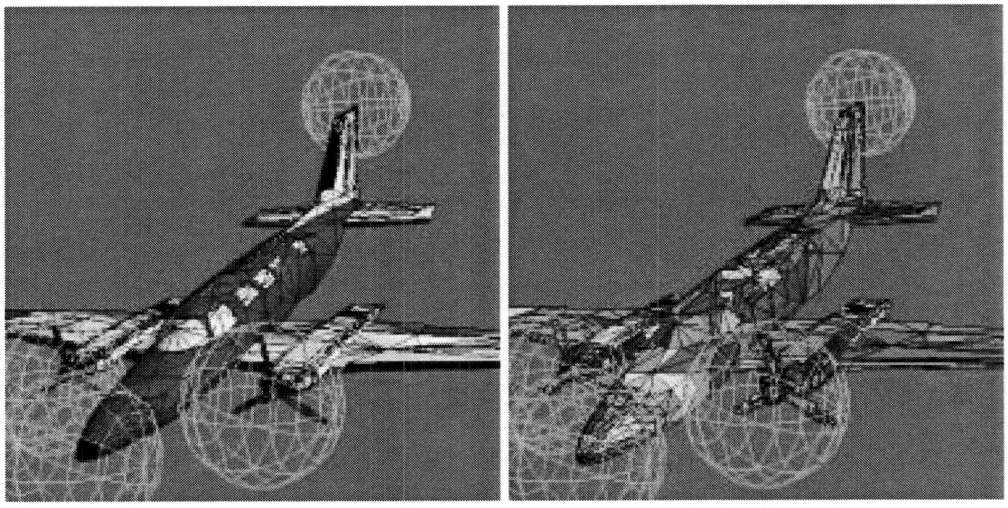
Table 7.6 also presents results for the application of an object-space RCF to the Cessna CRMR and the extraction cost for this example is similarly high. The volumes of significance in this case were defined to completely enclose the regions of interest in the model, rather than to lie inside the fragments of the CRMR in its resolution space, and hence ensured that certain parts of the CRMR were completely expanded. Figure 7.14a shows the four volumes of significance centred on the propellors, the nose and tail. This image demonstrates that these areas of the model were refined to a high degree while the main fuselage was retained at a much lower resolution.

Figure 7.14b shows that these volumes of significance lie outside the expanded fragments at their death resolutions. Although the projection of the vertices along their "current normals" is not particularly noticeable in this diagram, the propellor blades can be observed to be enlarged.

7.2.2.2 Screen-space RCFs

In this section we present the results of applying various screen-space RCFs to the Progressive Mesh CRMRs using the minimal surface algorithm.

Figure 7.15 presents selectively refined versions of the sphere CRMR. Figure 7.15a shows a selectively refined mesh which was obtained using a screen-space tolerance of 0.01%, together with frustum and back-face culling. This mesh is perceptually very similar to its CRMR's highest resolution version in Figure 7.3a while containing approximately half the number of triangles. Note that the triangles directly in front of the viewpoint are at a lower resolution than their surrounding facets because these triangles are almost perpendicular to the viewer and hence the screen-space RCF test described



(a) Mesh selectively refined with respect to the four object-space volumes of significance shown.

(b) Fragments of (a) drawn at their death resolutions in resolution space.

Figure 7.14: An object-space RCF applied to the Cessna.

on page 91 reduces the resolution required in that region. The plan view of Figure 7.15b demonstrates that the back-face culling component of the screen-space RCF has reduced the resolution at the back of the sphere. The final image in this top row shows the selectively refined mesh from the same viewpoint as (a) with the refinement fragments coloured randomly to illustrate the fact that the surface was constructed as a combination of a large number of fragments.

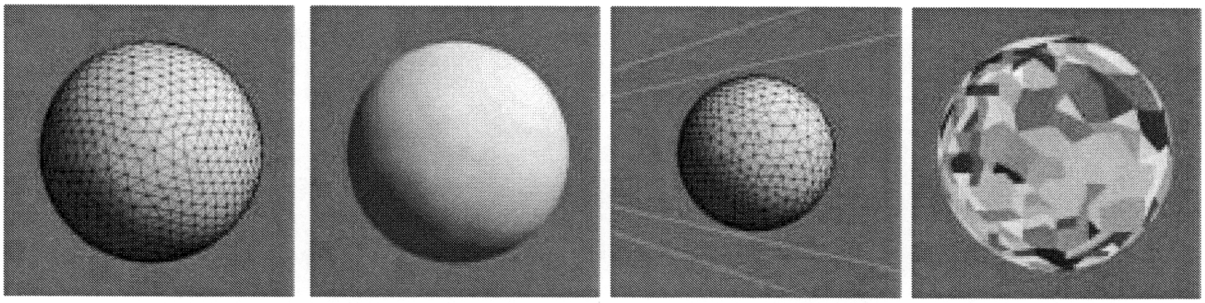
Figure 7.15d shows the effect of moving the viewpoint away from the selectively refined mesh of (a) while using the same RCF. The screen-space tolerance component of the RCF permits the triangle count to be reduced markedly as the model recedes into the distance.

The individual components of this RCF are demonstrated separately in Figures 7.15(e), (f) and (g), in a similar manner to Figure 7.12.

Table 7.7 details the statistics for the selective refinement depicted in Figure 7.15a together with other meshes obtained using alternative screen-space tolerances for the other Progressive Mesh-generated CRMRs. In all cases, the refinement operation extraction time is significantly faster than that for the object-space RCF examples. Also, the expansion and rendering time for the expanded surfaces is always faster than for the corresponding highest resolution versions.

The compromise between image quality and mesh generation/rendering time is apparent in this table, as it was for the selective refinement results of the Delaunay CRMRs.

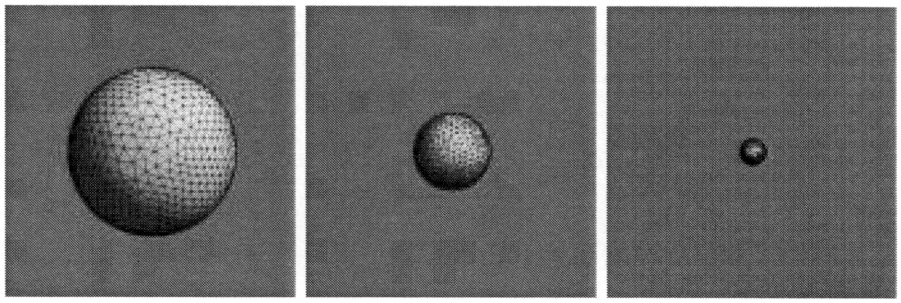
Figure 7.16 presents the Cessna selective mesh refinement results of Ta-



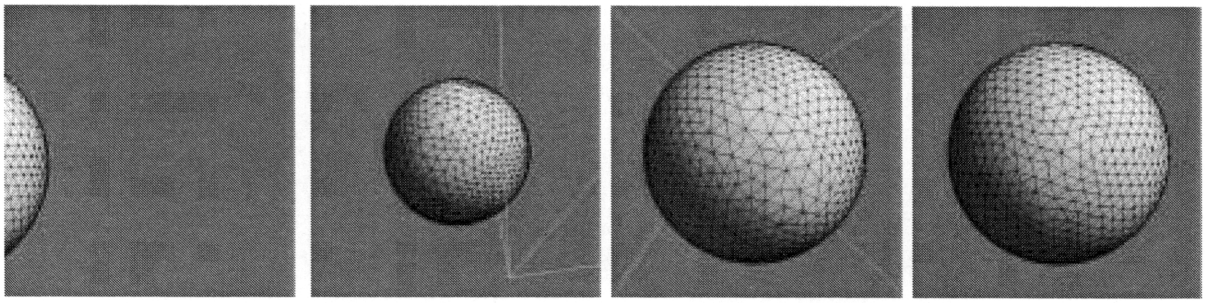
(a) Mesh selectively refined with 0.01% screen-space tolerance.

(b) Plan view of (a).

(c) Fragments of mesh in (a) coloured individually.



(d) A sequence of images produced using the same RCF as (a) while moving the viewpoint away from the model. Meshes contain 1612, 1292 and 712 triangles respectively.



(e) Mesh selectively refined using only frustum culling with respect to the frustum on the left (1826 triangles). The image on the right gives an overview of this scene and its frustum.

(f) Mesh produced using only back-face culling, viewed from the back (2800 triangles).

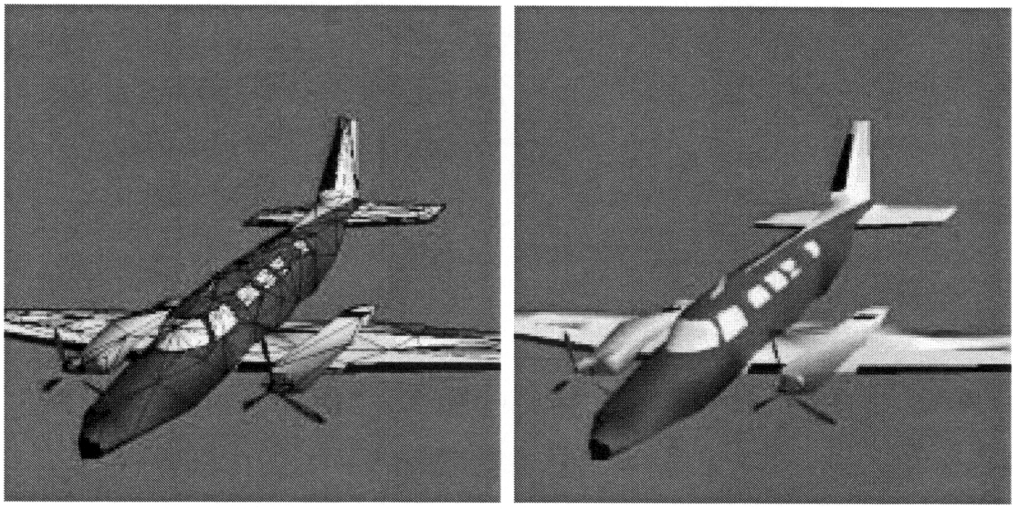
(g) Mesh selectively refined using only 0.01% screen-space tolerance constraint (2124 triangles).

Figure 7.15: Selective refinement using screen-space RCFs on the sphere CRMR. The top and middle rows present the results of applying a standard screen-space RCF with frustum and back-face culling enabled. The individual components of this RCF were used separately to produce the bottom images.

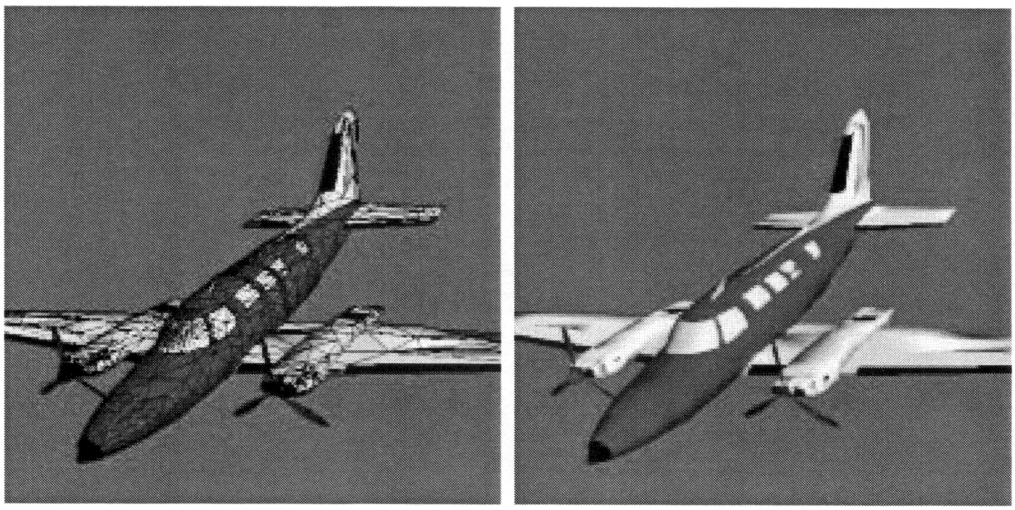
CRMR	Ref. op. extraction		Surface expansion			Total
Tolerance	$\|X\|$	Time(s)	$\|\hat{F}\|$	Triangles/fan	Time(s)	time(s)
Sphere						
0.010%	794	0.08	1730	1.41	0.05	0.13
0.005%	873	0.09	1894	1.42	0.06	0.15
Cessna						
0.010%	333	0.05	1384	1.53	0.04	0.09
0.001%	1865	0.32	5634	1.44	0.20	0.52
Apple						
0.500%	341	0.03	892	1.49	0.03	0.06
0.010%	544	0.05	1325	1.45	0.04	0.09
Statue						
0.010%	751	0.11	2052	1.49	0.06	0.17
0.001%	1859	0.32	4490	1.44	0.15	0.47
Mt St Helens						
0.033%	1208	0.23	3595	1.54	0.09	0.32
0.010%	2583	0.61	7945	1.48	0.23	0.84

Table 7.7: Results of applying the minimal surface algorithm to the Progressive Mesh CRMRs with screen-space RCFs.

ble 7.7 in pictorial form. This CRMR is significant in that its base mesh is a set of unconnected manifolds and hence it makes use of the ability of our refinement operations to "spawn" separate simplicial meshes (as described in Section 4.4.1). The lack of connectivity in the base mesh and the selectively refined versions of the CRMR is particularly visible around the propellor blades.



(a) Mesh selectively refined with 0.01% screen-space tolerance.

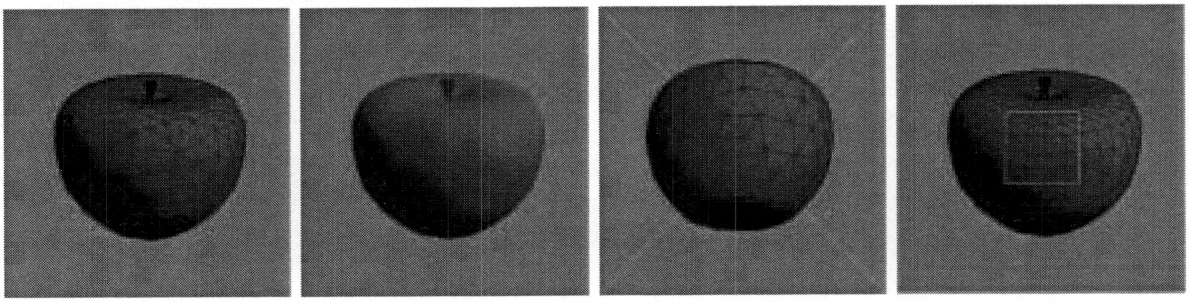


(b) Mesh selectively refined with 0.001% screen-space tolerance.

Figure 7.16: Cessna selectively refined with two screen-space tolerances.

Figures 7.17(a) and (b) show a standard 0.5% screen-space tolerance RCF applied to the apple CRMR. This was augmented by a screen-space Area Of Interest in the mesh shown in (c); the screen-space tolerance inside the yellow box was 0%. As can be seen, this caused the region of the selectively refined

mesh which projected into the AOI's box to be represented by triangles from the highest resolution mesh.



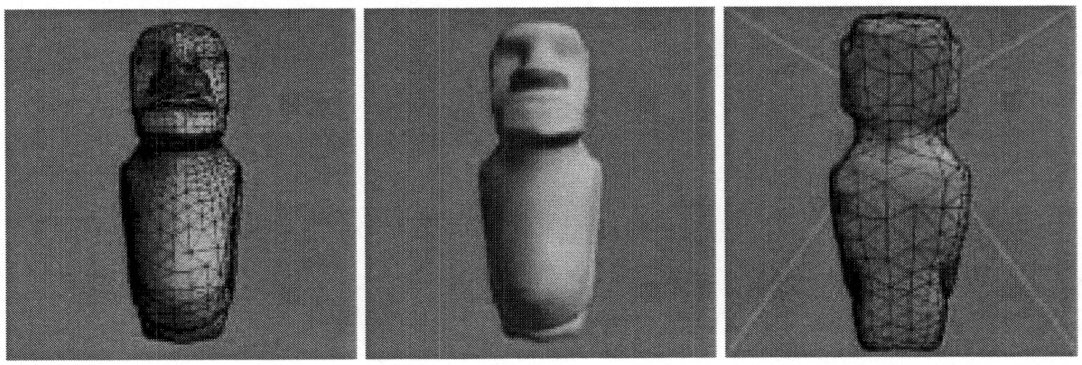
(a) Mesh selectively refined with 0.5% screen-space tolerance.

(b) Reverse view of (a).

(c) Screen-space AOI (950 triangles).

Figure 7.17: Apple selectively refined according to one screen-space tolerance and also using a screen-space Area Of Interest.

Figure 7.18 shows a selectively refined version of the statue CRMR. Of particular note here is that the version without wireframe lines is visually very similar to that in Figure 7.3d even though the selectively refined version contains only 37% of the original triangles. The reduction in detail was particularly drastic on the reverse side of the model in this case.



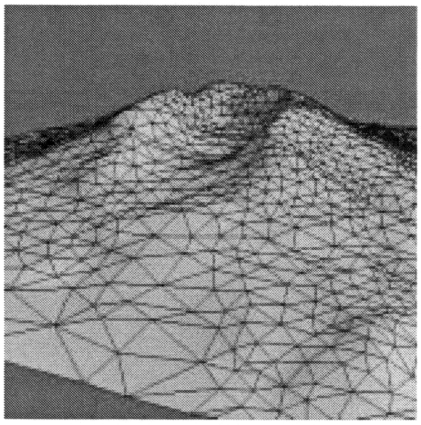
(a) Mesh selectively refined with 0.001% screen-space tolerance.

(b) Reverse view of (a).

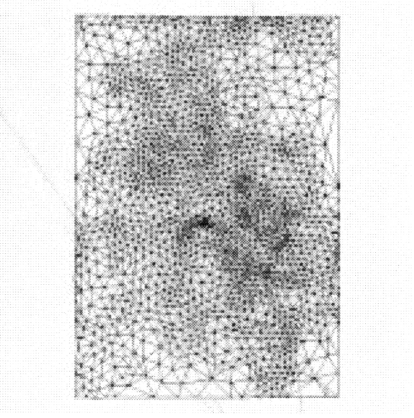
Figure 7.18: Statue selectively refined using a screen-space RCF.

Our final example of screen-space RCFs applied to Progressive Mesh CRMRs is on the Mt St Helens PM CRMR. Figure 7.19 presents examples of selectively refined meshes generated by two different screen-space tolerances. In comparison with the meshes generated from the Delaunay CRMR in Figure 7.9, the PM versions have more evenly-distributed points and smoother surfaces. This is due to the PM energy optimisation technique repelling points on the mesh during the PM approximation process.

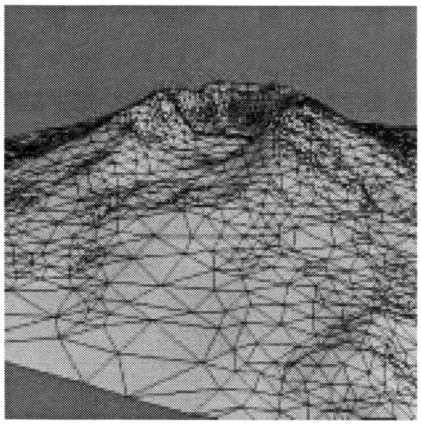
The statistics from these related selectively refined meshes (Tables 7.5 and 7.7) are remarkably similar, although the PM 0.01% tolerance mesh contains more fragments and triangles than its Delaunay equivalent.



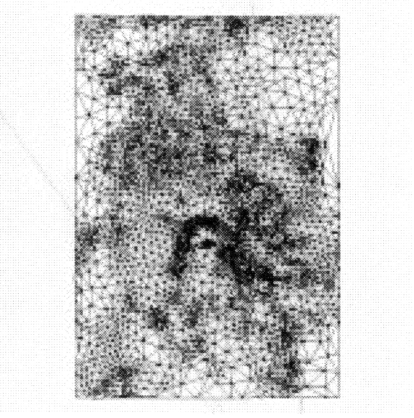
(a) Mesh selectively refined with 0.033% screen-space tolerance.



(b) Plan view of (a).



(c) Mesh selectively refined with 0.010% screen-space tolerance.



(d) Plan view of (c).

Figure 7.19: Screen-space selective refinements of a CRMR produced using the Progressive Mesh approximating technique applied to the Mt St Helens dataset (cf. Figure 7.9).

7.3 Geomorphing results

This section presents geomorphing results obtained by moving the viewpoint while viewing meshes which were selectively refined according to screen-space RCFs.

We first examine a traversal over the Mt St Helens Delaunay-generated CRMR. The starting point for this traversal was the scene in Figure 7.9a,

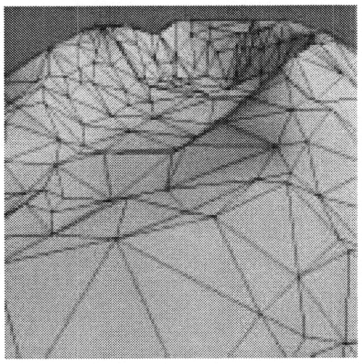


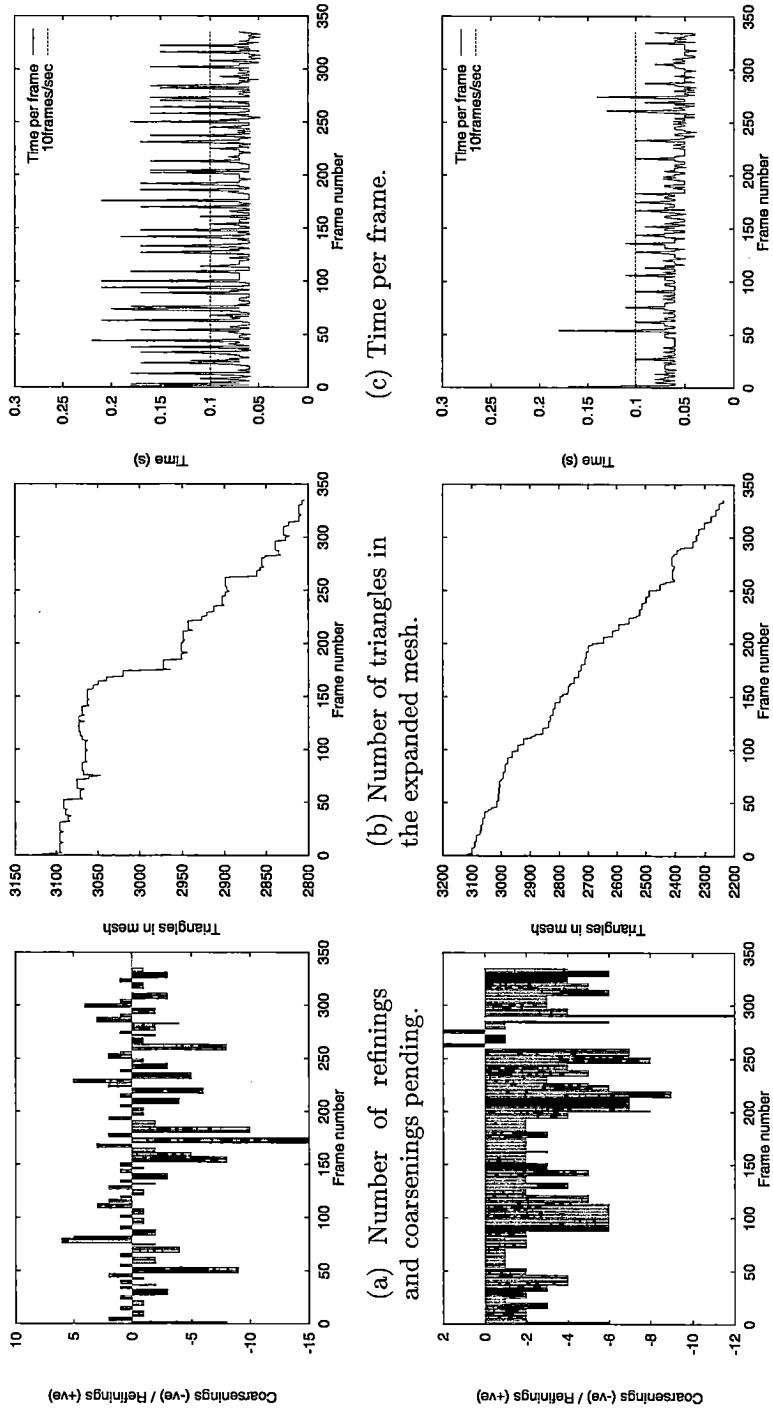
Figure 7.20: Selectively refined mesh of the Mt St Helens CRMR at the end point of the geomorphing traversal with 0.033% screen-space tolerance and no time limit on the geomorphing.

with a 0.033% screen-space RCF applied to the Mt St Helens CRMR. The viewpoint was then stepped forwards along the current view direction until the scene shown in Figure 7.20 was reached.

The numerical results obtained during this traversal are shown in the graphs on the top row of Figure 7.21. The numbers of refinement operation refinings and coarsenings which were pending during each frame are graphed in Figure 7.21a, where the number of refinings pending is on the positive axis and the number of coarsenings on the negative. The oscillating pattern of this graph reflects the nature of the geomorphing algorithm described in Chapter 6. That is, a sequence of coarsenings was applied, the next set of refinings was determined and applied, and then the whole process was repeated. Each refining and coarsening was performed by our own *MorphCoarsen()* and *MorphRefine()* operations, which required several frames to complete each local morph and thus each "bar" of this graph is more than one frame in width. It was frequently possible, though, to perform a number of these local morphs in parallel or to discard coarsenings and refinings due to frustum culling and hence the number of these operations which were pending could be reduced by more than a single element on many occasions (notably immediately after frame 150).

The number of triangles in the selectively refined meshes which resulted from the Mt St Helens traversal is given in Figure 7.21b. This illustrates that the triangle count was decreased at almost every frame during the traversal. By implication, this means that the effect of reducing the area of the mesh inside the view frustum was greater than the requirement to ensure that the screen-space tolerance criterion was met.

Figure 7.21c graphs the total time per frame of this traversal, together



(a) Number of refinings and coarsenings pending.
(b) Number of triangles in the expanded mesh.
(c) Time per frame.
(d) Number of refinings and coarsenings pending for time-limited geomorphing.
(e) Number of triangles in the expanded mesh.
(f) Time per frame.

Figure 7.21: Geomorphing results during two traversals over the Mt St Helens CRMR, the second one restricted by a maximum geomorphing period of 0.1s.

with a line indicating a rendering time which would permit 10 frames/sec for comparison. This shows that the time to display each selectively refined mesh was around 0.06s but that the additional time which was required to determine whether any refinings had to be invoked frequently raised the total time per frame to above 0.15s.

To demonstrate how our geomorphing process could avoid these frequent excessive frame times, the same traversal was also performed with a constraining maximum geomorphing period of 0.1s (Section 6.3). The results of this traversal are graphed in Figures 7.21(d)-(f). Firstly, note that the geomorphing during this traversal was almost entirely limited to coarsenings. This is because the refining process is only initiated if there is sufficient time remaining after any coarsenings have been performed in the current geomorphing period. The only occasions on which this was permitted are visible as the peaks in Figure 7.21f above 0.1s. As a result, the number of triangles in the resulting meshes was significantly less than those produced during the unconstrained geomorph and this assisted in reducing the overall time per frame.

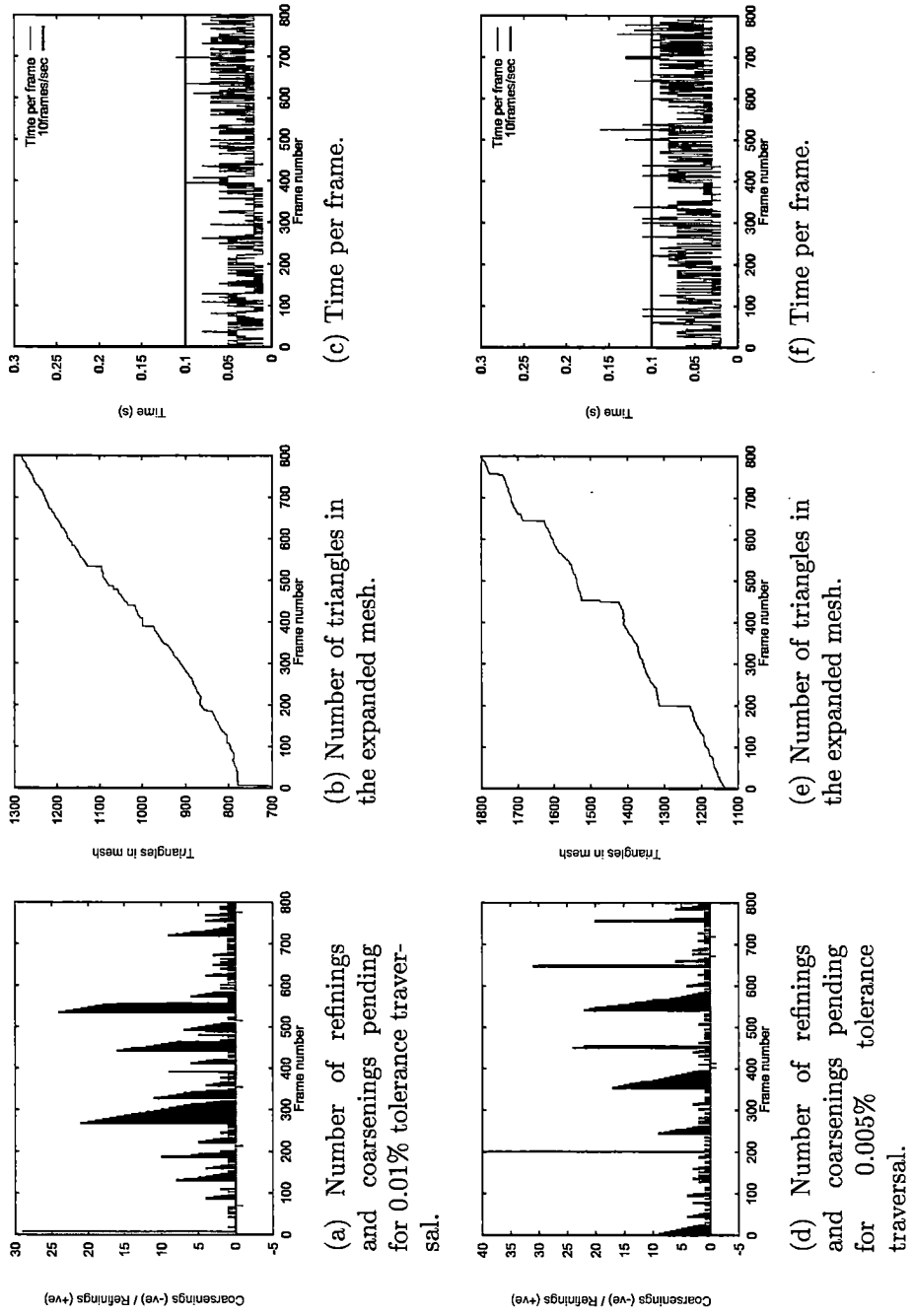
The results of two further geomorphing traversals are given in Figure 7.22. These traversals were performed on the sphere CRMR and involved moving the viewer closer to the model. The start and end points of the traversal are illustrated in the right and left images of Figure 7.15d respectively. The traversals were performed with screen-space tolerances of 0.01% and 0.005%. The coarsenings/refinings graphs of these traversals are noteworthy for their "saw-tooth" nature which reflects the smooth introduction of sets of refinement operations by a series of refinings. Also, the tall thin bars in these graphs indicate sets of refinement operations which had to be added to the current selectively refined mesh but which were back-face culled and hence did not require to be morphed into position. The other graphs of both traversals are closely related: the triangle count increases markedly in both cases and the time per frame is almost always lower than 0.1s.

The significant result which can be drawn from this section is that every set of local morphs which were necessitated by a movement of the viewpoint was a small percentage (less than 5%) of the number of refinement operations in the currently expanded mesh at that time. Thus geomorphing selectively refined meshes in the manner described in Chapter 7 is possible within reasonable time constraints.

7.4 Comparison of results

This section examines our timing results in relation to the few other comparable results in contemporary papers.

Xia and Varshney [XV96] presented the results of their Progressive Mesh-



(a) Number of refinings and coarsenings pending for 0.01% tolerance traversal.

(b) Number of triangles in the expanded mesh.

(c) Time per frame.

(d) Number of refinings and coarsenings pending for 0.005% tolerance traversal.

(e) Number of triangles in the expanded mesh.

(f) Time per frame.

Figure 7.22: Geomorphing results during two traversals over the sphere CRMR with differing screen-space tolerances.

style selective refinement approach for a number of datasets. Most of these are larger than the ones with which we experimented due to memory limitations. Xia and Varshney's results for their sphere dataset, though, contain figures which are very similar to our Cessna CRMR (and which were generated on a machine with comparable processing power to ours). An original model of 8192 triangles was reduced to 537 triangles on screen and displayed in 0.09s which compares with 7446 triangles of our Cessna model which were reduced to 1384 and displayed in a total time of 0.09s with a screen-space tolerance of 0.01%.

Hoppe [Hop97b] presented timings for a geomorphing fly-through of a PM representation which originally contained 400,000 faces and which was reduced to approximately 2,000-11,000 during the traversal. Hoppe's implementation gave a frame time in the region of 0.07-0.13s throughout the traversal. This dataset is much larger than any of our examples but we have to conclude that Hoppe's implementation of his PM approach is inevitably more efficient, though less flexible, than our SMR framework.

The only timings which have been presented for Puppo and De Florani's MultiTriangulation approach were given for a terrain dataset in [DMP97b]. The original dataset contained 14,767 vertices and 73,901 triangles. Selectively refined versions of this surface were produced which contained 6,322 and 12,402 triangles according to a simple object-space distance-to-viewpoint measure by algorithms operating on two alternative MT data structures. These surfaces were generated in 0.110s and 0.972s respectively. Again, we have no dataset of a comparable size but the Honolulu dataset of 11,893 triangles was reduced to 8,207 by the minimal surface algorithm operating on a more complex object-space RCF function in 0.64s.

7.5 Summary

In summary, the results presented in this chapter have demonstrated that selective mesh refinement and geomorphing of scalar field and three-dimensional manifold surfaces is possible within the SMR framework. Also, these operations can be performed with speeds which would permit a degree of interactivity with the rendered meshes. Optimisation of our implementation could lead to truly interactive selectively refined meshes.

Chapter 8

Conclusions

This dissertation has verified our primary thesis that a selectively refined version of a computer graphics model can be constructed from a series of single-resolution approximations to the model without regard to the original approximation technique. We have also demonstrated that a selectively refined mesh produced in this manner can be smoothly geometrically morphed into another mesh which satisfies modified resolution criteria.

The medium for this investigation has been presented as a novel framework within which the twin tasks of selective mesh refinement and geomorphing can be performed. The components and capabilities of this *SMR framework* can be summarised as follows.

- A model is represented by information obtained from applying a standard simplicial mesh approximation technique to that model at a range of resolutions. The differences between each resulting approximation can be treated as localised *refinement fragments* and inserted into our resolution-based representation of this model – its *Continuous Resolution Model Representation (CRMR).*

- The resolution criteria which specify the requirements of a selectively refined version of a model are provided as the components of a *Resolution Control Function (RCF).*

- A selectively refined version of a model can be obtained from its CRMR with respect to an RCF by applying either our *minimal surface* or *reduced extraction* algorithms. These first identify the set of refinement fragments which are necessary and sufficient to produce a mesh which meets the criteria presented by the RCF. This set of fragments can then be traversed to obtain a subset of their facets which will be a complete representation of the original model which satisfies the RCF.

- The extracted set of fragments can also be retained for future modification when the resolution criteria change. Our geomorphing algorithm

identifies refinement fragments which can be removed from, or added to, this original set of fragments while permitting a complete selectively refined mesh to be maintained during this process. The order of these modifications is such that the removals and additions of fragments can be performed using geometric morphing routines which are local to each fragment.

The results of applying the framework to various example CRMRs and RCFs were presented in the previous chapter. These results demonstrated that both selective mesh refinement and geomorphing could be performed as proposed on polygonal model fragments which were generated by existing approximation techniques.

The specific achievements of this framework are discussed in more detail in Section 8.1. Potential extensions to this framework and its implications for the wider field of computer graphics are presented in Section 8.2.

8.1 Specific achievements

This section reviews, in relation to previous work, the original aspects of our SMR framework.

The selective mesh refinement algorithms which were presented in Chapter 5 permit a selectively refined mesh to be generated from a CRMR of a model, which is a resolution-based representation of that model. This decoupling of the selective refinement process from the approximation technique which generated the CRMR extends the concept of Cignoni's HyperTriangulation (Section 3.3), which was restricted to handling the fragments of terrain surfaces generated by a Delaunay approximation process. In contrast, our algorithms can operate on n-dimensional approximations which have been generated by many existing techniques. Our algorithms also eliminate the non-decreasing constraint on the resolution of the output surface which was necessary for Cignoni's selective refinement algorithm.

The task of our minimal surface selective refinement algorithm can be directly related to the description of Puppo's MultiTriangulation algorithm [Pup96]. Puppo acknowledged the independent nature of our contemporary terrain refinement work [Bro96] in [PS97, Pup97].

Both of our selective mesh refinement algorithms are original, however, in the nature of their operation. They first identify those fragments from the CRMR which are necessary and sufficient to produce a complete selectively refined mesh which will satisfy the given resolution criteria. These fragments are then traversed using an advancing front algorithm to extract the simplices of the output selectively refined mesh. This means that the set of fragments from which an existing mesh has been constructed can be used for later operations, notably geomorphing, and that the process of extracting the

simplices of the output mesh can maintain the adjacencies of these simplices. Furthermore, the advancing front nature of this latter phase implies that the regions of greatest significance in a particular scene are extracted first and hence, if this process is being used for rendering, it can be interrupted at any point with the knowledge that the current mesh will contain the most significant features for the displayed view, even if it is not complete.

Our reduced extraction algorithm takes advantage of this separation of tasks in the selective refinement process by attempting to reduce the cost of the first, fragment identification, phase specifically for scalar field meshes and restricted resolution criteria. We have presented results which demonstrate that this objective was achieved in all except one of the example cases.

The ability of both of our selective mesh refinement algorithms to operate on the results of many existing approximation methods is dependent upon the nature of our Continuous Resolution Model Representation. Our specification of the fragments from which a CRMR can be constructed is designed to permit a CRMR to be generated from any n-dimensional simplicial mesh-generating approximation technique which makes repeated local modifications to a mesh. In addition, our CRMR fragments can introduce either holes in a mesh or additional simplicial meshes which means that a CRMR can represent a set of simplicial meshes. This ability was demonstrated in the Cessna model presented in the results chapter and in [Bro97b].

In contrast, Puppo and De Floriani's parallel MultiTriangulation representation (Section 3.4) has only recently been adapted to handle models which are not scalar fields [DMP97b, PS97] and cannot handle a set of manifolds such as the Cessna model.

The abstract structures which we use to represent our CRMR, the Hypermesh and the DAG, are also original. The data structure with which we implemented the Hypermesh for the $2\frac{1}{2}$- and 3-dimensional cases, our *fanedge* structure, is a novel extension of Guibas and Stolfi's quadedge data structure. The advantage of this is that existing quadedge algorithms for operations such as Delaunay retriangulation and point location can be easily adapted for use with the Hypermesh. The Hypermesh and DAG structures cannot be implemented as efficiently as Hoppe's Progressive Mesh representation (Section 3.2.2), but this is offset by the independence of our CRMR from any specific approximation technique.

Hoppe's PM approach also takes advantage of the simple nature of the dependencies between fragments in a PM representation to enable geomorphing between selectively refined Progressive Meshes but we have presented a geomorphing algorithm which can operate on a selectively refined mesh without regard to the underlying approximation method. This unique capability is made possible by the fact that our selective refinement algorithms generate the list of fragments from which a selectively refined mesh is produced.

The consistent fragment-based nature of our framework is also reflected in our resolution specifier, the Resolution Control Function. This permits object-space and screen-space dependencies to be combined in the specification of a particular scene.

This ability was demonstrated in the results of Chapter 7, in which our selective mesh refinement algorithms were applied to a range of terrain and three-dimensional manifolds with respect to various resolution criteria. This is the first quantitative examination of selective mesh refinement applied to resolution-based model representations generated by alternative approximation methods. Previously, Puppo and De Floriani have presented the results of their MultiTriangulation algorithms only for terrain surfaces approximated by Delaunay methods according to a simple object-space resolution criterion. Also, Hoppe has only demonstrated selective refinement of terrain and three-dimensional PM representations with respect to screen-space criteria.

The implementation of our algorithms and structures was designed as a "proof of concept" rather than an optimised selective mesh refinement generator, but our results have shown that:

- our Continuous Resolution Model Representation can handle the output of Delaunay approximations of terrain datasets as well as Progressive Mesh representations of terrain and three-dimensional manifold models;

- our selective refinement algorithms can generate meshes according to object-space or screen-space resolution criteria or a combination of both;

- the reduced extraction algorithm can reduce the cost of the initial phase of selective mesh refinement and also the overall cost of the process in almost all of the object-space terrain examples;

- all of the selectively refined meshes presented have been expanded in less time than their corresponding original models were rendered and they are all perceptually equivalent to their originals to a greater or lesser extent;

- the cost of the surface extraction phase of our selective refinement algorithms is proportional to the number of triangles being displayed and hence selective mesh refinement could be used in a rendering system to provide a trade-off between image quality and rendering speed;

- the cost of geomorphing a selectively refined mesh according to a changing view frustum is within the range required for the interactive display of a scene. In addition, the ability to constrain the time permitted for

geomorphing could be used to guarantee the interactive quality of a scene;

- the Progressive Mesh-generated CRMR of a terrain surface produces smoother selectively refined meshes than that from the corresponding Delaunay-generated CRMR. This is the first comparison of PM- and Delaunay-generated meshes.

8.2 Future work

This section discusses potential extensions and applications of the SMR framework.

The area which offers the greatest scope for immediate improvement of the framework is the optimisation of its implementation. All of the implementations of our algorithms and data structures have been designed primarily for extensibility and robustness rather than speed of operation. A fully-optimised version of the framework should be comparable in terms of speed with Hoppe's view-dependent Progressive Mesh implementation. To match the rendering speed of Hoppe's system, though, a means would be required for generating triangle strips (a generalisation of the triangle fans described in Section 5.1.2.1) during our surface expansion process.

Another feature of the OpenGL graphics library, apart from triangle fans and strips, which can be used to optimise rendered scenes is *display lists*. These are groups of OpenGL commands which are sent by an application to the graphics hardware and retained there for later, potentially optimised, execution. Thus display lists are particularly suited for static models and contribute little towards the dynamic tasks of selective mesh refinement and geomorphing. If selective mesh refinement is to become a useful tool for visualising large models then support for data structures and algorithms such as those proposed by our framework must be provided by graphics hardware. The simplicity of our algorithms means that they lend themselves to implementation in hardware.

In terms of space costs, though, a CRMR could never be compressed to the same degree as a PM representation because the specific nature of the PM approximation method allows a high degree of compression in its corresponding representation. It is this space efficiency which would be the major consideration if these representations were used to transmit a graphical model over a network. As Hoppe noted [Hop97b], a resolution-based model representation together with a selective mesh refinement algorithm offer the potential for models to be progressively transmitted since the base mesh of a model could be transmitted initially and then detail added as bandwidth and resolution requirements permit.

We discussed how our SMR framework related to the client-server model for transmitting 3D data in [Bro97b], where a server identifies the fragments of a CRMR which have to be transmitted to a client and the client constructs a selectively refined mesh from these fragments. Our framework is uniquely suited to this client-server approach because of the two-phase operation of our selective mesh refinement algorithms. The ability of our framework to handle a wide range of approximation techniques could be particularly applicable to the various encoding capabilities demanded by the proposed MPEG-4 SNHC (Synthetic/Natural Hybrid Coding)[1] standard. A quantitative examination of the speed of our framework applied to the progressive transmission problem is left for future exploration.

The ultimate aim of progressively transmitting three-dimensional models on the Internet should be to obtain a three-dimensional equivalent of the two-dimensional interlaced GIF images which can be handled by current Web browsers. An interlaced GIF is stored by reordering the lines of the original image such that a complete low resolution image can be retrieved quickly and finer detail can be added when further lines of information are obtained. The simplicity and generality of this approach cannot be matched by any existing three-dimensional approximation or selective refinement techniques. The Progressive Mesh approach, for example, is not sufficiently general to handle non-manifold meshes or permit topological modifications and its related approach which can handle these factors, Progressive Simplicial Complexes, has not received a similar level of acceptance.

With respect to the SMR framework, a first step towards further generality would be to develop a topology-modifying approximation method which produces manifold meshes. This would permit the CRMR's ability to handle topology-modifying fragments to be utilised. Such an approximation method could be an adaptation of Garland and Heckbert's quadric error measure method (Section 2.4.2). By restricting the vertex pair contractions which this method permits, we could ensure that the output meshes retain their manifold property while allowing some topological modifications.

Alternatively, the concept of the CRMR could be extended to handle non-manifold models. This would require each fragment to be treated as a group of simplices rather than as a simplicial mesh (since the simplices may be of mixed dimensionality). To contain these fragments in a Hypermesh, the linkages between fragments would have to allow links between elements of mixed dimensions. While the refinement operation extraction phase of our selective mesh refinement algorithms would remain unaffected by these changes, the advancing front nature of our surface expansion step may no longer be suitable.

It is this advancing front nature, though, which provides a final area

[1] http://www.es.com/mpeg4-snhc

for investigation. The potential which this offers to terminate a rendering process early while still displaying the significant features of a mesh could be used to cull facets outside the view frustum. In addition, a guaranteed rendering time could be enforced using an extension of Funkhouser's cost-benefit analysis for LOD usage [FS93].

Bibliography

[Bau75] Bruce G. Baumgart. A polyhedron representation for computer vision. In *1975 National Computer Conference*, volume 44, pages 589–596. AFIPS Conference Proceedings, AFIPS Press, 1975.

[Bro96] Peter J.C. Brown. A fast algorithm for selective refinement of terrain meshes. In *COMPUGRAPHICS 96*, pages 70–82. GRASP, December 1996. Paris, France. Also published in *Computer Networks & ISDN Systems*, 29(14):1587-1600, October 1997.

[Bro97a] Peter J. C. Brown. Selective mesh refinement for interactive terrain rendering. Technical Report TR No. 417, Computer Laboratory, University of Cambridge, New Museums Site, Cambridge, CB2 3QG, UK, February 1997.

[Bro97b] Peter J.C. Brown. Intelligent transmission of 3D polygonal models. In *SIGGRAPH 97 Visual Proceedings*, Computer Graphics, Annual Conference Series, page 176. ACM SIGGRAPH, August 1997.

[BS96] Chandrajit L. Bajaj and Daniel R. Schikore. Error-bounded reduction of triangle meshes with multivariate data. In *Symposium on Visual Data Exploration and Analysis*. SPIE, January 1996.

[CB97] Rikk Carey and Gavin Bell. *The Annotated VRML 2.0 Reference Manual*. Addison Wesley Longman, 1997.

[CL96] Daniel Cohen-Or and Yishay Levanoni. Temporal continuity of levels of detail in Delaunay triangulated terrain. In *7th IEEE Visualization Conference*. IEEE, October 1996.

[Cla76] James H. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10):547–554, October 1976.

[CMR90] Michael A. Cosman, Allen E. Mathisen, and John A. Robinson. A new visual system to support advanced requirements. In *IMAGE V Conference*, pages 370–380. Image Society Inc, June 1990.

[CPS95] P. Cignoni, E. Puppo, and R. Scopigno. Representation and visualization of terrain surfaces at variable resolution. In R. Scateni, editor, *Proceedings International Symposium on Scientific Visualization*, September 1995.

[CVM+96] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick Brooks, and William Wright. Simplification envelopes. In *SIGGRAPH 96*, Computer Graphics, Annual Conference Series, pages 119–128. ACM SIGGRAPH, August 1996. Also available as University of North Carolina at Chapel Hill Computer Science Technical Report TR 96-017.

[dD95] Mark de Berg and Katrin T. G. Dobrindt. On levels of detail in terrains. In *11th ACM Symposium on Computational Geometry*, pages 26–27, June 1995. Also published in longer version as Technical Report UU-CS-1995-12, Utrecht University, Dept. of Computer Science, April 1995.

[De 87] Leila De Floriani. Surface representations based on triangular grids. *The Visual Computer*, 3:27–50, 1987.

[De 89] Leila De Floriani. A pyramidal data structure for triangle-based surface description. *IEEE Computer Graphics and Applications*, pages 67–78, March 1989.

[DFNP82] Leila De Floriani, Bianca Falcidieno, George Nagy, and Caterina Pienovi. Yet another method for triangulation and contouring for automated cartography. In *American Congress on Surveying and Mapping*, pages 101–110, 1982.

[DFP83] Leila De Floriani, Bianca Falcidieno, and Caterina Pienovi. A Delaunay-based method for surface approximation. In *EUROGRAPHICS 83*, volume 2 of *Computer Graphics Forum, Conference Issue*, pages 333–350. Eurographics Association, 1983.

[DFP85] Leila De Floriani, Bianca Falcidieno, and Caterina Pienovi. Delaunay-based representation of surfaces defined over arbitrarily shaped domains. *Computer Vision, Graphics, and Image Processing*, 32:127–140, 1985.

[DL89] D. P. Dobkin and M. J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 4:3–32, 1989.

[DMP93] Leila De Floriani, Daniela Mirra, and Enrico Puppo. Extracting contour lines from a hierarchical surface model. In *EUROGRAPHICS 93*, volume 12 of *Computer Graphics Forum, Conference Issue*, pages 249–260. Eurographics Association, 1993.

[DMP97a] Leila De Floriani, Paolo Magillo, and Enrico Puppo. Building and traversing a surface at variable resolution. In *Proceedings IEEE Visualization 97*. IEEE, October 1997. Also available as Technical Report PDISI-97-02, Department of Computer and Information Sciences, University of Genova, Italy, 1997.

[DMP97b] Leila De Floriani, Paolo Magillo, and Enrico Puppo. Efficient encoding and retrieval of triangle meshes at variable resolution. Technical Report DISI-TR-97-01, Department of Computer and Information Sciences, University of Genova, Genova, Italy, 1997.

[DMP97c] Leila De Floriani, Paolo Magillo, and Enrico Puppo. VARIANT - processing and visualizing terrains at variable resolution. In *Proceedings 5th ACM Workshop on Advances in Geographic Information Systems*. ACM, November 1997.

[DMPB96] Leila De Floriani, Paolo Magillo, Enrico Puppo, and Michela Bertolotto. Variable resolution operators on a multiresolution terrain model. In *Proceedings 4th ACM Workshop on Advances in GIS (ACM-GIS '96)*. ACM, November 1996. Rockville, Maryland.

[Dou86] David H. Douglas. Experiments to locate ridges and channels to create a new type of digital elevation model. *Cartographica*, 23(4):29–61, 1986.

[DP92a] Leila De Floriani and Enrico Puppo. A hierarchical triangle-based model for terrain description. In *Lecture Notes in Computer Science*, volume N.639, pages 236–251. Springer-Verlag, 1992.

[DP92b] Leila De Floriani and Enrico Puppo. An on-line algorithm for constrained Delaunay triangulation. *CVGIP: Graphical Models and Image Processing*, 54(4):290–300, 1992.

[DP95] Leila De Floriani and Enrico Puppo. Hierarchical triangulation for multiresolution surface description. *ACM Transactions on Graphics*, 14(4):363–411, 1995.

[DPM96] Leila De Floriani, Enrico Puppo, and Paolo Magillo. A formal approach to multiresolution modeling. In *Theory and Practice of Geometric Modeling - Blaubeuren II*, October 1996. Tuebingen, Germany.

[DZ91] Michael J. DeHaemer, Jr. and Michael J. Zyda. Simplification of objects rendered by polygonal approximations. *Computers and Graphics*, 15(2):175–184, 1991.

[EC84] Atef A. Elassal and Vincent M. Caruso. USGS digital cartographic data standards – Digital Elevation Models. Technical Report Geological Survey Circular 895-B, US Geological Survey, 1984.

[EDD+95] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. Technical Report 95-01-02, University of Washington, Seattle, WA, 1995.

[FEKR90] R. L. Ferguson, R. Economy, W. A. Kelly, and P. P. Ramos. Continuous terrain level of detail for visual simulation. In *IMAGE V Conference*, pages 144–151. Image Society Inc, June 1990.

[FHMB84] O.D. Faugeras, M. Hebert, P. Mussi, and J.D. Boissonnat. Polyhedral approximation of 3-D objects without holes. *Computer Vision, Graphics and Image Processing*, 25:169–183, 1984.

[FL79] Robert J. Fowler and James J. Little. Automatic extraction of irregular network digital terrain models. In *SIGGRAPH 79*, Computer Graphics Proceedings, Annual Conference Series, pages 199–207. ACM SIGGRAPH, 1979.

[FS93] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, pages 247–254. ACM SIGGRAPH, 1993.

[FZPM93] John S. Falby, Michael J. Zyda, David R. Pratt, and Randy L. Mackey. NPSNET: Hierarchical data structures for real-time three-dimensional visual simulation. *Computers and Graphics*, 17(1):65–69, 1993.

[GGS95] Markus H. Gross, Roger Gatti, and Oliver G. Staadt. Fast multiresolution surface meshing. In *Visualization 95*. IEEE, October 1995.

[GH96] Michael Garland and Paul S. Heckbert. Fast and flexible polygonalisation of height fields. In *SIGGRAPH 96*, Visual Proceedings, Annual Conference Series, page 143. ACM SIGGRAPH, August 1996.

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH 97*, Computer Graphics, Annual Conference Series, pages 209–216. ACM SIGGRAPH, August 1997.

[GS77] P. J. Green and R. Sibson. Computing Dirichlet tesselations in the plane. *Computer Journal*, 12(2):168–173, 1977.

[GS85] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

[GSG96] Markus H. Gross, Oliver G. Staadt, and Roger Gatti. Efficient triangular surface approximations using wavelets and quadtree data structures. *IEEE Transactions on Visualization and Computer Graphics*, 2(2), June 1996.

[Gué96] André Guéziec. Surface simplification inside a tolerance volume. Technical Report RC 20440, IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, March 1996.

[HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *SIGGRAPH 93*, Computer Graphics, Annual Conference Series, pages 19–26. ACM SIGGRAPH, August 1993.

[Hec97] Paul S. Heckbert. Multiresolution surface modeling. In *SIGGRAPH 97*, Course Notes. ACM SIGGRAPH, August 1997. Course 25. To appear as a technical report of CS Dept, Carnegie Mellon University.

[HG94] Paul S. Heckbert and Michael Garland. Multiresolution modelling for fast rendering. In Wayne A. Davis and Barry Joe, editors, *Graphics Interface 94*, pages 43–50. Canadian Human-Computer Communications Society, May 1994.

[HG95] Paul S. Heckbert and Michael Garland. Fast polygonal approximation of terrains and height fields. Technical Report CMU-CS-95-181, Carnegie Mellon University, August 1995.

[HH93] Paul Hinker and Charles Hansen. Geometric optimization. In *Visualization 93*, pages 189–195. IEEE, October 1993.

[Hop94] Hugues Hoppe. *Surface Reconstruction from Unorganized Points.* PhD thesis, University of Washington, 1994.

[Hop96] Hugues Hoppe. Progressive meshes. In *SIGGRAPH 96*, Computer Graphics, Annual Conference Series, pages 99–108. ACM SIGGRAPH, August 1996.

[Hop97a] Hugues Hoppe. Personal correspondence, April 1997.

[Hop97b] Hugues Hoppe. View-dependent refinement of progressive meshes. In *SIGGRAPH 97*, Computer Graphics, Annual Conference Series, pages 189–198. ACM SIGGRAPH, August 1997.

[KLS96] Reinhard Klein, Gunther Liebich, and W. Straßer. Mesh reduction and error control. In *Visualization 96.* IEEE, November 1996.

[Kum94] Mark P. Kumler. An intensive comparison of triangulated irregular networks (TINs) and digital elevation models (DEMs). *Cartographica*, 31(2), 1994. Monograph 45.

[Law77] C. L. Lawson. Software for C^1 surface interpolation. In J. Rice, editor, *Mathematical Software III*, pages 161–194. Academic Press, 1977.

[LDW94] M. Lounsbery, T. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. Technical Report 93-10-05b, Dept. of Computer Science and Engineering, University of Washington, January 1994.

[LE97] David Luebke and Carl Erikson. View-dependent simplification of arbitrary polygonal environments. In *SIGGRAPH 97*, Computer Graphics, Annual Conference Series, pages 199–208. ACM SIGGRAPH, August 1997.

[Lis94] Dani Lischinski. *Graphics Gems*, volume IV, chapter titled "Incremental Delaunay Triangulation", pages 47–59. Academic Press, 1994.

[LKR+96] Peter Lindstrom, David Koller, William Ribarsky, Larry F. Hodges, Nick Faust, and Gregory Turner. Real-time, continuous level of detail rendering of height fields. In *SIGGRAPH 96*, Computer Graphics, Annual Conference Series, pages 109–118. ACM SIGGRAPH, August 1996.

[Lo85] S. H. Lo. A new mesh generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering*, 21:1403–1426, 1985.

[LT97] Kok-Lim Low and Tiow-Seng Tan. Model simplification using vertex-clustering. In *Symposium on Interactive 3D Graphics*, pages 75–81. ACM SIGGRAPH, April 1997.

[Moo92] Douglas William Moore. *Simplicial Mesh Generation with Applications.* PhD thesis, Cornell University, August 1992.

[Mue95] Carl Mueller. Architectures of image generators for flight simulators. Technical report, UNC Chapel Hill, February 1995.

[PD75] T. K. Peucker and D. H. Douglas. Detection of surface-specific points by local parallel processing of discrete terrain elevation data. *Computer Graphics and Image Processing*, 4:375–387, 1975.

[PH97] Jovan Popović and Hugues Hoppe. Progressive simplicial complexes. In *SIGGRAPH 97*, Computer Graphics, Annual Conference Series. ACM SIGGRAPH, August 1997.

[PS85] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction.* Springer Verlag, 1985.

[PS97] Enrico Puppo and Roberto Scopigno. Simplification, LOD and multiresolution principles and applications. In D. Fellner and L. Szirmay-Kalos, editors, *EUROGRAPHICS 97*, volume 16. Eurographics Association, September 1997.

[Pup96] Enrico Puppo. Variable resolution terrain surfaces. In *8th Canadian Conference on Computational Geometry*, August 1996. Ottawa, Canada. A longer version, titled "Variable Resolution Triangulations", was published as TR n. 6/96, Istituto per la Matematica Applicata – C.N.R., Genova, Italy, November 1996.

[Pup97] Enrico Puppo. Personal correspondence, June 1997.

[RB93] Jarek Rossignac and Paul Borrel. Multi-resolution 3D approximations for rendering complex scenes. In B. Falcidieno and T. Kunii, editors, *Modeling in Computer Graphics: Methods and Applications*, pages 455–465. Springer-Verlag, June 1993. Also available as IBM Research Report RC 17697 (# 77951), IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, February 1992.

[Rip92a] Shmuel Rippa. Adaptive approximation by piecewise linear polynomials on triangulations of subsets of scattered data. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1123–1141, September 1992.

[Rip92b] Shmuel Rippa. Long and thin triangles can be good for linear interpolation. *SIAM Journal on Numerical Analysis*, 29(1):257–270, February 1992.

[RR96] Rémi Ronfard and Jarek Rossignac. Full-range approximation of triangulated polyhedra. In *EUROGRAPHICS 96*, volume 15 of *Computer Graphics Forum, Conference Issue*, pages 67–76. Eurographics Association, 1996. Also available as IBM Research Report 20423, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, 1996.

[Sam84] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):187–260, June 1984.

[Sca90] Lori L. Scarlatos. A refined triangulation hierarchy for multiple levels of terrain detail. In *IMAGE V Conference*, pages 114–122. Image Society Inc, June 1990.

[Sch83] Bruce J. Schachter. *Computer Image Generation.* John Wiley and Sons, 1983.

[Sew96] Jonathan M. Sewell. *Managing Complex Models for Computer Graphics.* PhD thesis, University of Cambridge, March 1996.

[SP92] Lori Scarlatos and Theo Pavlidis. Hierarchical triangulaton using cartographic coherence. *CVGIP: Graphical Models and Image Processing*, 54(2):147–161, March 1992.

[Spa66] Edwin H. Spanier. *Algebraic Topology.* McGraw-Hill, 1966.

[SR94] Florian Schröder and Patrick Roßbach. Managing the complexity of digital terrain models. *Computers and Graphics*, 18(6):775–783, 1994.

[Sur85] Andrew P. Surany. A simple algorithm for determining whether a point resides within an arbitrarily shaped polygon. In Rae E. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics*, volume F17 of *NATO ASI Series*, pages 187–193. Springer-Verlag, 1985.

[Sut65] Ivan E. Sutherland. The ultimate display. In *Proceedings IFIP Congress*, volume 2, pages 506–508. IFIP, May 1965. New York.

[SZL92] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In *SIGGRAPH 92*, Computer Graphics, Annual Conference Series, pages 65–70. ACM SIGGRAPH, July 1992.

[TB94] David C. Taylor and William A. Barrett. An algorithm for continuous resolution polygonalizations of a discrete surface. In Wayne A. Davis and Barry Joe, editors, *Graphics Interface 94*, pages 33–42. Canadian Human-Computer Communications Society, May 1994.

[Tur92] Greg Turk. Re-tiling polygonal surfaces. In *SIGGRAPH 92*, Computer Graphics Proceedings, Annual Conference Series, pages 55–64. ACM SIGGRAPH, July 1992.

[Var94] Amitabh Varshney. *Hierarchical Geometric Approximations.* PhD thesis, University of North Carolina at Chapel Hill, 1994. Available as UNC Computer Science Technical Report TR 94-050, August 1994.

[Wer94] Josie Wernecke. *The Inventor Mentor.* Addison Wesley, 1994.

[WWHR97] Benjamin Watson, Neff Walker, Larry F. Hodges, and Martin Reddy. An evaluation of level of detail degradation in head-mounted display peripheries. *Presence: Teleoperators and Virtual Environments*, 6(6):630–637, December 1997.

[XV96] J.C. Xia and A. Varshney. Dynamic view-dependent simplification for polygonal models. In R. Yagel and G. Nielson, editors, *IEEE Visualization '96 Proceedings*, pages 327–334, 1996.