Voronoi Ball Models for Computational Shape Applications

by

Roger C. Tam

B.Sc. (Hons.), University of British Columbia, 1995
M.Sc., University of British Columbia, 1997

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

_____

_____

_____

_____

_____

The University of British Columbia

February 2004

Abstract

This thesis evaluates the suitability of *Voronoi ball models* (*VBMs*) as a multipurpose shape representation for applications in computer graphics, scientific visualization, and computer vision. The effectiveness of VBMs is judged with respect to six key properties, namely stability, flexibility, accuracy, complexity, efficiency, and intuitiveness. These properties have a significant impact on the range of applicability of a computational shape model. The ability of VBMs to support a number of core shape-driven operations, in particular shape extraction, simplification, matching, interpolation, manipulation, and surface reconstruction, is examined by determining the strength of the key properties in the representation. The general approach is to use VBMs in a number of representative applications, each requiring several of the shape operations being considered. These applications include image matching and interpolation, shape model extraction from image data, two and three-dimensional shape simplification, and polygonal surface reconstruction. The performance of VBMs in these applications is indicative of the extent to which each key property is present. The results of the experiments are very positive. They indicate that a VBM-based shape similarity measure can be effectively applied to quantify 2D shape differences and solve the 2D/3D shape correspondence problem. The findings also show that the VBM and the medial axis can be used together to take advantage of their complementary properties; the VBM gives the medial axis greater stability, while the axis adds connectivity and topological information to the VBM representation. The preservation of the topology of 3D shapes during processing is a particularly strong contribution of the thesis. In addition, the medial axis is shown to enhance the capabilities of the VBM for performing shape simplification and partitioning an object into parts. The experimental results also reveal that VBMs can be effectively used to extract shape information from images and reconstruct polygonal surfaces from point sample data. The primary conclusion made in this thesis is that VBMs are demonstrably capable of supporting a wide variety of shape operations. Additional research is warranted to further exploit the potential of the representation.

Table of Contents

Abstract ii

List of Tables viii

List of Figures ix

Dedication xiii

Acknowledgments xiv

1 Introduction 1

1.1 Motivation . . . 1

1.2 Shape and Shape Models . . . 2

1.2.1 Common Shape-Driven Operations . . . 2

1.2.2 Key Properties of a Computational Shape Model . . . 3

1.3 Voronoi Ball Models . . . 8

1.4 Primary Goals and Contributions . . . 10

1.4.1 Image Matching and Interpolation . . . 10

1.4.2 Shape Model and Threshold Extraction . . . 11

1.4.3 Shape Simplification Based on the Medial Axis . . . 11

1.4.4 Surface Reconstruction . . . 13

1.4.5 Thesis Statement . . . 13

1.5 Document Overview . . . 14

2 Related Work 15

2.1 Shape Representations . . . 15

2.1.1 Application-Oriented Models . . . 15

2.1.2 Visual Form Representations . . . 20
2.2 Shape Similarity Measures . . . 21
2.2.1 2D Similarity Measures . . . 22
2.2.2 3D Similarity Measures . . . 25
2.2.3 Matching Methods . . . 26
2.3 Application-Specific Research . . . 26
2.3.1 Computer Graphics and Visualization . . . 26
2.3.2 Computer Vision . . . 27

3 Computation and Approximation Properties of Voronoi Ball Models 30
3.1 Computation . . . 30
3.1.1 Voronoi Diagram . . . 31
3.1.2 Ball Classification . . . 33
3.2 Approximation Properties . . . 35
3.2.1 Union of Circles/Spheres . . . 35
3.2.2 Polar Balls . . . 37

4 Image Matching, Interpolation and Rigid Registration 38
4.1 Motivation . . . 38
4.1.1 VBM Properties . . . 39
4.2 Background . . . 40
4.3 Related Work . . . 41
4.4 Ranjan and Fournier's Matching Algorithm . . . 41
4.4.1 Simplification Algorithm . . . 41
4.4.2 Model Alignment . . . 43
4.4.3 Similarity Measure . . . 43
4.4.4 Matching Method . . . 44
4.5 Image Interpolation Algorithm . . . 44
4.5.1 Algorithm Overview . . . 44
4.5.2 Test Data . . . 45
4.5.3 Preprocessing . . . 46
4.5.4 Height Field Generation . . . 47
4.5.5 VBM Generation and Simplification . . . 47

4.5.6 VBM Matching and Parameter Selection . . . 47
4.5.7 VBM Interpolation . . . 49
4.5.8 Image Generation . . . 49
4.6 Results . . . 50
4.6.1 Processing Time . . . 53
4.7 Summary . . . 53
4.8 Observations . . . 54

5 Shape Model and Threshold Extraction **55**
5.1 Motivation . . . 55
5.2 Related Work . . . 57
5.3 Algorithm . . . 58
5.3.1 Boundary Point Generation . . . 59
5.3.2 VDM Generation and Simplification . . . 60
5.3.3 Shape Gradient Computation . . . 60
5.3.4 Shape Gradient Analysis . . . 62
5.3.5 Shape Model Generation . . . 63
5.4 Results . . . 63
5.5 Summary . . . 67
5.6 Observations . . . 67

6 Two-Dimensional Shape Simplification **68**
6.1 Motivation . . . 68
6.2 Background . . . 69
6.3 Related Work . . . 70
6.4 Algorithm . . . 72
6.4.1 Medial Axis Construction . . . 73
6.4.2 Area-Based Pruning . . . 74
6.4.3 Feature Reconstruction . . . 75
6.4.4 Shape Reconstruction . . . 79
6.5 Results . . . 79
6.5.1 Limitations . . . 82
6.6 Summary . . . 82

6.7 Observations . . . 83

7 Three-Dimensional Shape Simplification 84

7.1 Motivation . . . 84
7.2 Background . . . 85
7.3 Related Work . . . 86
7.3.1 Summary of Amenta and Kolluri's Algorithm . . . 87
7.4 Medial Axis Computation . . . 88
7.5 Simplification . . . 89
7.5.1 Parts Decomposition . . . 89
7.5.2 Pruning . . . 91
7.5.3 Topology Constraints and Pruning Order . . . 92
7.6 Surface Reconstruction . . . 94
7.7 Results . . . 94
7.8 Summary . . . 99
7.9 Observations . . . 99

8 Surface Reconstruction 100

8.1 Motivation . . . 100
8.2 The Power Crust . . . 101
8.3 Surface Reconstruction Using Singular Points . . . 101
8.3.1 Related Work . . . 102
8.3.2 Algorithm Overview . . . 102
8.3.3 Singular Point Computation . . . 103
8.3.4 Polygon Computation . . . 104
8.3.5 Undersampled Areas . . . 105
8.3.6 Results . . . 106
8.4 Summary . . . 108
8.5 Observations . . . 108

9 Conclusions and Future Work 110

9.1 Summary of Results and Observations . . . 110
9.1.1 Image Matching and Interpolation . . . 110

9.1.2 Shape Model and Threshold Extraction 111
9.1.3 Two-Dimensional Shape Simplification 111
9.1.4 Three-Dimensional Shape Simplification 112
9.1.5 Surface Reconstruction 112
9.2 Conclusions 113
9.3 Future Work 114
9.3.1 Representation Properties 114
9.3.2 Similarity Measure 115
9.3.3 Validation 115

Bibliography **116**

Index of Terms **128**

List of Tables

2.1 Comparison of common 3D representations used for shape operations, including the VBM 19

4.1 Parameter values used for our image matching experiments 50
4.2 Timing results for VBM image interpolation algorithm 53

6.1 Characteristics of the test objects used in our 2D medial axis pruning experiments 79

7.1 Processing times for 3D shape simplification 94

8.1 Number of polygons used by Geomview to render a single ball 100
8.2 Processing times for polygonal surface reconstruction from VBMs 106

List of Figures

1.1 The main idea behind this thesis: shape-driven modelling 1
1.2 Stability comparison between Voronoi disks and Delaunay triangles 4
1.3 Problematic test case for boundary-based similarity measures 4
1.4 Test cases for shape similarity measure flexibility 5
1.5 Relationship between shape model properties and the applications developed for this thesis 10
1.6 An example of the results produced by our image interpolation algorithm 11
1.7 An example of the results produced by our algorithm for extracting shape information from images . . 12
1.8 An example of the results produced by our 2D shape simplification algorithm 12
1.9 An example of the results produced by our 3D shape simplification algorithm 13
1.10 An example of the results produced by our algorithm for computing polygonal surfaces from VBMs . 14

2.1 (a) Medial axis of a rectangle (b) Instability of the medial axis 18
2.2 Example of Pizer's figural shape 18
2.3 Example of Hoffman and Richard's partitioning scheme 20
2.4 (a) Siddiqi and Kimia's Shape Triangle (b) Neck-based parts (c) Limb-based parts 21
2.5 Example for which an interior-based similarity measure may be better than a boundary-based one . . 22
2.6 Example for which a boundary-based similarity measure may be better than an interior-based one . . 23

3.1 Computation of a Voronoi Disk Model 31
3.2 (a) Voronoi diagram (b) Delaunay triangulation 32
3.3 Power diagram of eight disks 33
3.4 Comparison between Voronoi balls and Voronoi polar balls 35
3.5 An r-regular object 36

4.1 Graph showing VBM stability 39

4.2 Mathematical definition of sphericity . . . 42
4.3 Example of clustering for VDM simplification . . . 42
4.4 (a) Gradient vector from a disk to its neighbour (b) Feature distance between two disks . . . 44
4.5 Main algorithm steps for image interpolation using VBMs . . . 45
4.6 Test images for image interpolation using VBMs . . . 45
4.7 (a) Face image with pixel values inverted (b) Boundary points computed from the inverted pixel values 46
4.8 VBMs computed from the face images in Figure 4.6a . . . 47
4.9 VBMs computed from the CT images in Figure 4.6b . . . 48
4.10 Computing an image from an interpolated VBM . . . 49
4.11 Interpolated images from the faces in Figure 4.6a . . . 50
4.12 Interpolated images from the faces in Figure 4.6a, with manual feature specification for the lips . . . 51
4.13 Interpolated images from the CT slices in Figure 4.6b . . . 52
4.14 Rigid registration of Visible Man images using VBMs . . . 52

5.1 (a) Sharp ‘E’ image (b) Blurred ‘E’ image (c) Noisy ‘E’ image . . . 56
5.2 (a) Intensity histogram for blurred ‘E’ image (b) Intensity histogram for noisy ‘E’ image . . . 56
5.3 (a) Shape gradient plot for blurred ‘E’ image (b) Shape gradient plot for noisy ‘E’ image . . . 57
5.4 (a) Simple test image for shape model extraction algorithm (b) Overview of algorithm for shape model extraction via shape gradients . . . 58
5.5 (a) Volume created from the test image in Figure 5.4a (b) Boundary points created from intersecting the volume with a z-plane . . . 59
5.6 (a) VDM computed from the boundary points in Figure 5.5b (b) Simplified version of the VDM in (a) 60
5.7 VDMs computed from the image in Figure 5.4a at three intensity levels . . . 60
5.8 Shape gradient plot computed from the test image in Figure 5.4a . . . 62
5.9 (a) Brain MRI image (b) Shape gradient plot computed from the brain MRI image . . . 64
5.10 VDMs computed from the brain MRI (Figure 5.9a) at two levels . . . 64
5.11 (a) Lower abdomen CT image (b) Shape gradient plot computed from the lower abdomen CT image . 65
5.12 Gaussian smoothed version of the shape gradient plot in Figure 5.11b . . . 65
5.13 VDMs extracted from the abdomen CT image . . . 66
5.14 Shape gradient plot computed from the abdomen CT image using our modified similarity measure . . 66

6.1 Example of typical 2D medial axis pruning, showing lost features . . . 69
6.2 Medial axis of a pencil as computed by our algorithm . . . 70

6.3 Examples of medial axis pruning using Ogniewicz's algorithm 71
6.4 Applying our 2D medial axis noise removal algorithm to the maple leaf example 73
6.5 Area-based pruning of the 2D medial axis 74
6.6 Feature reconstruction of the 2D medial axis after pruning 76
6.7 Effect of smoothness constraints on feature reconstruction of the 2D medial axis 76
6.8 Axial gradient between two nodes of the 2D medial axis 77
6.9 Axial gradient at a branch node of the 2D medial axis 77
6.10 Branch extension for feature reconstruction of the 2D medial axis 78
6.11 Results of applying our 2D medial axis noise removal algorithm to Ogniewicz's leaf 80
6.12 Results of applying our 2D medial axis noise removal algorithm to Ogniewicz's goat 80
6.13 Results of applying our 2D medial axis noise removal algorithm to the lizard example 80
6.14 Results of applying our 2D medial axis noise removal algorithm to the brain MRI example 81
6.15 Results of applying our 2D medial axis noise removal algorithm to the ragged rectangle example . . . 81
6.16 Example of how a narrow neck can cause problems in our 2D medial axis noise removal algorithm . . 82

7.1 Example of medial axis disintegration caused by regularization 86
7.2 Processing pipeline for 3D shape simplification and surface reconstruction 87
7.3 Amenta and Kolluri's algorithm for computing the medial axis from the power shape 88
7.4 Illustration of how duplicate singular points are produced by Amenta and Kolluri's algorithm 89
7.5 Parts decomposition of the medial axis of a rectangular box 90
7.6 Feature of an object and the tetrahedra used to estimate its volume 92
7.7 Examples of 3D medial axis parts that form cycles 92
7.8 Graphs representing the topology of the parts in Figure 7.7 93
7.9 Topology graph of a three-holed torus 93
7.10 Bunny model at three levels of detail 95
7.11 Topology graph of simplified bunny model (middle bunny of Figure 7.10) 95
7.12 (a) Tweety model (b) Medial axis (c) Closeup of noise artifacts (d) Simplified medial axis (e) Closeup of clean model 96
7.13 (a) Max Planck model (b) Medial axis (c) Simplified medial axis (d) Simplified medial axis, ear removed (e) Max Planck model, ear removed 97
7.14 (a) Hip bone model (b) Medial axis (c) Simplified medial axis (d) Simplified hip bone model 98

8.1 Main steps of the power crust algorithm 101

8.2 (a) A union of balls and some of its singular points (b) The dual shape of the union of balls 102
8.3 Connecting the singular points of a union of disks to form a boundary 103
8.4 Singular point computation using the orthocentre of a simplex 104
8.5 Finding the right triangle neighbours and singular points during polygon computation 105
8.6 Surface reconstructed from the VBM of an apple 106
8.7 Surface reconstructed from the VBM of a mushroom 107
8.8 Surface reconstructed from the VBM of a heart 107
8.9 Surface reconstructed from the VBM of a two-holed torus 108

Dedicated to the memory of Dr. Alain Fournier.

Acknowledgments

Academic Advisors

Dr. A. Fournier

Dr. W. Heidrich

Dr. D. Lowe

Dr. D. Silver,
External Examiner

Examination Committee

Dr. A. Hodgson

Dr. K. Booth

Dr. M. van de Panne

Family

Dad

Mom

Big
Brother
Tandy

Shelby

Baby Lenny

Baby
Sister
Sandy

Patrick

...and Grandma.

Alex

Sally

Sarah

Hannah

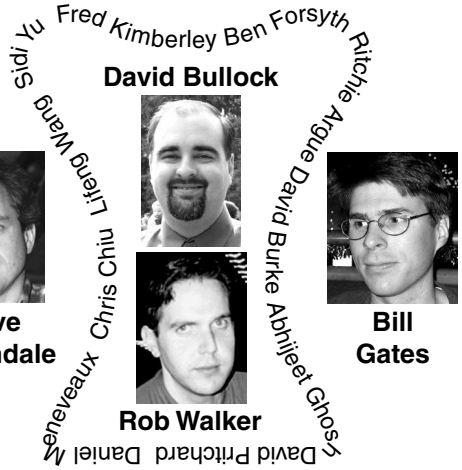
Leslie

Friends

Lisa
Streit

Rob
Scharein

Dr. P. Cahoon



David Bullock

Dave
Martindale

Rob Walker

Bill
Gates

ROGER C. TAM

University of British Columbia
February, 2004

Chapter 1

Introduction

1.1 Motivation

Shape analysis is a vital aspect of many areas of computer graphics, scientific visualization and computer vision. The shapes of objects in an image, volume or 3D scene provide high-level information that can be used for many tasks. Examples of applications that often utilize shape information include segmentation, level-of-detail modelling, registration and object recognition. The traditional approach to object modelling for shape-related applications focuses narrowly on the goals of the particular task at hand and often results in models that do not capture many shape properties that are useful for other applications. For example, in segmentation the model is usually designed to represent the boundary of the object, because the goal is to separate the object of interest from the rest of the image (*snakes* [75] are a classic example). In contrast, when designing a model for use in registration (*e.g.,* [163]), a representation that includes the interior of the object is usually advantageous.

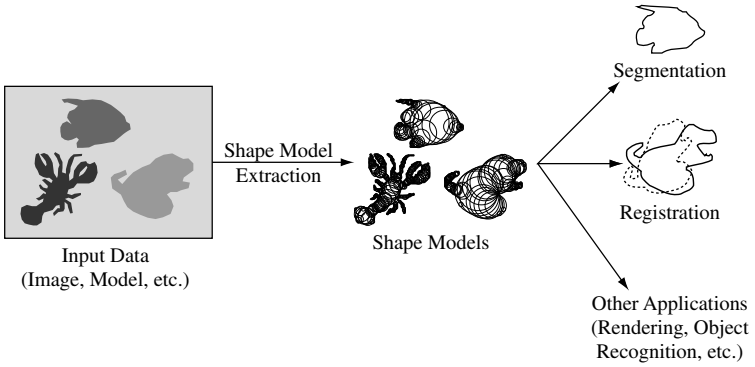


Figure 1.1: The main idea behind this thesis: shape-driven modelling

This thesis takes an alternative approach to object modelling that focuses primarily on shape representation. The ultimate goal of this project is to develop a shape model that is useful for a broad range of shape-driven tasks. Figure 1.1 illustrates this approach. Such a shape model has the potential to increase the efficiency of any graphics, visualization or vision pipeline that consists of more than one shape-related application. For example, a user may want to segment out an object from an image, register the object with another, then simplify the model before finally rendering it. A shape model that incorporates the common elements required by multiple applications can effectively reduce the amount of development time and speed up the functioning of the resulting shape processing pipeline.

1.2 Shape and Shape Models

Although shape is a very intuitive and commonly used property in everyday life, it is difficult to define precisely and even more difficult to quantify. We adopt a working definition from Lord and Wilson [92], who describe shape as "the characteristic way in which an object occupies space". Although seemingly vague, this definition can be used to elucidate several points. The shape of an object in this sense is a characteristic of both the boundary and the interior of the object, as the boundary on its own cannot fill space. In addition, this notion of shape is independent of translations, rotations and uniform scaling. These transformations only have meaning relative to the object's environment, so the location and size of the object are not characteristics of the object alone, and therefore are not shape properties.

Because the definition of shape itself tends to be imprecise, it is difficult to explicate what constitutes a good shape model. However, by understanding the intended purpose of the model, we can identify key properties that can contribute to its effectiveness. Our research focuses on the computational aspects of a shape model and the efficiency with which it can support operations for a variety of shape-based applications.

1.2.1 Common Shape-Driven Operations

There are a number of common shape-driven operations at the core of our research. These operations have been chosen because of their extensive usage in many applications:

- **Extraction.** This is the first operation in most shape processing pipelines, and consists of extracting shape models of objects from raw data such as voxel values (*e.g.*, CT/MRI scan) or surface samples (*e.g.*, laser scanner).
- **Simplification.** Simplification reduces the amount of detail in a given shape, and is frequently applied as a preprocessing step to increase the efficiency of subsequent operations.
- **Matching.** This operation is frequently used in applications such as object recognition and registration. The key component of an effective shape matching algorithm is the *shape similarity measure* used to quantify the differences between shapes.

- **Interpolation.** Shape interpolation provides a way to generate an "average" shape given two or more existing shapes, and is frequently used in *morphing* applications and statistical shape analysis.
- **Manipulation.** In many interactive modelling and animation applications, an efficient method for manually shaping an object is required.
- **Surface reconstruction.** This operation consists of generating an accurate surface of an object from its shape model. This process is particularly important for many graphics applications, because most graphics tools (*e.g.*, OpenGL® [168]) use surface polygons as the primary primitive.

There are many other operations that often require shape processing, such as rendering and object design, that are used by a smaller number of applications. Although it would be useful to consider these additional operations, we restrict our focus to the ones listed above in order to maintain a reasonable scope to the thesis.

1.2.2 Key Properties of a Computational Shape Model

We have identified a number of fundamental properties that affect the effectiveness of a multipurpose shape model. As discussed below, the presence of these properties determines the extent to which a shape model can provide efficient support for a wide range of data and commonly required operations.

The properties that we focus on are stability, flexibility, accuracy, complexity, efficiency, and intuitiveness. Among these, stability and flexibility are the most crucial in determining the range of applicability of the shape model, and therefore receive the most attention. It should be noted that these properties are *not* independent. For example, a representation that has a great deal of domain-specific knowledge built into it is typically both complex and inflexible.

Stability

The *stability* of a shape model refers to how much the model changes when the input data is altered. Ideally, the degree of change in the model should correspond to that occurring in the data; that is, a small change in the data should not cause the model to differ greatly. Stability is important because few sources of data are perfect in their precision or accuracy. For example, medical image data is often blurry or noisy. Small perturbations in the input data can cause large changes in an unstable model, and any operations performed on the model are consequently unstable. In addition, even if the representation itself is stable, instability can still arise from the operation being applied. For example, when computing the shape distance between two objects, a small change in one of the objects may cause an unstable similarity measure to output a radically different result.

Figure 1.2 illustrates the tradeoffs in terms of stability between two object representations. A comparison is done between the Delaunay triangulation [51], a common method for tessellating 2D objects, and the set of disks circumscribing the triangles. In this example, two almost circular objects are reconstructed using eight boundary

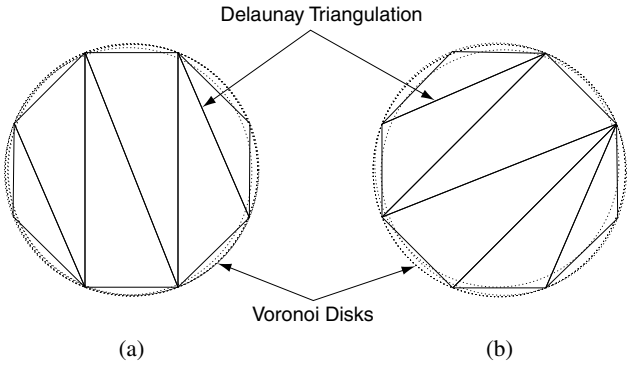


Figure 1.2: Stability comparison (the Voronoi disks are more stable than the Delaunay triangles)

points each. The vertices in Figure 1.2b are the result of perturbing the vertices in Figure 1.2a by small amounts. A Delaunay triangulation is formed from each set of vertices and a union of disks is then formed from each triangulation by computing the circumscribing disk of each triangle. These disks are called *Voronoi disks* because their centres are Voronoi vertices (more precise definitions are provided in Chapter 3). The two resulting triangulations are clearly quite different, whereas the two resulting unions of disks appear quite similar. However, the unions of disks clearly have a significant amount of redundancy. This example is meant to demonstrate several points:

- There are cases in which the Voronoi disk representation is clearly more stable than the triangle representation, and small changes in the input data result in much greater changes in the triangulation than in the Voronoi disks. Note that in all cases where the triangulation is stable, the Voronoi disks are stable as well.
- Even when a shape model appears to be stable, operations performed on it may be unstable. The redundancy in the two unions of disks can potentially cause instabilities in any similarity measure applied to the models. Examples of this type of stability are discussed in Chapter 5.
- Simplification can be an important process for increasing the stability of a shape model by reducing the amount of redundancy and removing the components associated with minor features in the input data. This provides the motivation for the simplification algorithms discussed in Section 4.4.1, and Chapters 6 and 7.

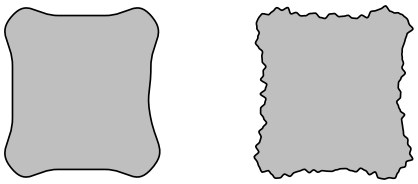


Figure 1.3: Problematic test case for boundary-based similarity measures

Figure 1.3 shows an example that would reveal the instability of many commonly used similarity measures. Even though the two objects are close in overall shape and differ only in minor perturbations of the boundary, this would cause problems for most similarity measures that only use boundary curvature or angles to determine shape differences (*e.g.*, measures that employ the *turning function*, such as [12, 166]). Curvature or angle-based measures are often unstable in that even small changes in the boundary can cause the measure to report large differences.

Flexibility

A multipurpose shape model is expected to be able to represent real and virtual objects from a wide variety of data sources. Because there is no predetermined source of data, the objects represented can be arbitrarily complex. Therefore, flexibility with respect to shape complexity is an essential property. Flexibility increases the variety of objects that can be represented as well as the types of operations that can be performed on them. A key to flexibility is to make as few assumptions as possible about the input data. For example, some shape models impose smoothness constraints on the object boundaries, which limit the types of objects that can be represented.

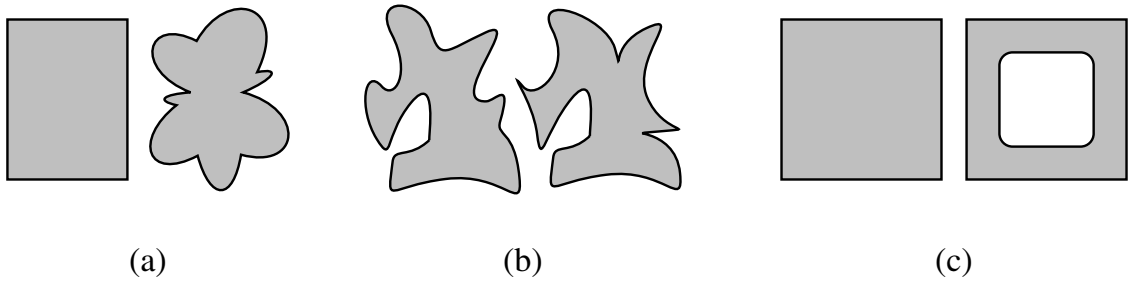


Figure 1.4: Three cases that would test the flexibility of a shape similarity measure

Figure 1.4 illustrates the flexibility requirement for similarity measures. The three test cases shown require a similarity measure that is more flexible than most of the commonly used methods. In Figure 1.4a, the two objects differ significantly in the amount of curvature in their boundaries and the number of potential landmark points. This makes comparing the two shapes difficult for measures that are polygon (*e.g.*, [12]) or landmark-based (*e.g.*, [30]). Also, most primitive-based methods would represent each of these objects with a different number of primitives, which is problematic for similarity measures that can only operate on models with the same number of primitives (*e.g.*, *figural shape* [122]). Figure 1.4b would be a difficult case for similarity measures associated with shape models that require a certain degree of smoothness, typically defined by differentiability, in the object's contour. Some deformable models [147] fall into this category. In this case the second object has a number of cusps that would contradict the differentiability requirement. In Figure 1.4c, the two objects are of different topology. Most shape similarity measures only deal with objects of genus zero, and would have trouble with the hole in the second object.

Accuracy

Most shape representations cannot capture the shapes of all objects exactly, and usually they must form an approximation of the space occupied by the objects instead. For example, deformable models [147] tend to smooth out the sharp features of an object. Another example are primitive-based models, where the shape of the primitive (*e.g.*, polygons vs. quadric patches) has a strong effect on the amount of approximation error. This error is usually expressed as a distance between points on the boundary of the object and the corresponding points on the border of the model. The amount of acceptable error depends on the application, but computable and reasonably tight bounds on the error are desirable in most cases. Another important consideration is how the amount of error changes as the sampling density increases. Ideally, as the number of samples gets larger, the resulting model should become an increasingly more faithful representation of the original object. Sometimes, accuracy can be sacrificed to facilitate certain operations. For example, a surface may need to be smoothed to make it differentiable.

In addition to approximation error, another important aspect of accuracy is related to the resolution to which the similarity measure is able to discriminate between shapes. When used for matching tasks, an accurate similarity measure gives reasonable correspondences with relatively few false matches. Unfortunately, the accuracy of shape matches is notoriously difficult to validate formally, and the acceptability of the final results is often determined by the objective judgment of the users.

Complexity

The geometric and/or mathematical complexity of a shape model affects its applicability in a number of important ways. Very simple models, such as contours used to represent 2D boundaries, have the advantages of adaptability and low storage requirements, but suffer from a lack of support for shape computations. For example, contours can be used as a starting point for practically any shape operation, but require augmentation of the model for many (even simple) applications, particularly those in which the interior of the object is important. Models that are very complicated tend to be so because they are specially designed for a narrow range of applications and lack adaptability as a result. In addition, computing complicated models can be a time-consuming task. For example, many component partitioning schemes have been proposed to compute parts-based models for object recognition (examples are given in [126, 150]). Another aspect of a model's complexity is mathematical complexity. In general, the models that are designed to work with numerical methods are more mathematically complex (*e.g.*, *levels sets* [98, 139]), whereas the ones designed for predominantly combinatorial solutions tend to be less complex.

From experience, we are able to derive a few basic guidelines that, when adhered to, should result in a representation that has a moderate degree of complexity:

- Assuming the source data is clean, generating a model should be automatic. This process should take advantage

of any stability and flexibility properties present in the model.

- There should be basic support for operations such as shape matching, interpolation, and simplification.
- The model and basic operations should be adaptable for a wide variety of applications. For example, the model should facilitate component partitioning for object recognition if desired. Ideally, the model should allow for the addition of application-specific features without changing the main features.
- The core model should not have any computationally intensive features that are needed by only a few applications. For example, hierarchical decomposition schemes that break up an object's parts and protrusions into a tree-like structure (*e.g.*, [89]) are not necessary for most purposes.
- The storage requirements for a shape model are also an important consideration for many applications. For example, databases that use shape information, such as some designed for facial recognition (*e.g.*, [25]), may store millions of entries, and the compactness or compressibility of the representation is a significant factor.

Efficiency

We use the term *efficiency* to refer to the effect of the model's design on the computing costs and development time of new applications. An efficient representation results in faster algorithms and is more likely to be adopted for implementation. The efficiency of a shape model is affected by most of the other key properties discussed. Although the correctness of our methods takes priority over efficiency, there are a number of primary issues that are of interest to us:

- The effect of the shape model's properties on the running times of algorithms is the most common focus of researchers concerned with efficiency. The stability, flexibility and complexity of the representation can strongly affect the amount of processing power required to generate and perform operations on the models.
- Another aspect of efficiency concerns the ease with which new applications employing the shape model can be implemented. A representation that requires a large amount of overhead, such as those with high mathematical complexity or low intuitiveness (defined in the next section), may hinder the development process and be more difficult to validate.

Intuitiveness

We use the term *intuitiveness* to refer to the degree that a potential algorithmic designer or end-user will find the shape model easy to learn and work with. A representation that is parameterized in a way that is descriptive in terms of shape tends to be more user-friendly. For example, contours are usually parameterized by border length, which is not very descriptive. In contrast, skeleton models are generally relatively intuitive and easy to use.

A useful resource for enhancing the intuitiveness of a shape model are studies on how humans perceive shape. Although human shape perception is not within the scope of this dissertation, we take into account some well-established theories of shape perception to guide our research and algorithmic development. The power of the human shape processing system makes it an attractive model to use in the design of a computer-based system. However, for a number of reasons, it is inappropriate to base a computational shape model entirely on human perception. First, even though visual form has been the subject of research for many years (*e.g.*, [24, 67, 79, 87, 144, 146]), there is still no universally accepted theory of human shape perception. Therefore, there are aspects of shape perception that are simply not understood well enough to implement on a computer. In addition, many useful mathematical theories of shape have been shown to contradict human perception [108, 160]. For these reasons, we only exploit a few high-level theorems on human shape perception in our work.

1.3 Voronoi Ball Models

Ball models have been used in numerous applications in graphics and visualization, such as collision detection [56, 70], surface simplification [61], and molecular modelling [88]. This dissertation presents the results of a number of experiments performed with a type of ball model we term the *Voronoi disk/ball model* (*VDM/VBM*). A VDM/VBM is defined as a subset (not necessarily proper) of the Voronoi disks/balls that are inside of the boundary of the object being represented. VDMs are typically used for 2D applications, whereas VBMs are used in 3D. There are a number of methods for classifying a disk/ball as being inside or outside. Also, as shown by the results in this thesis, various applications require ways to further filter the set of interior disks/balls. Chapter 3 gives the formal definitions of the primary geometric constructions used for forming VDM/VBMs and discusses the approximation properties of the resulting models.

Some of the algorithms in this thesis incorporate and extend the work of Ranjan and Fournier [129, 130, 131], who were the first to investigate the use of VDM/VBMs for shape matching and quantification of shape differences. We design our experiments to focus on issues undocumented or lightly treated in their work. For example, some of our test data is selected to test the flexibility of the similarity measure, which is a property that was previously unexamined.

Ranjan and Fournier called their models *Unions of Circles/Spheres*; we use the terms “disks” and “balls” instead of “circles” and “spheres” to be consistent with the terminology commonly used in computational geometry, theories of which are prevalent in parts of our work, and to emphasize the fact that our representation is intended to include the interior of the shape. In addition, where confusion is unlikely, we use “VBM” as a general term to include both VDMs and VBMs, because disks can be viewed as 2D balls.

As a consequence of their work, Ranjan and Fournier made the following observations:

- A basic VBM is easy to generate from any data that can be point-sampled.
- VBMs are *complete* in the sense that they represent the interior of an object as well as the boundary.
- The VBM representation is stable with respect to changes in the input data when Ranjan and Fournier's similarity measure (defined in Section 4.4.3) is used to mark the changes in the models.
- The *clustering* simplification algorithm (defined in Section 4.4.1) is an effective method for reducing the complexity of models while retaining their shape. The simplified models tend to have greater stability.
- The VBM representation has well-defined error bounds (discussed in Section 3.2.1).

However, our research has revealed a number of shortcomings of Ranjan and Fournier's methods. The findings that have been used to motivate our own experiments include:

- There is no connectivity between primitives, which makes tasks such as parts decomposition, topological operations, and interpolation control difficult.
- Contrary to the common belief at the time, their method for constructing the VBM results in balls that do not converge to the medial axis transform as the sampling density increases to infinity. As discussed in Section 3.1.2, convergence can be usefully exploited for stabilizing VBMs. This is the primary reason for switching to Amenta's method of VBM computation, also explained in Section 3.1.2, for our work with the medial axis.
- The similarity measure is dependent on scale, position and orientation. This contradicts the definition of shape as stated in Section 1.2. To compensate for this limitation, Ranjan and Fournier used a prealignment procedure that sometimes needs to be manually performed for complex shapes.
- The similarity measure does not make effective use of unmatched primitives.
- Simply rendering the balls is an inefficient method of displaying an object represented by a VBM when the goal is to convey the overall shape to the viewer. Even for modern graphics hardware, any VBM with more than several thousand balls can cause the frame rate to become non-interactive.
- The claim that algorithms developed in 2D for VDMs can be easily extended to 3D is only true under limited circumstances. For example, as discussed in Chapters 6 and 7, when parts-decomposition or topological processing is required, the 3D methods tend to be much more complex.

1.4 Primary Goals and Contributions

This thesis explores the relationships between the core operations and key shape model properties introduced above, as pertained to VBMs. Although the focus is on the shape representation rather any particular application, a computational shape model can only be considered useful and properly validated when applied in practical implementations. Therefore, the approach of the thesis is to develop a number of representative applications to investigate the key properties that we are interested in. Each application requires a subset of the key operations and is designed to expose the strengths and weaknesses of VBMs in the context of the specified shape model properties. Figure 1.5 summarizes the relationships explored in this thesis. The remainder of this section describes the applications that we have developed, along with some examples of the results produced by our algorithms.

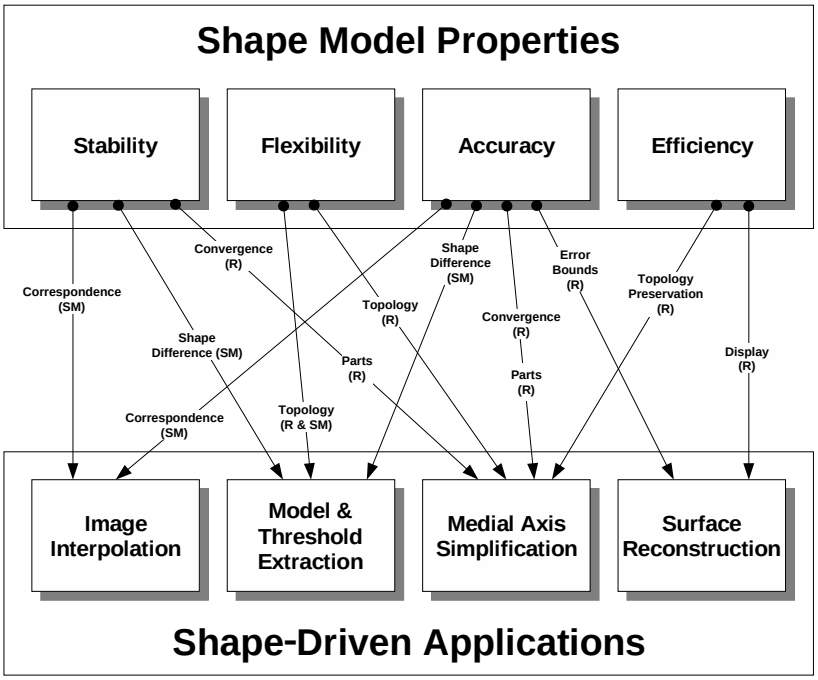


Figure 1.5: Relationship between shape model properties and the applications developed for this thesis ("R" = representation, "SM" = similarity measure)

1.4.1 Image Matching and Interpolation

In this application, we use VBMs to represent 2D images and use Ranjan and Fournier's similarity measure to establish shape-based correspondences between image features. The matches are then used to interpolate or rigidly register the

images. This application is used to establish a performance baseline for the matching method. This work goes beyond that by Ranjan and Fournier because they focused largely on matching with 2D models; our application uses 3D VBMs and more challenging data. The primary goal of the experiment is to determine whether VBMs can be used to form stable and accurate correspondences between 3D object features. The details of this work are found in Chapter 4 and [153]. Figure 1.6 shows an example of an image interpolation produced by our algorithm.

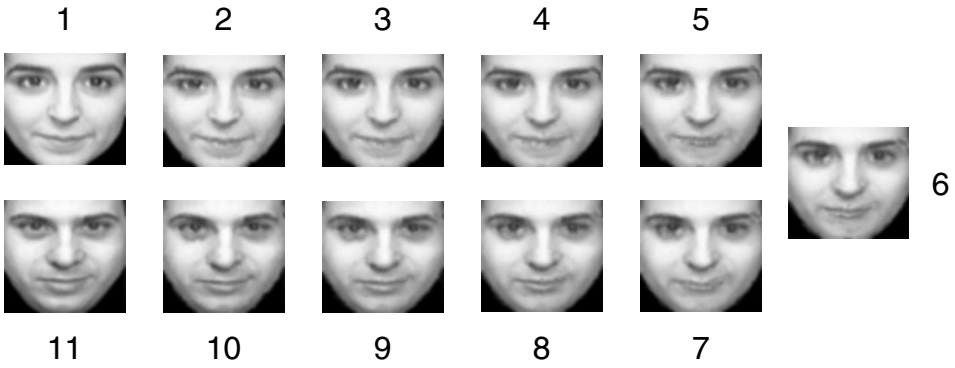


Figure 1.6: An example of the results produced by our image interpolation algorithm; the first image and the last image are original, the rest are interpolated

1.4.2 Shape Model and Threshold Extraction

This application uses VDMs to extract shape information from 2D images. The algorithm detects and extracts significant objects from images with no *a priori* knowledge of the objects being represented, and is designed to test the ability of the similarity measure to accurately quantify the differences in 2D shapes in the presence of significant variations in shape complexity and topology; this is markedly different from Ranjan and Fournier's experiments, in which the shapes used were largely similar (*e.g.*, animal silhouettes of the same topology). While the evaluation of the results of our image matching experiments is largely based on subjective judgment, a more objective and quantitative analysis is done for this work. Our experimental results for this application are discussed in Chapter 5 and [154]. Figure 1.7 shows an MRI image and two shape models extracted by our algorithm.

1.4.3 Shape Simplification Based on the Medial Axis

Our experiments with image interpolation and evidence from Ranjan's dissertation [129] show that grouping primitives to provide connectivity and topological information can be an important aspect of shape modelling. To this end, we implement two applications, one working in 2D and the other in 3D, to explore how the *medial axis*, a shape model widely used to concisely represent topological information, can be used in conjunction with the VBM. The applications

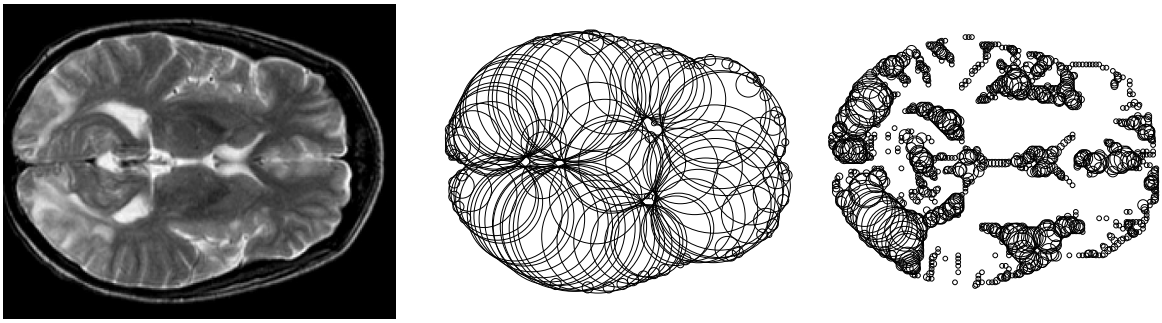


Figure 1.7: An MRI image and two shape models extracted by our algorithm

are designed to exploit the properties of the two representations in a complementary manner. In both cases, shape simplification is performed while preserving the topology of the object. The medial axis is computed from the VBM, then used to enforce topological constraints while the model is simplified.

In addition, both applications demonstrate how a partitioning scheme can be applied to a VBM to facilitate operations on groups of primitives. In the 2D case, we use local shape information defined on the VDM of a 2D object to remove noise-type artifacts from the boundary of the object without sacrificing the fine but significant features. This algorithm is discussed in Chapter 6 and [155]. While no explicit partitioning is applied, the medial axis is used to determine how much of the shape information computed in one area of the object can be used in another area of the object. Figure 1.8 shows an example of applying our 2D simplification algorithm to a leaf model.

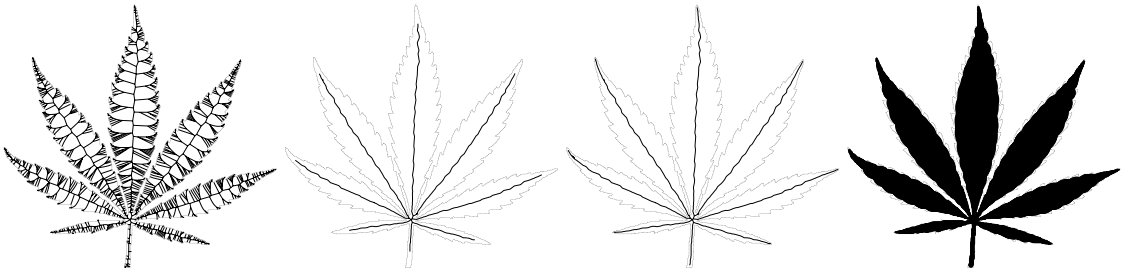


Figure 1.8: An example of the results produced by our 2D shape simplification algorithm (a) A leaf and its medial axis (b) Denoised medial axis (c) Denoised medial axis after feature reconstruction to recover fine details (d) Denoised shape with original boundary superimposed

In the 3D case, the medial axis is decomposed into distinct components before simplification, which helps to define the features of the object, and makes maintaining the topology of the axis more efficient. The preservation

of the topology of 3D shapes during processing is a difficult task in general, and the fact that our representation incorporates this ability is a particularly strong contribution of the thesis. This algorithm is discussed in Chapter 7 and [156]. Figure 1.9 shows an example of how our simplification algorithm is able to effectively reduce the amount of detail in a bunny model.

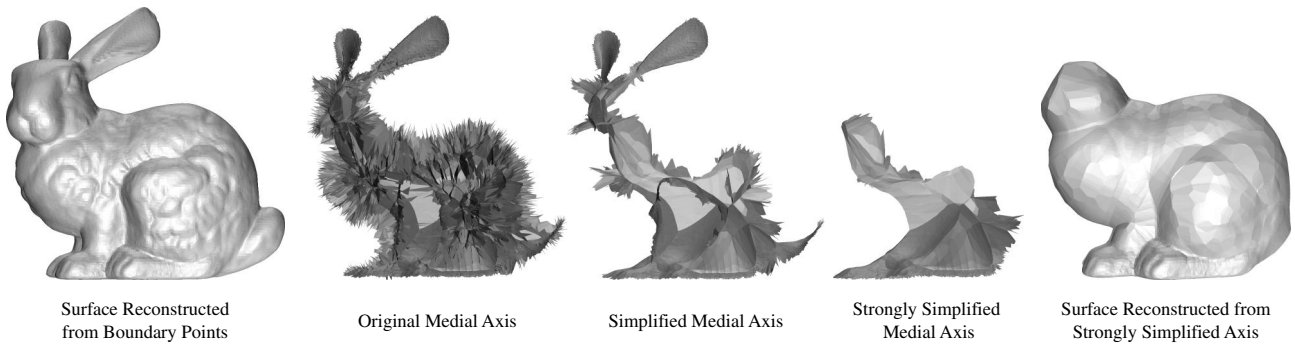


Figure 1.9: The results of applying our 3D shape simplification algorithm to a bunny model

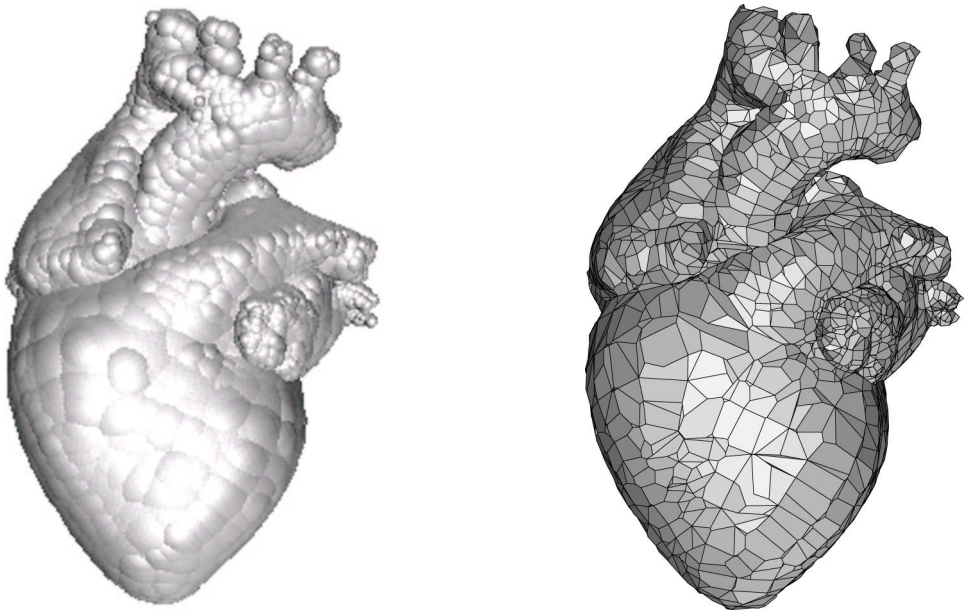
1.4.4 Surface Reconstruction

While VBMs can be shown to have good approximation properties (Chapter 3), a polygonal surface of the model is often desired to enhance its visual quality and display efficiency. In addition, the goal of applications such as segmentation is to generate an accurate boundary of the object. In Chapter 8, we discuss the two algorithms that we use for surface reconstruction from VBMs:

1. We use Amenta *et al.*'s *power crust* algorithm [7] in combination with our medial axis-based shape simplification method to produce polygonal models of the simplified objects. In addition to the VBM of the object, the power crust algorithm requires a union of balls covering the outside of the object, typically computed from a set of sample boundary points, in order to compute a surface.
2. We present a novel, compact algorithm for generating a surface from a VBM without the sample points or other additional information. Figure 1.10 shows an example of the results produced by our surface reconstruction algorithm.

1.4.5 Thesis Statement

The primary goal of this thesis is to show that Voronoi Ball Models, by virtue of the existence of a number of key properties, have the capabilities to be broadly applicable in shape-driven applications in computer graphics, scientific



(a) **(b)**

Figure 1.10: (a) VBM of a heart (b) Surface reconstructed from the VBM of a heart

visualization and computer vision. The main approach employed is the development of a number of applications requiring commonly-used shape operations and designed to be challenging for any shape model.

1.5 Document Overview

This dissertation consists of nine chapters. This chapter is the first. The second chapter provides a broad survey of the related work in computational shape modelling. Chapter 3 outlines our methods for computing VBMs and the approximation properties of the resulting models. The fourth chapter presents our work with using VBMs for image matching and interpolation. The fifth chapter analyzes our use of VDMs for extracting shape information from images. The sixth and seventh chapters discuss our experiments with using the medial axis and VBM together in 2D and 3D. Chapter 8 explains how we compute polygonal surfaces from VBMs. The final chapter presents a summary and our conclusions.

Chapter 2

Related Work

This chapter is an overview of the work done by other researchers on topics that are related to this thesis. There has been a vast amount of research done in the areas of shape representations, shape similarity measures and shape-related applications. Therefore, it is impractical to mention every publication specifically and we only summarize a number of key papers and refer the reader to survey papers for a broader perspective where appropriate. However, for the five applications that we have developed for this thesis, we provide a more detailed summary of the related work in the appropriate chapters. For the reader interested in a general overview of shape analysis techniques, a good starting point is the survey by Loncaric [91].

2.1 Shape Representations

The numerous approaches to modelling shape can be roughly divided into two categories. One category deals with the design and use of shape models to perform certain computational tasks, and the other consists of work that attempts to replicate how humans perceive shape. These categories overlap significantly and there is much current research that incorporates both of these elements of shape modelling. It should be noted that some of the models discussed in this chapter are not, strictly speaking, "shape" models as defined in Chapter 1; they are, however, used to perform shape-related operations and are therefore included for completeness.

2.1.1 Application-Oriented Models

Point-Based Representations

A point-based model consists of samples on the surface or interior of an object. These models are very simple and are often converted to other types of models (*e.g.*, via some method of tessellation) for further processing. Landmark-

based models are a particular type of point-based representation in which the points are located according to some application-specific rule. For example, points in an object's contour where the curvature is high are often used as landmarks. Some areas of shape analysis, such as *morphometrics* (*e.g.*, [30, 132]), use landmark-based methods exclusively.

Boundary Representations

Boundary representations, or *B-reps*, only represent the contours (2D) or surfaces (3D) of objects. B-reps are composed of interconnected vertices, edges and faces. Each face usually has a compact mathematical representation in that it lies on a single planar, quadratic, or parametric surface. The most common face type is the planar triangle. Triangular meshes are heavily used in graphics rendering. See Mäntylä [99] for a more thorough description of B-reps and associated data structures.

Spatial Subdivision Representations

Spatial subdivision representations are computed by dividing the area or volume occupied by the object into cells. Hoffmann and Rossignac [68] classify subdivision representations into two types: *boundary conforming* and *boundary approximating*. Examples of boundary conforming subdivision models include meshes (*e.g.*, tetrahedral) and *binary space partitioning trees* (*BSP's*) [110]. Examples of boundary approximating representations are grids and *octrees* [133]. There is a wide range of applications for spatial subdivision models, including finite element analysis (tetrahedral meshes), hidden surface removal (BSP's), collision detection (octrees), and various other graphics applications (*e.g.*, [53]).

Deformable and Implicit Models

Deformable models can be curves, surfaces or solid models, although the amount of research done on deformable curves and surfaces far exceeds the work done on solid ones. The book by Singh *et al.* [147] includes a representative collection of papers on deformable models. McInerney and Terzopoulos provide a comprehensive, though somewhat outdated, survey of deformable models in medical image analysis [102].

The central idea of these methods is the minimization of energy to satisfy the following Euler-Lagrange equation (for simplicity, the 2D version of the equation is shown here; the 3D version is similar):

$$\underbrace{\overbrace{-\frac{\partial}{\partial s}(w_1(s)\frac{\partial \mathbf{v}}{\partial s})}^{\text{stretching}} + \overbrace{\frac{\partial^2}{\partial s^2}(w_2(s)\frac{\partial^2 \mathbf{v}}{\partial s^2})}^{\text{bending}}}_{\text{internal forces}} + \underbrace{\nabla P(\mathbf{v}(s))}_{\text{external forces}} = \mathbf{0}$$

where $\mathbf{v}(s) = (x(s), y(s))^\top$ is a parameterization of the object's contour, $w_1(s)$ and $w_2(s)$ are variable weights controlling the contour's "tension" and "rigidity", respectively, P is a scalar potential function defined on the image plane, and ∇ is the gradient operator. The first two terms of the equation represent the internal stretching and bending forces, respectively, and the third term represents the external forces that couple the model to the image data.

Snakes (2D) [75] and *superquadrics* (3D) [158] are classic examples of deformable models. Although successfully used for many applications (especially in medical image analysis, where segmentation, registration and motion tracking are the primary uses of these models), most deformable models suffer from the drawback of not being able to handle topological changes during model generation or evolution. McInerney and Terzopoulos extend standard snakes and deformable surfaces to make them topologically adaptive [103, 104].

Implicit models [27] are a class of deformable models that are parameterized implicitly. An example is the *level sets* approach [98, 139] that is most commonly used for segmentation and surface blending. Implicit models can overcome some of the limitations of traditional parametric deformable models, because they naturally handle topology changes well, and can represent surfaces with sharp corners and cusps. The main problem with implicit models is that the typical formulation makes the imposition of topological and geometric constraints more difficult. In addition, the parameterization does not correspond to an intuitive notion of shape

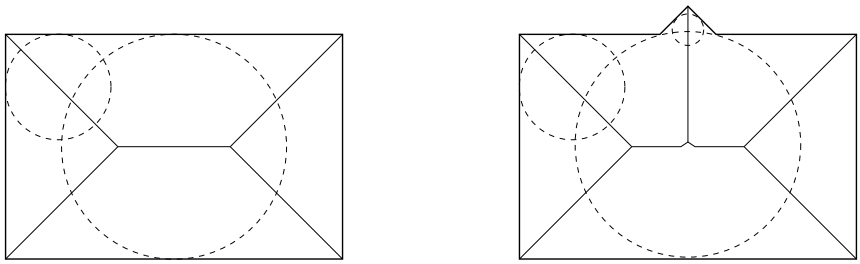
Medial Axis Representations

Another group of popular shape models are based on a *medial axis* (often called *medial surface* in 3D) or *skeleton* representation pioneered by Blum and Nagel [28, 29]. In 2D, the medial axis of an object is defined as the closure of the loci of the centres of disks that are maximally inscribed within the object's region. The medial surface is similarly defined with balls. The medial axis and the maximal disks/balls are collectively referred to as the *medial axis transform* (*MAT*). Figure 2.1a shows the medial axis and two maximal disks of a rectangle.

There are many papers on the computation and application of 2D skeletons. Examples include the work by Lee [82] and Ogniewicz [112]. The 3D skeleton is not nearly as well-studied. Notable papers include those by Brandt [31], Attali and Montanvert [13, 15], and Amenta and Kolluri [9, 10]. Other algorithms for calculating the 3D medial axis include those by Sheehy *et al.* [141] and Sherbrooke *et al.* [142].

One of the major problems with the medial axis representation is its instability with respect to local perturbations in the object's boundary. These perturbations can result in large branches in the medial axis, even if the changes in the boundary are small. Figure 2.1b shows the result of changing the boundary in Figure 2.1a by a small amount. Consequently, a large branch in the medial axis appears. We summarize the related research on the stabilization and simplification of the medial axis, as well as present our own algorithms, in Chapters 6 and 7.

The most comprehensive medial-based approach for shape analysis to date is by Pizer *et al.*, who propose a



(a) **(b)**

Figure 2.1: (a) Medial axis of a rectangle (b) Instability of the medial axis

shape model that can be used for various applications in medical image processing (segmentation, registration and measurement of shape variation [122]) and computer graphics (multiscale modelling and rendering [125]). Pizer's approach represents shapes using interconnected *figures*, where a figure is a whole object, the main part of an object, or a protrusion or indentation in another figure (Figure 2.2). Notable features of this representation include a measure for the quantification of shape variation, four levels of coarseness for multiscale processing, and positional tolerance for enhanced stability. One of the drawbacks of the Pizer model is its mathematical complexity [123] and consequent computational costs. Another limitation is that the similarity measure only works with objects having the same number of figures.

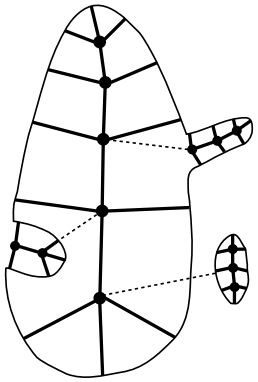


Figure 2.2: Example of Pizer's figural shape, with four figures

Comparison of Shape Models

Table 2.1 summarizes the strengths and weaknesses of a number of well-known models. We include the VBM for a direct comparison. The ratings for the VBM are derived from the results of our experiments, presented in subsequent chapters, while the ratings for the other representations are estimated from a survey of the related literature. In this table, **stability**, **flexibility**, **accuracy**, **complexity**, and **intuitiveness** are key properties discussed in Section 1.2.2. **Generation** refers to the ease with which the model can be generated. This takes into account the theoretical complexity of each model, the computational overhead of the algorithms used to create the model, and the amount of user intervention normally required. **Shape Operations** refers to the amount of support the representation has for the shape operations discussed in Section 1.2.1. The similarity measure(s) most commonly associated with the model, as well as the completeness of the model (*i.e.*, whether the model represents the interior as well as the boundary), are among the important considerations. It should be noted that of all the models in the table, only the last three (deformable models, figural shape, and VBMs) are designed with shape operations in mind.

Property / **Representation**	Generation	Stability	Flexibility	Accuracy	Complexity	Intuitiveness	Shape Operations
Surface Points	G+	P	G	G	G+	P	P
Surface Mesh	G	P	G	G	G	M-	P
Octree	G	P	M	M-	M	M	P
Tetrahedra	G	M-	G	G	G	M	P
Deformable Superquadric	M	M+	M	G	M	M+	M
Figural Shape	M-	M+	M-	G	M-	G	G
Voronoi Ball Model	G-	G	G	G	G	M+	G

'G' = Good, 'M' = Medium, 'P' = Poor

Table 2.1: Comparison of common 3D representations used for shape operations, including the VBM

2.1.2 Visual Form Representations

There has been much research done into how humans perceive shape visually. In this section, we provide a sparse but representative sampling of the work done in this field by describing a number of shape models related to perception. The most common application of the computational modelling of visual form is object recognition, with the most popular approach being based on the decomposition of an object into its parts. Therefore, the work cited in this section reflects an emphasis in this area. Siddiqi and Kimia provide an overview of shape partitioning techniques [144] . For a broader perspective of visual perception and recognition, the reader is referred to [11, 23, 91, 94, 100, 173] and the references therein.

- Biederman's theory of human perception and recognition [24] states that objects are represented as an arrangement of simple convex shape primitives called *geons*. Examples of geons include bricks, cylinders, wedges, and cones.
- Koenderink and van Doorn observe that a 3D shape is perceived as a composition of elliptical regions and suggest a decomposition of the shape along parabolic lines [79].
- Hoffman and Richards propose an approach [67] in which the decomposition into parts is not based on the shape of primitives, but rather on the curvature properties of the object boundaries (Figure 2.3).

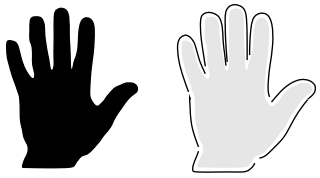


Figure 2.3: Example of Hoffman and Richard's partitioning scheme

- Leyton models parts of objects as historical *processes* [87]; that is, the parts of an objects are the consequences of the object growing and evolving.
- Burbeck *et al.* provide evidence that humans recognize the shape of a figure by linking opposing boundaries in the figure [35, 36]. This is part of the motivation for the medial-based representation known as *cores*. This representation leads naturally to a partitioning of the shape along branch points of the axis.
- Siddiqi *et al.* propose a model for shape perception [144, 146] that is based on a continuum between the *parts* (*e.g.*, Biederman) and *protrusions* (*e.g.*, Leyton) extremes. This continuum is part of the *shape triangle* (Figure 2.4a), whose third node is the *bends* node. Their model is based on *shocks* [78], or entropy-satisfying singularities, formed during the evolution of curves. They present a partitioning scheme that focuses on the parts node of the shape triangle and involves two types of parts, *neck-based* and *limb-based* (Figures 2.4b and 2.4c).

In addition, they propose the use of *shock graphs* [145], which in addition to parts also represent bends and protrusions, for shape matching.

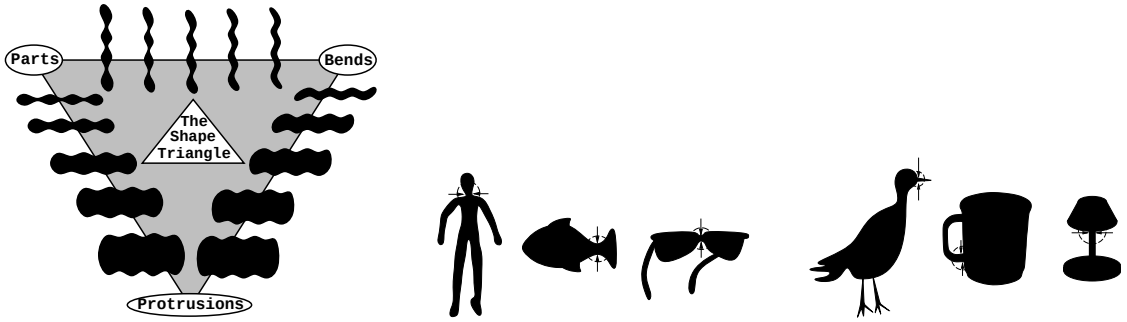


Figure 2.4: (a) Siddiqi and Kimia's Shape Triangle (b) Neck-based parts (c) Limb-based parts

2.2 Shape Similarity Measures

The area of shape similarity measures is very broad and in the case of 2D measures, very well-studied. A significant amount of work has been done in the use of global shape parameters for determining similarity and for matching. Methods using this approach typically describe an object by decomposing the data into a number of feature vectors. The similarity measure or matching algorithm then operates in this feature space. Examples of the types of features used include *moments* (*e.g.*, [39]), *principal components*/*eigenvectors* (*e.g.*, [134]), and *curvature scale space* (*e.g.*, [106]). The main problem with these feature vectors is that they provide the user with little geometric intuition. The similarity measures that we are most interested in are the ones that operate in geometric space. These measures can be roughly divided into two types, *statistical* similarity measures and *geometric* similarity measures. Statistical similarity measures often have little to no basis in human perception, and rely on the large number of samples used to compensate for their sensitivity to outliers. In contrast, geometric similarity measures, which are the focus of our research, usually possess more properties such as stability and flexibility that make them more effective for a smaller number of samples and a wider variety of data. However, the difference between statistical and geometric measures is not always distinct, as there are a number of similarity measures that are used for both types of analysis. For more information on statistical shape analysis, the papers by Bookstein [30], Lorenz and Krahnstöver [93] and Cootes and Taylor [42] are good starting points.

2.2.1 2D Similarity Measures

The vast majority of the research on shape similarity measures has been on 2D point sets, curves and polygons. In this section, we summarize the major issues addressed by other researchers and their approaches for developing similarity measures.

Basri *et al.* provide a survey [19, 20] of methods for determining the similarity between deformable shapes. They describe a number of possibly desirable constraints for similarity measures. The authors observe that several of their constraints are incompatible with each other and cannot all be satisfied at the same time, which points out the difficulty of formulating a similarity measure that works well in all situations. Some of the proposed constraints are designed to help capture the parts-based nature of objects without requiring the explicit decomposition of shapes into parts. Among the other constraints proposed are continuity (*i.e.*, if the shapes change smoothly, so should the measure), invariance (with respect to certain classes of transformations), and metric properties. The focus of their research is on *elastic matching* (*e.g.*, [37, 73]), an approach that computes the similarity of two shapes as the sum of local deformations required to change one shape into the other. They propose several cost functions, using physical models of the object boundaries that incorporate *stretching* and *bending* components, that meet most of the constraints.

An important question that is brought up by the authors is whether shapes are best compared by their interiors or their boundaries. Figure 2.5, copied from [20], shows an example of where a similarity measure using the elastic

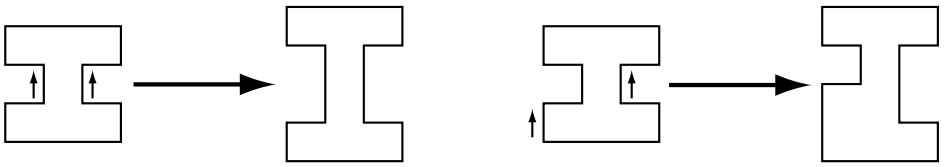


Figure 2.5: Example for which an interior-based similarity measure may be better than a boundary-based one

matching approach may fail. Two different shapes are produced from a single shape by applying the same deformations to different parts of the object's contour. In each case, the boundary is vertically stretched in two places. Because the local distortions required to produce the two shapes are the same, a similarity measure based on local deformations may report them as being equally different from the original shape, even though perceptually the shape on the right is likely more distorted for most viewers. In this case, a measure that compares the interiors of the shapes is likely to have a more accurate result. However, in some cases using the boundary may be a better approach. Figure 2.6, similar to one shown in [20], illustrates this point. The left-most object and the middle one differ very little in terms of local deformations to the interior. There is greater distortion in the interior when comparing the left-most and right-most objects. A similarity measure that compares shapes based on their interiors may report the middle object to be closer in shape to the left-most object, even though most observers would say that the right-most object is closer. A boundary-

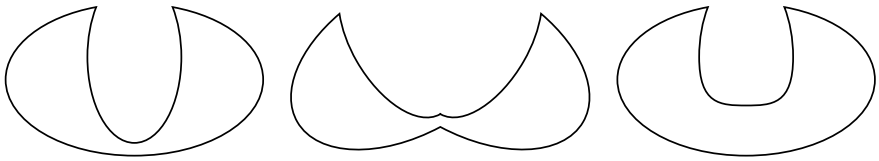


Figure 2.6: Example for which a boundary-based similarity measure may be better than an interior-based one

based measure may be preferable in this case. These two examples can be used to illustrate another important point, which is the fact that local deformations alone are often inadequate for making judgments in shape similarity. Whether we are using a boundary-based or interior-based measure, taking into account global distortions and properties such as symmetry and collinearity could affect the results, and may give answers that are more perceptually intuitive.

Mumford gives a summary and comparison of six very different approaches to defining similarity measures [108]. These approaches are (given that $A, B \subset \mathbf{R}^2$ are the shapes to be compared):

1. Hausdorff metric:

$$d_{\text{Hausdorff}}(A,B) = \sup_{x_1 \in A}\left[\inf_{x_2 \in B} \|x_1 - x_2\|\right] + \sup_{x_2 \in B}\left[\inf_{x_1 \in A} \|x_1 - x_2\|\right]$$

   This commonly-used distance is very sensitive to outlier points.

2. *Template metric* (*area of symmetric difference*) (*e.g.*, [3]):

$$d_{\text{template}}(A,B) = \text{area}(A - B) + \text{area}(B - A)$$

   In contrast to the Hausdorff metric, the template metric is very resistant to outliers.

3. *Transport metric* [54]:

$$d_{\text{transport}}(A,B) = \inf_{\rho} \int_A \int_B \|x_1 - x_2\| \cdot d_\rho(x_1, x_2)$$

   where ρ is a probability measure on $A \times B$ such that

$$\int_A \int_{U_B} d_\rho(x_1, x_2) = \frac{\text{area}(U_B)}{\text{area}(B)}, \quad U_B \subseteq B \text{ and } U_B \not\subset A$$

$$\int_{U_A} \int_B d_\rho(x_1, x_2) = \frac{\text{area}(U_A)}{\text{area}(A)}, \quad U_A \subseteq A \text{ and } U_A \not\subset B$$

   A physical analogy for this metric is to think of shape A as being filled uniformly with a mass and the metric calculates the amount of work needed to move the mass so that it fills B uniformly. The transport metric is regarded as a good compromise between the Hausdorff and template metrics.

4. *Optimal diffeomorphism*:

$$d_{\text{od}}(A,B) = \inf_{\phi}\left[\int_A \|J\phi\|^2 + \int_B \|J(\phi^{-1})\|^2\right]$$

where $\phi : A \to B$ is a $1-1$, onto differentiable map with differentiable inverse ϕ^{-1} and J is the matrix of first derivatives. The map ϕ can be chosen so that the similarity measure models the energy of elastic deformations. By definition, the shapes to be compared must be topologically equivalent, or the measure reports them to be infinitely far apart.

5. *Maps with tears*:

$$d_{mwt}(A, B) = \inf_{S \subset A \times B} \text{area}(S)$$

where S is a surface that represents an invertible map between A and B. S has "tears" where there are mismatches between A and B. These "tears" allow this similarity measure to compare objects of differing topology. When comparing objects of equivalent topology, this measure gives similar results to that of the transport metric.

6. *Graph matching* (*e.g.*, [89]):

$$d_{gm}(A, B) = \text{bpm}(\Gamma_A, \Gamma_B)$$

where Γ_X is a graph attached to the shape X where the nodes represent the parts of X and the edges represent adjacency or inclusion of the parts, and bpm is a measure of the best partial match between the two given graphs. The main problem with this approach is that the parts decomposition process can be very unstable in that small changes can result in a major reorganization of the parts-graph.

Mumford asserts that any successful theory of shape description must include considerations of the features of the boundary as well as the interior of the object. In addition, he emphasizes the importance of multiscale analysis. He states that satisfying these two conditions tends to give the associated measures the ability to handle much more effectively the variations in data caused by noise, changes in perspective, partial occlusion, *etc*.

Veltkamp and Hagedoorn summarize a number of similarity measures for 2D polygons, curves and regions [161, 162]. The emphasis of this work is on techniques from computational geometry. Among the measures discussed are relatively popular ones such as the *L_p-distance* (*e.g.*, $p = 2$ gives the Euclidean distance), *bottleneck distance*, and *Fréchet distance* (*e.g.*, [5]), as well as lesser-known ones such as the *reflection metric* [62]. They rate many of the commonly-used measures on their stability with respect to small deformations, blurring, cracking, and noise. The authors also provide a list of desirable qualities for similarity measures, and give a set of constructions for manipulating measures, such as remapping and normalization methods.

2.2.2 3D Similarity Measures

The amount of research done on 3D similarity measures has been far less than that done for 2D. Some of the 2D measures described above, such as the bottleneck and Hausdorff distances, extend naturally to 3D, while others, such as ones based on the turning function, cannot be extended directly because of the lack of an appropriate parameterization. In this section, we describe a number of notable methods designed to work in 3D.

- Some researchers use a voxel-based approach to defining a similarity measure in order to avoiding the parameterization problem. For example, Bribiesca propose a measure [32] that can be seen as a 3D discretized version of the transport metric.

- Methods for matching deformable models typically consist of the extraction of characteristic contours or curves, followed by the application of an algorithm (*e.g.*, elastic matching) to find corresponding pairs of the extracted parts. This is the most popular approach for the shape matching of volumetric data in medical image analysis. For example, Bajcsy and Kovačič use a multiscale technique [18] for brain scan registration.

- Some measures are based on skeletal models. For example, Pizer *et al.* use their figural shape representation to determine shape similarity [122] by first matching the figures between objects, then summing the differences in local shape parameter values in the matched figures. They propose this method for use in template-based segmentation and non-rigid registration.

- Ohbuchi *et al.* propose a number of different techniques [115, 116, 117] for determining the shape similarity of 3D "polygonal soup" models. One such approach utilizes distances of the object's surface from the principal axes to form a feature vector, while another uses alpha shapes [47] to derive a multiresolution shape descriptor.

- A number of researchers use *topology matching* to quantify shape differences. The main idea of this approach is to partition each object in a way such that a graph of the parts can be built to represent the topology of the object. The nodes of the graphs of the objects to be compared are then matched to estimate the similarity in shape. Examples of algorithms that use topology matching include that by Hilaga *et al.* [66] and Bespalov *et al.* [22].

- Some methods are designed to facilitate fast similarity estimation, for tasks such as 3D database queries. An abstracted *shape descriptor* in the form of a function defined on a canonical domain is often used to facilitate the comparison of various aspects of shape. For example, a number of approaches use spherical functions (*e.g.*, *directional histogram* [90], curvature distribution [143, 171]). Kazhdan *et al.* [76] use *spherical harmonics* to transform rotationally dependent spherical shape descriptors into rotationally invariant ones.

- Osada *et al.* [119] represent the signature of a polygonal object as a *shape distribution* sampled from a *shape function* measuring global geometric properties of an object. The shape difference between two objects is then computed via a comparison of their shape distributions. Examples of shape functions include the distance between two random points on the surface, the angle between three random points on the surface, and the area of the triangle formed by three random points on the surface.

2.2.3 Matching Methods

The distance functions described above measure the shape difference between two objects, but some of them do not solve the *correspondence* problem (*i.e.*, provide a matching between parts). There is a wide range of methods for matching, including some well-known approaches such as graph building, *geometric hashing* [167], and the *alignment method* [72]. For surveys on matching techniques, the reader is referred to the papers by Alt and Guibas [4] and Veltkamp and Hagedoorn [161].

2.3 Application-Specific Research

This section gives an overview of the work done in shape modelling for several popular applications in graphics, visualization and vision. For certain applications, such as segmentation, object recognition, and registration, only survey papers are cited because a thorough summary is out of the scope of this dissertation. As mentioned previously, for the five applications that we have developed for this thesis, we provide a more detailed overview of the related work in the appropriate chapters.

2.3.1 Computer Graphics and Visualization

Object Metamorphosis

- Examples of work on 2D shape blending of polygons include that by Sederberg *et al.* [135, 136].
- Papers on blending of 3D surfaces include that by Kent *et al.* [77] (polyhedral objects) and DeCarlo and Metaxas [43] (deformable models).
- Alt and Guibas summarize a number of techniques [4] for shape matching and interpolation from computer graphics and computational geometry.
- Lazarus and Verroust survey the major techniques [81] for 3D shape metamorphosis used in computer graphics. The approaches discussed are categorized based on the object representation (volume-based, boundary-based, and elevation maps).

- Treece *et al.* propose a method [159] to perform voxel-based object metamorphosis using spheres to compute the correspondences between regions.

Image Matching and Interpolation

- Shape interpolation is sometimes utilized by image morphing techniques. Wolberg provides a survey [164, 165] of image morphing methods. Practically all of the techniques discussed require explicit manual feature matching. A more detailed summary of image interpolation techniques is given in Section 4.3.
- Feature matching and image morphing are often used for panorama *stitching* in image-based rendering applications (*e.g.*, [34, 105, 152]).

Other Work

- Pizer *et al.* [125] use figural shape to represent objects for object design, rendering and physically based operations.
- Storti *et al.* discuss the use of skeleton-based modelling operations on solids [149], such as level-of-detail control, shape interpolation and shape synthesis.
- Free-form deformations are a popular group of methods for the manipulation of object geometry. Early examples include work by Sederberg and Parry [137] and Hsu *et al.* [69].
- There are a number of papers, such as those by Frisken *et al.* [53] and Adams and Dutré [2], that present models that enable efficient operations such as inside/outside and proximity tests, boolean operations, blending, *etc.* The range of operations supported is typically narrow, and most of these models are not designed to support shape matching.
- Hart surveys the methods [63] for controlling connectedness for shape modelling when using recurrent (fractal) models and implicit surfaces.

2.3.2 Computer Vision

Segmentation

For a broad survey of image segmentation techniques, the reader is referred to the papers by Acharya and Menon[1] and Pham *et al.* [121]. Although the focus of these papers is on biomedical images, most of the methods discussed are generally applicable. Segmentation techniques range from very simple and general techniques, such as thresholding, to very complex and domain-specific methods, such as expert-systems. Deformable models are the most popular type of

shape model used for segmentation. Xu *et al.* have written a tutorial/review [170] on the use of deformable models for medical image segmentation. The use of shape models for segmentation purposes is discussed further in Section 5.2.

Object Recognition

Shape models for recognition are often based on theories of human perception. The general consensus seems to be that parts-based representations are the most appropriate for matching during recognition, especially in the face of occlusion, deletion of portions of objects, *etc*.

- The survey paper on computational strategies for object recognition by Suetens *et al.* [150] is well-written but somewhat outdated now. The methods discussed are divided into four classes: simple strategies that use feature vector classification, methods that work on reliable data and complex models, methods that work on noisy data and simple models, and combinations of these strategies.

- Ponce *et al.* have a paper [126] discussing the role of shape models in object recognition. This paper focuses on the use of *generalized cylinders* and *evolving surfaces*, which is an approach that analyzes the singularities of a surface as it evolves under some family of deformations.

- Dickinson's survey [85] discusses several key components of an object recognition system (object representation, feature extraction, database organization, model indexing). He provides a comparison of several major object representations (*e.g.*, points, contours, surfaces, deformable models, superquadrics, geons and generalized cylinders), focusing on how their properties such as primitive complexity affect the recognition process.

- Forsyth *et al.* examine the role of primitives and perceptual organization in object recognition [50]. They discuss issues in object recognition such as alignment theories, viewpoint dependence, volumetric primitives for structural representation, and 2D versus 3D representations.

- Campbell and Flynn survey the major representations and techniques [38] used for the recognition of free-form objects. They review the construction methods and the strengths and weaknesses of parametric and implicit surfaces, superquadrics, generalized cylinders, and polygonal meshes. They also discuss various *appearance-based* recognition systems, as well as techniques that match objects in range data using geometric features.

Registration

The most exhaustive surveys on image registration are by Brown [33] and Maintz and Viergever [97]. Maintz and Viergever focus on medical image data, and mainly discuss papers published after 1993. Brown's paper summarizes publications before 1992, and is more general. Brown views registration methods as different combinations of choices

of four components: a feature space, a search space, a search strategy, and a similarity measure, and she groups registration methods into four categories: correlation and sequential methods, Fourier methods, point mapping, and elastic model-based matching. The methods discussed by Maintz and Viergever can be classified as follows:

1. Landmark-based
    i. Anatomical
    ii. Geometrical
2. Segmentation-based
    i. Rigid models
    ii. Deformable models
3. Voxel property-based
    i. Reduction to scalars/vectors (moments, principal axes)
    ii. Using full image content

Because we are focusing on the use of shape models, the approaches that are the most closely related to our work are segmentation-based methods using non-rigid models.

Chapter 3

Computation and Approximation Properties of Voronoi Ball Models

In this chapter, we explain how Voronoi ball models are computed. We discuss the primary methods of construction used in this thesis and their approximation properties. Many of our algorithms used are grounded in theory from computational geometry. As evident in later chapters, the result of choosing a geometric approach is that many of our methods are combinatorial in nature and provably correct, and avoid the problems in precision that plague many numerically-based methods.

3.1 Computation

VBM computation consists of three basic steps, and starts with a set of sample points on the boundary of an object. The first step is to compute the Delaunay tessellation (defined in Section 3.1.1) of the point set. The second step is to compute the circumscribing ball of each tetrahedron (triangle in 2D). The last step is to discard all balls that are "outside" of the object, which requires an inside/outside test. The remaining balls form the VBM. All of our algorithms employ this basic procedure for computing VBMs; the only significant differences are in the inside/outside test. Figure 3.1 illustrates the process in 2D; a VDM model is computed from the boundary points outlining the shape of the province of British Columbia. In this case, a disk is considered inside the object if the corresponding Delaunay triangle is inside.

We assume that the boundary points of a given object are readily available or easily computed. For example, laser scanners can produce dense points that can be used directly. Other common sources of data include surface data (*e.g.*, polygonal meshes), volumetric data (*e.g.*, CT or MRI scans), and 2D image data. For polygonal meshes,

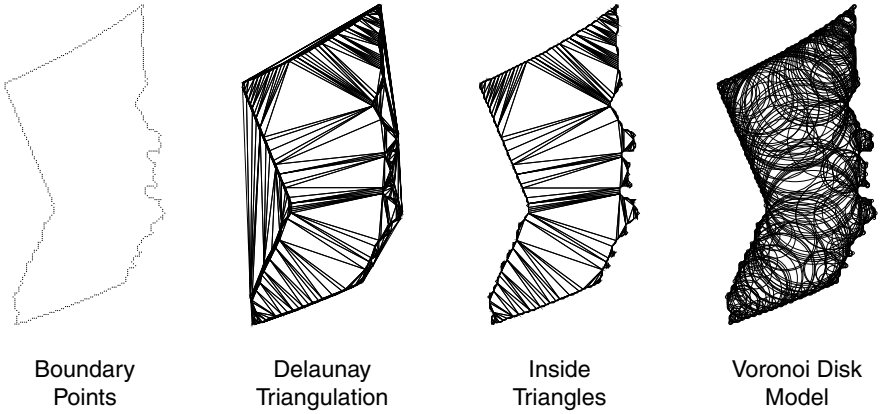


Figure 3.1: Computation of a Voronoi Disk Model

the vertices of the mesh can be used, with additional points computed by subdivision if necessary. For other types of surface data, such as those composed of curved patches, points can be generated by ray-casting the model from different directions. For volumetric data, boundary points are typically computed by interpolation between the inside and outside voxels. The two types of data primarily used in this thesis are volumetric and laser scan data.

3.1.1 Voronoi Diagram

Let $P = \{p_1, \ldots, p_k\}$ be a finite set of distinct points in $\mathcal{R}^n$, with the assumption that no four points are cospherical.[1] Let $\mathbf{x_1}, \ldots, \mathbf{x_k}$ be the location vectors of the points in P. The region defined by

$$V(p_i) = \{\mathbf{x} | d(\mathbf{x}, \mathbf{x_i}) \leq d(\mathbf{x}, \mathbf{x_j}), \forall\, j \neq i\}$$

where d is the Euclidean distance function, is called the *Voronoi region* of the point p_i. Simply put, the Voronoi region of p_i is the set of all points in R^n that are closer to p_i than another other point in P. Some researchers choose to define Voronoi regions as open sets; we choose to define them as closed. The union of the Voronoi regions

$$\mathcal{V}(P) = \bigcup_{i=1}^{k} V(p_i)$$

is called the Voronoi diagram of the point set P. Figure 3.2a shows the 2D Voronoi diagram of a set of 12 points. A good overview of Voronoi diagrams and their application in many areas can be found in the work by Okabe *et al.* [118].

The *Delaunay tessellation* $\mathcal{D}(P)$ is defined to be the straight line dual of the Voronoi diagram. A *Delaunay edge* exists between two points p_i and p_j if and only if $V(p_i)$ and $V(p_j)$ share an edge in the Voronoi diagram. Figure 3.2b

[1] Where necessary, this constraint can be overcome by numerical or symbolic perturbation of the point locations [44].

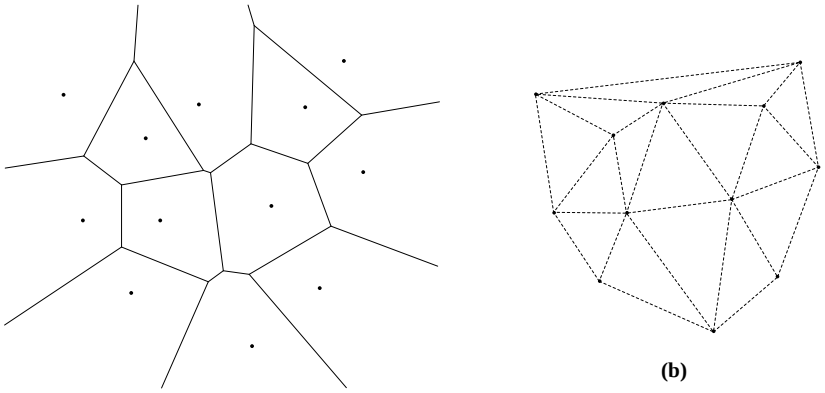


Figure 3.2: (a) Voronoi diagram (b) Delaunay triangulation

shows the dual of the Voronoi diagram in Figure 3.2a. A very useful property of the Delaunay triangulation is the *empty circle property*, which states that the circumscribing circle of a Delaunay triangle does not contain any points of P other than the three vertices of the triangle. This property is also true for the higher dimensions.

Power Diagram

The *power diagram* [17] is a generalized Voronoi diagram that is computed on a set of weighted points. The dual of the power diagram is the *weighted* or *regular* Delaunay triangulation. The weighted Delaunay triangulation is used in the algorithms presented in Chapters 7 and 8, where it is applied to VBMs by using the radii of the balls as weights.

In a power diagram, the *power distance* is used in place of the Euclidean distance. Given a set of n balls $\{b_1, \ldots, b_n\}$ with centres $\{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ and radii $\{r_1, \ldots, r_n\}$, the power distance $\pi(b_i, b_j)$ between any two balls b_i and b_j is defined as

$$\pi(b_i, b_j) = \|\mathbf{x_i} - \mathbf{x_j}\|^2 - {r_i}^2 - {r_j}^2.$$

Two balls b_i and b_j are called *orthogonal* if $\pi(b_i, b_j) = 0$. Figure 3.3 shows a power diagram computed from eight disks.

The (weighted) Delaunay triangulation has sometimes been avoided because of its theoretically high computational cost. The tessellation of a set of n points in 3D can have size and running time of $\Theta(n^2)$ in the worst case. However, in practice, when using surface point data, the size and running time are typically observed to be close to linear, suggesting a very reasonable $O(n)$ behaviour in most cases. We use the Computational Geometry Algorithms Library (CGAL), which implements a version [44] of the randomized incremental algorithm [41] for Delaunay triangulations. This implementation has been documented to have linear behaviour [40].

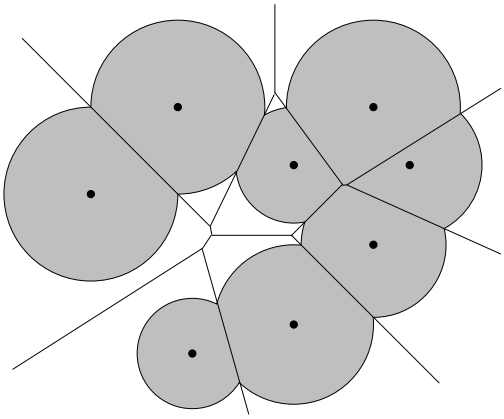


Figure 3.3: Power diagram of eight disks

3.1.2 Ball Classification

This process involves filtering out all of the Voronoi balls that should be considered outside of the object's boundary. In this section, we describe the two major approaches that we use for classifying balls. One is by Ranjan [129], the other by Amenta *et al.* [6, 9].

Unions of Spheres

Ranjan's approach [129] used a different inside/outside test for each type of data. We describe his ball labelling methods for volumetric data, range data and polygonal meshes:

- For volumetric data, if the computed boundary points are sufficiently dense, then each ball can only contain either inside or outside voxels, because by the empty sphere property of the Delaunay triangulation no ball can contain any boundary points in its interior. Therefore, a simple inside/outside test can be performed whereby the ball receives the same label as any one of its voxels (typically the voxel nearest the centre of the ball is used). We employ this method for producing the VBMs used in the image interpolation algorithm discussed in Chapter 4.
- For range/laser-scanned data, the use of visibility information is required to classify the balls. We do not use this procedure in our work (we use the polar ball method, described in the next section), but we include it here for completeness. For each ball, the four points used to define it are processed in the following manner:
  1. The scan direction for each point is found.
  2. The vector from the centre of the ball to each point is computed. This is an estimated surface normal.
  3. For each point, take the dot product of the scan direction with the surface normal. If any of the four dot products are positive, the ball is classified as outside.

This process eliminates most of the outside balls, but occasionally a ray-casting procedure is used in a followup phase to check that any point on the visible surface of the model should be within a certain distance of a sampled boundary point.

- For polygonal meshes, the test is similar to that for range data. The same three steps listed above are followed, except that instead of the scan direction, the surface normals of the polygons at the four sample points used to define the ball are used to provide the visibility information. The surface normals are assumed to be oriented consistently.

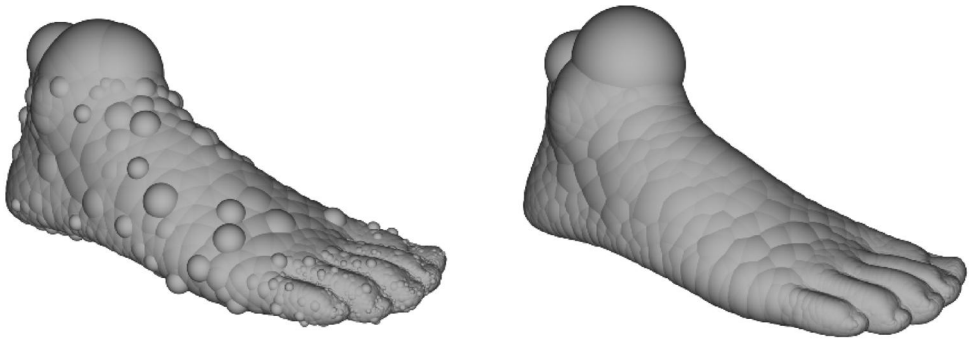
Polar Balls

Amenta *et al.* [6, 9] have developed a robust method for computing a VBM, using the *poles* of the sample points. The poles of a boundary sample s are defined as the farthest Voronoi vertex from s in the interior of the object and the farthest Voronoi vertex from s in the exterior of the object. Thus, each sample normally has an *inner pole* and an *outer pole*. In 2D, all Voronoi vertices are poles, but not in 3D. The balls centred at the poles are called *polar balls*. To find the poles, the Voronoi vertex of s furthest from s is selected as the first pole p_1; the second pole is selected from the remaining vertices v such that $\angle v s p_1 > \frac{\pi}{2}$. If the orientation of the object's surface is known, it can be used to determine which pole is inside or outside. If the orientation is unknown (*e.g.*, only the sample points are given), then they use the following procedure:

1. To avoid dealing with infinity, they add four points to the sample set. The added points form a large bounding box around the object, and result in polar balls that are known to be outside the object.
2. According to the theory by Amenta and Bern [6], given sufficiently dense sampling the interior and exterior polar balls should intersect only very shallowly, if at all. This condition is defined more precisely in Section 3.2.2. Therefore, the balls can be labelled by starting with the known exterior balls and their interior counterparts, then propagating the labels appropriately to their neighbours.

This method of ball labelling is typically more robust and efficient than most of the inside/outside tests mentioned above. In addition, it has been shown that the interior polar balls actually give a provably better approximation of the object's boundary than the full set of interior Voronoi balls computed by most other methods [9]. Figure 3.4 illustrates this difference. The VBM in Figure 3.4a uses the entire set of interior balls, and shows the typical artifacts of such a construction; the "warts" on the foot are the result of discretization, and would appear at *any* finite sampling density. Figure 3.4b, computed from the same data set but composed only of polar balls, has a much smoother appearance, many fewer balls, and is likely to be more usable for many shape-driven applications.

In addition to giving a better representation of the object's boundary, the polar balls of an object also allow for the more accurate approximation of the medial axis of the object. Given sufficient sampling, the α-shape [47] and



(a) **(b)**

Figure 3.4: (a) Voronoi balls (10,903) (b) Voronoi polar balls (2,931) (images from [9])

the medial axis of the interior polar balls have both been proven to be homotopy equivalent to the original object [9]. In fact, the centres of the balls converge to the true medial axis of the object as the sampling density increases [8]. A number of previous attempts at using the Voronoi diagram to approximate the medial axis rely on the incorrect claim [59] that convergence occurs even if the full set of interior balls is used. Because of its robustness, efficiency and convergence properties, we use the polar ball approach in our algorithm discussed in Chapter 7.

3.2 Approximation Properties

It has been shown by a number of researchers that the approximation errors of VBMs are bounded, given certain conditions on the sampling density of the boundary points with respect to the complexity of the shape. The types of data used for our work allow us to ensure quite easily that the models are adequately sampled. We provide the error bounds here in order to show that using VDMs/VBMs in our algorithms does not introduce an undue amount of error, and to establish a foundation for future work in the detection and handling of undersampling.

For the following discussion, we use this notation: $\mathcal{W}$ is the object being approximated, W is the boundary of $\mathcal{W}$, $S = \{s_i, i = 1, \ldots, n\}$ is the set of n sample points on W used to construct the VBM, and ϵ is the approximation error, defined to be the maximum distance between any point on the boundary of the VBM and its closest point on W.

3.2.1 Union of Circles/Spheres

The concept of the r-*regular* object, introduced in mathematical morphology [138], is utilized in Ranjan's work as a way to characterize the complexity of the object's shape. An object is said to be r-regular if it is morphologically open and closed with respect to a disk of radius $r > 0$. Simply put, this means that the curvature of the object's boundary

cannot be greater than $\frac{1}{r}$ anywhere, and the object cannot have a narrowing smaller than $2r$. Figure 3.5 shows an example of an r-regular object.

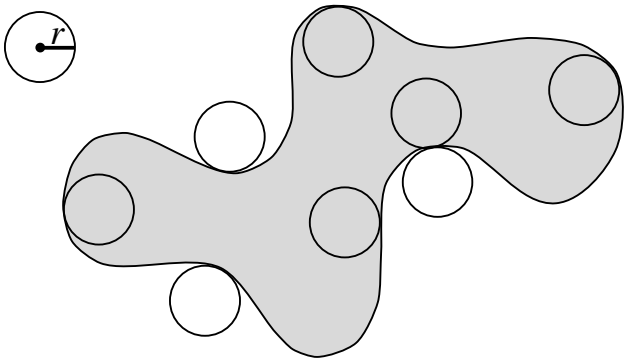


Figure 3.5: An r-regular object

For the 2D case, the following result has been proven for VDMs. Using δ to denote the sampling density of S (defined to be the maximum distance between sample points):

Theorem 3.1 (Ranjan [129]) *If the underlying object being approximated is r-regular, and the sampling density condition $\delta < r$ holds, then the error ϵ in the VDM is less than or equal to δ.*

Clearly, as the shape complexity increases (r gets smaller), δ must get smaller in order to maintain the same amount of error. For 3D, the following gives the error bound for a VBM computed from volumetric data:

Theorem 3.2 (Ranjan [129]) *Given an r-regular object $\mathcal{W}$, let there be a volumetric sampling of $\mathcal{W}$ on a regular grid of cell size ($c \times c \times c$), where each voxel is classified as "inside" or "outside". Let the boundary of the object be sampled by computing points between all neighbouring (in the six-connected sense) inside and outside pairs of voxels. If $c <= \frac{2\sqrt{3}}{3} r$, then the VBM approximation error ϵ is less than or equal to $\sqrt{3}c$.*

Our algorithms presented in Chapters 4 and 5 use the method described in Section 4.4.1 to simplify VBMs. The error introduced by this simplification process is also bounded:

Theorem 3.3 (Ranjan [129]) *Given an object $\mathcal{W}$, let $\mathcal{V}$ be a VBM representing $\mathcal{W}$ with the approximation error ϵ. Let $\mathcal{V}'$ be the simplified VBM computed by clustering $\mathcal{V}$ with a sphericity of σ. The error bound (maximum distance between the boundaries of the original and simplified models) for $\mathcal{V}'$ is $\epsilon + 2r_l\frac{1-\sigma}{\sigma}$, where r_l is the radius of the largest ball in $\mathcal{V}$.*

3.2.2 Polar Balls

This section summarizes the work by Amenta and Bern [6] and Amenta and Kolluri [9], who compute the error bounds for VBMs composed of polar balls. Their approach is based on proving that as sampling density increases, the interior polar balls become increasingly more accurate approximations of the maximal balls of the medial axis transform. With infinite sampling, the boundary of the medial balls form the exact boundary the object. Before the error bound is given, some definitions are needed:

Definition: The *Local Feature Size* at a point $w \in W$ ($LFS(w)$) is the distance from w to the nearest point p_w of the medial axis of W. The $LFS(w)$ can also be thought of as the radius of the medial ball centred at p_w and touching w. The *LFS* is a local measure of the "level of detail". The *LFS* decreases when the curvature of the boundary is high or when two patches of the surface are close together. Amenta and Bern then define their measure of sampling density r as a function of the *LFS*:

Definition: The set of points $S \subseteq W$ is called an r-*sample* if the distance from any point $x \in W$ to its closest sample in S is at most a constant fraction r times $LFS(x)$.

Alternatively stated, as r gets smaller, the distance between any point on the boundary of the object and its closest sample point also gets smaller, relative to the *LFS*. For this to be true for all of the points on W, the sampling density must increase. The dependence on the *LFS* is Amenta and Bern's way of specifying the sampling constraint as a function of the local shape complexity. As the *LFS* decreases, the sampling should be increased. This use of the local shape is in contrast to Ranjan's approach, which imposes a lower limit on the sampling density for the entire surface.

For the following two theorems, it is assumed that the inner and outer polar balls are computed from an r-sample S, with $r \leq \frac{1}{10}$.

Theorem 3.4 (Amenta and Kolluri [9]) *Let U_i and U_o be the boundaries of the unions of the inner and outer polar balls, respectively. The distance ϵ from a point $u \in U_i$ or $u \in U_o$ to its closest point x on the surface W is $O(r)LFS(x)$.*

As mentioned in Section 3.1.2, given sufficient sampling (*i.e.*, S is an r-sample), the inner and outer polar balls only intersect shallowly, if at all. This condition is defined more precisely by the following:

Theorem 3.5 (Amenta *et al.* [8]) *Let B_I be an inside polar ball and B_O be an outside polar ball. If B_I and B_O intersect, and α is the angle of intersection, then $\alpha = O(r)$.*

Chapter 4

Image Matching, Interpolation and Rigid Registration

4.1 Motivation

Shape matching deserves special consideration in the design of a multipurpose shape model, because it is the core process of many shape-driven applications. The shape model used directly affects the type of similarity measure that can be applied. For example, landmark-based representations frequently have similarity measures, such as the Hausdorff distance (*e.g.*, [71]), that are very sensitive to outliers. Because the similarity measure computes the shape distance between objects or parts of objects, it determines the matches and is therefore critical to the performance of the matching algorithm.

This chapter describes an algorithm that we have developed to test the capabilities of VBMs for solving the 3D shape correspondence problem. This algorithm, originally presented with preliminary results in [153], uses Ranjan and Fournier's similarity measure to determine the feature matches in order to perform image interpolation (*i.e.*, "morphing") and rigid registration. A 3D representation of each image is created by constructing a height field from the pixel values. We use image interpolation because it is a popular application that has proved challenging for many methods. In addition, mismatches are usually easily noticed in the images, unlike in VBM model space, where correspondence errors can sometimes be hidden, especially where there is a large number of primitives. Through this experiment we qualitatively assess the similarity measure's ability to determine accurate and stable correspondences between two given shapes.

Our work extends that by Ranjan and Fournier in a number of significant ways. Their work focused largely on using 2D models for shape matching; although a number of simple 3D experiments were done, the test cases used made

mismatches difficult to identify. We use 3D models to perform matching, without Ranjan and Fournier's prealignment step discussed in Section 4.4.2. In addition, our data is considerably more challenging, especially considering that image matching is not a purely shape-driven application. We also examine the degree of control the user has over the matching process via the selection of parameter values.

4.1.1 VBM Properties

A number of properties make the use of VBMs in shape matching and interpolation advantageous as compared to other representations, and we exploit these properties in our image interpolation algorithm. Ranjan showed that VBM models are measurably stable with respect to small changes in the input data when using their similarity measure [129]. Figure 4.1 shows an example of how a VBM changes as its boundary points are perturbed. In this case, the model is computed from 15,277 boundary points. The points are then randomly perturbed by amounts ranging from 0.1% to 1.0% of the shortest side of the bounding box. For each point set, the VBM is computed (resulting in about 40,000 balls each), then simplified to about 100 balls. The similarity measure is then applied to determine the amount of shape change resulting from the induced noise.

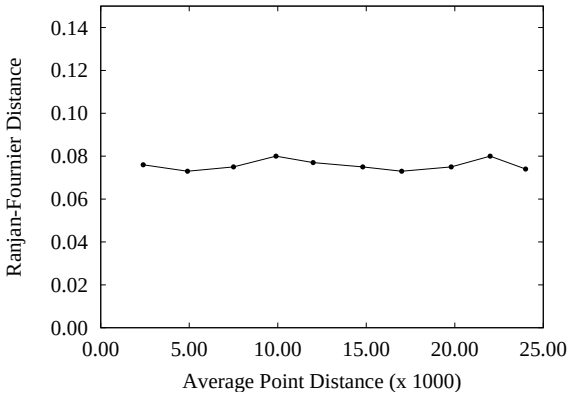


Figure 4.1: Example of VBM stability (data taken from [129])

In addition to random perturbation of the boundary points, the VBM representation is also measurably stable with respect to small changes caused by shape-preserving transformations, and distortions such as shearing. Even though no representation is immune to very large changes (features can actually appear or disappear as a result of such changes), in most of the cases that we have studied the VBM representation allows for the reliable and intuitive prediction of how the model will change in response. Having determined that VBMs are generally stable with the given measure, we focus on testing the accuracy of the correspondences and their stability with respect to the parameters used in the similarity measure.

In addition to stability, Ranjan noted several other advantages to using VBMs for shape interpolation [129]:

- VBMs can be simplified, so the interpolation can be made at various levels of detail.
- Using a union of closed primitives eliminates the problems of tearing and self-intersection associated with surface-based methods.
- The number of primitives in the intermediate shapes are on the same order as the original models.
- The intermediate shapes are not wildly distorted compared to the original objects.
- The similarity measure allows some control over the interpolation.

4.2 Background

The general objective of 2D image interpolation is to derive a number of intermediate images between a beginning image and an end image. There are numerous applications for image interpolation, ranging from aesthetic purposes (*e.g.*, morphing) to scientific visualization (*e.g.*, 3D volumetric reconstruction from 2D slices). Although the requirements for an interpolation algorithm vary somewhat with the application, there are a number of elements considered desirable in most cases. The interpolation of images requires a matching of features that can be humanly identified as having a certain degree of similarity. Most people would consider the shapes of objects to be an important criterion for feature matching. Therefore, in order for an algorithm to generate "good" interpolations, it should take into account similarities in shape. Partly because of their importance to image interpolation, the representation and matching of shapes have been the focal points of intense study. Many successful algorithms have been developed to interpolate between objects, but many of these methods rely on the user to specify the features and often the correspondences between them. Manual specification of features can be a labour-intensive task, especially for complex images. On the other hand, there are algorithms that attempt to automatically extract all of the matchable features from images. Some commonly used features include points, edges, corners, and skeletons. A common problem with feature extraction methods is instability with respect to changes in the input data. For example, rescaling the intensities in a greyscale image can cause an edge detection algorithm to output a different set of edges. Ideally, an algorithm should require little or no user assistance in forming a representation of the image features, but should have enough stability to be able to handle reasonably large variations in input data. We present an approach to image interpolation that focuses on stability and accurate matching. The main idea is to use VBMs to represent and match image features.

4.3 Related Work

There is a large amount of literature that is related to image matching and interpolation, so only a summary of the major techniques is given here. Although some of approaches mentioned are not, strictly speaking, shape-based, they are included for completeness. A good starting point for information on image morphing techniques are the surveys by Wolberg [164, 165]; most of the techniques he discusses require manual feature matching.

The most popular methods of image interpolation use specialized matching primitives drawn by the user, such as points [148], line segments [21, 84], or curves [83]. In some cases (*e.g.*, [83]), energy-minimizing splines are used to assist the user in marking image features. After the user specifies the matches, a *warp function* is generated to interpolate the positions of the marked features. Such functions typically operate on a control lattice such as a thin plate spline (*e.g.*, [64]). In general, these algorithms are effective when dealing with object features that are well-defined and relatively straightforward to specify interactively. The facial images in Fig. 4.6a are examples of images with such features. The CT images in Fig. 4.6b are examples of images for which the interactive specification of features can be problematic and labour-intensive, because some of the objects are very small and have fuzzy boundaries.

A number of the more recent methods in image interpolation, including ours, aim to minimize the amount of user input. Many such methods borrow ideas from related fields in computer vision, such as image registration [33, 97]. An example of such an approach is by Gao and Sederberg [58]. Since the original publication of our algorithm [153], a number of powerful techniques for image matching have been proposed. Most recently, Lowe [95] presents a method for extracting highly distinctive features from images for use in matching and recognition. These features are invariant to image scale and rotation, and have been shown to provide robust matching.

4.4 Ranjan and Fournier's Matching Algorithm

In this section, we summarize Ranjan and Fournier's method for shape matching using VBMs. The central idea is to represent each object by a VBM, and apply the similarity measure to form correspondences between balls in the two VBMs being matched. The similarity measure has a number of parameters to give the user some control of the matching process. A simplification process that reduces redundancy is usually applied before the matching in order to increase stability. More details on the basic approach can be found in [129, 130].

4.4.1 Simplification Algorithm

The simplification algorithm is aimed at reducing the number of primitives while preserving the shape features as much as possible. The algorithm, called *clustering*, works by replacing groups of balls within the VBM with larger,

encompassing balls. The degree of simplification is controlled by a user-set parameter called *sphericity*, which is a measure of how well a set of balls can be modelled by a single ball. Mathematically, the sphericity of a cluster of balls is defined as the ratio of the radius of the largest ball in the cluster to the radius of the smallest ball containing all balls in the cluster. Fig. 4.2 illustrates how sphericity is defined for disks; the extension to balls is trivial.

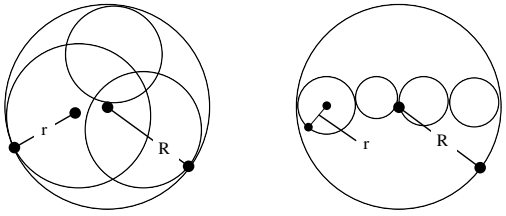


Figure 4.2: Definition of sphericity ($\frac{r}{R}$) (a) Large sphericity (b) Low sphericity

The clustering algorithm processes the balls in order of decreasing size. In each iteration, the algorithm takes the largest unprocessed ball **a**, and calculates the smallest ball encompassing **a** and as many other unprocessed balls as possible, under the constraint that the cluster must have a sphericity greater than or equal to the user-chosen threshold. The cluster is then replaced by the newly formed encompassing ball. As discussed in Section 3.2.1, the distance between the surface of a simplified VBM and the original point set is bounded; therefore, clustering is guaranteed not to distort the original shape features beyond what is expected at a given sphericity.

Figure 4.3 shows a 2D example of clustering. The original VDM has 425 disks. There is a significant amount of redundancy caused by the density of the boundary points used for calculating the VDM. In contrast, the simplified VDM, the result of clustering with a sphericity of 0.95, only has 112 disks, and is a much more efficient representation. It is worth noting that small disks are still present where necessary to preserve shape, such as in the corners. In areas of relatively low detail, such as in the centre of the object, the number of disks is greatly reduced.

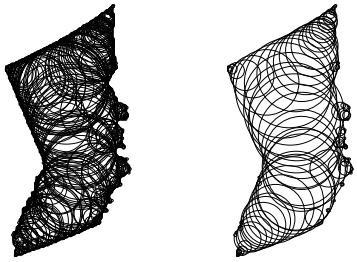


Figure 4.3: Example of clustering (sphericity = 0.95)

4.4.2 Model Alignment

In order to satisfy the constraint that the results of shape matching be independent of translations, rotations and uniform scaling, Ranjan and Fournier's method requires an alignment process before applying their similarity measure, which itself is not independent of the given transformations. The models to be compared are translated so that their centres of masses coincide and scaled such that their volumes or linear dimensions match. The models are then rotated so that their local axes established from second order *moments* [127] are aligned. This process is useful for simple objects, and can be done automatically. For more complex objects, manual alignment is usually necessary.

4.4.3 Similarity Measure

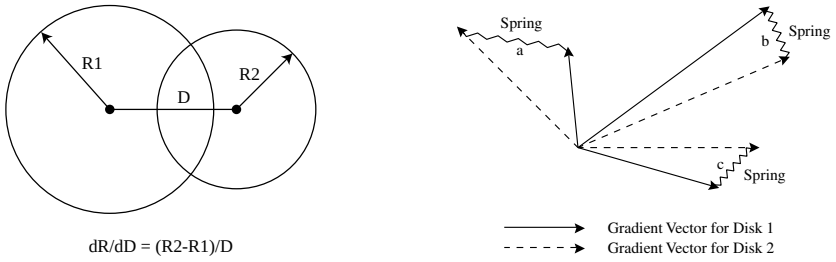
In this section, we give the definition of the similarity measure. The shape difference between two VBMs is computed by matching balls between the two models using a specially defined distance measure. The shape distance between the two VBMs is taken to be the average of the distances between all matched pairs of balls. This method not only gives a quantitative shape difference between two VBMs; it also determines a correspondence between the primitives in the two models.

The first step in the matching process is the calculation of the distances $d(a,b)$ between every a and b, where a is a ball in the first VBM, and b is a ball in the second. Given that a has centre (x_a, y_a) and radius r_a and b has centre (x_b, y_b) and radius r_b, the distance function is given by:

$$d(a,b) = w_p d_p(a,b) + w_s d_s(a,b) + w_f d_f(a,b) \tag{4.1}$$

where $d_p(a,b) = (x_a - x_b)^2 + (y_a - y_b)^2$, $d_s(a,b) = (r_a - r_b)^2$ and $d_f(a,b)$ is the *feature* distance between a and b, as described below. The weights w_p (position), w_s (size), and w_f (feature) are chosen by the user, who selects the values based on the application and data at hand. The definition of a feature in this case is a mathematical relationship between a ball and its four largest neighbours (in 2D, the three largest neighbours are used). Four neighbours are used because in an unsimplified VBM, each ball has a maximum of four neighbours (corresponding to the number of neighbouring Delaunay tetrahedra); we use the same number in a clustered model for the sake of consistency. Between the ball and each neighbour, we take the gradient $\frac{dR}{dD}$, where dR is the signed difference between the radius of the ball and the radius of the neighbour, and dD is the Euclidean distance between the centres of the balls. Figure 4.4a illustrates the definition of a gradient vector in 2D. If a ball has less than four neighbours, the value for each missing neighbour is set to $-\infty$, because in this direction the neighbouring ball shrinks to 0 for any distance moved. The $\frac{dR}{dD}$ value is then mapped to the range [0,2], where $-\infty$ is mapped to 0, 0 to 1, and $+\infty$ is mapped to 2. The gradients in the directions of the four largest neighbours form the feature of the ball. The feature distance between two balls can be best explained using a physical analogy. If the two features have a common centre and are free to rotate around it,

and between the ends of each pair of gradient vectors (in the directions of the neighbours) there is a spring that has a pulling force proportional to its length, then the system will be at rest when the potential energy is at a minimum. The sum of the residual distances between the vector ends in this minimum energy state is taken to be the feature distance between the two balls. Figure 4.4b shows a 2D example. Thus, the cost function defines the distance between two balls as a function of the differences in their positions, sizes, and neighbourhood information.



(a) **(b)**

Figure 4.4: (a) Gradient vector from a disk to its neighbour (b) Feature distance (a + b + c) between two disks

4.4.4 Matching Method

After the distances between all balls in the two VBMs have been calculated, a weighted bipartite graph is built where the nodes correspond to the balls, and the weights on the edges are the distances between them. A maximal match is computed such that the sum of the distances between all matched pairs is a minimum. The final shape distance is the average distance calculated over all of the matched pairs in the maximal match.

4.5 Image Interpolation Algorithm

4.5.1 Algorithm Overview

As illustrated in Figure 4.5, the main steps of our algorithm for image interpolation using VBMs are:

1. (Optional) Preprocess each image for input into the interpolation algorithm (*e.g.*, scale pixel intensities, noise removal, *etc*.).
2. Generate a height field for each image from the pixel intensity data and use the resulting point set as the boundary points of a volume.
3. Generate VBMs from the two boundary point sets.

4. Simplify the VBMs by clustering.
5. Match balls between the two simplified VBMs by using the shape similarity measure.
6. Generate intermediate VBMs (one for each intermediate frame in the interpolation).
7. Generate image data from each intermediate VBM.

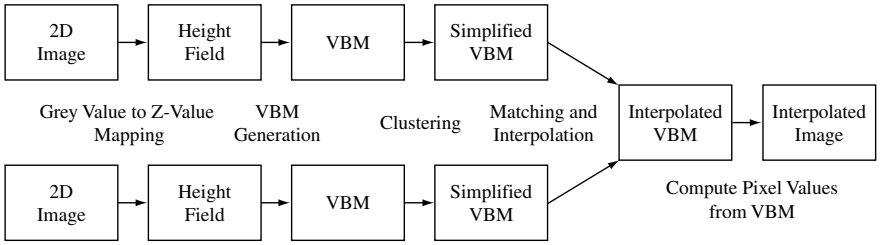


Figure 4.5: Main algorithm steps for image interpolation using VBMs

4.5.2 Test Data

Figure 4.6 shows the images used as test cases for our experiments. For simplicity, we are only dealing with greyscale images. We choose these particular test cases to provide a range of challenging input data for our matching experiments. The data set consists of images from three different imaging modalities.

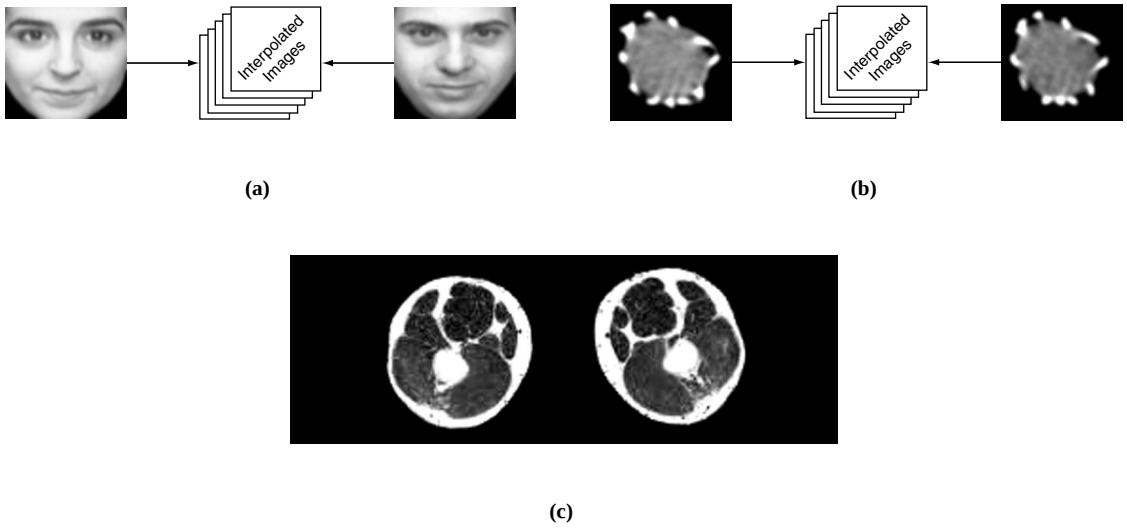


Figure 4.6: Test images (a) Faces (b) CT slices (c) Visible Man legs

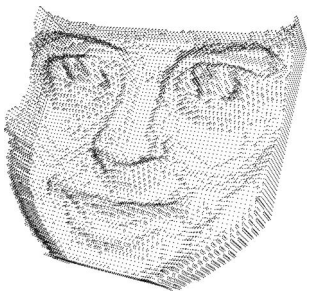
Fig. 4.6a shows two images of faces; the left image (we call this Face 1) will undergo a morph to become the right image (Face 2). We use faces because morphing faces is generally considered a difficult task; even minor

artifacts in the interpolated images are easily noticeable by a human observer. Fig. 4.6b shows two consecutive slices from a computed tomography (CT) data set. Each slice shows a cross-section of a blood vessel surrounded by the wire supports of a stented graft implant. These supports are visible as small, bright white patches around the circumference of the blood vessel. Comparing the second slice to the first, we can see that some of the wires move closer together, while others move farther apart. The shapes of some of the wires also change. Fig. 4.6c shows a cross-sectional view of a man's upper thighs. This image, from the Visible Man[1] data set, is a photograph of a physical cross-section of a cadaver. To test the capabilities of the matching method, we take the mirror image of the left leg and allow the algorithm to automatically register it with the right leg.

4.5.3 Preprocessing

The amount of preprocessing required is dependent on the application and the original image characteristics. For example, in the facial images shown in Figure 4.6a, we notice that most of the important features (*e.g.*, eyes, eyebrows, outline of the nose, mouth, *etc.*) are lower in pixel intensity than the rest of the face. To ensure that these features are well-represented in the VBMs, we invert the pixel values. Figure 4.7a shows Face 1 from Figure 4.6a with the pixel intensities inverted. The height field generated from this image would have peaks where the important features are. In contrast, the CT slices in Figure 4.6b do not need to be preprocessed at all for input into our algorithm. In that case, the most important features to be matched are the graft wire supports that appear as bright white spots, which naturally become peaks in the height field.

(a)



(b)

Figure 4.7: (a) Face 1 with pixel values inverted (b) Boundary points computed from the inverted pixel values

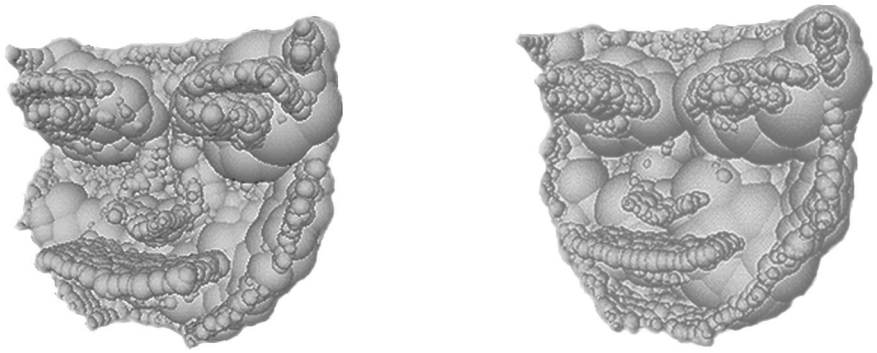
[1] `http://www.nlm.nih.gov/research/visible/visible_human.html`

4.5.4 Height Field Generation

The generation of the height field is relatively straightforward. All that is required is a mapping from the intensity value at each pixel to a z-value in the height field. A simple linear mapping works well in many cases. The points in the height field are then used as the boundary points of a volume. This volume is bounded by the image plane. Figure 4.7b shows the boundary points generated from the Face 1 image. The important facial features mentioned above are visible as peaks in the height field.

4.5.5 VBM Generation and Simplification

After the boundary points are generated, the VBM model can be formed using the method described in Section 3.1. In this case, the voxel closest to the centre of each ball is tested to label the ball as being inside or outside. We then use the clustering process defined in Section 4.4.1 to reduce the number of balls while preserving the features as much as possible. Eliminating excess balls has a strong stabilizing affect on the model and also increases the speed of the matching and visualization processes. Figure 4.8 shows the clustered VBMs of Face 1 and Face 2. The clustering process is able to preserve the smaller details where necessary, such as the areas around the eyes and mouth.



(a) **(b)**

Figure 4.8: VBMs computed from the face images in Figure 4.6a

4.5.6 VBM Matching and Parameter Selection

The next step is to establish correspondences between the two VBMs to be interpolated. Equation 4.1 is employed as the distance measure in the matching process. Determining appropriate values for the parameters w_p, w_s, and w_f is more important in this application than in Ranjan and Fournier's experiments, because in their work the models are

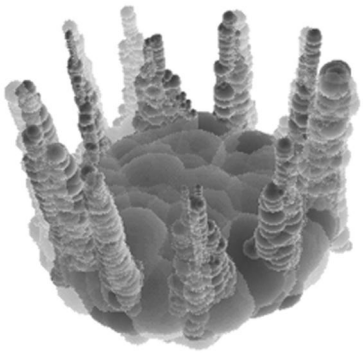


Figure 4.9: VBMs computed from the CT images in Figure 4.6b

prealigned and scaled before the similarity measure is applied. In our case, we are aiming to test the capabilities of the similarity measure without the preprocessing, as well as to estimate the degree of control that the user can have in the matching process.

The user selects the weights by experimentation and visualizing the VBMs. For example, Figure 4.9 shows the VBMs computed from the CT data shown in Figure 4.6b. In this figure, the two VBMs are superimposed, with the transparency of one set to 50% to give the user some idea of how the balls should be matched. In this case, we would like to form correspondences between the peaks, because they represent the centres of the wire supports. For this example, w_p and w_s should be relatively large (0.8 to 1.0), with w_f somewhat lower (0.3 to 0.5), because we can see the balls that should be matched are quite close in position and size. In contrast, getting the desired matches between the two VBMs shown in Figures 4.8a and 4.8b would require a higher feature weight, because the neighbourhood context of the balls is more important for matching in that case. After matching with some initial values for the weights, a number of interpolated VBMs can be formed and visualized, allowing the user to make adjustments if necessary. We find that animating the interpolation is especially useful in helping the user find appropriate values. Normally, suitable values are attained within several iterations.

After all the distances between balls have been calculated, a weighted bipartite graph as described in Section 4.4.4 is built. We use the Cost Scaling Algorithm (CSA) [60] to construct the graph. If the numbers of balls in the two VBMs are not the same, as is usually the case, there will be a number of unmatched balls on one side. These balls can be dealt with in a number of ways. For example, they can simply be matched to their nearest neighbours. In other cases, the number and/or locations of the balls may be such that they do not affect the appearance of the derived image; in such cases the balls can be discarded.

4.5.7 VBM Interpolation

The interpolation step comes next in our algorithm. For each pair of matched balls a and b, a number of intermediate balls are produced. The position and size of each intermediate ball is obtained by linear interpolation from the matched balls. In addition, because the features of a and b have specific orientations, the intermediate ball should be rotated to reflect the change in orientation. The degree of rotation is also linearly interpolated. We constrain the axis of rotation of each ball to be coincident with the z-axis to prevent gaps in the image that may otherwise appear in the image generation phase.

4.5.8 Image Generation

The final step in our image interpolation algorithm is the generation of the images from the interpolated VBMs. The basic idea is to associate a pixel with a particular location on a ball, and track the movement of that pixel as the ball moves, scales, and rotates across frames in the interpolation. This is done by projecting the image onto the upper surface of the VBM. For each pixel in the interpolated image, the algorithm finds the corresponding pixels in the original two images by comparing the locations, sizes, and orientations of the associated balls. The final value for that pixel is linearly interpolated from the two values in the original images. Figure 4.10 shows a simple example of how this process works. Consider the blot in the centre of the interpolated image. The 2D projection of the ball at that location is shown as a dotted circle. The 2D projections of the matched balls are shown in Images 1 and 2. The ball from Image 1 moves to the right and down, gets smaller, and rotates about the z-axis as we move through the frames of the interpolation. The pixel values of the blot in the centre image are interpolated from the blots in Images 1 and 2.

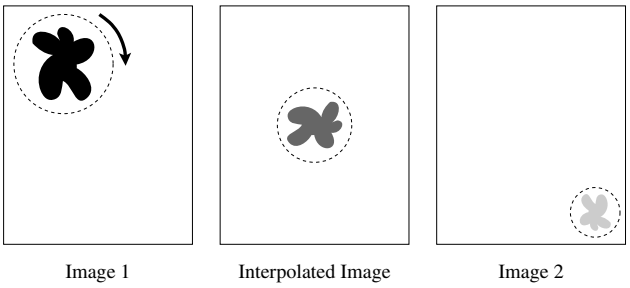


Figure 4.10: Computing an image from an interpolated VBM

4.6 Results

This section summarizes the results of our image interpolation and registration experiments. Table 4.1 gives the parameter values (weights for the similarity measure and the sphericities used for clustering) for the three data sets. To determine appropriate values for the similarity measure, the parameters are adjusted up or down by a set value (0.5) in an ordered sequence while observing the effect. We find that for the cases tested changing any number of the matching weights by ≤ 0.05 does not result in any difference in the visible results, and changing any *one* of the matching weights by ≤ 0.1 from the specified values does not alter the results appreciably. This gives evidence that the matches are stable with respect to changes in the parameter values.

Data	w_p	w_s	w_f	Sphericity
Faces	0.40 ± 0.05	0.50 ± 0.05	0.80 ± 0.05	0.65
Aorta	0.80 ± 0.05	0.90 ± 0.05	0.50 ± 0.05	0.70
Legs	0.80 ± 0.05	0.80 ± 0.05	0.90 ± 0.05	0.70

Table 4.1: Parameter values used for our image matching experiments

Figure 4.11 shows the facial images produced with our interpolation algorithm without any user specification of features. Frame 1 in the sequence is Face 1, and Frame 11 is Face 2. A viewer normally focuses on areas such as the eyes, eyebrows, nose, mouth, and curvature of the face, all of which are reasonably well-interpolated, as shown in the intermediate frames (2-10). For example, the nose gets larger gradually, the eyes get smaller, the eyebrows change shape and move toward the eyes in a smooth manner, and the face gets thinner without getting jagged. This is a good result, especially considering that no user-specification of features is used.

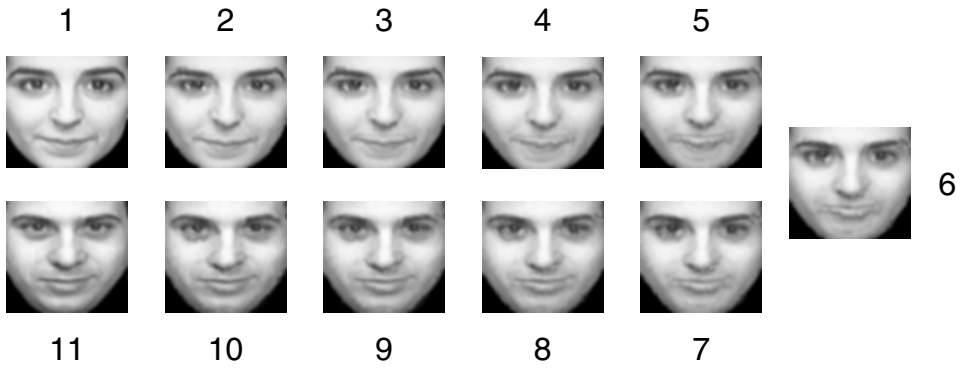


Figure 4.11: Interpolated images from the faces in Figure 4.6a

However, a number of artifacts are visible. The most noticeable problem is that the upper lip of Face 1 gets matched to the area between the lips of Face 2, causing a strange "flipping over" of the upper lip in the interpolated faces. The main reason is that Face 1 has a prominent upper lip but a very small lower lip, and Face 2 has very visible upper and lower lips that define the area between the lips very well. Most automatic methods would have problems with this type of situation, because most algorithms do not know the difference between an upper and lower lip. In our case, the problem can be corrected by a small amount of manual feature specification. The mouth can be forced to match properly simply by increasing the pixel intensities in the area between the lips to highlight this region in both faces. In our case, we use an image editing program to "paint" a white line between the lips in the two original faces. Figure 4.12 shows the interpolation done with this minor modification. The lips are now interpolated nicely.

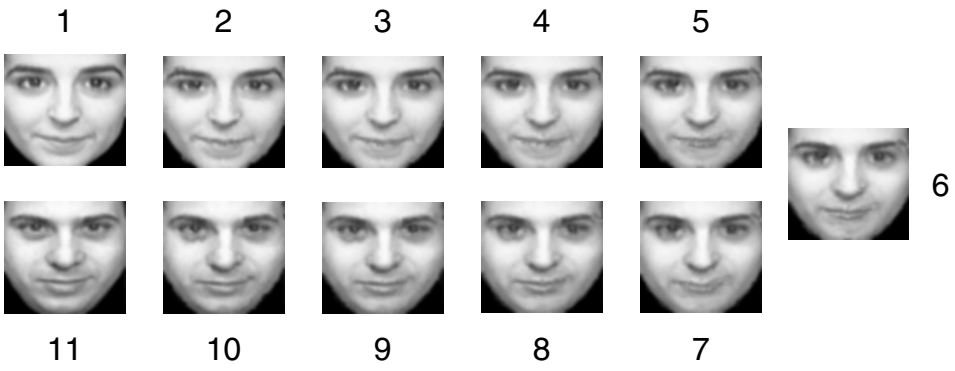


Figure 4.12: Interpolated images from the faces in Figure 4.6a, with manual feature specification for the lips

Another significant artifact in the interpolated frames (especially noticeable in Frame 6) is that some of the finer features, especially the pupils, do not have well-defined boundaries. The main reason for this fuzziness is that each pupil (or other feature) is represented by a group of small balls, and some of the balls representing the pupils in Face 1 are getting matched to balls outside of the pupils in Face 2. This "migration" problem is also documented in Ranjan's work [129]. A potential solution to this problem is a method for partitioning a VBM into parts. In this case, such a capability would be useful for grouping primitives so that distinct features such as the pupils can be treated as separate objects, thereby preventing balls from migrating from one object to another. Ranjan briefly explored VBM partitioning in [129]; we present our experiments with this concept in Chapter 7.

For the CT data, the most important matches are in the graft wire supports that appear as small, bright white patches around the circumference of the aorta in the original images. As can be seen in Figure 4.6b, some of the wires move toward one another, while others move apart. In addition, the aorta and some of the wires change shape between the two images. Figure 4.13 shows the results of applying our algorithm. Frame 1 is the first original slice, Frames 2 to 10 are interpolated images, and Frame 11 is the second original slice. Even with no manual feature specification,

our algorithm effectively interpolates between the two original slices. However, the fuzziness of the features observed in some of the interpolated faces is also present here. In this case, these artifacts are most noticeable in the boundaries of several of the wire supports.

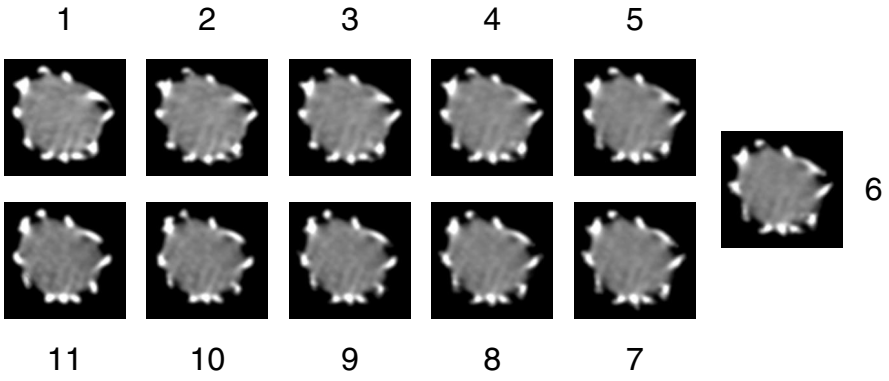
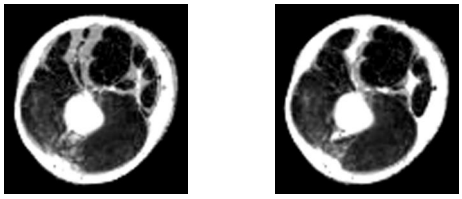


Figure 4.13: Interpolated images from the CT slices in Figure 4.6b

The data for our third test case is the image of the two legs shown in Figure 4.6c. The mirror image of the left leg and the image of the right leg are the input to our algorithm. Our software forms VBM representations of the images, performs the ball matching, then calculates the transformations that should be applied to the left side to register the two images. Figure 4.14a shows the mirror image of the left leg manually superimposed onto the image of the right leg. It is evident that one of the transformations should be a clockwise rotation of the left leg about the z-axis. The transformations computed by our algorithm are a translation and a rotation (6.6°) about the z-axis, which result in the image shown in Figure 4.14b. Qualitatively speaking, the two legs are well-registered, with similar features very close together. As with the CT data, no manual feature specification is required for the matching process.



(a) **(b)**

Figure 4.14: Visible Man legs (a) Unregistered (b) Registered

4.6.1 Processing Time

This section gives an example of the amount of processing time required by our algorithm to create a simplified VBM from an image, perform feature matching between two VBMs, and render a number of interpolated frames. The facial images shown in Figure 4.6a are used for the timing tests. Each image is of size $72 \times 66 \times 16$ (width $\times$ height $\times$ depth) bits. The number of balls in each unsimplified model is about 7,000, and the number of balls in each simplified model is about 700. The final output of the algorithm consists of eleven images, nine of which are interpolated. The computer used is a Silicon Graphics Indigo II Impact workstation with an R10000 CPU. Table 4.2 summarizes the timing results. Although the hardware used is now considered obsolete, these statistics should still give the user some idea of the speed of the algorithm. The main bottleneck are the matching and interpolation processes, because they are typically repeated a number of times while the user determines suitable parameter values. Although the running times seem slow, they are competitive for a method that does not require manual feature specification and does not use any graphics hardware.

Process	Time (seconds)
Height field generation	7 (per image)
VBM generation	104 (per image)
Clustering	39 (per model)
Distance calculation	4
Matching and interpolation	20
Rendering from interpolated VBMs	65 (per frame)

Table 4.2: Timing results for VBM image interpolation algorithm

4.7 Summary

In this chapter we have presented a method for image interpolation and rigid registration using VBMs. In this algorithm, VBMs are used to represent the images and matching is performed with the Ranjan-Fournier similarity measure. We used three test cases, one of facial images and two of medical data, to demonstrate the capabilities and limitations of the method. We have shown that this method can interpolate between images effectively with minimal preprocessing.

4.8 Observations

- The VBM approach to shape matching produces visually accurate results when used for interpolating and rigidly registering greyscale images of average complexity, even with little or no manual intervention.
- Determining the parameter values (size, position, and feature weights) of the similarity measure requires an iterative, trial-and-error approach. However, the stability of the matching method with respect to the parameters makes determining appropriate values straightforward.
- Before clustering, the VBM models generated from our images contain many extra balls, which can cause mismatches and other problems during the matching process. These problems are significantly reduced by clustering. This is an example of how simplification can be a useful method for stabilization.
- The lack of topological constraints during matching and interpolation can be problematic in that a feature represented by a set of balls can split into two or more pieces when a subset of the balls gets matched and migrates to another feature. A method for grouping primitives and enforcing intra and intergroup topological constraints would be a useful feature for VBMs. This is partly the motivation for the work presented in Chapters 6 and 7.
- Although linear interpolation of the matched balls gives reasonable results in our experiments, there is actually little control over the intermediate shapes. Further investigation into transition control is likely needed for more complex objects and applications.
- The CSA graph matching routine is computationally intensive ($O(n^2)$ in space, $O(n^3)$ in time, where n is the number of balls in one model). As shown by Martindale [101], using an alternate algorithm, such as that found in the Library of Efficient Data types and Algorithms (LEDA), can have a significant positive impact on efficiency.

Chapter 5

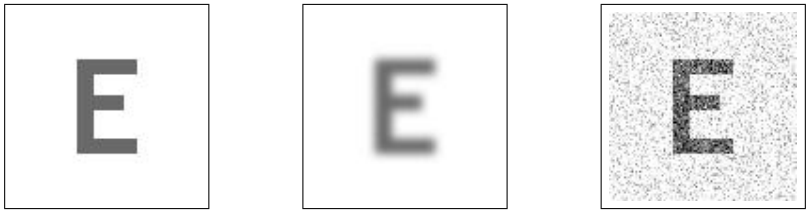
Shape Model and Threshold Extraction

5.1 Motivation

The efficient extraction of shape features from raw data such as pixel or voxel values requires a representation that is both stable and flexible. In this chapter, we present the results of an experiment in which we use VDMs to explore and extract shape information from 2D greyscale images. From an image, our algorithm outputs the intensity range spanned by each significant object as well as a VDM of the object. There are two main purposes for this experiment. The first is to see if the VDM is a flexible enough representation to be used for shape extraction with no *a priori* knowledge of the objects in the images. We use data that spans a wide range of shape complexity and topology. This is in contrast to Ranjan's experiments [129], which used largely similar shapes (*e.g.*, fish, calf, cow, giraffe). Our second primary goal is to perform an analysis of the stability and accuracy of the similarity measure for quantifying shape differences.

The algorithm discussed in this chapter, originally presented in [154] with earlier results, is a largely automatic method that is designed to reveal the strengths and limitations of the VDM representation and similarity measure. The main idea of the method is to measure the differences in the shapes of objects in an image as we vary the intensity threshold applied to the image. An *object* in this case can be comprised of two or more spatially distinct components of the same intensity range. For example, two kidneys in an image would be considered collectively as one object. The novelty of this method is primarily in the use of the *shape gradient*, the amount of shape difference caused by a given change in threshold value, to determine the occurrence of significant shape change *events* in the given intensity range. These events determine the thresholds that we should use for computing VDMs that represent objects of potential interest to the user.

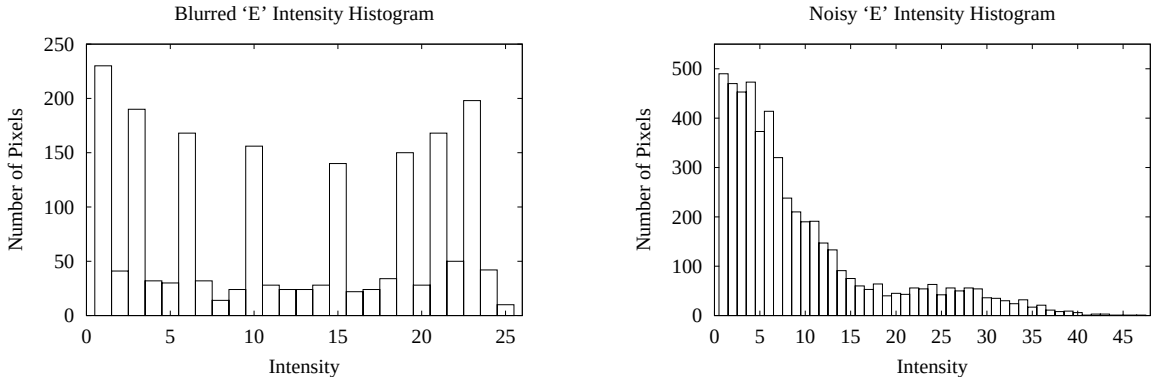
Figure 5.1 provides a motivating example that illustrates how the shape gradient method compares to the most commonly used method for determining thresholds, histogram analysis (*e.g.*, [111]). The 'E' on the left has an intensity of 25, with the background having an intensity of 0.[1] Because the boundary of the 'E' is sharp, the intensity histogram for this image is very simple and finding thresholds for this object is trivial. In contrast, Figures 5.1b and 5.1c show a blurred and noisy version of the 'E', respectively. The blurring is done with a Gaussian filter of radius 2.5, and the noise has a double Gaussian distribution, with means at 0 and 25.



(a) **(b)** **(c)**

Figure 5.1: (a) Sharp 'E' (b) Blurred 'E' (c) Noisy 'E'

Figure 5.2 shows the histograms of the blurred and noisy images. Even though the images show essentially the same shape, their histograms are very different, and it is not at all obvious what thresholds would best define the object's intensity range, especially if the blurring and noise characteristics are not known.



(a) **(b)**

Figure 5.2: (a) Intensity histogram for blurred 'E' (b) Intensity histogram for noisy 'E'

[1]The pixel intensities of the three images in Figure 5.1 have been scaled and inverted to increase visibility for the reader.

Figure 5.3 shows the graphs of how the shape gradient varies with intensity for the blurred and noisy ‘E’ images. As explained in Section 5.3.4, a minimum in a shape gradient plot marks the lower threshold of a significant object. The plots in this case have obvious minima at 14 for the blurred ‘E’ and at 16 for the noisy ‘E’. These minima indicate good lower thresholds to use for extracting shape models of the object.

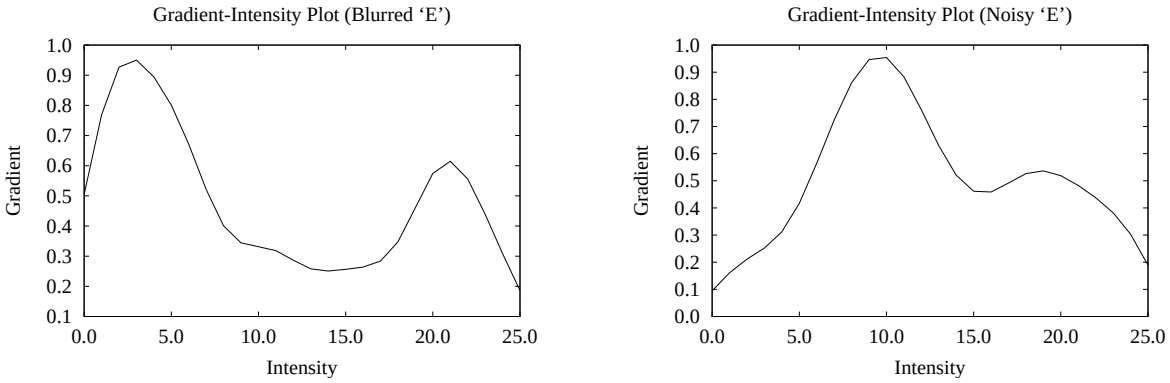


Figure 5.3: (a) Shape gradient plot for blurred ‘E’ (b) Shape gradient plot for noisy ‘E’

5.2 Related Work

The work presented here is related to many interesting papers dealing with shape modelling for feature extraction. We list a few representative examples here. At the low end of the complexity spectrum are feature clustering techniques that do not use a higher-level shape model. The idea is to first extract from the image some simple primitives, most commonly edges, then connect them together to form representations of the objects in the image. Acharya and Menon [1] and Xu *et al.* [170] discuss many examples in their reviews of segmentation methods. The advantage of using only simple primitives is that they incorporate very few assumptions about the objects represented and are thus very flexible. The most common problem with these techniques is instability. For example, edge-based approaches depend strongly on the characteristics of the edge detector used.

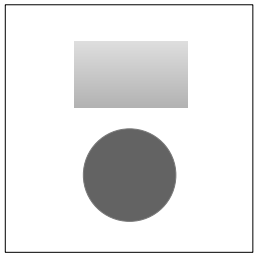
The most popular and generally effective group of shape models used for feature extraction are deformable models [102, 147]. Examples include energy minimizing snakes [75, 103] and 3D deformable surfaces [104, 158]. A limitation of most of these techniques is that they have to impose smoothness constraints on the models, which limit the types of objects that can be represented with any given set of parameters. On the other hand, these constraints can

sometimes help to stabilize the extraction process by bridging gaps or ignoring spurious edges. The main problem is that if there is little or no *a priori* information about the image, such as in an exploratory application, it may be difficult to estimate suitable parameter values.

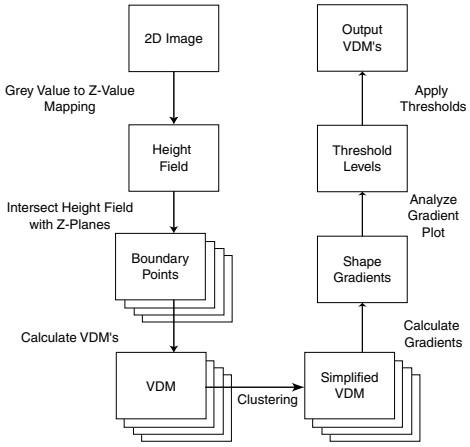
Another group of shape models used for feature extraction are based on the medial axis representation ([28, 29]). The most significant approach for using the medial axis for segmentation is by Pizer *et al.*, who propose a shape model that can be used for various applications in medical image processing [122]. As explained in Section 2.1.1, Pizer's approach represents shapes using interconnected figures. One of the significant shortcomings of the Pizer model is that the similarity measure only works with objects having the same number of figures and medial primitives, which limits its flexibility.

5.3 Algorithm

This section describes the details of our algorithm for extracting shape models from images. The simple image shown in Figure 5.4a is used as an example to explain the processes involved. This image (150×150, 8 bits per pixel) contains two objects, one having uniform intensity (100), the other having a linear intensity gradient ($175 - 225$).



(a)



(b)

Figure 5.4: (a) Simple test image (b) Overview of shape model extraction algorithm

As illustrated in Figure 5.4b, the main steps of our algorithm are:

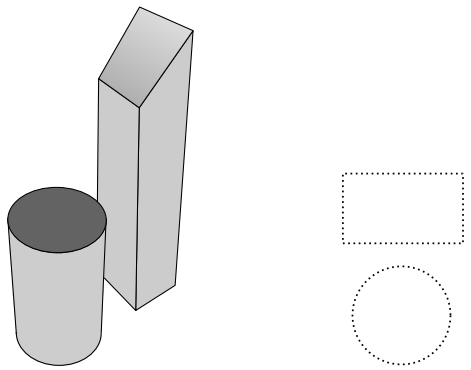
1. Generate boundary points.
    i. Generate a height field from the image's intensity data by mapping the value at each pixel to a z-value.

ii. Intersect the height field with n planes $z = I_i$, $i = 1, \ldots, n$, where I_i is an intensity value. This results in a set of boundary points for each of n grey levels.

2. Calculate a VDM for each set of boundary points.
3. Simplify each VDM by clustering.
4. Calculate the shape gradient between successive levels using the 2D Ranjan-Fournier similarity measure.
5. Determine the thresholds of interest from the maxima and minima in the shape gradient data.
6. Use the VDMs from Step 3 and the thresholds from Step 5 to compute VDMs of the objects in the image.

5.3.1 Boundary Point Generation

Creating a height field from a greyscale image is relatively straightforward. All that is required is a mapping from the intensity value at each pixel to a z-value in the height field. The simplest case would be a linear mapping. The points in the height field are then used as the boundary points of a volume that is bounded below by the image plane. Figure 5.5a shows the volume created from the example in Figure 5.4a using a linear mapping. The resulting volume



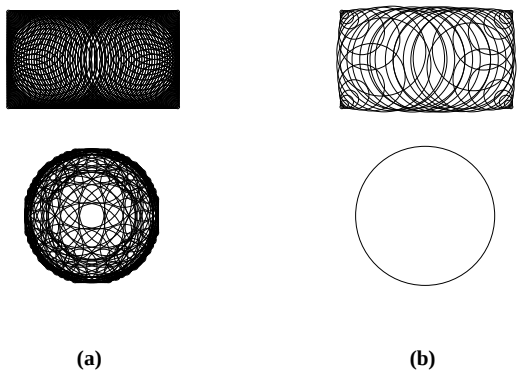
(a) **(b)**

Figure 5.5: (a) Volume created from the test image in Figure 5.4a (b) Boundary points created from intersecting the volume with a z-plane

is then intersected with n different z-planes. The values used for z are $\{I_i,\ i = 1, \ldots, n : I_{min} \leq I_i \leq I_{max},\ I_{i+1} = I_i + I_{incr}\}$, where I_{min} and I_{max} are the minimum and maximum intensity values of the image, and I_{incr} is the intensity increment from one level to the next. The intersection of each plane with the volume results in a set of boundary points at that level. Figure 5.5b shows the boundary points created by intersecting the test volume with the plane $z = 50$.

5.3.2 VDM Generation and Simplification

The boundary points are then used to compute a VDM for each level, using the method described in Section 3.1. Figure 5.6a shows the VDM generated from the boundary points in Figure 5.5b. This model has 358 disks. After



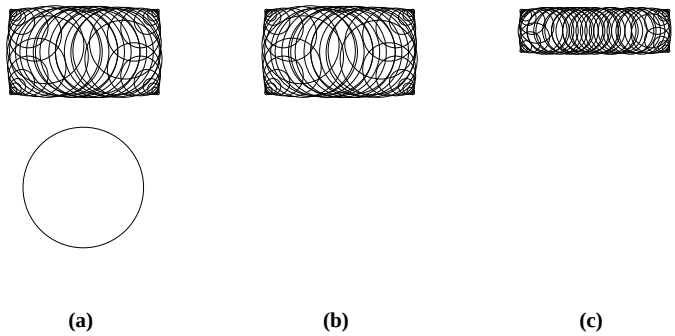
(a) **(b)**

Figure 5.6: (a) VDM (358 disks) generated by intersecting the volume in Figure 5.5a with the plane $z = 50$ (b) Simplified version of the model, with only 46 disks

each VDM is generated, it is simplified using the clustering method described in Section 4.4.1. Figure 5.6b shows a simplified version of the VDM shown in Figure 5.6a. The resulting VDM only has 46 disks.

5.3.3 Shape Gradient Computation

The next step in our algorithm is the computation of shape gradients between VDMs on successive levels. The shape difference between two VDMs is computed by matching disks between the two models using the similarity measure discussed in Section 4.4.3. Figure 5.7 shows three VDMs computed from the image in Figure 5.4a at three different levels.



(a) **(b)** **(c)**

Figure 5.7: VDMs computed from Figure 5.4a at three intensity levels (a) 50 (b) 125 (c) 200

An analysis of how the shape gradient changes as the threshold varies allows the user to determine the ranges of intensity where one or more objects in the image are actively changing shape, as well as the ranges where the shapes are relatively stable. A graph of gradient versus intensity, in which for each threshold value I_i the shape difference between the VDMs at I_i and I_{i+1} is plotted, allows the thresholds of interest to be determined easily. We refer to this graph as the *matched gradient* plot.

In most cases, the number of disks does not remain constant across levels. Therefore, the matching process can leave a number of disks unmatched at each level. A disk that exists at one level but "disappears" at the next is considered to have shrunk in place to a radius of zero. A plot of $\bar{r^2}$ versus intensity, where $\bar{r^2}$ is the average of the radii squared of unmatched disks at each level, may contain additional shape change information not captured by the matched gradient plot. This *unmatched gradient* plot is especially useful in cases where one or more significant objects in the image have (close to) uniform intensity, as these objects can disappear between successive levels without affecting the matches in the other objects. An unmatched gradient plot would have very distinct spikes at such intensities. Thus, even though for many real-world images, the matched gradient plot alone is sufficient for threshold determination, the unmatched gradient plot is sometimes useful for providing complementary information.

Modification of the Similarity Measure

As mentioned in Section 4.4.2, the Ranjan-Fournier similarity measure is not independent of scale, position, or orientation. This can be problematic in certain situations, particularly where the prealignment procedure cannot be easily done. Although we expect the similarity measure to perform well for the current application, because the image is not moved or scaled between thresholds, we conjecture that modifying the similarity measure to be less dependent on the given transformations may result in more accurate matches and subsequently greater stability.

The original shape distance between two disks a and b, as explained in Section 4.4.3, is:

$$d(a,b) = w_p d_p(a,b) + w_s d_s(a,b) + w_f d_f(a,b)$$

where $d_p(a,b) = (x_a - x_b)^2 + (y_a - y_b)^2$ is the square of the Euclidean distance between the centres, $d_s(a,b) = (r_a - r_b)^2$ is the square of the difference in radii, $d_f(a,b)$ is the feature distance, and the w's are the weights determined by the user. The feature distance is already independent of the given transformations, so we only need to modify d_p and d_s. In place of d_p and d_s, we suggest defining two new functions:

$$d_{p'}(a,b) = (|x'_a| - |x'_b|)^2 + (|y'_a| - |y'_b|)^2$$

where (x'_a, y'_a) and (x'_b, y'_b) are the positions of the disk centres relative to the centres of mass of the respective VDMs, and

$$d_{s'}(a,b) = (r'_a - r'_b)^2$$

where r'_a and r'_b are the radii of the disks divided by the square root of the areas of the respective VDMs. These changes make the measure independent of translation and rotation, and reduces the dependence on scale ($d_{p'}$ still varies with scale). We discuss the results of applying this new measure to a challenging test case in the Results section.

5.3.4 Shape Gradient Analysis

The following observations are useful when analyzing a matched gradient plot:

- Minima indicate relative stability in the shapes of objects as the threshold is varied.
- Maxima indicate shape change events. A sudden rise in the matched gradient occurs when the threshold reaches a point where a small increase in intensity causes a significant object to breakdown and/or distort.
- From the above, we can conclude:
    1. A minimum or the point at the beginning of a peak marks the lower threshold of a significant object.
    2. The point immediately at the end of a local peak marks the upper threshold of a significant object.
- A wide peak indicates the object spans a relatively large intensity range; a sharp spike means the object spans a narrow intensity range.

Figure 5.8 shows the shape gradient plot computed from the image in Figure 5.4a. The matched gradient clearly shows a significant object, in this case the rectangle, in the range $175 \leq I \leq 225$. The unmatched gradient in this range confirms the result. The matched gradient increases here because as the rectangle gets thinner, the decrease in width becomes more significant relative to its size. The unmatched gradient decreases because the unmatched disks

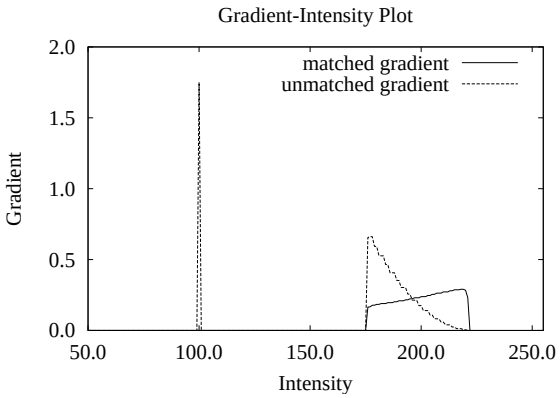


Figure 5.8: Shape gradient plot computed from the test image in Figure 5.4a

are getting smaller. The unmatched gradient also shows a peak at $I = 100$. This means there is an object of uniform intensity, in this case the disk, at that level. The height of the peak indicates that the object is of significant size.

In this example, the maxima and minima are very obvious and distinct from each other. However, this is often not the case with real images. This brings into question how we should decide which maxima and minima are to be considered significant. These decisions currently require user input and are based on the level of detail at which the user wishes to analyze the image. Applying a Gaussian filter to smooth out small maxima and minima in the plot to is a helpful step.

5.3.5 Shape Model Generation

Having determined the upper and lower thresholds of a significant object, we can compute a shape model for that object by using the VDMs generated during the gradient computation process. As demonstrated in the Results section, for some applications it is best to simply use the VDM at the lower threshold as the object's shape model. In other cases, we take the lower threshold VDM, and remove from it any disks or parts of disks that would lie within the upper threshold VDM if the two VDMs were superimposed. If the resulting VDM has partial disks in its boundary, it is retriangulated to form a new VDM. The result is a VDM of the object in the appropriate intensity range.

5.4 Results

This section presents the results of applying our technique to two of the test cases used in our experiments. We use a linear mapping for the height field and an increment of 1.0 between threshold levels for both examples. For simplification, we have found that a sphericity of 0.96 works well for most of the images we have tested to date, which include a wide variety of CT and MRI images. The three weights w_p, w_s, and w_f in the cost function of the distance measure are set to 1.0 for the shape gradient calculation.

The first test case, shown in Figure 5.9a, is an MRI image of a human brain. This image has 128 grey levels. Figure 5.9b shows the shape gradient plot computed from this image. Using the analysis method outlined in Section 5.3.4, we can clearly identify three significant intensity ranges. The first ($20 \leq I \leq 40$) is the range for the fluid surrounding the brain and is of limited interest for shape analysis. The other two ranges, labelled R1 and R2 in Figure 5.9b, are for the whole brain and the grey matter inside the brain.

Figure 5.10 shows the two VDMs representing the shapes identified by our algorithm as being stable at $I = 43$ and $I = 100$. The unmatched gradient confirms shape stability at these intensities, which are the lower thresholds for the whole brain and the grey matter inside. In this case, the VDMs for the upper thresholds are not used. The reason we do not use the upper threshold for the whole brain is that the VDM at $I = 43$ is a good representation of the shape

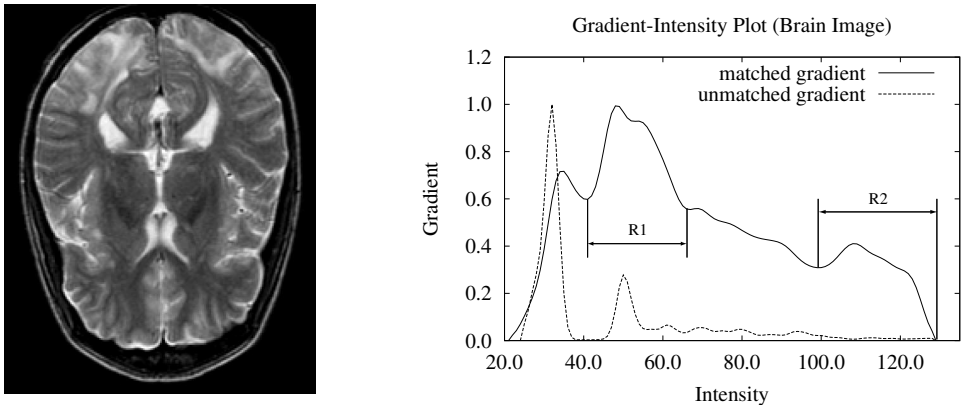


Figure 5.9: (a) Brain MRI image (b) Shape gradient plot computed from the brain MRI image

of the brain as a solid mass, and this is likely to be of more use in the analysis of the overall shape of the brain than the same model with parts of its interior taken out. We do not need to use an upper threshold for the grey matter, because R2 is at the top of the image's intensity range. Therefore, we use the VDMs at the lower thresholds as the final shape models for the two objects.

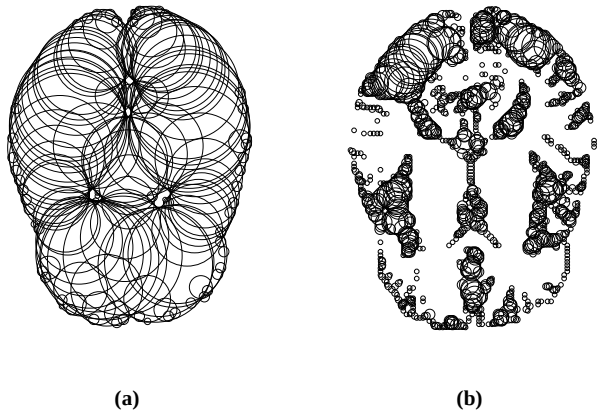


Figure 5.10: VDMs computed from the brain MRI (Figure 5.9a) at two levels (a) 43 (whole brain) (b) 100 (grey matter)

The second test case, shown in Figure 5.11a, is a CT image of a person's lower abdomen. This image has 128 grey levels. The most noticeable structures present are the liver, kidney, small intestine, and spine. This image is a more challenging case than the brain MRI because it has multiple objects with overlapping intensity ranges. The

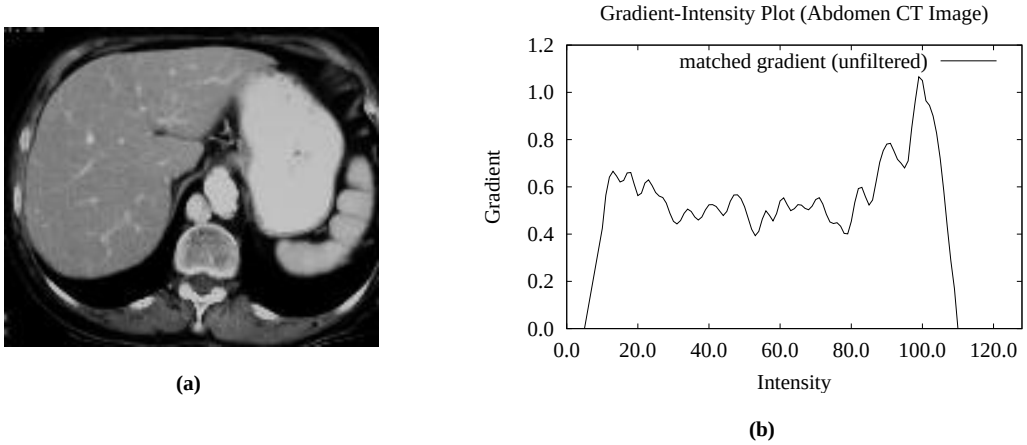


Figure 5.11: (a) Lower abdomen CT image (b) Shape gradient plot computed from the lower abdomen CT image

matched gradient plot from this test case (Figure 5.11b) shows instabilities not present in the brain MRI case. The many smaller peaks and valleys are a symptom of the complex shape changes that occur in the image as the threshold is varied, as well as a possible sign that the similarity measure has some instability resulting from mismatches.

In order to extract the most salient features from the shape gradient data, we apply a Gaussian filter to smooth out the plot. The result, shown in Figure 5.12, is a graph that reveals only the major shape change events. Dividing the graph using the minima results in four intensity ranges. The first ($5 \leq I \leq 30$) corresponds to the fluid and soft tissue surrounding the organs. The second ($31 \leq I \leq 51$) corresponds to the liver. The third ($52 \leq I \leq 76$) is associated

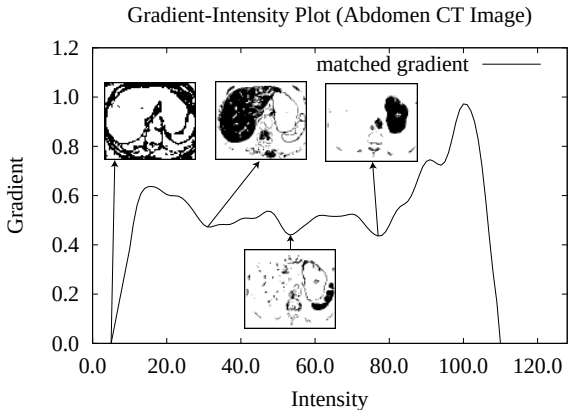


Figure 5.12: Gaussian smoothed version of the shape gradient plot in Figure 5.11b

with the small intestine, and the last ($77 \leq I \leq 109$) is the intensity range for the kidney. Figure 5.12 shows the CT image with each of the four sets of upper and lower thresholds applied. Again, the algorithm has resulted in effective thresholds for the significant objects in the image. However, there is not a distinguishable maximum for the spine, which points out a weakness of the method. The intensity range of the spine overlaps with those of the liver and small intestine, and because the spine is significantly smaller than the other two structures, it is essentially lost. This kind of problem is common to all algorithms using global thresholds.

Figure 5.13 shows the extracted shape models of the liver, kidney, and small intestine computed using their respective upper and lower thresholds, with some manual removal of disks not connected with the objects. Visually speaking, these VDMs are accurate shape representations of the objects in the image.

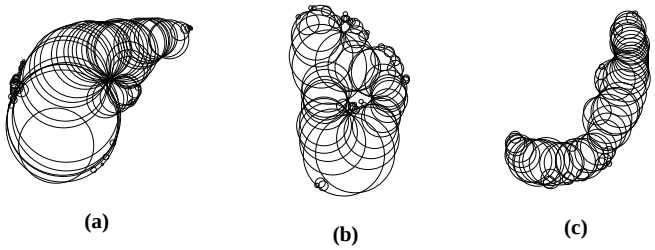


Figure 5.13: VDMs extracted from the abdomen CT image (a) liver (b) kidney (c) small intestine

For this test case, we also compute the gradients using our modified similarity measure defined in Section 5.3.3. Figure 5.14 shows the unfiltered shape gradient plot. Compared to the gradient plot in Figure 5.11b, this graph suggests that the new similarity measure is more stable while being able to identify the same thresholds within a tolerance of ± 1. While the difference between the two results is not dramatic, it is significant enough to warrant further investigation.

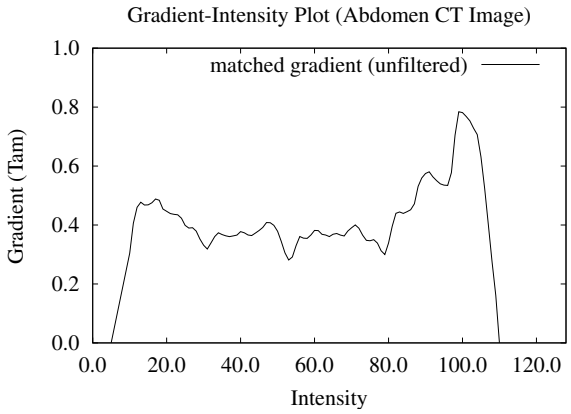


Figure 5.14: Shape gradient plot computed from the abdomen CT image using our modified similarity measure

5.5 Summary

We have presented a novel method for exploring shape information in a greyscale image. Our algorithm uses the shape gradient to effectively extract thresholds and shape models of significant objects. The VDM representation is used to compute the gradient and represent the extracted shapes. We presented the test results of applying our algorithm to artificial and real-world images.

5.6 Observations

- The extracted thresholds and VDMs are qualitatively accurate representations of the shape data present in the test images, which indicates that the similarity measure is forming appropriate correspondences.
- The evidence suggests that our modifications to make the similarity measure less dependent on translation, rotation, and scaling can further improve the reliability of the shape distance function.
- The flexibility of the VDM and similarity measure are strong advantages in an exploratory application such as this one. The shapes in our test images undergo many changes in complexity and topology as the threshold is varied, but the VDMs of the objects are still easily computed at each level without manual intervention, and the similarity measure is still able to output reasonable results.
- Mismatches in the number of primitives between VDMs being compared can be a significant source of information that can be used as an independent measure to augment the data in the matches. Perhaps the information from the matched and unmatched disks can be incorporated into a single measure.
- While the extracted VDMs appear to be reasonably accurate representations of the objects, further processing is likely needed for certain applications. For example, for segmentation purposes a method for computing a polygonal or spline-based boundary from a VDM would be useful. This is partly the motivation for the work presented in Chapter 8.

Chapter 6

Two-Dimensional Shape Simplification

6.1 Motivation

In the work presented in this chapter and the next, we investigate how the medial axis and the VBM can be used together effectively. The goal is to enhance medial axis and VBM applications by taking advantage of the desirable and complementary properties of both representations. As mentioned in Section 2.1.1, the medial axis transform (MAT) is a shape model that represents an object by the set of maximal balls that are completely contained within the object. For a continuous object this set is infinite. The medial axis consists of the centres of the balls, and can be intuitively thought of as the skeleton of the object. Skeletal representations have numerous applications in visualization (*e.g.*, [109, 120, 122, 151]), computer graphics (*e.g.*, [26, 57, 169]), and computer vision (*e.g.*, [112, 145]). One of the main reasons the medial axis is considered an attractive model is that it is a visually intuitive representation, as evidenced by some studies on human shape perception (*e.g.*, [35, 36, 87]). Another strength of the medial axis is that it forms a concise representation of the object's topology, a shape property that is considered important for many applications such as path planning and molecular modelling.

One of the primary drawbacks of the medial axis is that it is very sensitive to minor perturbations of the object's boundary, such as that caused by discretization, segmentation errors, image noise, and so forth. The goal of most medial axis *pruning* techniques is the removal of branches associated with these artifacts, typically resulting in a much cleaner and more usable medial axis. This chapter discusses a technique, originally presented in [155], for 2D medial axis pruning. This application is an example of how the VDM and medial axis can be used to mutually enhance the other's stability. We use the shape information in the VDM to prune the medial axis, and in turn use the structure of the medial axis to determine which disks should be preserved in order to maintain the topology of the object. The method introduced uses a VDM of an object to compute the medial axis, and defines a shape-based *significance measure*

based on the VDM to remove the branches generated by noise or other artifacts on the object's boundary without losing the fine features that are often altered or destroyed by other current pruning methods. The algorithm consists of an intuitive threshold-based pruning process, followed by an automatic feature reconstruction phase that effectively recovers lost details without reintroducing noise. The result is a technique that is robust and easy to use. Our tests show that the method works well on a variety of objects with significant differences in shape complexity, topology, and noise characteristics.

6.2 Background

Most current medial axis pruning algorithms suffer from the problem that when excess branches are removed, other branches that correspond to fine but perceptually significant features of the object are excessively shortened. This is primarily due to the fact that most pruning methods use a global significance measure (*e.g.*, feature size or frequency) to discern between data and noise. Unfortunately, for most measures there is a significant overlap between what is considered noise and data, and when the noise is removed some data is taken with it. Figure 6.1 provides a simple, motivating example. The unprocessed medial axis (Figure 6.1b) has many spurious branches, largely because of discretization artifacts. Figure 6.1c shows the typical result of pruning with a global threshold. In this case the significance measure is noise size. The result is that all of the branches associated with noise are gone, but the remaining branches are also shortened, causing the tip of the pencil to become rounded (Figure 6.1d).

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

Figure 6.1: Pencil (a) Original object (b) Original axis (c) Typical pruned axis (d) Object reconstructed from typical pruned axis

So far, proposed solutions to address this issue have proven inadequate. Attempts to overcome the noise/data overlap problem by developing more complicated global measures frequently result in a fuzzy relationship between parameter values and how they correspond to changes in object features, thereby making the estimation of an appropriate threshold more difficult. Another general approach is to recover lost details by *unpruning* the remaining branches after the noisy branches are removed. Current algorithms using this approach typically do not work well because they depend on a global threshold for the unpruning process as well, thereby subjecting it to the same overlap problem.

In this chapter, we summarize a novel approach for medial axis denoising that removes unwanted artifacts while preserving fine features, such as sharp corners and thin limbs. Our method first prunes the axis by using an intuitive

global threshold based on noise size, then *automatically reconstructs* the fine features by extending the remaining branches. The use of a simple pruning method based on a physically meaningful parameter followed by an effective feature reconstruction process makes the technique robust and easy to use. The user can determine an appropriate threshold simply by estimating the size of the noise to be removed and in most cases, rough estimates are adequate because the reconstruction process can automatically correct many errors caused by overly aggressive pruning.

Our feature reconstruction algorithm extends each branch by using local shape information defined on the VDM and does not depend on a global threshold. This process localizes the discernment between data and noise to the feature level which significantly reduces the overlap problem. As demonstrated in our results, this localization allows each branch to be extended to an appropriate length and in the correct direction so that each feature is reconstructed accurately without reintroducing noise. For example, Figure 6.2a shows the medial axis of the pencil after processing with our algorithm; the noise is gone and the tip of the pencil is still sharp.

(a) **(b)**

Figure 6.2: Pencil (a) Medial axis processed by our denoising algorithm (b) Our reconstructed object

Although simple in its design, we can show that our technique works well for removing artifacts of various sizes and characteristics from objects of arbitrary shape complexity and topology. We have tested our algorithm on a wide variety of data, a number of examples of which are included in this chapter to demonstrate the effectiveness of our method.

6.3 Related Work

Our method for the construction of the medial axis is one of a number of algorithms that use the Voronoi diagram of a set of sample points regularly spaced along the object's boundary to form a discrete approximation of the skeleton. The main idea of such algorithms, examples of which include [13, 31, 114], is to first compute the Voronoi diagram of the points, then extract a subgraph to form the skeleton. For example, the subgraph can be extracted by taking only the Voronoi vertices that are inside the boundary of the object.

Given a model or an image of an object, there are two main approaches for producing a clean medial axis. The first approach performs some form of preprocessing on the image or model before computation of the medial axis. Such preprocessing usually consists of blurring (*e.g.*, [124]) or boundary smoothing (*e.g.*, [107]) of the original object

to reduce spurious branches. Blurring and smoothing techniques can result in undesirable structural changes to the medial axis [113, 140]. In addition, these operations typically use a global scale measure (*e.g.*, size of smoothing kernel) to filter out noise, and smaller object features are often altered or destroyed during preprocessing.

The other main approach is to start with the complete axis and prune the branches using some heuristic (*e.g.*, [16, 31]). The general idea is to have a significance measure that assigns an importance value to each branch. During pruning, this value is compared to a user-given threshold to determine how much of each branch gets cut. With an ideal significance measure and threshold, only the parts of the axis associated with noise would be removed, and the rest of the axis would remain unaltered. However, for currently available measures there is usually an overlap between data and noise; a threshold value that completely removes the branches associated with noise will usually shorten the remaining branches as well, often to an undesirable degree. Thus, finding a good threshold value often requires striking a delicate balance between noise removal and feature preservation. In addition, the complexity of some measures makes them seem ad hoc and adds to the difficulty of finding an appropriate threshold. In some cases, even multiple parameters are required (*e.g.*, [14]). To overcome the difficulties in estimating parameter values, a completely automatic method for threshold selection is proposed in [112, 113]. The method is able to determine an appropriate value for many shapes, but there are instances in which the algorithm strongly oversegments the shape, resulting in large missing features. Figures 6.3a and 6.3b, generated with the algorithm from [113], show two examples. In Figure 6.3a, the stem of the leaf is missing from the axis; in Figure 6.3b, two of the goat's legs are among the larger features not represented.

(a) **(b)**

Figure 6.3: Examples of medial axis pruning using Ogniewicz's algorithm, showing large missing features (a) Leaf (b) Goat

A number of researchers have proposed methods that utilize a postprocess to recover small details destroyed by pruning. Such methods add an unpruning process that extends the branches that remain (*e.g.*, [112, 140]). The typical

approach is to use the same significance measure as used for pruning and simply apply a different threshold to extend the branches. This approach forces the user to select two thresholds, and still the problem of overlap between data and noise is not solved. This frequently results in some branches being overextended (*i.e.*, noise is reintroduced) while others are still too short.

Our algorithm is designed to address the problems described above, and consists of the following two main processes:

1. A threshold-based pruning technique with a single parameter and a simple significance measure that gives the user intuitive control.
2. An automatic feature reconstruction process that extends each branch using local shape information and does not depend on a global threshold.

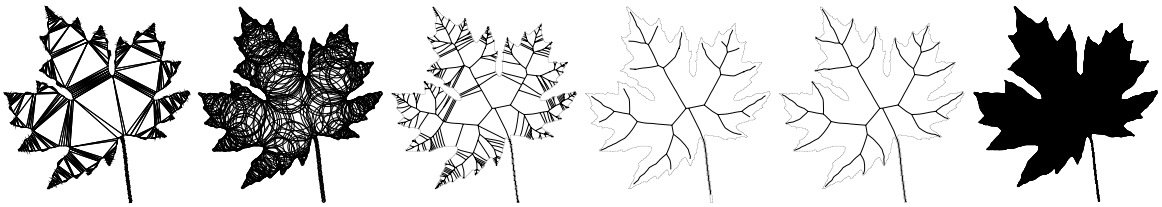
The result is an algorithm that gives the user the freedom to select a threshold that completely removes all noise while providing a reliable feature reconstruction process that brings back the right amount of detail at each branch. This technique gives the user some control, so that large features are not accidentally removed, but hides the more complex data/noise discernment algorithm inside an automatic process so that the user is not burdened with a complicated significance measure.

It should be noted that some pruning methods, such as [112], are hierarchical in nature and can produce results at multiple levels of detail. Thus, at coarser levels, the loss of fine features is considered acceptable, even appropriate. In contrast, our algorithm is designed to remove artifacts of a given size, while preserving as much detail in the rest of the object as possible. However, at finer levels of detail, the goals of the algorithms are essentially the same.

6.4 Algorithm

Given the boundary points of an object, the main steps of our algorithm for medial axis noise removal are as follows:

1. Construct the medial axis from the boundary points (Figures 6.4a-c).
2. Prune the spurious branches (Figure 6.4d) by using a user-determined global threshold.
3. Extend the remaining branches to recover small details (Figure 6.4e) by using a local measure of shape smoothness to distinguish between data and noise.
4. Reconstruct the object with the clean medial axis (Figure 6.4f).



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figure 6.4: Maple leaf example (a) Delaunay triangulation (b) VDM (c) Original medial axis (d) Pruned medial axis (e) Pruned medial axis with details recovered (f) Denoised shape

6.4.1 Medial Axis Construction

Like most Voronoi-based methods for medial axis construction, our method assumes that the sample points are spaced with sufficient density along the boundary of the object. Our method for the computation of the medial axis from a boundary point set consists of these two main steps:

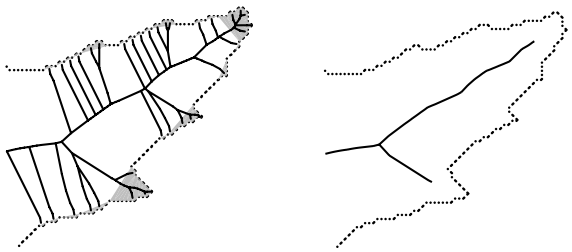
1. Compute the set of Voronoi disks that are inside the object (Figures 6.4a-b), as described in Section 3.1.
2. Construct the medial axis by connecting the centres of the disks (Figure 6.4c).

We represent the medial axis as a directed graph[1] whose root node is the centre of the largest disk in the VDM. Construction of the graph begins by creating a line segment (called an *axial segment*) between the root node and each of its neighbours (two disks are *neighbours* if their corresponding triangles share an edge). The root node is the *parent* node and the neighbours are *child* nodes. This process is then repeated with the neighbours as parent nodes until all disks in the VDM are linked. The result is a medial axis that can be traversed recursively by starting at the largest disk and following the child nodes until they reach the boundary of the object. With this construction method, each node in the axis has a corresponding disk in the VDM, which in turn has a corresponding triangle in the Delaunay triangulation. We refer to a node that has no children as an *end node*. A node that has more than one child is called a *branch node*. A *branch* is defined as any chain of nodes that has a single branch node, located at the beginning, and an end node at the end.

[1]For objects of genus zero, the graph is naturally a tree. For objects with holes, we break each cycle by imposing an appropriate breakpoint in the loop. This is only for the purposes of traversing the graph without running into infinite loops. In order to preserve the topology of the original object, cycles are never pruned.

6.4.2 Area-Based Pruning

The purpose of the pruning process is to remove parts of the medial axis that are associated with noise on the object's boundary. A significance measure is needed for determining whether a feature of the object should be considered as noise. In the context of pruning, a feature can be defined as the set of triangles associated with any subtree of the axis graph. Our significance measure assigns an importance value to a feature based on the surface area it covers. An example is shown in Figure 6.5a, which shows a part of the maple leaf from Figure 6.4. In this figure, the shaded



(a) **(b)**

Figure 6.5: Area-based pruning (a) Original axis (b) Pruned axis

regions are the features that are smaller in area than the user-given significance threshold. The significance value of a feature can be determined by summing the areas of all triangles in the subtree associated with that feature. Any subtree that has a value below the threshold is pruned. Each feature can be seen as being *supported* by the branches of the subtree, so when the subtree is pruned, the feature is eliminated.

This significance measure has the following two main advantages:

- The pruning is guaranteed not to disconnect the graph, because a parent always has a higher significance value than its child.
- Area is a simple and intuitive significance measure and an appropriate threshold can be estimated via a typically straightforward analysis of the data acquisition method. Even when knowledge of the acquisition method is insufficient, a suitable value can be found by visual inspection of the data more easily than most heuristics-based measures that have a less direct physical meaning.

The result of the pruning process is that all noise below the threshold size is eliminated. With an appropriate threshold, the only branches that remain are associated with significant features of the object. As mentioned, a side effect of pruning with a fixed global threshold is that the remaining branches are typically shorter than they should be and fine but important details are often lost.

Noise Model and Threshold Selection

Like practically all medial axis pruning methods, our noise model focuses on artifacts in the form of relatively small protrusions from a larger body. Pruning techniques are the most effective when applied to this type of noise, called *additive noise*, because they work by removing and shortening branches. A common example of additive noise are the artifacts originating from the dark current in CCD cameras. We define two conditions that must be satisfied in order for a feature to be classified as noise:

1. The size of the feature is smaller than the user-determined threshold. This is the condition used in the pruning phase.
2. The feature is not *smoothly connected* to the rest of the object. The transition into a protrusion is considered smooth if the abruptness of the narrowing does not exceed the changes in width in the parts of the object leading up to the feature. This condition is used in the feature reconstruction phase and is defined more precisely in the next section.

To select an appropriate threshold for pruning, consideration must be given to the data and application at hand. If the noise characteristics are known, an estimate of the artifact size can be made, and selection of the threshold is relatively simple. Otherwise, the value can be set by visual inspection of the data. Our implementation is such that the threshold can be set as an absolute size or as a percentage of the total area of the object. In most of our examples, the value is determined interactively. The feature reconstruction process, as described in the next section, is robust enough to allow a fairly imprecise threshold selection, and a range of appropriate values exists for most objects.

The most important guiding principle in selecting an appropriate threshold is to ensure that each significant protrusion in the object has a single supporting branch. The reason for this is best described by Leyton's Symmetry-Curvature Duality Theorem [87]:

Theorem 6.1 (Symmetry-Curvature Duality) *Any section of curve, that has one and only one curvature extremum, has one and only one symmetry axis. This axis is forced to terminate at the extremum itself.*

Figure 6.5b shows an example of how each significant feature is supported by a single branch. The result of pruning should be that each significant extremum in the border has a single remaining branch.

6.4.3 Feature Reconstruction

The purpose of the feature reconstruction process is to recover significant parts of the object that have been pruned because they fall below the size threshold. For example, Figure 6.6 shows part of the maple leaf before and after reconstruction. The shaded areas show features that would be removed. The axis in Figure 6.6b clearly represents the

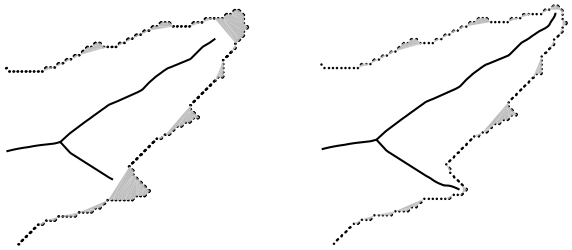


Figure 6.6: Feature reconstruction (a) Before reconstruction (b) After reconstruction

shape of the object better than the axis in Figure 6.6a.

Our reconstruction algorithm works on one branch at a time, and its main idea is to use the shape information present in the remaining branches and disks to calculate local smoothness constraints that determine how far each branch can be extended to recover fine features without reintroducing noise. In this scheme, what is classified as noise varies from branch to branch. Figure 6.7 shows an example in which the branch in one feature (B) is extended further than in another (A), even though the tips of the features have the same angle and both branches reach the boundary in the original axis. In this case, the small sharp point in Feature A is regarded as noise because it falls below the size threshold and violates the local smoothness constraints.

The measure of smoothness that we use is termed the *axial gradient*, which measures the change in the width of the object per unit length of the axis. Each axial segment has an axial gradient value that is mathematically defined as the signed difference in radius between the child disk and the parent disk, divided by the Euclidean distance between the two nodes (Figure 6.8). Note that this definition is similar to the gradient used in the feature distance of the similarity measure described in (Section 4.4.3), with the main difference being that the direction of the axial gradient is determined by the medial axis, rather than the three largest neighbouring disks.

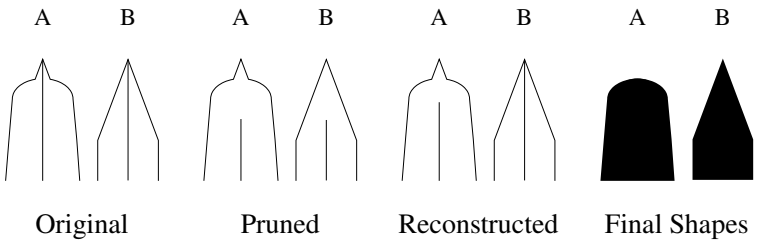


Figure 6.7: Effect of smoothness constraints on feature reconstruction

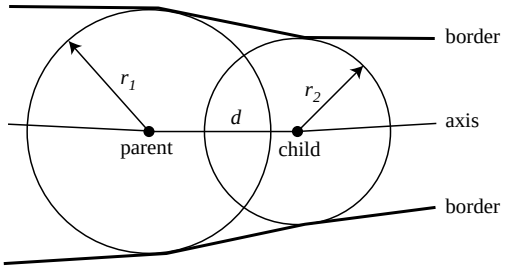


Figure 6.8: Axial gradient between two nodes ($g = \frac{r_2 - r_1}{d}$)

The reconstruction algorithm works by starting at the root node and following the axis until it reaches the current end node of the branch in question, while keeping track of the greatest absolute axial gradient value ($|g_{\max}|$) encountered along the path. This value is used to determine how far the branch can be extended. The reasoning is that if a feature at the end of the branch is below the threshold size and is marked by a narrowing that is more abrupt than any other change in width along the path, then the feature is most likely noise. Because $|g_{\max}|$ is calculated individually for each branch, the data/noise overlap problem associated with global thresholds is significantly reduced.

When calculating $|g_{\max}|$ along a path, special consideration must be given to branch nodes, because the degree of *continuity* of a feature across a branch node depends strongly on the branching angles at these nodes. Figure 6.9 shows an example in which one feature (B) has stronger continuity across a branch node than another (A). In this case, the shape information along the path before the branch node is more relevant to Branch B than Branch A. For any given path, the branching angle at a branch node is a good indicator of how much of the maximum gradient encountered before the node should be "carried over" past the branch node. Intuitively, an angle of 0 degrees (maximum continuity) should impose no change to the maximum gradient, whereas an angle of 90 degrees or greater (no continuity) should

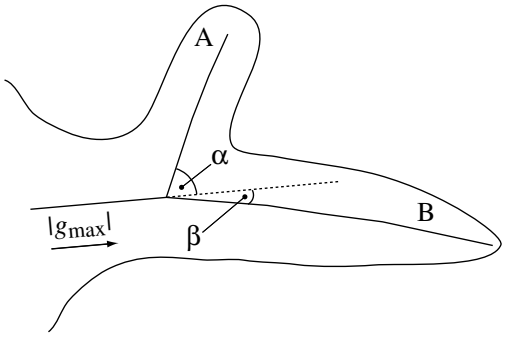


Figure 6.9: Axial gradient at a branch node (because β is smaller than α, $|g_{\max}|$ is more relevant to Branch B than Branch A)

cause $|g_{\max}|$ to become 0 at the branch node. For an angle between 0 and 90, we use this formula:

$$g_{\max}(\theta) = |g_{\max}| \times \frac{90 - \theta}{90}$$

where $|g_{\max}|$ is maximum axial gradient before the branch node and $g_{\max}(\theta)$ is the maximum axial gradient at the branch node for the segment with angle θ.

Once $|g_{\max}|$ for a given path has been calculated, the next step is to extend the branch at the end of the path. Two main issues need to be considered at this point. First, if there is more than one direction for possible extension (*i.e.*, the current end node is a branch node in the original axis), a decision needs to be made to determine which segment to follow. The second issue is how far to extend the branch.

The first issue is addressed in consideration of the second part of Leyton's theorem, which says that the symmetry axis of a feature should terminate at the extremum of that feature. Given that the objects we are considering have many minor extrema due to noise, we need to distinguish these from the extrema associated with significant features. Again, we use the axial gradient for this purpose. If there is more than one path for possible extension, the segment with the lowest axial gradient is chosen. This method is essentially a greedy algorithm for finding the smoothest path. As shown in the Results section, this gives a high likelihood of reaching the correct extremum. Figure 6.10 illustrates an example.

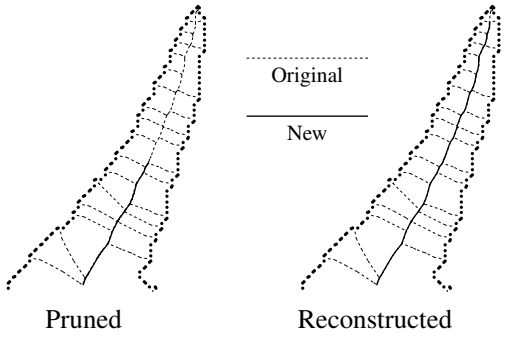


Figure 6.10: Branch extension (at each branch node, the segment with the lowest axial gradient is chosen, which extends the branch toward the appropriate extremum)

The second issue of how far to extend the branch is addressed by comparing the absolute value of the axial gradient of the chosen candidate segment ($|g_{\text{cand}}|$) with the $|g_{\max}|$ of the branch. If $|g_{\text{cand}}| \leq |g_{\max}|$, the segment is added to the end of the branch. Note that because our noise model defines noise as smaller protrusions from a larger body, a feature can be classified as noise only if $g_{\text{cand}} < 0$. Segments are added until $g_{\text{cand}} < 0$ and $|g_{\text{cand}}| > |g_{\max}|$, or there are no more candidate segments.

As demonstrated by the examples in the Results section, our branch extension algorithm is very effective in automatically reconstructing features to their appropriate degree of sharpness without reintroducing noise.

6.4.4 Shape Reconstruction

An advantage to using the VDM method for medial axis construction is that there is a one-to-one correspondence between the nodes of the axis and the set of disks. For densely sampled models, we can simply take the VDM associated with the new axis to reveal the final object shape after pruning and feature reconstruction. For more sparsely sampled cases, a boundary reconstruction step may be desirable.

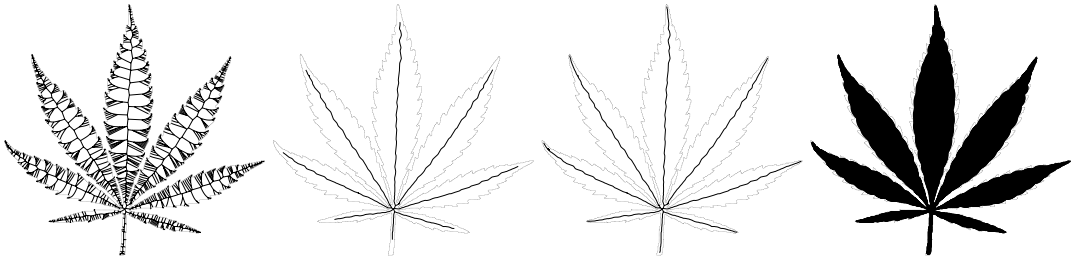
6.5 Results

We have tested our algorithm on many objects with various amounts of noise. Our data sources include map data, medical images, aerial photographs, and specially designed test models. In this section, a number of examples are used to illustrate the effectiveness of our method. Some of the current limitations of our algorithm are also discussed. The examples include four synthetic objects and one object from an MRI image. The characteristics of the objects are shown in Table 6.1. All thresholds (t) are specified as a percentage of the total area of the original object.

Object	Figure	Dimensions	Threshold (t)
Leaf	6.11	479×462	0.3%
Goat	6.12	254×344	0.4%
Lizard	6.13	443×446	2.0%
Brain	6.14	255×293	0.1%
Rectangle	6.15	339×238	2.0%

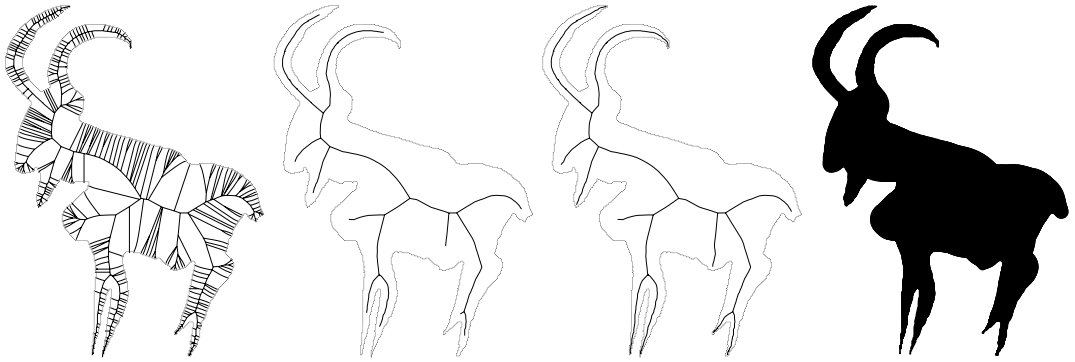
Table 6.1: Characteristics of our test objects

Figures 6.11 (Leaf) and 6.12 (Goat) are good examples of how the algorithm can reconstruct fine features after removing unwanted artifacts. In the Leaf, the small variations along the border are removed, resulting in a much smoother shape. However, the fine features such the tips of the leaflets and the thin stem are nicely reconstructed. This example can be compared to the result by Ogniewicz in Figure 6.3a, where the stem is missing completely. The Goat shows how the branch extension method can use local shape information to reconstruct features to differing degrees of sharpness where appropriate. For the sharper features, such as the horns, goatee, and legs, the branches are extended to the tips. For the more rounded features, such as the mouth, chest, belly, and tail, the branches are extended enough to fully reconstruct the features, but not so far as to reintroduce noise.



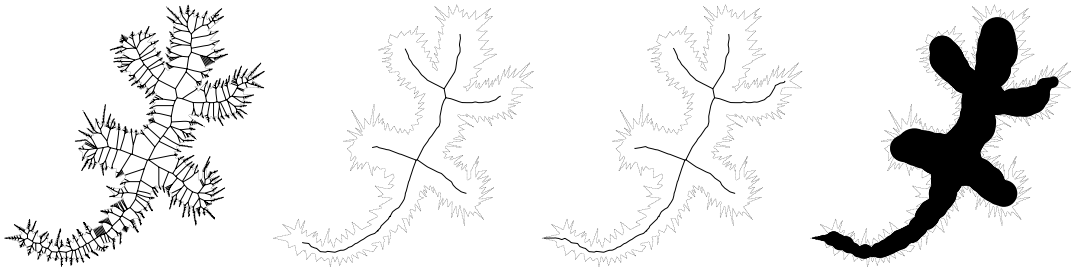
(a) **(b)** **(c)** **(d)**

Figure 6.11: Leaf (479×462, $t = 0.3\%$) (a) Original axis (b) Pruned axis (c) Features reconstructed (d) Final shape with original boundary superimposed



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

Figure 6.12: Goat (254×344, $t = 0.4\%$) (a) Original axis (b) Pruned axis (c) Features reconstructed (d) Final shape

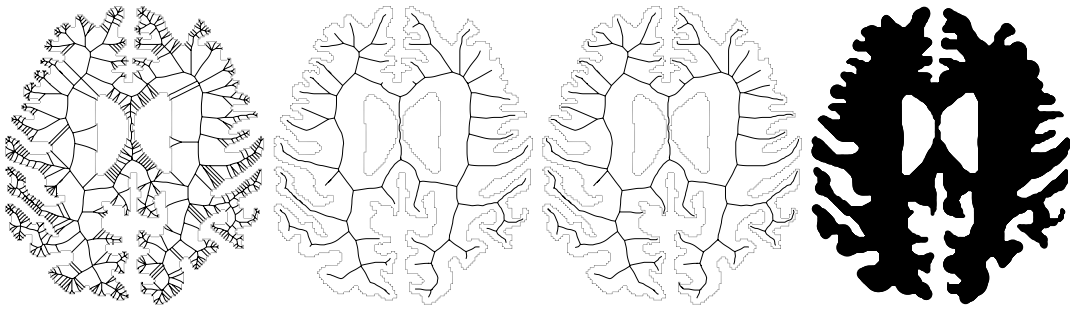


(a) **(b)** **(c)** **(d)**

Figure 6.13: Lizard (443×446, $t = 2.0\%$) (a) Original axis (b) Pruned axis (c) Features reconstructed (d) Final shape with original boundary superimposed

Figure 6.13 (Lizard) is an example where the "noise" artifacts are quite large. Our algorithm still results in a nicely simplified shape in this case. In the head and the four legs, the reconstructed branches do not extend into any of the spurious spikes along the border. The shape of the tail causes its supporting branch to be extended to the tip. It is somewhat debatable whether the spike at the end should be considered as a significant feature or noise, but in this case its inclusion seems appropriate.

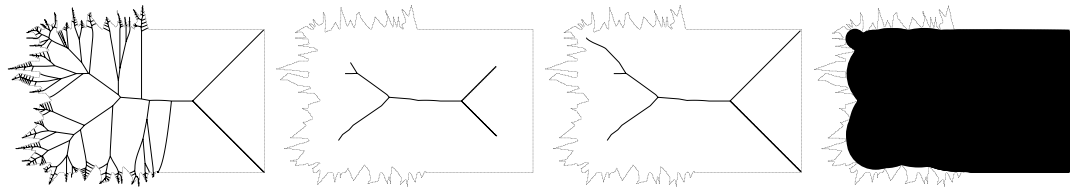
Figure 6.14 (Brain) shows an object from an MRI image. The noise in this case is a combination of image noise, segmentation errors, and discretization artifacts. The shape of this object is significantly more complex than in the other examples. This object is also of a different topology in that it has two holes. Our algorithm is able to effectively remove the various types of noise from all areas of the object.



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

Figure 6.14: Brain (255×293, $t = 0.1\%$) (a) Original axis (b) Pruned axis (c) Features reconstructed (d) Final shape

Figure 6.15 (Rectangle) is an example of an object that has a heterogeneous distribution of noise. In this case, the left side of the rectangle is very noisy, whereas the right side is clean. Again, the reconstruction algorithm is able to perform well, extending branches to their appropriate lengths so that the corners on the right side are sharp, while the artifacts on the left side are removed.

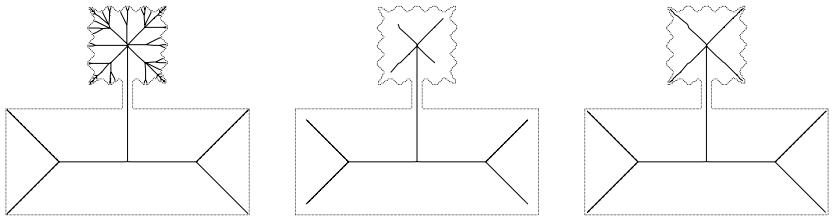


(a) **(b)** **(c)** **(d)**

Figure 6.15: Rectangle (339×238, $t = 2.0\%$) (a) Original axis (b) Pruned axis (c) Features reconstructed (d) Final shape with original boundary superimposed

6.5.1 Limitations

Our use of the maximum axial gradient as a local smoothness constraint inherently assumes that the axial gradient does not vary greatly within a single feature, at least not relative to the gradient of any noise at the end of the branch. Although such variations are not encountered frequently, they certainly can exist. Figure 6.16 shows an example of such a situation. In this case, a thin *neck*, or local narrowing, in the object causes the branches in the noisy square to be overextended into the corners. A possible solution to this problem is to impose a limit on the length of the path used for computing the maximum gradient. For example, instead of only having one root node from which to start, we can break the graph down into subgraphs using features such as necks to do the division. Although theoretically straightforward, this has not been implemented at the time of writing of this document.



(a) **(b)** **(c)**

Figure 6.16: Narrow neck example (a) Original axis (b) Pruned axis (c) Axis after feature reconstruction, with the branches extended too far

The branch extension algorithm may not work well in the case of a very short branch with a large branching angle, because there would be very few axial segments and, therefore, a very limited amount of local shape information with which to calculate an appropriate maximum axial gradient. Increasing the sampling density of the boundary points may be a possible solution.

Although our tests show it to be largely effective, the greedy algorithm for finding the smoothest path for branch extension is not *guaranteed* to reach the correct extremum. A solution would be to search further into the tree before making a path selection.

6.6 Summary

We have presented a novel algorithm for 2D noise removal using the VDM and medial axis. Our algorithm consists of a threshold-based pruning method that uses a simple significance measure, followed by an automatic feature reconstruction process that extends the remaining branches to recover fine features without reintroducing noise. We

demonstrated the effectiveness of the method with a number of examples varying in shape complexity.

6.7 Observations

- The 2D medial axis can be used effectively to preserve the topology of a VDM during simplification and other processing.
- The branching angles of the 2D medial axis can be used effectively to determine the degree of continuity of an object's features. This can be seen as a method for the "partial partitioning" of a VDM.
- Adding the 2D medial axis makes the visual discernment between noise and data easier to perform, compared to using only the VDM.
- The 2D medial axis and the VDM can be used effectively in combination.

Chapter 7

Three-Dimensional Shape Simplification

7.1 Motivation

The work in the last chapter investigates how the 2D medial axis and the VDM can be used in combination. In this chapter, the study is extended to 3D. Again, the goal is to use the medial axis and the VBM in a complementary manner. It has been documented that the construction of the 3D medial axis is much more stable when using an approximating union of balls as an intermediate representation, rather than building the axis directly from the boundary of the object [15]. In turn, as discussed in Section 3.1.2, convergence of the ball centres to the medial axis is a useful criterion for computing a VBM that has a relatively small number of balls while attaining an accurate approximation. We take advantage of this mutually beneficial relationship in our application.

In addition, we explore how the medial axis can be used to preserve the topology of a VBM during processing. As shown in Section 7.5.3, this goal is considerably more difficult than in 2D. Also, we experiment with using the medial axis to partition a VBM, a concept also explored in the 2D case. In this algorithm, the medial axis is explicitly broken up into components before further processing.

In this chapter, we discuss a method, originally presented in [156], for simplifying the shape of 3D objects by manipulating a VBM that approximates the medial axis transform (MAT). From an unorganized set of boundary points, our algorithm computes the VBM and the medial axis, decomposes the axis and VBM into parts, then selectively removes a subset of these parts in order to reduce the complexity of the overall shape. The result is a simplified medial axis and VBM that can be used for a variety of shape operations. In addition, a polygonal surface of the resulting shape can be directly generated from the filtered VBM using a robust surface reconstruction method. The algorithm presented is shown to have a number of advantages over other existing approaches.

7.2 Background

Motivated by the usefulness of the medial axis and its unfortunate sensitivity to boundary noise and other artifacts, many *regularization* methods have been proposed to filter out the spurious components. Some of these methods also aim to preserve the topology of the axis during the pruning process. As shown in Chapter 6, the preservation of topology during processing of the 2D medial axis is relatively easy to achieve because the axis is one-dimensional and structurally hierarchical, thereby providing a natural processing sequence when progressing from the outer branches toward the inside. In addition, any cycles in the axis indicate the presence of real loops in the shape of the object and can be easily preserved. As a result, a number of very effective algorithms exist, and the problem is more or less considered solved. In 3D, the situation is more complicated. The relationship between components of the medial axis is much more complex. There are many cycles that do not represent loops in the object. There is no natural processing order and there are usually many different deletion sequences possible. In addition, there is often a mutual dependency between skeletal components where the removal of one component can change the topological relationship between others. The result is that the simplification algorithm must impose a processing order and perform explicit topology checks as components are removed. A number of regularization schemes have been proposed (*e.g.*, [7, 15, 46, 109]), each with their own advantages and limitations.

We present an algorithm that has a number of advantages over other existing approaches. Our parts-based approach, described in Section 7.5, allows the medial axis to be simplified to a much greater degree without certain undesirable effects such as the disintegration caused by some methods that operate on lower order primitives. This makes the approach suitable for a variety of applications ranging from noise removal to manual modelling. In addition, we can use the connectivity of the parts to efficiently preserve the topology of the axis during simplification, a goal unmet by most other 3D medial axis techniques. Also, our method allows the user to control the degree of simplification using simple, visually intuitive parameters. Finally, we have designed our algorithm to fit very well into an existing surface reconstruction framework, so that the filtered VBM can be used directly to generate an accurate polygonal representation. The reconstruction algorithm, called the *power crust* ([7]), uses an approximate MAT to compute an interpolating surface from boundary point samples.

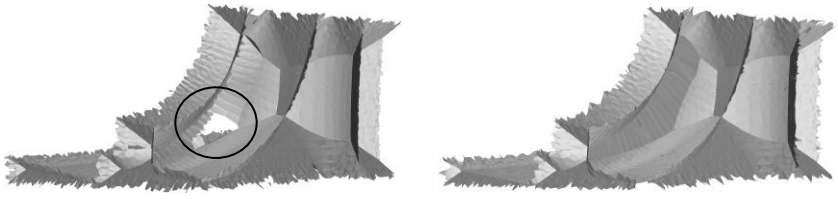
It is important to note that the goals of our algorithm differ significantly from the many *mesh simplification* algorithms (see [65, 96] for examples) whose primary aim is to minimize the number of polygons in a model given certain constraints such as storage size and image resolution. Such algorithms do not necessarily simplify the shape of the object. In contrast, our algorithm focuses on the reduction of shape features in the object by removing parts of the underlying shape model (*i.e.*, the VBM). The primary goal of our work is to produce a *feature-based* simplification algorithm that does not adversely affect the integrity of the remaining components.

7.3 Related Work

As in the 2D case, many researchers use the Voronoi diagram (and/or the dual Delaunay tetrahedralization) of a set of sample points on the object's boundary to approximate the medial axis. This method is suitable for many applications as sample points are typically readily available (*e.g.*, laser scans) or easily derived. Our algorithm follows this approach. Other methods for computing the medial axis include volume thinning (*e.g.*, [55]) and wave propagation (*e.g.*, [86]).

There are only a few methods for 3D medial axis regularization that preserve the topology of the object. Because of the lack of a hierarchical structure, a *peeling* approach is usually employed in which the outermost components are removed one layer at a time. Two of the most notable techniques are by Attali and Montanvert [15] and Näf *et al.* [109]. Both of these methods begin with the entire set of interior Delaunay tetrahedra, and delete them one layer at a time according to some criteria for maintaining topological consistency. The main problem with such an approach is that unlike in 2D, the Voronoi vertices (circumcentres of the tetrahedra) in 3D do *not* converge to the medial axis as the sampling density approaches infinity [8]. Therefore, regardless of sampling density, there are many tetrahedra that are not even close to the medial axis that are being used for enforcing topological constraints. This can often hinder the regularization process.

A number of recent approaches are designed to guarantee convergence. For example, Dey and Zhao [46] have a method for computing approximations that converge to the medial axis by applying certain filter conditions to the Voronoi diagram. These filters are also used to eliminate noisy components. The strongest advantage of their technique is that the filter parameters are independent of scale and density. Another approach is by Foskey *et al.* [52], who present an efficient method for computing a simplified medial axis using a spatial subdivision scheme and graphics hardware. The primary advantage of their approach is speed. The greatest drawback of most of these techniques is that topology is ignored, and in many cases a disintegration effect is seen in which holes appear in the axis (*e.g.*, Figure 7.1). Such errors are particularly prevalent when simplification is being aggressively applied. In addition to being visually distracting, these artifacts make subsequent use of the axis more difficult.



Fandisk Medial Axis (Dey and Zhao)

Fandisk Medial Axis (Our Algorithm)

Figure 7.1: Example of the holes that are prevented by our algorithm

As described in Section 3.1.2, Amenta *et al.* propose an approach that utilizes a relatively small subset of the Voronoi vertices called the poles [8]. They have proven that the inner poles converge to the interior medial axis. Amenta *et al.* use the power diagram of the poles (using the polar ball radii as weights) to compute an approximate medial axis, called the *power shape* [7]. The power shape method is robust and gives visually reasonable results. Unfortunately, for most objects the power shape is largely composed of very flat tetrahedra instead of 2D faces, and this geometry complicates tasks such as parts decomposition. In addition, their proposed method of simplification can result in an approximation that can diverge quite dramatically from the true medial axis as the level of detail decreases.

Amenta and Kolluri use the power shape to produce a more accurate axis composed only of 2D components [10]. Our experimentation with this algorithm reveals that it produces many duplicate vertices, causing cracks in the resulting medial surface. This again makes the axis difficult to work with. Our algorithm for computing the medial axis builds on the work by Amenta and Kolluri. We make improvements to eliminate the degeneracies and add a simplification method that preserves topology. We choose to work with Amenta and Kolluri's method because of its convergence guarantees and because of the existence of the power crust algorithm, which can take a set of filtered polar balls and reconstruct a polygonal surface. Figure 7.2 shows how our algorithm complements the power crust pipeline.

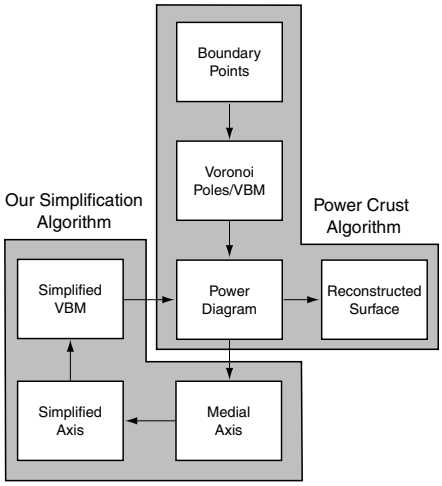


Figure 7.2: Processing pipeline for 3D shape simplification and surface reconstruction

7.3.1 Summary of Amenta and Kolluri's Algorithm

As mentioned above, our method for computing the medial axis builds on the work by Amenta and Kolluri. We briefly summarize their algorithm here. As shown in Figure 7.2, Amenta and Kolluri's algorithm computes the medial axis from the power diagram of the Voronoi poles. This computation can be broken down into several steps, as shown in Figure 7.3:

1. The power shape of the object, a subcomplex of the dual of the power diagram, is computed by keeping only those simplices whose vertices are all inner poles.
2. For each regular triangle in the power shape, a *singular point* is computed. The three polar balls centred at the vertices of the triangle form two intersection points. One of these points lies on the surface of the union of polar balls, and the other lies in the interior. The surface point is the singular point.
3. A Voronoi diagram of the singular points is computed. The subcomplex of the Voronoi diagram that intersects the power shape is computed as the medial axis.

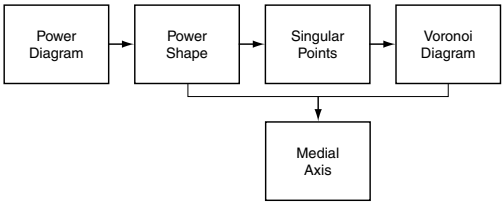


Figure 7.3: Amenta and Kolluri's algorithm for computing the medial axis from the power shape

More details on the algorithms by Amenta *et al.*, including the theoretical derivation, sampling assumptions, and convergence guarantees can be found in [8].

7.4 Medial Axis Computation

For use in geometric processing, the most significant limitation of Amenta and Kolluri's algorithm is that it produces many duplicate vertices in the medial axis. These vertices cause double edges that show up in the form of cracks in the medial surface. By solving this problem we can produce a medial axis that has much cleaner geometry for further processing. We take a combinatorial approach because removing duplicate vertices numerically is computationally expensive and subject to errors in precision. From our analysis, there are two primary causes of the duplicate vertices:

1. Many duplicate singular points are generated. This results in duplicate vertices in the medial axis because the vertices of the axis are Voronoi vertices computed from the singular points.
2. Many of the singular points are cospherical. The Voronoi vertices are the circumcentres of the dual Delaunay tetrahedralization of the singular points, so when more than four points are cospherical, duplicate circumcentres are produced, resulting in multiple identical medial vertices.

To address the first problem, we note that Amenta and Kolluri's algorithm computes a singular point for *every* regular triangle in the power shape. However, we observe that two or more neighbouring triangles can often produce identical singular points. We use Figure 7.4 to illustrate the 2D analogy of a situation that happens frequently in real

data sets. In this case, A–E are five boundary points of the object, the dotted lines represent the Delaunay triangulation of the boundary points, P1–P3 are the polar balls computed from the triangles, and S1, S2 are two simplices of the power shape (edges in 2D, triangles in 3D). The construction leads to S1 and S2 producing the same singular point, located at boundary point B, because P2 intersects both P1 and P3 at that point. We can efficiently identify and remove duplicate singular points produced in this manner by checking which polar balls have corresponding tetrahedra that share the same boundary point.

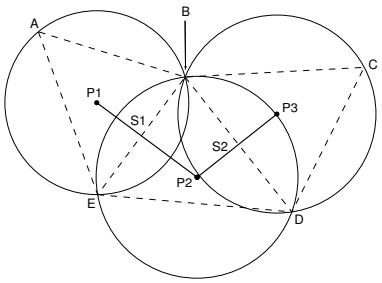


Figure 7.4: Simplices S1 and S2 produce two identical singular points, co-located at point B

The second problem, cospherical singular points, can be attributed to the simple fact that the points are computed by intersecting balls in the VBM. So it should not be surprising to find cases in which more than four singular points lie on the surface of a polar ball. We can quickly identify which singular points are cospherical by keeping track of which polar balls are intersected to form which singular points. We can thus find and eliminate duplicate medial vertices very efficiently.

7.5 Simplification

As with most 3D medial axis regularization methods that preserve topology, we utilize a peeling approach in which the outer layers of components are removed over a number of iterations. At the beginning of each iteration, we decompose the medial axis into parts. We then assign a significance value to each part that is a candidate for removal. An ordered pruning process then removes all parts that have a significance value in a given range and can be deleted while maintaining topological consistency. The number of iterations performed depends on the complexity of the shape and the degree of simplification required.

7.5.1 Parts Decomposition

After generating the VBM and medial axis, we decompose the axis into parts before further processing. We begin by triangulating all faces to make implementation easier. We then form parts by grouping triangles. Our decomposition

scheme takes advantage of the fact that the 3D medial axis can be viewed as a composition of 2D surfaces (sometimes called *boundary sheets*). The result of the decomposition is a set of connected 2-manifold parts embedded in 3D space. As shown in Section 7.5.3, this is very useful for simplifying the topological representation used during pruning.

The parts creation process begins at the boundaries of the medial axis and works inwardly on a level-by-level basis. The first level parts are at the boundaries of the axis, the second level parts are inward neighbours of the first level, and so on. Each first level part starts with a randomly chosen seed triangle that has an edge with no neighbours (*i.e.*, it is at a boundary of the axis) and does not yet belong to any part. The part begins to grow by gathering neighbouring triangles of the seed triangle. For each of the triangle's edges, if that edge is shared with only one other triangle, then the part grows into that neighbour. This growth process proceeds recursively until no more triangles can be added to this part. The resulting part is a 2D surface. Another boundary triangle is then selected to begin another first level part.

After all parts of level 1 have been formed (*i.e.*, there are no more unused triangles at the axis boundaries), parts of level 2 are computed. For a part of level i, where $i > 1$, a randomly chosen triangle that neighbours a part of level $i - 1$ is used to begin the part. The growth process is the same as that for level 1. In this manner a number of levels of parts are created. Figure 7.5 illustrates an example using a simple medial axis, in this case computed from the boundary points of a rectangular box. The numbers in the figure indicate the levels of the respective parts.

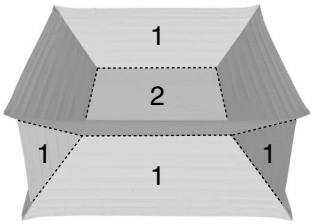


Figure 7.5: Medial axis showing level 1 and level 2 parts (the dotted lines show the boundaries between parts)

The user can limit the number of levels created depending on the application. During each pruning iteration, only parts of level 1 are removed, and parts of level > 1 are only used for enforcing topological constraints. Therefore, if topological preservation is considered unimportant for the current application, only the first level parts need to be created.

The decomposition of the medial axis leads naturally to a decomposition of the VBM, because each part of the axis is associated with a set of polar balls. To make this clear, we make use of two observations:

Observation 1 (Amenta and Kolluri [10]) *Every vertex of the power shape (*i.e.*, an inner pole) must lie on an edge or a vertex of the medial axis.*

Observation 2 *Since the medial axis is the intersection between the power shape and the Voronoi diagram of the singular points, every part of the axis must contain a subset of the inner poles.*

Thus, each part of the axis has a number of inner poles lying on its edges and/or vertices. We can combinatorially determine the set of poles that lie in a given part by keeping track of which polar balls intersect to produce which singular points, and tracking which singular points are subsequently used to produce which medial vertices and edges.

7.5.2 Pruning

The general simplification strategy is to remove one layer of the outermost (*i.e.*, level 1) parts at a time. This allows us to check for and prevent undesirable changes in topology. In order to determine which parts are to be removed, we apply two significance measures to evaluate the importance of any given part. In each iteration, the user selects a significance measure and a threshold value. Every level 1 part that falls below the threshold and satisfies all topological constraints is removed.

We have designed our significance measures to be efficient, versatile, and intuitive to the average user. The first measure is simply the number of triangles in the part. We use triangle count because it allows us to filter out a large number of insignificant parts with very little computation. The other measure that we use is the volume of the feature of the object that would be removed as a result of pruning the part. We use the volume because it is an intuitive and visually meaningful property of each component. Useful thresholds can be easily determined by examining the size of features that the user wants to remove. To make this measure independent of scale, we divide the volume of each part by the total volume of the object to give a relative value.

To estimate the volume for each part, we note that the set of Delaunay tetrahedra computed from the object's boundary points makes a visually reasonable approximation of the object's interior. Since these tetrahedra are available from earlier Voronoi computations, it would be efficient to reuse them for volume estimation. To do so we note from the previous section that each part is associated with a set of inner poles. Having the set of inner poles immediately gives us a set of tetrahedra, because each pole is computed as a circumcentre from the Delaunay tetrahedralization. We use the sum of the volumes of the tetrahedra associated with a part to estimate the volume of the object feature corresponding to that part. Figure 7.6 shows a feature of an object (the top of the Tweety model in Figure 7.12a) and the group of tetrahedra used to estimate its volume.

Although the two significance measures can theoretically be used in any order or combination, we have found the process generally more effective and easier to control if the triangle count measure is used first to remove the very small outer layer parts, followed by using the volume measure to remove the more significant parts underneath. The reasoning is that for most data sets, practically all of the very small parts in the first few layers are discretization or

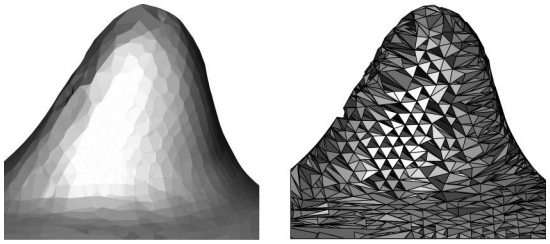


Figure 7.6: Feature of an object and the tetrahedra used to estimate its volume

noise artifacts. After these are removed, the volume measure can be applied iteratively in a manner appropriate for the data and application. Indeed, as discussed in Section 7.7, our experiments have shown this to be a useful strategy.

7.5.3 Topology Constraints and Pruning Order

Our algorithm is able to preserve the topology of the medial axis during pruning. The topological properties that concern us are the number of connected components and the number of loops. Each loop in the axis represents a tunnel going through the object. Given a set of connected parts representing the medial axis, we need to determine whether any given level 1 part can be safely removed without disconnecting the remaining parts and without changing the number of existing loops.

We deal first with the loops. The part removal method described in the previous section guarantees that new loops are never created, because only parts at the borders of the axis are candidates for pruning. In order to save existing loops, we first determine which parts join together to form loops in the axis. We can then preserve these parts during pruning. In 3D, finding loops is not just a simple matter of detecting cycles, because there are many cycles that do not form loops. For example, the three connected parts on the left in Figure 7.7 form a cycle (A→B→C→A) but do not make a loop. Removal of any of the three parts would not change the topology of the object. In contrast, the three parts on the right in Figure 7.7 do make a loop.

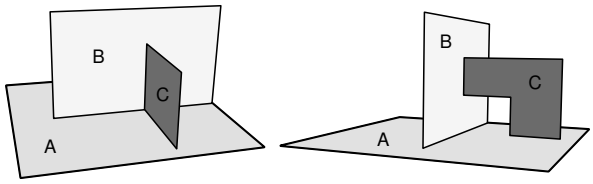


Figure 7.7: The three parts on the left do not form a loop, while the three parts on the right do.

To detect loops efficiently, we take advantage of the simplicity of our parts-based representation to build a topology graph that concisely captures the connectivity information between the parts. In this graph, each node represents a "point" of contact between two neighbouring parts. This "point", called a *contact curve*, takes the form of

an unbroken polyline shared by the touching parts. The edges of the topology graph represent the parts themselves. In order to eliminate cycles that are not loops, two nodes are collapsed into one if the corresponding contact curves are connected. For example, Figure 7.8 shows the graphs representing the topology of the two sets of parts in Figure 7.7.

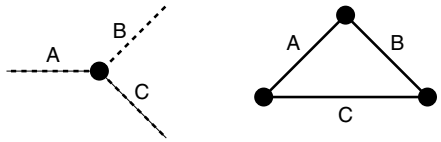
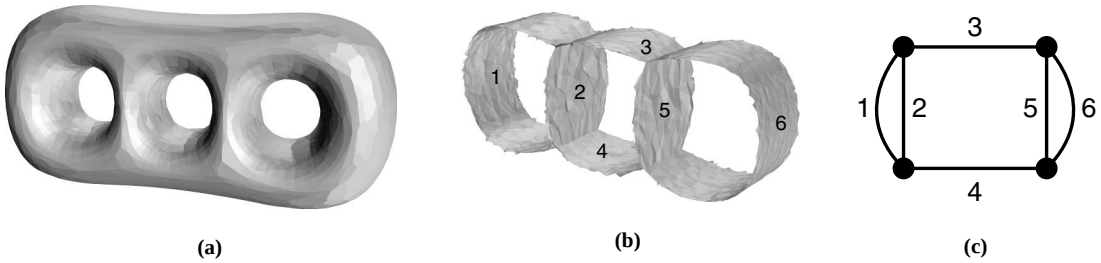


Figure 7.8: Graphs representing the topology of the parts in Figure 7.7

In the first graph, there is only one node because the contact curves between parts A and B, B and C, and A and C are all connected. The edges of this graph do not connect to any other nodes. The second graph has a loop in the configuration because the contact curves do not touch each other. Figure 7.9 shows a less trivial example computed by our algorithm. The topology graph has four nodes interconnected by six parts.



(a) **(b)** **(c)**

Figure 7.9: (a) Object (b) Medial axis (c) Topology graph

With the loops preserved, maintaining topology then becomes a matter of ensuring that the number of connected components stays the same. We satisfy this constraint by checking that the removal of a part does not disconnect any of the neighbouring parts.

The lack of a natural processing order and the fact that there are no theoretical results known about the effect of the deletion sequence on the skeleton mean that we need to impose a pruning order. Our algorithm takes the simple approach of sorting the parts in order from lowest to highest value, using the significance measures defined in the previous section. The reasoning is that in general, we want to remove the less important parts first whenever possible so that they do not prevent the more significant parts that typically represent more salient features from being pruned.

7.6 Surface Reconstruction

After pruning, we use the power crust algorithm [7] to reconstruct a surface of the simplified object. The algorithm works by computing a piecewise linear boundary between the inner and outer polar balls. Before simplification, each boundary point sample has an inner and outer pole. Afterwards, only a subset of the inner poles remain, because we discard the balls associated with the pruned parts of the axis. For most data sets, we only need to remove the inner poles. However, as mentioned in [7], for objects with sharp corners, more accurate reconstructions can be had by removing the corresponding outer poles as well. The polar balls associated with the remaining poles are used to compute the surface. We discuss the power crust algorithm further in the next chapter, Section 8.2.

7.7 Results

This section describes the results of testing our algorithm with a number of data sets. In all of the examples shown, it is easily noticed that our approach can greatly reduce the complexity of the axis without creating holes or breaks in the remaining components. Table 7.1 lists the examples presented, along with the processing times for computing and simplifying their medial axis. A Pentium 4 processor running at 2.0 Ghz is used. Our implementation makes extensive use of the CGAL and LEDA libraries. Although some of the models take several minutes to process, we feel confident that our algorithm is more efficient than the other current topology preserving methods (*e.g.*, [15, 109]), because of our use of the convergence property (Section 7.3) and parts-based topology graph (Section 7.5.3) to reduce the number of topology checks required.

Model	Points	Axis Generation (sec.)	Pruning (sec.)
Tweety	48668	186	205
Max Planck	25044	57	92
Hip bone	70688	348	354
Bunny	34835	103	136

Table 7.1: Processing times for 3D shape simplification

The typically pruning scheme that we use is to first apply the triangle count significance measure for one to three iterations with a very low value (≤ 5 triangles) and without topology checks. This usually removes many very small and visually unimportant parts with little computation. Then we apply the volume measure with topology checks for one or more iterations as required to achieve the desired level of detail.

We use the Bunny model (Figure 7.10) to demonstrate how our algorithm can automatically remove features

of various sizes. In this figure, the middle bunny is the result of applying four pruning iterations ($2 \times t_t, t_t = 5$, $1 \times t_v, t_v = 1.0\%$, $1 \times t_v, t_v = 5.0\%$) to reduce the overall detail of the model without losing the large features. The topology graph of the second bunny is shown in Figure 7.11. The third bunny is also the result of four iterations ($2 \times t_t, t_t = 5$, $1 \times t_v, t_v = 1.0\%$, $1 \times t_v, t_v = 10.0\%$), but with a greater final threshold to remove the larger features. The resulting object has lost all of its small details, such as the eyes, as well as most of the large features, such as the ears and tail. The reconstructed surface is considerably simpler than the original.

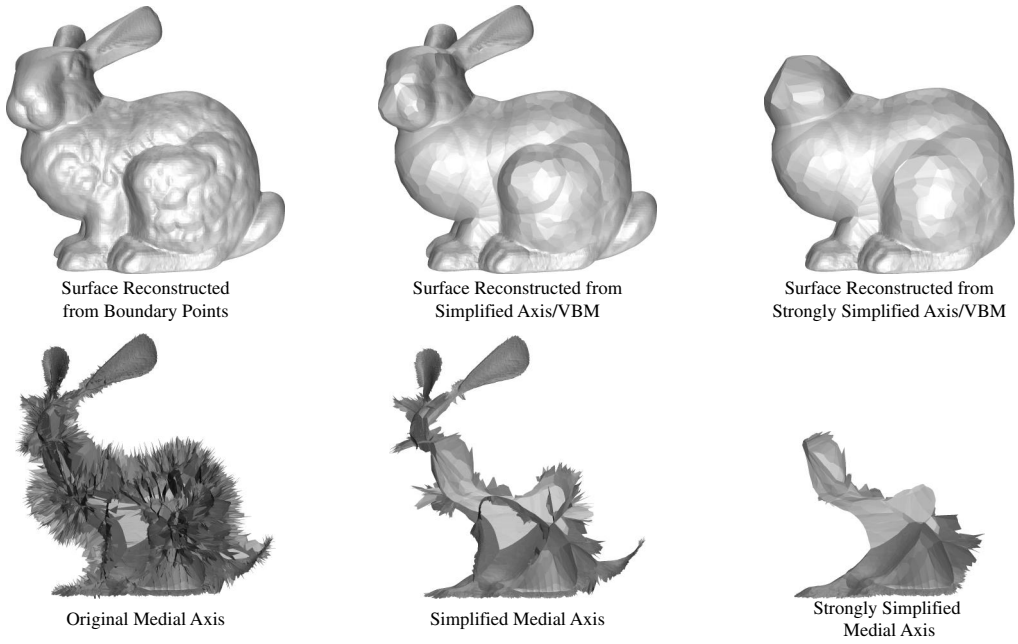


Figure 7.10: Bunny model at three levels of detail

However, this example also shows a limitation of the algorithm: our simple decomposition method does not always completely divide the parts in the way a human would. In this case, the feet of the bunny are not separated from the rest of the body, and cannot be thresholded out. Consequently, the feet are only mildly simplified compared to the rest of the object. In such cases, more sophisticated *mesh segmentation* techniques (*e.g.,* [172]) should prove useful for further decomposition of the axis.

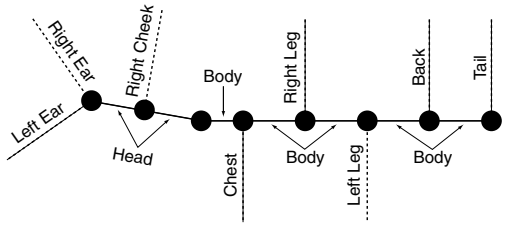


Figure 7.11: Topology graph of simplified bunny model (middle bunny of Figure 7.10)

The Tweety model and its medial axis are shown in Figures 7.12a and 7.12b, respectively. This example illustrates how our algorithm can be used to remove small noise-type artifacts. Figure 7.12d shows the medial axis after five iterations of pruning ($3 \times t_t, t_t = 5$, $2 \times t_v, t_v = 1.0\%$, where t_t is the triangle count threshold, and t_v is the volume threshold). The simplified axis is clearly much cleaner, and more useful for applications such as shape matching. Figures 7.12c and 7.12e show a closeup of the original model and the simplified model, respectively. The simplified model is clean of the small bumps seen on the back and leg of the original model.

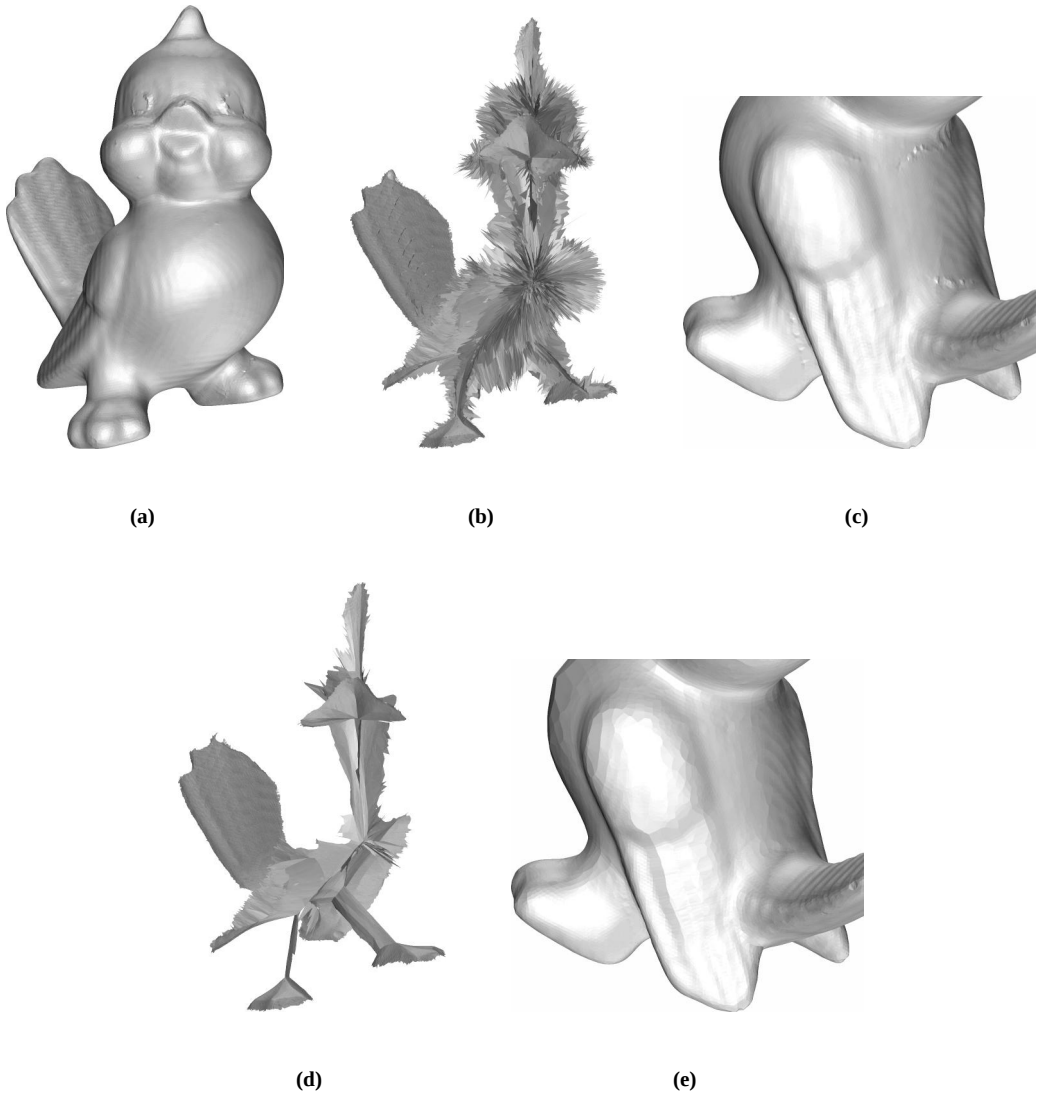


Figure 7.12: (a) Tweety model (b) Medial axis (c) Back of Tweety model, with obvious noise artifacts on the back and leg (d) Simplified medial axis (e) Back of simplified model, with noise clearly reduced

The Max Planck model, shown in Figure 7.13a, is given as an example of how an object can be edited by manually specifying parts of the medial axis to be removed. After four pruning iterations ($2 \times t_t, t_t = 5, 2 \times t_v, t_v = 1.0\%$), we have a relatively clean axis to work with (Figure 7.13c). We can manually select the left ear and its stump by simply specifying a single triangle in each part. The medial axis without the left ear is shown in Figure 7.13d and the resulting surface is shown in Figure 7.13e. The ear is removed without appreciable distortion to the surrounding area.

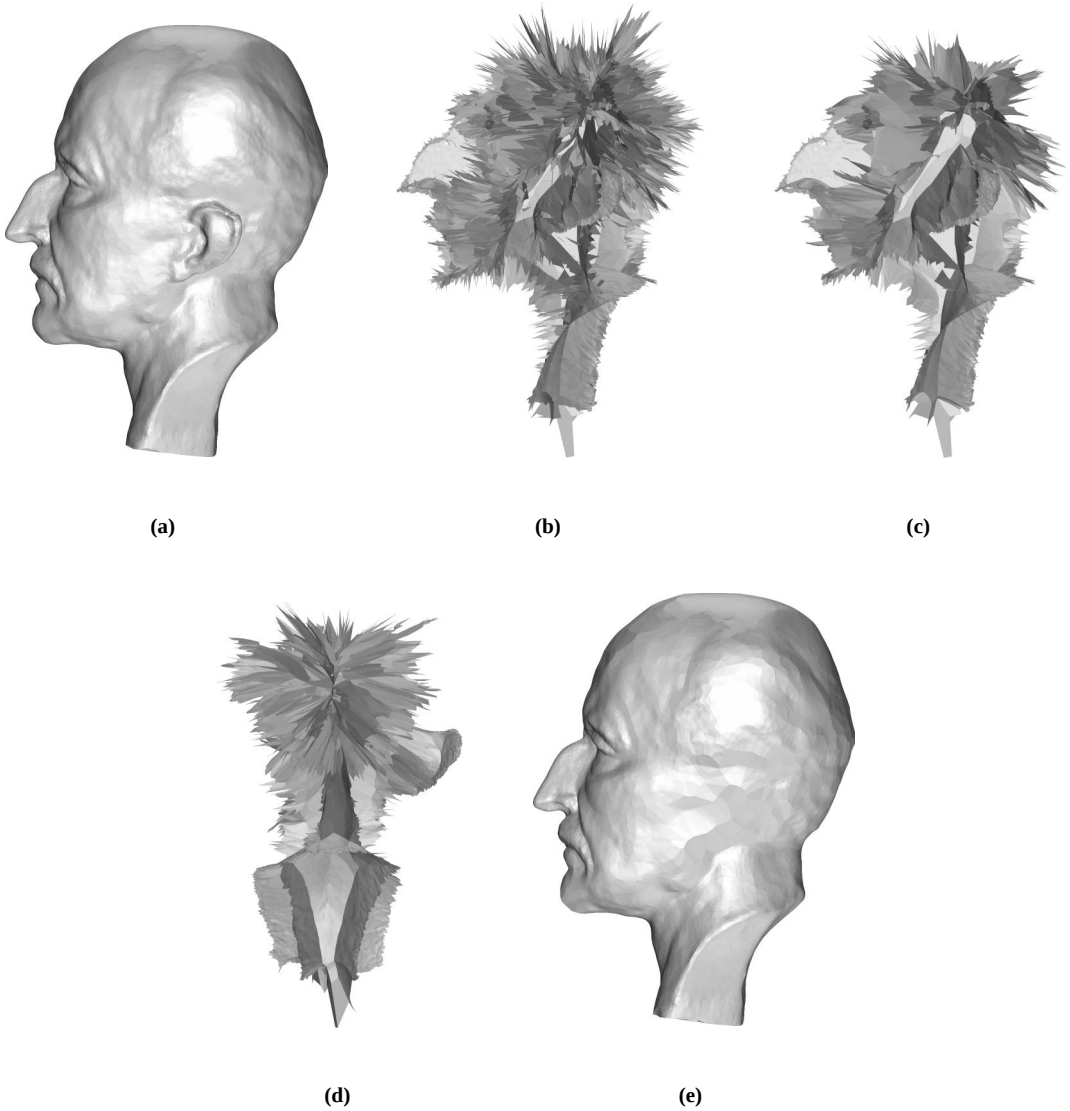
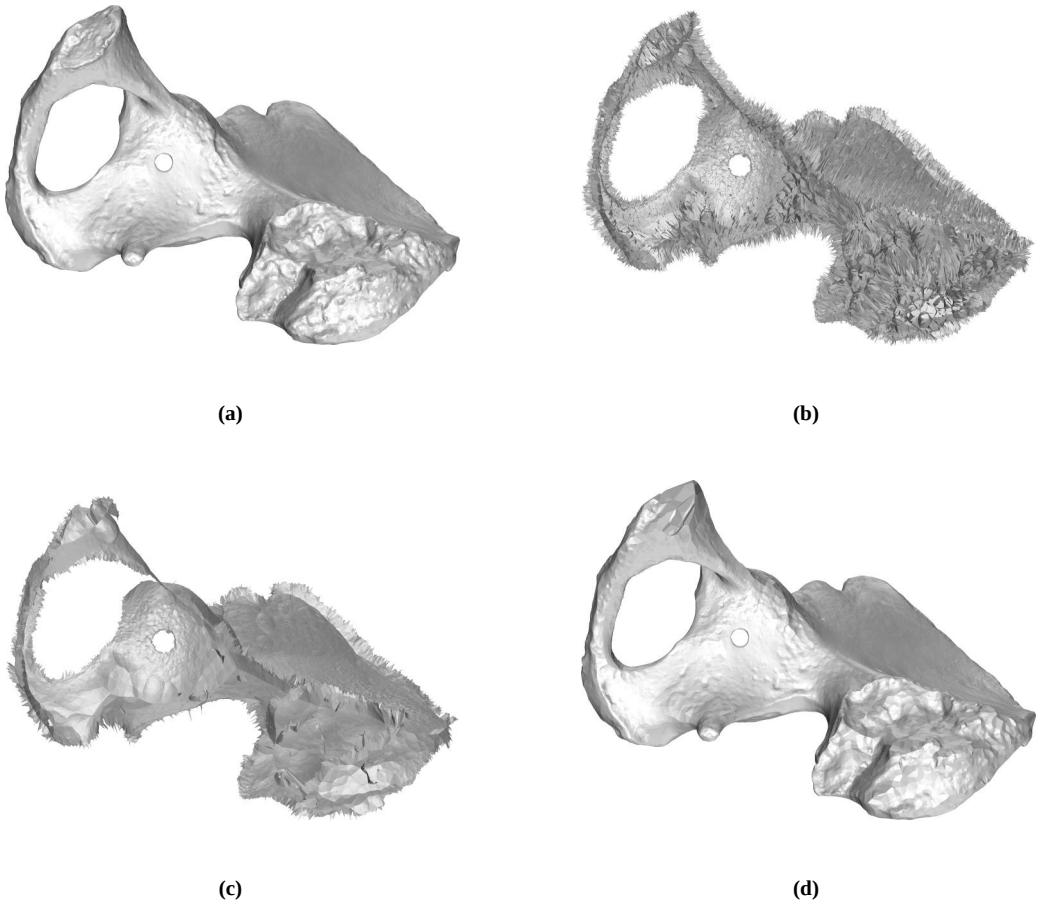


Figure 7.13: (a) Max Planck model (b) Medial axis (c) Simplified medial axis (d) Simplified medial axis, ear removed (e) Max Planck model, ear removed

The hip bone model (Figure 7.14a) is an example of where topology preservation becomes very useful. Figure 7.14b shows the original medial axis, which is full of small details. Figure 7.14c shows the medial axis after six pruning iterations ($3 \times t_t, t_t = 5$, $3 \times t_v, t_v = 5.0\%$). The algorithm greatly reduces the amount of small details, but is able to preserve the narrow arch in the axis. Many other existing pruning methods would break or disintegrate this loop, particular where it is thin. The reconstructed simplified object with the narrow loop clearly intact is shown in Figure 7.14d.



(a) **(b)**

(c) **(d)**

Figure 7.14: (a) Hip bone model (b) Medial axis (c) Simplified medial axis (d) Simplified hip bone model, with narrow loop preserved

7.8 Summary

In this chapter, we have presented an algorithm for simplifying 3D shapes by pruning their medial axes and associated VBMs. The approach is particularly novel in that the axis and VBM are decomposed into parts before simplification. We showed that this feature-based approach has a number of advantages over other existing techniques. Our results demonstrate that the algorithm is able to greatly reduce the amount of detail in an object without negatively affecting the remaining components. The power crust method is used to reconstruct a polygonal surface from the filtered VBM.

7.9 Observations

- The medial axis computed using our version of Amenta and Kolluri's algorithm can be effectively used to decompose a VBM into parts for tasks such as shape simplification. However, as mentioned in the Results section, even though our method of simply using the number of triangle neighbours to divide the axis into parts works well in general, there are cases that warrant a more advanced parts decomposition technique.
- The connectivity of the medial axis parts can be used to efficiently preserve the topology of the VBM during processing.
- The 3D medial axis makes it easy to manually specify parts to be removed, compared to using only the VBM.
- The prevention of the disintegration effect shown in Figure 7.1 is a positive indication that the use of the medial axis can keep groups of balls connected during processing.
- We have assumed that the input data is adequately sampled. While this is usually the case for data sources such as laser scanners, other sources of data may result in undersampled point sets. More research in the detection of undersampling, based on previous work (*e.g.*, [6, 45]), should be done to improve the robustness of the algorithm.
- Using the balls of the VBM instead of the Delaunay tetrahedra to estimate the volume of parts may result in greater accuracy. Such an approach can make use of previous work on computing properties of unions of balls (*e.g.*, [88]).

Chapter 8

Surface Reconstruction

8.1 Motivation

Generating a polygonal surface from a given union of balls can be very useful. For example, if a VBM is used in a segmentation application, such as that in Chapter 5, a reconstructed boundary may be required as the final output. Surface models are often desirable even if they are only needed for display purposes, because the balls are sometimes found to be visually distracting. Although a sufficiently sampled VBM is accurate in terms of approximation error, a polygonal surface model typically "looks better" to a human viewer. A common criticism of Ranjan's work is the lumpy appearance of the models [128]. In addition, a high quality rendering of a union of balls typically requires a large number of approximating polygons. For example, Table 8.1 shows the number of polygons used by Geomview, a popular polygonal model viewer, to render a single ball at different mesh resolutions. These numbers show that rendering a large union of balls at interactive rates would take substantial processing power.

Patch Dicing	Number of Polygons
10 (default)	648
20 (good onscreen quality)	2888
30 (good print quality)	6728

Table 8.1: Number of polygons used by Geomview to render a single ball

In this chapter, we discuss two methods for constructing a polygonal surface from a VBM. The first, called the power crust, is used in our 3D shape simplification algorithm, as described in Chapter 7. The second is a new reconstruction method [157] that we have developed.

8.2 The Power Crust

The main idea of the power crust algorithm is, given a set of boundary points, to first approximate the medial axis transform of the object, then use it to compute an interpolating surface. Alternatively, the method can be seen as computing the interface between two unions of balls, one approximating the interior medial balls (VBM) and the other approximating the exterior medial balls. The algorithm in Chapter 7 demonstrates how we can remove shape features by removing the corresponding medial balls.

The main steps for computing a power crust are (Figure 8.1 shows a 2D example):

1. Compute the Voronoi diagram of the sample points (Section 3.1.1).
2. Compute the poles of each sample point (Section 3.1.2).
3. Compute the power diagram of the poles (Section 3.1.1).
4. Classify the poles as inside or outside (Section 3.1.2).
5. Output the power diagram faces separating the cells of inside and outside poles as the power crust.

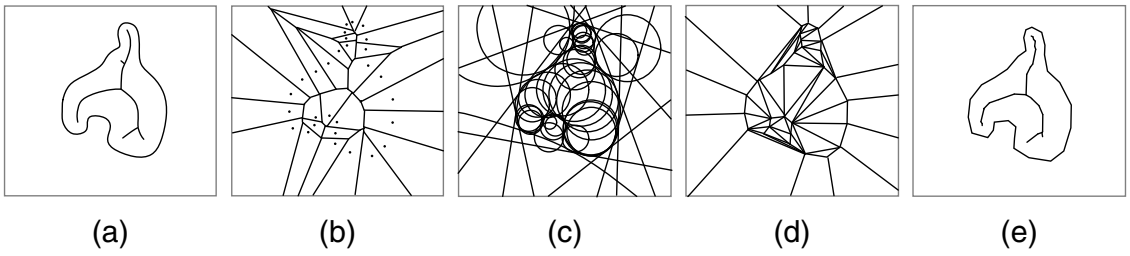


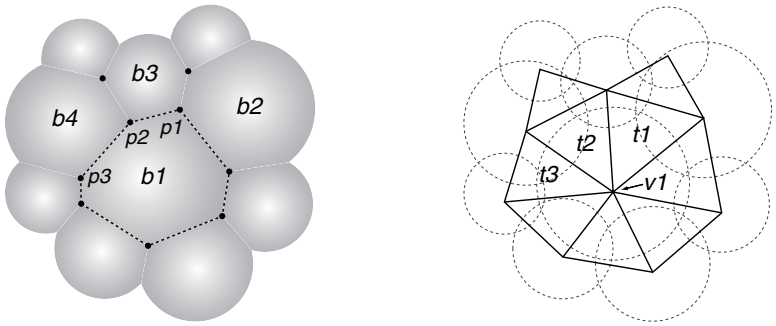
Figure 8.1: Power crust algorithm in 2D (figure adapted from [8]) (a) An object with its medial axis (b) The Voronoi diagram of a set of sample points on the boundary (c) The inner and outer polar balls (the outer polar balls with centres at infinity are halfspaces) (d) The power diagram cells of the poles (e) The power crust and approximate medial axis

8.3 Surface Reconstruction Using Singular Points

Although the power crust algorithm is a robust performer, it requires both the inner and outer polar balls to be present in order to compute a surface. In addition, it produces a large number of faces relative to the number of balls present. We need a method that can produce a surface given only the interior balls, and we would like the number of polygons created to be on the same order as the number of balls. In this section, we present a new algorithm for computing a polygonal surface from a union of balls. The focus is on developing a relatively simple and efficient method.

As mentioned in Section 7.3.1, any location on the surface of a union of balls where three or more balls intersect is called a singular point. In a reasonably dense union of balls, such as a typical VBM, there are many singular points.

Figure 8.2a shows a union of balls and some of its singular points. Our algorithm connects these points to form a polygonal surface and incorporates a method for dealing with undersampled areas in which the balls do not intersect.



(a) **(b)**

Figure 8.2: (a) A union of balls and some of its singular points (b) The dual shape of the union of balls

Our algorithm uses the weighted Delaunay triangulation of the union of balls to compute the surface. Given adequate sampling, the subcomplex consisting of the interior Delaunay triangles has been proven to be homotopy equivalent to the medial axis of the union of balls and therefore accurately reflects the topology of the shape [8]. We take advantage of this property to compute surfaces that are topologically correct.

8.3.1 Related Work

An algorithm for the *skinning* of unions of balls has been developed by Edelsbrunner *et al.* [48, 49]. The skins are typically composed of parts of the balls connected by hyperbolic and spherical patches to make the surface tangent continuous. The skin is then adaptively triangulated to form a polygonal surface. Kruithof and Vegter [80] extend the method to approximate C^2 surfaces. There are two primary drawbacks to the skinning approach. First, even a small number of balls can generate a complicated skin. Second, there are always concave patches between the balls, sometimes resulting in a bumpy appearance. Our method is designed to be considerably more lightweight, and is more suitable for use in applications where speed is a concern.

8.3.2 Algorithm Overview

Our algorithm consists of three main steps:

1. Compute the weighted Delaunay triangulation (Section 3.1.1), and discard all triangles that are outside of the union of balls. The resulting complex is called the *dual shape* [47] of the union of balls.

2. For all regular and singular triangles (defined in the next section) in the dual shape, compute the singular points, except for duplicates.
3. Compute faces from the singular points using the connectivity of the dual shape.

The main idea of the method is to traverse the hull of the dual shape in an ordered fashion, connecting the singular points on the way. Figure 8.3 illustrates how the method works in 2D (*i.e.*, computing a polyline border from a union of disks). In this example, the s's are the regular and singular simplices of the dual shape, the p's are the singular points, and the dotted line is the resulting approximating boundary. The idea is that as we traverse the simplices in order along the hull (s1 → s2 → s3 → s4 → ...), we connect the corresponding singular points (p1 to p2 to p3 to p4, *etc.*).

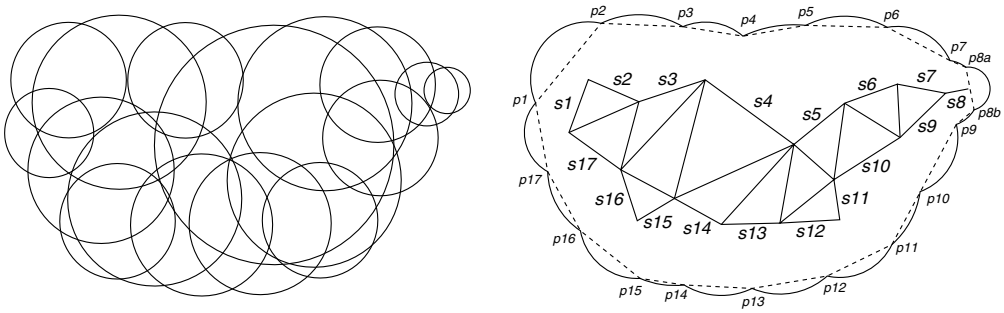


Figure 8.3: Connecting the singular points of a union of disks to form a boundary, represented by the dotted line

8.3.3 Singular Point Computation

Each triangle in the dual shape can be classified as one of three types: *interior*, *regular*, or *singular*. This classification is based on the number of tetrahedra to which the triangle belongs. An interior triangle is the interface between two neighbouring tetrahedra, a regular triangle belongs to only one tetrahedron, and a singular triangle is not part of a tetrahedron. For most data sets, the majority of triangles are faces of tetrahedra, and therefore are either regular or interior triangles. Each type of triangle produces a different number of singular points. An interior triangle produces no singular points, a regular triangle produces one singular point, and a singular triangle produces two singular points.

Given a simplex of dimension one or greater in the triangulation, the *orthocentre* of the simplex is the centre of the smallest ball that is orthogonal, as defined by the power distance (Section 3.1.1), to each ball of the simplex. For a given triangle, we compute its singular point(s) by first locating the orthocentre. The intersection points lie on a line orthogonal to the simplex and going through the orthocentre. Once we find the orthocentre, we only need to compute the correct distance to find the singular point(s). Figure 8.4 shows the 2D analog; the simplex s (an edge in 2D, a triangle in 3D) produces two potential singular points, each a distance d from the orthocentre.

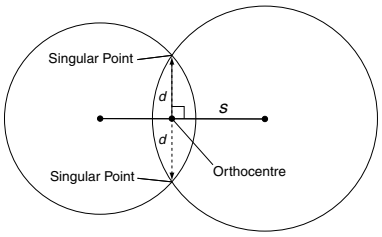


Figure 8.4: Computation of the singular points of the simplex s

It is important to note that in order to avoid degenerate faces, we need to prevent duplicate singular points from being computed. Multiple identical singular points are often produced when four or more balls intersect at the same point, as is frequently the case when the union of balls is a set of Voronoi balls. This issue is discussed in Section 7.4, with an example shown in Figure 7.4.

8.3.4 Polygon Computation

After the singular points have been computed, they must be connected properly so that the resulting surface has the correct topology and well-formed faces. For each vertex in the dual shape, we traverse the incident faces on the hull and connect the singular points in order around the vertex. Figure 8.2b shows a projection of the hull of the dual shape of the union of balls in Figure 8.2a. Traversing around v1 (t1 → t2 → t3 → ...) results in connecting p1 to p2 to p3, *etc.* in Figure 8.2a. The final result of going around this vertex is the face shown by the dotted line in Figure 8.2a.

This method of traversing around the vertex is equivalent to taking the ball centred at the vertex, and using the surface arcs formed by the ball intersecting with its neighbours as "paths" to find the appropriate sequence for the singular points. For example, the path from p1 to p2 in Figure 8.2a is determined by the intersection between the balls b1 and b3. This duality is apparent if we consider the facts that the arcs lie on the Voronoi walls of the power diagram of the balls, and the singular points lie on the intersections between the Voronoi walls.

While traversing the triangulation, there are two primary issues that need to be addressed. The first is that if a triangle has more than one neighbour on a given edge, we need a method for determining which neighbour to use in order to stay on the proper side of the hull. The second is that when we encounter a singular triangle, we need to decide which of the two singular points is the proper one to use next.

Finding the Right Neighbour

Figure 8.5a illustrates a scenario in which we need to decide which neighbour of a triangle we should proceed to next. In this case, singular point p1, computed from triangle t1, has just been added to the current face. We need to determine which of t2, t3, or t4 is the correct triangle. Making a wrong choice (either t3 or t4) would cause the traversal to go

right through the hull. To find the right neighbour (t2), we form a triangle t1′ using the given edge and the current singular point (p1). The first triangle encountered when rotating in the direction from t1 to t1′ is the correct neighbour.

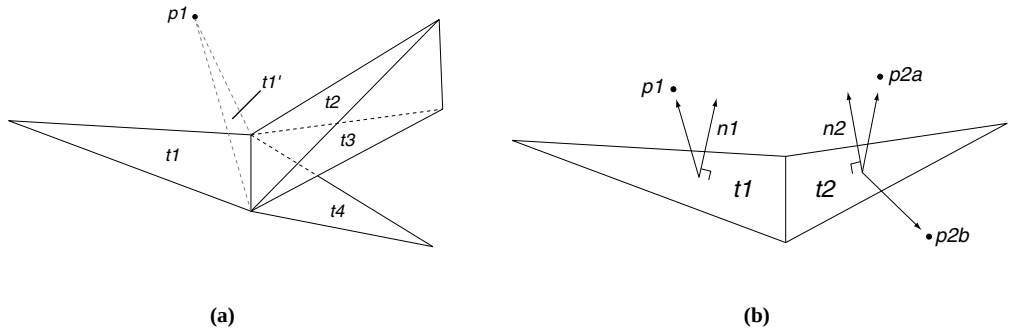


Figure 8.5: (a) Using the direction from t1 to t1′ to find the right neighbour (t2) (b) Using the normal vectors to find the right singular point

Finding the Right Singular Point

When we reach a singular triangle, we need to determine which of the two singular points is the correct one to use. Choosing the wrong one would cause the constructed face to cut across the interior of the shape. In order to find the right singular point, we compute a number of vectors to determine if choosing a particular singular point would cause a flip in orientation to the other side of the hull, relative to the singular point used for the previous triangle.

Figure 8.5b illustrates the vectors used. The normal vectors (n1, n2) for the previous (t1) and current (t2) triangles are computed. In addition, a vector originating from the centroid of the previous triangle pointing to the previous singular point (p1) is computed. Similarly, two vectors directed toward the two singular points of the current triangle (p2a, p2b) are derived. The sign of the dot product between the normal vector and the singular point vector of the previous triangle t1 gives us a reference to determine which of p2a or p2b should be used to preserve the current orientation. In this case p2a is the correct one.

8.3.5 Undersampled Areas

For densely sampled VBMs, such as those computed from laser scans, undersampling is rarely a problem. However, in some data sets, there are areas in which the balls are close enough for the computed Delaunay triangles to be largely inside the shape, but the balls do not actually intersect. This happens most frequently when one ball intersects two other ones, and the other two are close to each other but do not actually intersect. Our method of computing the singular points using the orthocentre is beneficial in such cases, because the orthocentre can be computed without an

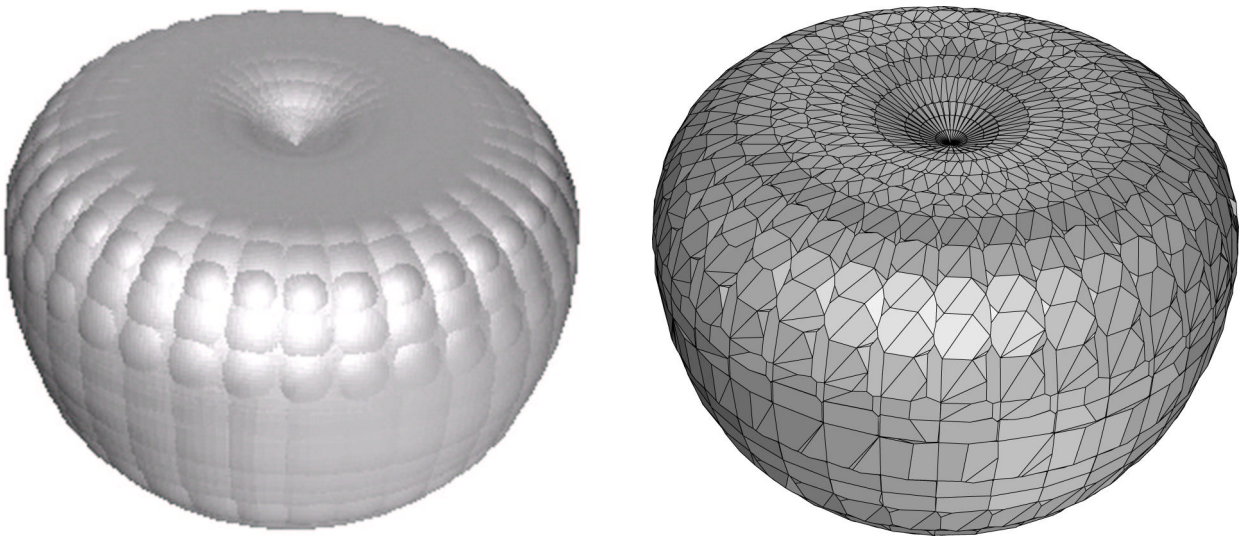
intersection. This is in contrast to using, for example, great circles for determining intersection points. To compute a "fake" singular point to add to the surface, we simply estimate an appropriate orthogonal offset from the orthocentre by averaging the radii of the three balls connected in the triangle.

8.3.6 Results

To demonstrate the effectiveness of our algorithm, we show several examples of varying shape complexity, topology, and sampling density in Figures 8.6 to 8.9. All of the unions of balls are VBMs computed using Amenta's polar ball method (Section 3.1.2). Table 8.2 shows the processing times to compute the surfaces. The number of balls and faces are also shown for each case. A Pentium 4 processor running at 2.0 Ghz is used. Our implementation makes extensive use of the CGAL and LEDA libraries.

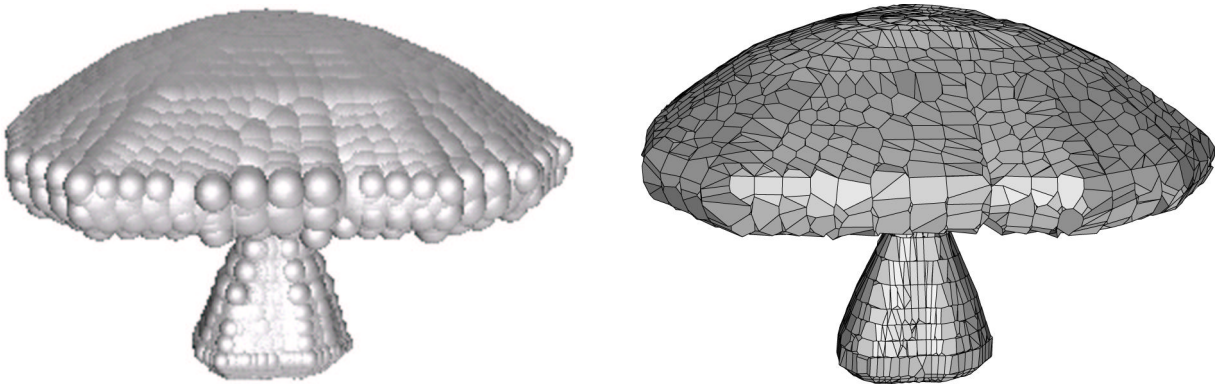
Model	Balls	Time (sec.)	Faces
Apple	3095	4.6	4179
Mushroom	3609	5.0	4963
Heart	3405	4.6	3900
Torus	5613	6.2	7154

Table 8.2: Processing times for polygonal surface reconstruction from VBMs



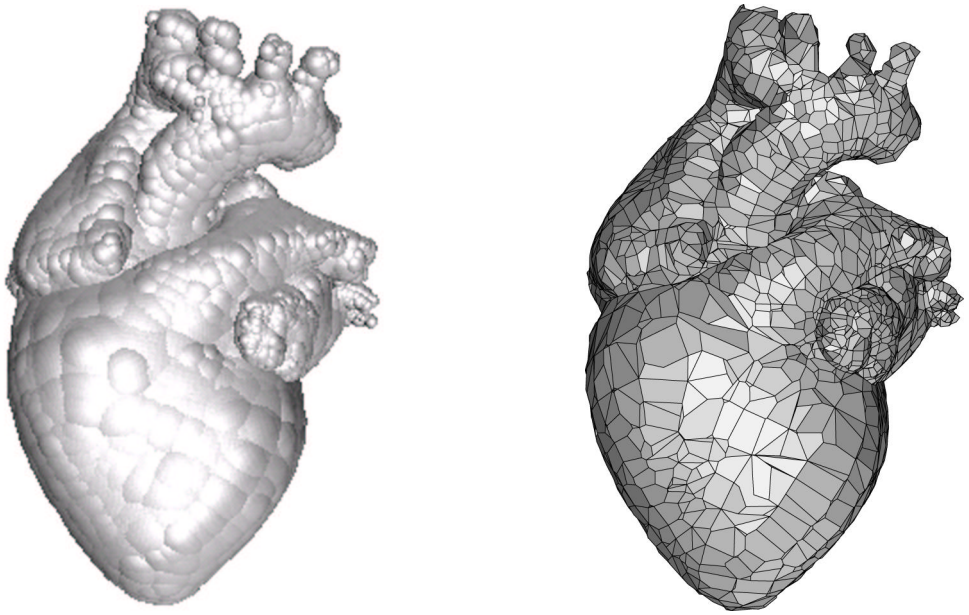
(a) **(b)**

Figure 8.6: (a) VBM of an apple (3095 balls) (b) Surface reconstructed from the VBM of an apple (4179 faces)



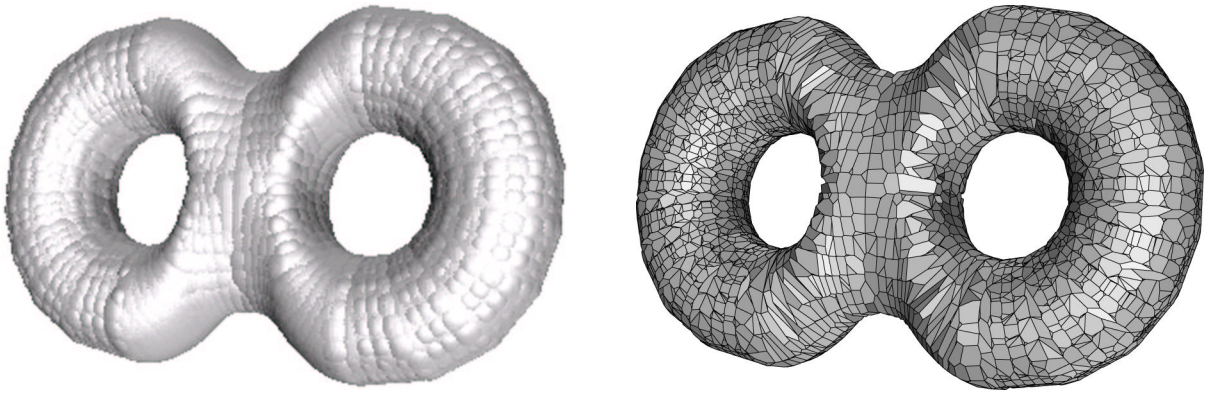
(a) **(b)**

Figure 8.7: (a) VBM of a mushroom (3609 balls) (b) Surface reconstructed from the VBM of a mushroom (4963 faces)



(a) **(b)**

Figure 8.8: (a) VBM of a heart (3405 balls) (b) Surface reconstructed from the VBM of a heart (3900 faces)



(a) **(b)**

Figure 8.9: (a) VBM of a two-holed torus (5613 balls) (b) Surface reconstructed from the VBM of a two-holed torus (7154 faces)

8.4 Summary

In this chapter, we have briefly reviewed the power crust algorithm, as well as presented our own novel method for constructing an approximating boundary of a union of balls. We have shown the results of applying our algorithm to several VBMs of varying shape complexity and topology.

8.5 Observations

- Our algorithm is robust, and has not failed for the many data sets that we have tried. This is due to the fact that the algorithm is simple and based on well-established geometric theorems and constructions.
- Our method works for unions of balls in general, not just VBMs, because there are no assumptions made in our algorithm with regards to the construction method of the ball model.
- While our algorithm produces surfaces that the average viewer would say are faithful reconstructions, geometrically speaking they are only approximations. A derivation of the error bounds would be informative, but judging by the methods of construction we can probably assume that the amount of error is greater than in the skins approach. In addition, while we conjecture that the reconstructed surface converges to the envelop of the union of balls as the sampling density tends to infinity, this has not been rigorously proven.

- The polygons produced by our algorithm are non-planar in general. Depending on the application, this may be problematic for subsequent processing. A conversion step to retessellate the surface into planar faces (*e.g.*, triangles) would be useful.
- We note that the singular points can be used as input into the power crust algorithm to produce an interpolating surface. In our experiments, this approach tends to create smoother surfaces. However, as noted earlier, the power crust produces a large number of faces for the number of balls present.

Chapter 9

Conclusions and Future Work

This thesis shows that Voronoi Ball Models possess the characteristics to make them broadly-applicable in computer graphics, scientific visualization and computer vision. Five representative applications are used to demonstrate the capabilities and limitations of the VBM. The full potential of VBMs is too great to be completely explored in one thesis, but, as documented in this dissertation, some important steps have been taken. In this last chapter, we summarize the major conclusions and some directions for future work.

9.1 Summary of Results and Observations

We summarize the applications developed for this thesis, and our main results and observations related to the key properties that we identified in Chapter 1. Some of the observations have been made from the results of more than one application, but are only listed once for conciseness.

9.1.1 Image Matching and Interpolation

We presented an algorithm for image interpolation and rigid registration using VBMs. In this approach, VBMs are used to represent the images and a similarity measure is applied to form feature correspondences between the VBMs to be matched. A number of interpolated VBMs are then formed, from which the intermediate images are computed. The major results and observations are:

- The method is able to form correspondences between similar image features, even with no explicit feature extraction and little or no manual intervention.
- Clustered VBMs are stable with respect to changes in the positions of the sample boundary points.

- The correspondences formed by the matching method are stable with respect to changes in the user-determined parameter values (matching weights).
- The appearance of artifacts in the interpolated images indicates that connectivity information between primitives would be a valuable addition to the representation. This is a motivating factor for our research done on exploring the use of the medial axis with VBMs.

9.1.2 Shape Model and Threshold Extraction

We discussed our largely automatic algorithm for extracting thresholds and VDMs of significant objects from greyscale images. The method entails using a VDM-based similarity measure to quantify the shape changes of the objects in an image as the intensity threshold is varied. Plots of the shape gradient are then used to detect the intensity ranges in which significant objects lie. The major results and observations are:

- The extracted thresholds and VDMs are visually accurate representations of the shape information contained in the test images.
- Our proposed modification to the similarity measure enhances its stability by making it independent of translation and rotation, and less dependent on scaling.
- When computing the shape distance between two VDMs, the unmatched disks can be used to reveal large differences not evident in the matches.
- The similarity measure can be used to compare objects of different shape complexity and topology. This flexibility is particularly advantageous in exploratory applications.
- Reconstruction of the object boundaries would be a useful step to add. This points to the need for surface reconstruction methods such as those discussed in Chapter 8.

9.1.3 Two-Dimensional Shape Simplification

We developed an algorithm that uses the VDM and the medial axis in combination to remove noise-type artifacts from the borders of 2D objects. The method prunes the spurious branches of the axis without sacrificing the fine features that are usually lost with other techniques. The major results and observations are:

- The 2D medial axis is an efficient representation for preserving the topology of a VDM during processing.

- The branch nodes of the 2D medial axis make effective partitioning points for localizing shape information defined on the VDM.
- Adding the 2D medial axis makes visual discernment between noise and significant features easier to perform compared to using only the VDM.

9.1.4 Three-Dimensional Shape Simplification

We presented a method for simplifying 3D shapes by using the medial axis and VBM together. The algorithm computes a VBM and medial axis of the object, uses the medial axis to decompose the VBM into distinct components, then iteratively removes layers of the medial axis and VBM until the desired amount of simplification is achieved. A surface of the object is then reconstructed from the VBM using the power crust algorithm. The major results and observations are:

- The convergence of polar balls to the medial axis transform provides a way to filter out many balls resulting from discretization, thereby enhancing the stability of the VBM.
- The 3D medial axis can be efficiently used to represent and preserve the topology of a VBM during processing.
- The 3D medial axis can be used to partition a VBM into meaningful components.
- The 3D medial axis and our parts-decomposition scheme make the manual selection of features for removal easier to perform than using only the VBM.

9.1.5 Surface Reconstruction

We developed an efficient algorithm for computing a polygonal surface from a union of balls. Our lightweight approach connects the singular points of a union of balls to construct surfaces that are topologically correct. The resulting surfaces have a small number of polygons compared to those produced by other current techniques. The major results and observations are:

- Our surface reconstruction algorithm is robust and can handle objects of varying shape complexity, topology and sampling density.
- The technique works for unions of balls in general, not just VBMs. For example, the method can be used to compute surfaces from clustered VBMs.
- The amount of processing time to compute a surface for each model is small relative to the gain in display efficiency.

9.2 Conclusions

In this section, we summarize our major conclusions regarding the suitability of VBMs for shape-driven applications. As described below, most of the core shape operations that we are focusing on (*i.e.*, extraction, simplification, matching, interpolation, manipulation, and surface reconstruction) can be performed effectively with VBMs.

- VBMs make stable, accurate, and efficient shape representations of all objects for which a dense sampling of the entire boundary can be computed. They are most suitable for use with data from which the boundary of each object can be independently extracted.

- The VBM-based similarity measure forms accurate shape correspondences between 3D object features. The matches have been shown to be stable with respect to changes in the positions of the sample boundary points, and to changes in the measure's parameter values. The method has sufficient capabilities to perform well for matching tasks such as (rigid/non-rigid) registration.

- VBMs can be used for shape interpolation by using a similarity measure to establish correspondences between balls, then interpolating the positions and sizes of matched balls to derive intermediate shapes. In our current interpolation method, the balls are interpolated linearly, and there is no connectivity between primitives, so there is little control over the intermediate shapes. As a result, the approach is feasible for applications such as aesthetic morphing, but the interpolation control may require further development for more demanding tasks.

- The VDM-based 2D similarity measure has sufficient discriminatory power to quantify a large range of shape differences accurately. The makes the method suitable for applications such as shape database queries or template matching. Some cases can result in unmatched primitives, which can be a significant source of information that can be used to complement the data from the matches. More research in the use of unmatched primitives is likely to further increase the accuracy of the method.

- The VDM-based 2D similarity measure effectively quantifies shape differences between objects that vary significantly in topology. The flexibility of the similarity measure makes it well-suited for exploratory applications such as shape extraction from image data, particularly where manual intervention is impractical.

- The VBM can be used to accurately approximate the medial axis transform, as well as simplify and stabilize the medial axis by removing components associated with minor features. This makes the VBM applicable to the large group of shape processing algorithms designed for the medial axis. For example, a number of animation methods use a skeletal structure as the primary representation. In addition, the VBM can use the medial axis to efficiently preserve the topology of an object during shape processing.

- The VBM can be used to partition an object into parts for further processing by using medial segments to link balls together into groups. We have proven this approach to be particularly effective for shape simplification via feature size thresholding or manual specification of components for removal. Parts-decomposition is used in many object recognition systems, because it is a strategy known to improve robustness in the face of occlusion. Therefore, partitioned VBMs can likely be used for recognition in cluttered environments.

- Using the VBM and the medial axis together theoretically gives the VBM greater visual relevance. Our experiments provide evidence to support this statement. For example, our 2D simplification algorithm shows that Leyton's Symmetry-Curvature Duality Theorem, which is based on psychological evidence, can be used to effectively distinguish between the noise and significant features of an object.

- The VBM is a good intermediate shape representation for reconstructing polygonal surfaces from boundary point samples. The VBM can be used to stabilize the shape by filtering out spurious components before computation of the surface. Amenta's surface reconstruction method is accurate in that it results in a surface that interpolates the sample points, but requires both inner and outer polar balls. We developed a lightweight algorithm that, while not as accurate, requires only the inner polar balls, and produces many fewer polygons. Both methods result in surfaces with provably correct topology.

- Currently, VBMs are most useful for applications in which the final results are either quantitative, or in the form of balls, points, or polygons. The reason is that the coupling of the VBM with other types of data (*e.g.*, voxels) has not been thoroughly researched. In addition, for some applications that are not purely shape-driven (*e.g.*, image matching), the use of other information to complement shape information would likely enhance accuracy and robustness.

9.3 Future Work

The results of this thesis motivate further work in many directions. Foremost is the development of other applications using VBMs, examples of which include non-rigid registration, 3D shape extraction, object recognition, interactive modelling, and animation. We outline some of other topics for future work in this section.

9.3.1 Representation Properties

- We would like to compare the approximation properties of Amenta's method for stabilizing VBMs (convergence to the medial axis) with Ranjan and Fournier's approach (clustering).

- We would like to analyze the approximation properties of Amenta's polar ball VBMs that have been further simplified by clustering. In particular, we would like to derive tight error bounds and determine whether clustering preserves the convergence property.
- We would like to further examine the issue of sampling. So far, we have only used densely sampled data. The detection and handling of undersampled areas would make our algorithms more robust.
- We are interested in doing more work on using the VBM, medial axis and/or other representations together. For example, "wrapping" a deformable model around a VBM may be useful for shape extraction applications. An example of an approach that combines medial-based and deformable models is presented by Joshi *et al.* [74].

9.3.2 Similarity Measure

- A more rigorous study on how the matching weights affect the correspondences formed by the similarity measure should be done.
- Different methods of handling unmatched balls and incorporating the resulting information into the similarity measure should be investigated.
- We would like to experiment more with our modifications to the Ranjan-Fournier measure. One of the primary goals is to make the measure fully independent of scaling, translation and rotation.
- We would like to incorporate the use of the medial axis into the quantification of shape differences. This approach can be used to add topological and parts-based information, and can make use of the substantial body of previous related work (*e.g.*, [145, 151]).
- We would like to further investigate multiscale processing. The results of matching at different levels of detail can be potentially combined for greater stability.

9.3.3 Validation

- Some of the validation done for this thesis has been subjective and done by visual inspection. More objective and quantitative validation methods would further strengthen our claims.
- Alternative similarity measures should be used for the validation of our matching results.
- More direct comparisons with other shape models using the same applications and data should be done to evaluate the relative advantages and limitations of VBMs.

Bibliography

Note: each entry is followed by the page(s) on which the reference is cited.

[1] R. Acharya and R.P. Menon. A review of biomedical image segmentation techniques. In A. Singh, D. Goldgof, and D. Terzopoulos, editors, *Deformable Models in Medical Image Analysis*, pages 140–161. IEEE Computer Society Press, 1998. Cited on page(s): 27, 57

[2] B. Adams and P. Dutré. Interactive boolean operations on surfel-bounded solids. *Computer Graphics (Proc. SIGGRAPH 2003)*, July 2003. Cited on page(s): 27

[3] H. Alt, U. Fuchs, G. Rote, and G. Weber. Matching convex shapes with respect to the symmetric difference. In *Algorithms ESA '96, Proc. 4th Annual European Symposium on Algorithms*, pages 320–333, Barcelona, Spain, September 1996. Cited on page(s): 23

[4] H. Alt and L. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation - a survey. Technical Report 96-11, Freie Universität Berlin, Fachbereich Mathematik und Informatik, 1996. Cited on page(s): 26

[5] H. Alt, C. Knauer, and Wenk C. Matching polygonal curves with respect to the Fréchet distance. In *Proc. 18th International Symposium on Theoretical Aspects of Computer Science*, pages 63–74, 2001. Cited on page(s): 24

[6] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, 1999. Cited on page(s): 33, 34, 37, 99

[7] N. Amenta, S. Choi, and R.K. Kolluri. The power crust. In *Proceedings of the ACM Symposium on Solid Modeling and Applications*, pages 249–260, Ann Arbor, Michigan, June 2001. Cited on page(s): 13, 85, 87, 94

[8] N. Amenta, S. Choi, and R.K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19(2-3):127–153, 2001. Cited on page(s): 35, 37, 86, 87, 88, 101, 102

[9] N. Amenta and R.K. Kolluri. Accurate and efficient unions of balls. In *Proc. ACM Symposium on Computational Geometry*, pages 119–128, Hong Kong, June 2000. Cited on page(s): 17, 33, 34, 35, 37

[10] N. Amenta and R.K. Kolluri. The medial axis of a union of balls. In *Proc. Canadian Conference on Computational Geometry*, pages 111–114, Fredericton, New Brunswick, August 2000. Cited on page(s): 17, 87, 90

[11] C. Arcelli, L.P. Cordella, and G. Sanniti di Baja, editors. *Visual Form: Analysis and Recognition*. Plenum Press, 1992. Cited on page(s): 20

[12] E. Arkin, L. Chew, D. Huttenlocher, K. Kedem, and J. Mitchell. An efficient computable metric for comparing polygonal shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(3):209–215, March 1991. Cited on page(s): 5

[13] D. Attali and A. Montanvert. Semicontinuous skeletons of 2D and 3D shapes. In *Proc. International Workshop on Visual Form*, pages 32–41, Capri, 1994. World Scientific. Cited on page(s): 17, 70

[14] D. Attali and A. Montanvert. Modeling noise for a better simplification of skeletons. In *Proc. International Conference on Image Processing*, volume III, pages 13–16, Lausanne, Switzerland, 1996. Cited on page(s): 71

[15] D. Attali and A. Montanvert. Computing and simplifying 2D and 3D continuous skeletons. *Computer Vision and Image Understanding*, 67(3):161–273, 1997. Cited on page(s): 17, 84, 85, 86, 94

[16] D. Attali, G. Sanniti di Baja, and Thiel E. Pruning discrete and semicontinuous skeletons. In C. De Floriani, C. Braccini, and G. Vernazza, editors, *Lecture Notes in Computer Science, Image Analysis and Processing*, volume 974, pages 488–493. Springer-Verlag, 1995. Cited on page(s): 71

[17] F. Aurenhammer. Power diagrams: properties, algorithms, and applications. *SIAM Journal on Computing*, 16(1):78–96, 1987. Cited on page(s): 32

[18] R. Bajcsy and S. Kovačič. Multiresolution elastic matching. *Computer Vision Graphics and Image Processing*, 46:1–21, 1989. Cited on page(s): 25

[19] R. Basri, L. Costa, D. Geiger, and D. Jacobs. Determining the similarity of deformable shapes. In *IEEE Workshop on Physics-Based Modeling in Computer Vision*, pages 135–143. IEEE Press, 1995. Cited on page(s): 22

[20] R. Basri, L. Costa, D. Geiger, and D. Jacobs. Determining the similarity of deformable shapes. *Vision Research*, 38:2365–2385, 1998. Cited on page(s): 22

[21] T. Beier and S. Neely. Feature-based image metamorphosis. *Computer Graphics (Proc. SIGGRAPH '92)*, 26:35–42, 1992. Cited on page(s): 41

[22] D. Bespalov, A. Shokoufandeh, W.C. Regli, and W. Sun. Scale-space representation of 3D models and topological matching. In *Proceedings of the ACM Symposium on Solid Modeling and Applications*, Seattle, Washington, June 2003. Cited on page(s): 25

[23] I. Biederman. Human image understanding: Recent research and a theory. *Computer Vision, Graphics and Image Processing*, 32:29–73, 1985. Cited on page(s): 20

[24] I. Biederman. Recognition by components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987. Cited on page(s): 8, 20

[25] V. Blanz and T. Vetter. Face recognition based on fitting a 3D morphable model. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(9):1063–1074, September 2003. Cited on page(s): 7

[26] J. Bloomenthal. Medial-based vertex deformation. In *Proceedings of the ACM SIGGRAPH Symposium on Computer Animation*, San Antonio, Texas, July 2002. Cited on page(s): 68

[27] J. Bloomenthal, C. Bajaj, and J. Blinn, editors. *Introduction to Implicit Surfaces*. Morgan Kaufmann, 1998. Cited on page(s): 17

[28] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*. MIT Press, Cambridge, MA, 1967. Cited on page(s): 17, 58

[29] H. Blum and R. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10:167–180, 1978. Cited on page(s): 17, 58

[30] F.L. Bookstein. Shape and the information in medical images: A decade of the morphometric synthesis. *Computer Vision and Image Understanding*, 66(2):97–118, 1997. Cited on page(s): 5, 16, 21

[31] J.W. Brandt and V.R. Algazi. Continuous skeleton computation by Voronoi diagram. *CVGIP: Image Understanding*, 55(3):329–337, 1992. Cited on page(s): 17, 70, 71

[32] E. Bribiesca. Measuring 3D shape similarity using progressive transformations. *Pattern Recognition*, 29(7):1117–1129, 1996. Cited on page(s): 25

[33] L.G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, 1992. Cited on page(s): 28, 41

[34] M. Brown and D.G. Lowe. Recognising panoramas. In *Proc. International Conference on Computer Vision*, Nice, France, October 2003. Cited on page(s): 27

[35] C.A. Burbeck and S.M. Pizer. Object representation by cores: Identifying and representing primitive spatial regions. *Vision Research*, 35(13):1917–1930, 1995. Cited on page(s): 20, 68

[36] C.A. Burbeck, S.M. Pizer, B.S. Morse, D. Ariely, G.S. Zauberman, and J.P. Rolland. Linking object boundaries at scale: a common mechanism for size and shape judgments. *Vision Research*, 36(3):361–372, 1996. Cited on page(s): 20, 68

[37] D.J. Burr. Elastic matching of line drawings. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 3(6):708–713, 1981. Cited on page(s): 22

[38] R. Campbell and P. Flynn. A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding*, 81:166–210, 2001. Cited on page(s): 28

[39] C.C. Chen. Improved moment invariants for shape discrimination. *Pattern Recognition*, 26(5), 1993. Cited on page(s): 21

[40] S. Choi and N. Amenta. Delaunay triangulation programs on surface data. In *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms*, pages 135–136, 2002. Cited on page(s): 32

[41] K. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. *Computational Geometry*, 3(4):185, September 1993. Cited on page(s): 32

[42] T.F. Cootes and C.J. Taylor. Statistical models of appearance for computer vision. Unpublished report available from urlhttp://www.isbe.man.ac.uk/ bim/ (valid February 23, 2001). Cited on page(s): 21

[43] D. DeCarlo and Metaxas D. Blended deformable models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(4):443–448, 1996. Cited on page(s): 26

[44] O. Devillers. Improved incremental randomized Delaunay triangulation. In *Proc. Symposium on Computational Geometry*, pages 106–115, 1998. Cited on page(s): 31, 32

[45] Tamal K. Dey and J. Giesen. Detecting undersampling in surface reconstruction. In *Proceedings of the 17th Annual Symposium on Computation Geometry*, pages 257–263, 2001. Cited on page(s): 99

[46] Tamal K. Dey and Wulue Zhao. Approximate medial axis as a Voronoi subcomplex. In *Proceedings of the ACM Symposium on Solid Modeling and Applications*, Saarbrucken, Germany, June 2002. Cited on page(s): 85, 86

[47] H. Edelsbrunner. The union of balls and its dual shape. In *Proc. 9th Annual ACM Symposium on Computational Geometry*, pages 218–231, 1993. Cited on page(s): 25, 34, 102

[48] H. Edelsbrunner. Deformable smooth surface design. *Discrete & Computational Geometry*, 21:87–115, 1999. Cited on page(s): 102

[49] H. Edelsbrunner and A. Üngör. Relaxed scheduling in dynamic skin triangulation. In *Proc. Japan Conference on Discrete and Computational Geometry*, Tokyo, Japan, December 2002. Springer-Verlag. Cited on page(s): 102

[50] D. Forsyth, J. Malik, M. Fleck, and J. Ponce. Primitives, perceptual organization and object recognition. Technical report, Computer Science Division, University of California at Berkeley, Berkeley, CA, February 1997. Cited on page(s): 28

[51] S. Fortune. Voronoi diagrams and Delaunay triangulations. In D.-Z. Du and F. Hwang, editors, *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*. World Scientific, 1992. Cited on page(s): 3

[52] M. Foskey, M.C. Lin, and D. Manocha. Efficient computation of a simplified medial axis. In *Proceedings of the ACM Symposium on Solid Modeling and Applications*, Seattle, Washington, June 2003. Cited on page(s): 86

[53] S.F. Frisken, R.N. Perry, A.P. Rockwood, and T.R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. *Computer Graphics (Proc. SIGGRAPH 2000)*, pages 249–254, 2000. Cited on page(s): 16, 27

[54] D. Fry. *Shape Recognition using Metrics on the Space of Shapes*. PhD thesis, Department of Mathematics, Harvard University, 1993. Cited on page(s): 23

[55] N. Gagvani and D. Silver. Parameter-controlled volume thinning. *Graphical Models and Image Processing*, 61(3), May 1999. Cited on page(s): 86

[56] N. Gagvani and D. Silver. Shape-based volumetric collision detection. In *Proc. IEEE Volume Visualization and Graphics Symposium*, Salt Lake City, Utah, October 2000. Cited on page(s): 8

[57] N. Gagvani and D. Silver. Animating volumetric models. *Graphical Models*, March 2002. Cited on page(s): 68

[58] P. Gao and T.W. Sederberg. A work minimization approach to image morphing. *The Visual Computer*, 14(8/9):391–400, November 1998. Cited on page(s): 41

[59] J. Goldak, X. Yu, A. Knight, and L. Dong. Constructing discrete medial axis of 3D objects. *International Journal of Computational Geometry and Applications*, 1(3):327–339, 1991. Cited on page(s): 35

[60] A.V. Goldberg and R. Kennedy. An efficient cost scaling algorithm for the assignment problem. *Mathematical Programming*, 71(2):153–177, 1995. Cited on page(s): 48

[61] A. Guéziec. Locally toleranced polygonal surface simplification. *IEEE Trans. Visualization and Computer Graphics*, 5(2):178–199, June 1999. Cited on page(s): 8

[62] M. Hagedoorn and R.C. Veltkamp. Metric pattern spaces. Technical Report UU-CS-1999-03, Department of Computer Science, Utrecht University, 1999. Cited on page(s): 24

[63] J.C. Hart. Computational topology for shape modeling. In *Shape Modeling International*, pages 36–45, University of Aizu, Japan, 1999. Cited on page(s): 27

[64] A.E. Hassanien and M. Nakajima. Image morphing of the facial images transformation based on Navier elastic body splines. In *Computer Animation '98*, pages 119–125, Geneva, Switzerland, 1998. IEEE Computer Society Press. Cited on page(s): 41

[65] P. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. Technical report, Department of Computer Science, Carnegie Mellon University, 1997. Cited on page(s): 85

[66] M. Hilaga, Y. Shinagawa, T. Kohmura, and T.L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. *Computer Graphics (Proc. SIGGRAPH 2001)*, pages 203–212, August 2001. Cited on page(s): 25

[67] D.D. Hoffman and W.A. Richards. Parts of recognition. *Cognition*, 18:65–96, 1985. Cited on page(s): 8, 20

[68] C.M. Hoffmann and J.R. Rossignac. A road map to solid modeling. *IEEE Trans. Visualization and Computer Graphics*, 2(1):3–10, 1996. Cited on page(s): 16

[69] W. Hsu, J. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. *Computer Graphics (Proc. SIGGRAPH '92)*, 26(2):177–184, 1992. Cited on page(s): 27

[70] P. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996. Cited on page(s): 8

[71] D. Huttenlocher, G. Klauderman, and W. Rucklidge. Comparing images using the Hausdorff-distance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15:850–863, 1993. Cited on page(s): 38

[72] D. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proc. International Conference on Computer Vision*, pages 102–111, London, 1987. Cited on page(s): 26

[73] A.K. Jain, Y. Zhong, and S. Lakshmanan. Object matching using deformable templates. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(3):267–278, 1996. Cited on page(s): 22

[74] S. Joshi, S. Pizer, P. Fletcher, A. Thall, and G. Tracton. Multiscale 3D deformable model segmentation based on medial description. In *Proc. Information Processing in Medical Imaging*, volume 2082 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001. Cited on page(s): 115

[75] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987. Cited on page(s): 1, 17, 57

[76] M. Kazhdan, T. Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In L. Kobbelt, P. Schröder, and H. Hoppe, editors, *Proc. Eurographics Symposium on Geometry Processing*, 2003. Cited on page(s): 25

[77] J.R. Kent, W.E. Carlson, and R.E. Parent. Shape transformations for polyhedral objects. *Computer Graphics (Proc. SIGGRAPH '92)*, 26(2):47–54, 1992. Cited on page(s): 26

[78] B.B. Kimia, A.R. Tannenbaum, and S.W. Zucker. Shapes, shocks, and deformations I: the components of two-dimensional shape and the reaction-diffusion space. *International Journal of Computer Vision*, 15(3):189–224, 1995. Cited on page(s): 20

[79] J.J. Koenderink and A.J. van Doorn. The shape of smooth objects and the way contours end. *Perception*, 11:129–137, 1982. Cited on page(s): 8, 20

[80] N. Kruithof and G. Vegter. Approximation by skin surfaces. In *Proc. International Conference on Shape Modelling and Applications*, pages 86–95, Seattle, June 2003. Cited on page(s): 102

[81] F. Lazarus and A. Verroust. Three-dimensional metamorphosis: a survey. *The Visual Computer*, 14(8/9):373–389, November 1998. Cited on page(s): 26

[82] D.T. Lee. Medial axis transformation of a planar shape. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 4:363–369, 1982. Cited on page(s): 17

[83] S. Lee, K.-Y. Chwa, S.Y. Shin, and G. Wolberg. Image metamorphosis using snakes and free-form deformations. *Computer Graphics (Proc. SIGGRAPH '95)*, pages 439–448, 1995. Cited on page(s): 41

[84] T. Lee, Y.-C. Lin, L. Lin, and Y. Sun. Fast feature-based metamorphosis and operator design. *Computer Graphics Forum (Proc. Eurographics '98)*, 17(3):15–22, 1998. Cited on page(s): 41

[85] E. Lepore and Z. Pylyshyn, editors. *Rutgers University Lectures on Cognitive Science*, chapter 10: object representation and recognition, pages 172–207. Basil Blackwell Publishers, 1999. Chapter written by Dickinson, S. Cited on page(s): 28

[86] F. Leymarie and B. Kimia. Computation of the shock scaffold for unorganized point clouds in 3D. In *Proc. Conference on Computer Vision and Pattern Recognition*, volume 1, pages 821–827, Madison, Wisconsin, June 2003. Cited on page(s): 86

[87] M. Leyton. Shape and causal-history. In C. Arcelli, L.P. Cordella, and G. Sanniti di Baja, editors, *Visual Form: Analysis and Recognition*, pages 379–388. Plenum Press, 1992. Cited on page(s): 8, 20, 68, 75

[88] J. Liang, H. Edelsbrunner, P. Fu, P.V. Sudharkar, and S. Subramaniam. Analytic shape computation of macromolecules I: molecular area and volume through alpha shape. *Proteins: Structure, Function, and Genetics*, 33:1–17, 1998. Cited on page(s): 8, 99

[89] T-L. Liu and D. Geiger. Approximate tree matching and shape similarity. In *Proc. International Conference on Computer Vision*, Kerkyra (Corfu), Greece, September 1999. Cited on page(s): 7, 24

[90] X. Liu, R. Sun, S.B. Kang, and H.-Y. Shum. Directional histogram model for three-dimensional shape similarity. In *Proc. Conference on Computer Vision and Pattern Recognition*, Madison, Wisconsin, June 2003. Cited on page(s): 25

[91] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998. Cited on page(s): 15, 20

[92] E.A. Lord and C.B. Wilson. *The Mathematical Description of Shape and Form*. Ellis Horwood Limited, Chichester, England, 1984. Cited on page(s): 2

[93] C. Lorenz and N. Krahnstöver. 3D statistical shape models for medical image segmentation. In *Second International Conference on 3D Imaging and Modeling*, Ottawa, Canada, 1999. Cited on page(s): 21

[94] D.G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, 1985. Cited on page(s): 20

[95] D.G. Lowe. Distinctive image features from scale-invariant keypoints. To appear in *International Journal of Computer Vision*, 2004. Cited on page(s): 41

[96] D. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications*, 21(3):24–35, 2001. Cited on page(s): 85

[97] J.B.A. Maintz and M. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36, 1998. Cited on page(s): 28, 41

[98] R. Malladi, J.A. Sethian, and B.C. Vemuri. Shape modeling with front propagation: a level set approach. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995. Cited on page(s): 6, 17

[99] M. Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, Maryland, 1988. Cited on page(s): 16

[100] D. Marr. *Vision*. W.H. Freeman and Co., San Francisco, 1982. Cited on page(s): 20

[101] D. Martindale. *A pipelined framework for multiscale image comparison*. PhD thesis, University of British Columbia, 2002. Cited on page(s): 54

[102] T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: A survey. *Medical Image Analysis*, 1(2):91–108, 1996. Cited on page(s): 16, 57

[103] T. McInerney and D. Terzopoulos. Topology adaptive deformable surfaces for medical image volume segmentation. *IEEE Trans. Medical Imaging*, 18(10):840–850, 1999. Cited on page(s): 17, 57

[104] T. McInerney and D. Terzopoulos. T-Snakes: Topology adaptive snakes. *Medical Image Analysis*, 4(2):73–91, 2000. Cited on page(s): 17, 57

[105] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics (Proc. SIGGRAPH '95)*, 29:39–46, 1995. Cited on page(s): 27

[106] F. Mokhatarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. In *Proc. First International Workshop on Image Databases and Multi-Media Search*, pages 35–42, Amsterdam, 1996. Cited on page(s): 21

[107] F. Mokhtarian and A. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(8):789–805, 1992. Cited on page(s): 70

[108] D. Mumford. Mathematical theories of shape: Do they model perception? In *Geometric Methods in Computer Vision*, volume 1570 of *Proceedings of SPIE*, pages 2–10. SPIE Press, 1991. Cited on page(s): 8, 23

[109] M. Näf, G. Székely, R. Kikinis, M.E. Shenton, and O. Kübler. 3D Voronoi skeletons and their usage for the characterization and recognition of 3D organ shape. *Computer Vision and Image Understanding*, 66(2):147–161, May 1997. Cited on page(s): 68, 85, 86, 94

[110] B. Naylor. Binary space partitioning trees as an alternative representation of polytopes. *Computer Aided Design*, 22, 1990. Cited on page(s): 16

[111] D.R. Ney, E.K. Fishman, and D. Magid. Volume rendering of computed tomography data: Principles and techniques. *IEEE Computer Graphics & Applications*, 10(2):24–32, 1990. Cited on page(s): 56

[112] R.L. Ogniewicz. Skeleton-space: A multiscale shape description combining region and boundary information. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 746–751, Seattle, WA, 1994. Cited on page(s): 17, 68, 71, 72

[113] R.L. Ogniewicz. Automatic medial axis pruning by mapping characteristics of boundaries evolving under the euclidean geometric heat flow onto Voronoi skeletons. Technical Report 95-4, Harvard Robotics Laboratory, 1995. Cited on page(s): 71

[114] R.L. Ogniewicz and M. Ilg. Voronoi skeletons: Theory and applications. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 63–69, Champaign, Illinois, 1992. Cited on page(s): 70

[115] R. Ohbuchi, T. Minamitani, and T. Takei. Shape-similarity search of 3D models by using enhanced shape functions. In *Proc. Theory and Practice of Computer Graphics*, pages 97–104, Birmingham, U.K., June 2003. Cited on page(s): 25

[116] R. Ohbuchi and T. Takei. Shape-similarity comparison of 3D models using alpha shapes. In *Proc. Pacific Graphics*, Canmore, Canada, October 2003. Cited on page(s): 25

[117] R. Ohbuchi, O. Tomo, M. Ibato, and T. Takei. Shape-similarity search of three-dimensional models using parameterized statistics. In *Proc. Pacific Graphics*, pages 265–274, Beijing, China, October 2002. Cited on page(s): 25

[118] A. Okabe, B. Boots, and K. Sugihara. *Concepts and Applications of Voronoi Diagrams*. Wiley, 1992. Cited on page(s): 31

[119] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Transactions on Graphics*, 21(4):807–832, October 2002. Cited on page(s): 26

[120] D.S. Paik, C.F. Beaulieu, R.B. Jeffrey, G.D. Rubin, and S. Napel. Automated flight path planning for virtual endoscopy. *Medical Physics*, 25(5):629–37, 1998. Cited on page(s): 68

[121] D.L. Pham, C. Xu, and J.L. Prince. Current methods in medical image segmentation. *Annual Review of Biomedical Engineering*, 2, 2000. Cited on page(s): 27

[122] S. Pizer, D. Fritsch, P. Yushkevich, V. Johnson, and E. Chaney. Segmentation, registration and measurement of shape variation via image object shape. *IEEE Trans. Medical Imaging*, 18(10):851–865, 1999. Cited on page(s): 5, 18, 25, 58, 68

[123] S.M. Pizer, D. Eberly, D.S. Fritsch, and B.S. Morse. Zoom-invariant vision of figural shape: The mathematics of cores. *Computer Vision and Image Understanding*, 69(1):55–71, 1998. Cited on page(s): 18

[124] S.M. Pizer, W.R. Oliver, and S.H. Bloomberg. Hierarchical shape description via the multiresolution symmetric axis transform. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9(4):505–511, 1987. Cited on page(s): 70

[125] S.M. Pizer, A.L. Thall, and D.T. Chen. M-reps: A new object representation for graphics. Technical Report TR99-030, University of North Carolina at Chapel Hill, September 1999. Cited on page(s): 18, 27

[126] J. Ponce, M. Cepeda, S. Pae, and S. Sullivan. Shape models and object recognition. In *Shape, Contour and Grouping in Computer Vision*, volume 1681 of *Lecture Notes in Computer Science*, pages 31–57. Springer-Verlag, 1999. Cited on page(s): 6, 28

[127] R.J. Prokop and A.P. Reeves. A survey of moment-based techniques for unoccluded object representation and recognition. *CVGIP: Graphical Model and Image Processing*, 54(5):438–460, 1992. Cited on page(s): 43

[128] V. Ranjan. Personal communication. Cited on page(s): 100

[129] V. Ranjan. *A Union of Spheres Representation for 3D Objects*. PhD thesis, University of British Columbia, Vancouver, Canada, 1996. Cited on page(s): 8, 11, 33, 36, 39, 40, 41, 51, 55

[130] V. Ranjan and A. Fournier. Volume models for volumetric data. *IEEE Computer, Special Issue on Volume Visualization*, 27(7):28–36, 1994. Cited on page(s): 8, 41

[131] V. Ranjan and A. Fournier. Matching and interpolation of shapes using unions of circles. *Computer Graphics Forum (Proc. Eurographics '96)*, 15(3):35–42, 1996. Cited on page(s): 8

[132] F.J. Rohlf. On the use of shape spaces to compare morphometric methods. *Hystrix*, 11(1):1–17, 1999. Cited on page(s): 16

[133] H.J. Samet. *Design and Analysis of Spatial Data Structures: Quadtrees, Octrees, and Other Hierarchical Methods*. Addison-Wesley, New York, 1989. Cited on page(s): 16

[134] S. Sclaroff and A. Pentland. Modal matching for correspondence and recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(6):545–561, 1995. Cited on page(s): 21

[135] T.W. Sederberg, P. Gao, G. Wang, and H. Mu. 2D shape blending: An intrinsic solution to the vertex path problem. *Computer Graphics (Proc. SIGGRAPH '93)*, 27:15–18, 1993. Cited on page(s): 26

[136] T.W. Sederberg and E. Greenwood. A physically-based approach to 2D shape blending. *Computer Graphics (Proc. SIGGRAPH '92)*, 26:25–34, 1992. Cited on page(s): 26

[137] T.W. Sederberg and S.R. Parry. Free-form deformation of solid primitives. *Computer Graphics (Proc. SIGGRAPH '86)*, 20(4):151–160, 1986. Cited on page(s): 27

[138] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982. Cited on page(s): 35

[139] J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, Cambridge, UK, 1999. Cited on page(s): 6, 17

[140] D. Shaked and A.M. Bruckstein. Pruning medial axes. *Computer Vision and Image Understanding*, 69(2):156–169, February 1998. Cited on page(s): 71

[141] D.J. Sheehy, C.G. Armstrong, and D.J. Robinson. Shape description by medial surface reconstruction. *IEEE Trans. Visualization and Computer Graphics*, 2(1):62–72, 1996. Cited on page(s): 17

[142] E.C. Sherbrooke, N.M. Patrikalakis, and E. Brisson. An algorithm for the medial axis transform of 3D polyhedral solids. *IEEE Trans. Visualization and Computer Graphics*, 2(1):44–61, 1996. Cited on page(s): 17

[143] H.-Y. Shum, M. Hebert, and K. Ikeuchi. On 3D shape similarity. In *Proc. Computer Vision and Pattern Recognition*, pages 526–531, San Francisco, June 1996. Cited on page(s): 25

[144] K. Siddiqi and B.B. Kimia. Parts of visual form: Computational aspects. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(3):239–251, 1995. Cited on page(s): 8, 20

[145] K. Siddiqi, A. Shokoufandeh, S.J. Dickinson, and S.W. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35(1):13–32, 1999. Cited on page(s): 21, 68, 115

[146] K. Siddiqi, K.J. Tresness, and B.B. Kimia. On the anatomy of visual form. In *Proc. International Workshop on Visual Form*, pages 507–521, Capri, 1994. World Scientific. Cited on page(s): 8, 20

[147] A. Singh, D. Goldgof, and D. Terzopoulos, editors. *Deformable Models in Medical Image Analysis*. IEEE Computer Society Press, 1998. Cited on page(s): 5, 6, 16, 57

[148] D.B. Smythe. A two-pass mesh warping algorithm for object transformation and image interpolation. Technical Report 1030, ILM Computer Graphics Department, Lucasfilm, San Rafael, California, 1990. Cited on page(s): 41

[149] D.W. Storti, G.M. Turkiyyah, M.A. Ganter, C.T. Lim, and D.M. Stal. Skeleton-based modeling operations on solids. In *Proceedings of Solid Modeling '97*, pages 141–154, Atlanta, GA, 1997. ACM Press. Cited on page(s): 27

[150] P. Suentens, P. Fua, and A.J. Hanson. Computational strategies for object recognition. *ACM Computing Surveys*, 24(1):5–61, 1992. Cited on page(s): 6, 28

[151] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton-based shape matching and retrieval. In *Proc. International Conference on Shape Modelling and Applications*, Seoul, Korea, May 2003. Cited on page(s): 68, 115

[152] R. Szeliski and H.Y. Shum. Creating full view panoramic image mosaics and environment maps. *Computer Graphics (Proc. SIGGRAPH '97)*, 31:251–258, August 1997. Cited on page(s): 27

[153] R. Tam and A. Fournier. Image interpolation using unions of spheres. *The Visual Computer*, 14(8/9):401–414, November 1998. Cited on page(s): 11, 38, 41

[154] R. Tam and A. Fournier. Shape model and threshold extraction via shape gradients. In *Proc. Vision, Modeling and Visualization*, pages 481–489, Stuttgart, Germany, 2001. Cited on page(s): 11, 55

[155] R. Tam and W. Heidrich. Feature-preserving medial axis noise removal. In *Proc. European Conference on Computer Vision*, volume 2351 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002. Cited on page(s): 12, 68

[156] R. Tam and W. Heidrich. Shape simplification based on the medial axis transform. In *Proc. IEEE Visualization*, pages 481–488, Seattle, October 2003. Cited on page(s): 13, 84

[157] R. Tam and W. Heidrich. Computing polygonal surfaces from unions of balls. In *Computer Graphics International*, 2004. (Accepted). Cited on page(s): 100

[158] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: deformable superquadrics. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):703–714, 1991. Cited on page(s): 17, 57

[159] G. Treece, R. Prager, and A. Gee. Volume-based three-dimensional metamorphosis using sphere-guided region correspondence. *The Visual Computer*, 17(7):397–414, 2001. Cited on page(s): 27

[160] A. Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977. Cited on page(s): 8

[161] R.C. Veltkamp and M. Hagedoorn. State-of-the-art in shape matching. Technical Report UU-CS-1999-27, Department of Computer Science, Utrecht University, 1999. Cited on page(s): 24, 26

[162] R.C. Veltkamp and M. Hagedoorn. Shape similarity measures, properties, and constructions. In *Advances in Visual Information Systems*, volume 1929 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000. Cited on page(s): 24

[163] Y. Wang and L.H. Staib. Physical model-based non-rigid registration incorporating statistical shape information. *International Journal of Medical Image Analysis*, 4(1):7–20, 2000. Cited on page(s): 1

[164] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, 1990. Cited on page(s): 27, 41

[165] G. Wolberg. Image morphing: a survey. *The Visual Computer*, 14(8/9):360–372, 1998. Cited on page(s): 27, 41

[166] H.J. Wolfson. On curve matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(5):483–489, May 1990. Cited on page(s): 5

[167] H.J. Wolfson and I. Rigoutsos. Geometric hashing: An overview. *IEEE Computational Science and Engineering*, pages 10–21, Oct-Dec 1997. Cited on page(s): 26

[168] M. Woo, J. Neider, T. Davis, D. Shreiner, and OpenGL Architecture Review Board. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*. Addison-Wesley, third edition, 1999. Cited on page(s): 3

[169] B. Wyvill, J. Bloomenthal, T. Beier, J. Blinn, A. Rockwood, and G. Wyvill. Modeling and animating with implicit surfaces. Siggraph Course Notes Volume 23, 1990. Cited on page(s): 68

[170] C. Xu, D.L. Pham, and J.L. Prince. Medical image segmentation using deformable models. In J.M. Fitzpatrick and M. Sonka, editors, *SPIE Handbook on Medical Imaging – Volume III: Medical Image Analysis*. SPIE Press, May 2000. Cited on page(s): 28, 57

[171] D. Zhang and M. Hebert. Multi-scale classification of 3D objects. In *Proc. Computer Vision and Pattern Recognition*, pages 864–869, Puerto Rico, June 1997. Cited on page(s): 25

[172] Y. Zhang, J.K. Paik, A. Koschan, M.A. Abidi, and Gorsich D. A simple and efficient algorithm for part decomposition of 3D triangulated models based on curvature analysis. In *Proceedings of the International Conference on Image Processing (ICIP)*, volume 3, pages 273–276, Rochester, N.Y., September 2002. Cited on page(s): 95

[173] L. Zusne. *Visual Perception of Form*. Academic Press, 1970. Cited on page(s): 20

Index of Terms

additive noise, 75

bipartite graph, 44

clustering, 41
completeness, 9
contact curve, 92
Cost Scaling Algorithm, 48
CSA, 48

Delaunay triangulation, 31
    regular, 32
    weighted, 32
dual shape, 102

empty circle property, 32
event, shape change, 55

feature distance, 43
figural shape, 18

gradient
    axial, 76
    neighbourhood, 43
    shape, 55

histogram analysis, 56

LFS, 37
Local Feature Size, 37

matched gradient, 61
moments, alignment by, 43

orthocentre, 103
orthogonal balls, 32

peeling, 86
polar balls, 34
poles, 34
power diagram, 32
power shape, 87
pruning, medial axis, 68

r-regular, 35
r-sample, 37
regular triangle, 103
regularization, medial axis, 85

shape, definition, 2
significance measure, 69
singular point, 88, 101
singular triangle, 103
sphericity, 42
Symmetry-Curvature Duality Theorem, 75

topology graph, 92

Union of Spheres, 8
unmatched gradient, 61
unpruning, 69

Voronoi diagram, 31