

ALGORITHMS IN 3D SHAPE SEGMENTATION

by

Patricio D. Simari

A thesis submitted in conformity with the requirements
for the degree of Ph.D. in Computer Science
Graduate Department of Computer Science
University of Toronto

Copyright © 2009 by Patricio D. Simari

Abstract

Algorithms in 3D Shape Segmentation

Patricio D. Simari

Ph.D. in Computer Science

Graduate Department of Computer Science

University of Toronto

2009

Surfaces in 3D are often represented by polygonal meshes and point clouds obtained from 3D modeling tools or acquisition processes such as laser range scanning. While these formats are very flexible and allow the representation of a wide variety of shapes, they are rarely appropriate in their raw form for the range of applications that benefit from their use. Their decomposition into simpler constituting parts is referred to as shape segmentation, and its automation remains a challenging area within computer science.

We will present and analyze different aspects of shape segmentation. We begin by looking at useful segmentation criteria and present a categorization of current methods according to which type of criteria they address, dividing them into affinity-based, model-fitting, and property-based approaches.

We then present two algorithmic contributions in the form of a model-based and a property-based segmentation approaches. These share the goals of automatically finding redundancy in a shape and propose shape representations that leverage this redundancy to achieve descriptive compactness. The first is a method for segmenting a surface into piece-wise ellipsoidal parts, motivated by the fact that most organic objects and many manufactured objects have large curved areas. The second is an algorithm for robustly detecting global and local planar-reflective symmetry and a hierarchical segmentation approach based on this detection method.

We note within these approaches the variation in segmentations resulting from differ-

ent criteria and propose a way to generalize the segmentation problem to heterogeneous criteria. We introduce a framework and relevant algorithms for multi-objective segmentation of 3D shapes which allow for the incorporation of domain-specific knowledge through multiple objectives, each of which refers to one or more segmentation labels. They can assert properties of an individual part or they can refer to part interrelations. We thus cast the segmentation problem as an optimization minimizing an aggregate objective function which combines all objectives as a weighted sum.

We conclude with a summary and discussion of the contributions presented, lessons learned, and a look at the open questions remaining and potential avenues of continued research.

Dedication

To my family.

Thank you for all the love, support and encouragement throughout the years.

Acknowledgements

I would like to thank my advisor, Karan Singh, for all his guidance, help and support since the beginning of my Ph.D. studies, as well as my advisory committee members Eugene Fiume, Aaron Hertzmann and external examiner Richard Zhang for all of their comments and feedback which improved this work enormously.

I am also grateful to Richard Zhang in addition to Frank Liu, Alla Sheffer and Vladislav Kraevoy for providing the results of applying their algorithms for comparison.

My thanks go out to my fellow lab members Derek Nowrouzezahrai and Evangelos Kalogerakis for their friendship and collaboration in research.

I would also like to thank Diego Macrini, Pablo Sala and Andrés Lagar Cavilla for all of their friendship, lively discussions, and much needed distraction.

Finally, I would like to thank my family for all of the love, support and encouragement they gave me throughout the years in spite of the distance. I couldn't have done it without them.

Contents

1	Introduction	1
1.1	Shape segmentation: motivation and applications	1
1.2	Segmentation criteria	3
1.3	Segmentation parameters	5
1.4	Part labeling	6
1.5	Terminology	7
1.6	Contributions of this thesis	7
2	Related work	13
2.1	Image segmentation	13
2.2	Approaches to 3D shape segmentation	16
2.2.1	Affinity-based approaches	17
2.2.2	Model-fitting approaches	19
2.2.3	Property-based approaches	21
2.2.4	Other considerations when segmenting shape	22
2.3	Summary of shape segmentation methods and typical results	23
2.3.1	Affinity-based approaches	23
2.3.2	Model-fitting approaches	30
2.3.3	Property-based approaches	34

3	A model-fitting approach: Ellipsoidal segmentation	47
3.1	Motivation and background	47
3.2	Error metric for ellipsoidal surface regions	50
3.3	Error metric for ellipsoid volume	54
3.4	Ellipsoidal segmentation	57
3.4.1	Smoothing segmentation boundaries	58
3.4.2	Impact of different metrics	59
3.5	Remeshing of ellipsoidal representations	60
3.5.1	Using ellipsoidal region boundaries	60
3.5.2	Remeshing with parametric tessellation of surfaces	62
3.5.3	Remeshing by iterative vertex addition	63
3.6	Limitations	65
3.7	Summary	67
4	A property-based approach: Local approximate symmetry-based segmentation	68
4.1	Motivation and background	68
4.2	Symmetric region detection	73
4.3	Folding trees	77
4.3.1	Definitions	77
4.3.2	Folding tree construction and unfolding	78
4.4	Results	80
4.5	Limitations	83
4.6	Summary	86
5	Generalizing to heterogenous criteria: A multi-objective approach to shape segmentation and labeling	87
5.1	Motivation and background	88

5.2	Multi-objective shape segmentation	89
5.3	Parameterizing segmentations	91
5.4	Initial center placement	95
5.5	Assigning labels to segments	95
5.5.1	Optimal unary labeling	96
5.5.2	Complexity of n -ary labeling	96
5.5.3	Evolutionary labeling	97
5.6	Segmentation optimization	100
5.7	Applications	101
5.7.1	Semantically meaningful segmentations	102
5.7.2	Bounding volume hierarchies for path tracing	105
5.7.3	Automatic character rigging	107
5.8	Limitations	113
5.9	Summary	116
6	Conclusion	117
6.1	Summary	117
6.2	Discussion	119
6.3	Future directions	121
	Bibliography	124

List of Figures

1.1	Segmentation applications	4
1.2	Results of two segmentation criteria on the same model	5
2.1	Katz and Tal	25
2.2	Katz <i>et al.</i>	26
2.3	Yamauchi <i>et al.</i>	27
2.4	Lai <i>et al.</i>	28
2.5	Liu and Zhang	28
2.6	Liu and Zhang	29
2.7	Golovinsky and Funkhouser	30
2.8	Lukács <i>et al.</i>	31
2.9	Attene <i>et al.</i>	31
2.10	Cohen-Steiner <i>et al.</i>	32
2.11	Julius <i>et al.</i>	33
2.12	Yan <i>et al.</i>	34
2.13	Lien <i>et al.</i>	35
2.14	Kraevoy <i>et al.</i>	36
2.15	Lu <i>et al.</i>	37
2.16	Mitra <i>et al.</i>	37
2.17	Mitra <i>et al.</i>	38
2.18	Podolak <i>et al.</i>	39

2.19	Podolak <i>et al.</i>	40
2.20	Thrun <i>et al.</i>	41
2.21	Mortara <i>et al.</i>	42
2.22	Liu <i>et al.</i>	42
2.23	Lien	43
2.24	Skeleton approaches	45
3.1	Ellipsoidal representation	48
3.2	Surface-to-ellipsoid distances	52
3.3	Negative ellipsoids	53
3.4	From volume to surface approximation using ellipsoid	55
3.5	Ellipsoidal volume segmentation	56
3.6	Ellipsoidal surface segmentation	58
3.7	Impact of distance metrics to ellipsoidal segmentation	59
3.8	Ellipsoidal representation remeshing using stereographic projection . . .	62
3.9	Ellipsoidal representation remeshing results	64
3.10	Medical curve initialization of ellipsoids	66
4.1	GM robust cost estimator	74
4.2	Local symmetry detection convergence	76
4.3	Folding dino	78
4.4	Folding tree construction	79
4.5	Folding horse	81
4.6	Local symmetries and folding trees	82
4.7	More local symmetries	84
4.8	Local symmetry detection failure	85
5.1	Influence of multiple objectives	91
5.2	Using MDS to disentangle	94

5.3	Multi-objective segmentation of birds	99
5.4	Hierarchical multi-objective segmentation	102
5.5	Multi-objective segmentation of quadrupeds	103
5.6	Multi-objective segmentation comparison to human segmentations	104
5.7	Multi-objective segmentation for bounding volume hierarchy computation	106
5.8	Hierarchical multi-objective seg. of humanoids for automatic rigging . . .	109
5.9	Segmentation objectives for each model	110
5.10	Automatic fitting of clothing	112
5.11	Spatial parametrization and polygon soup	113
5.12	Shapes with non-zero genus and complicated segmentation boundaries . .	114
5.13	Multi-objective initialization failure	115

Chapter 1

Introduction

1.1 Shape segmentation: motivation and applications

Digital representations of 3D shapes have become increasingly common-place in recent years as a product of both artist modeling and acquisition processes such as laser range scanning. Among the most common formats for representing such data are polygonal meshes and point clouds. A point cloud represents a shape by a set of points in 3D space. Each such point represents the 3D coordinates of the surface at a sample location. Often other information such as surface normal and texture information are stored as well. A polygon mesh also represents a surface by a collection of points. In this case, however, they are the vertices of a tessellation and the surface is represented as a set of polygons (most often triangles or quadrilaterals.)

Digital 3D models, represented as polygon meshes or point clouds, are extensively used in engineering design, animation, games and medicine. These models are invariably structured into parts dictated by their application: limbs for articulated character rigging, functional components for engineering simulations, tight bounding volumes for efficient rendering or labeled anatomic structures for medical visualization. The partitioning of a model into such parts is referred to as shape segmentation, and its automation for various

applications is an important area of ongoing research.

There are many varied applications relevant to all of the major areas of computer graphics that directly benefit from automated surface segmentation. One such application is the automatic creation of animation skeletons [49], which are graphics tools which allow an animator to deform a shape and pose it so that it can be animated for a computer game, film, *etc.* A segmentation approach for this application automatically decomposes a character's shape into limbs and other relevant anatomical parts and places the bones and joints of the animation skeleton in the appropriate places.

Another application assists artists in the creation of models by combining elements from several pre-existing ones which have been automatically segmented into relevant parts [28, 55]. Novices who are unfamiliar with the modeling process, or those who simply want to create models more quickly, can take parts automatically found across different models and combine them to create a novel model with only a fraction of the effort which would be required to model the shape from scratch.

Range scanning devices can be used to capture 3D data from the point of view of the capturing device, but data for parts of the objects which are occluded will not be present in the scan. Segmentation algorithms can be used to automatically partition the scanned data by properties such as local symmetry, such that the missing data from the occluded parts can be inferred [109].

Models acquired using digital scanning processes are often of a very high resolution. Here, too, shape segmentation can play a role in discovering regions which can be simplified and represented with a much smaller set of surface elements [20].

Scanned shapes also contain varying degrees of noise which can be corrected by smoothing algorithms. Here, segmentation can play a role by automatically decomposing the surface into regions which can be locally smoothed while preserving surface details [115].

Digital modeling tools are often used in the design of manufactured objects whose

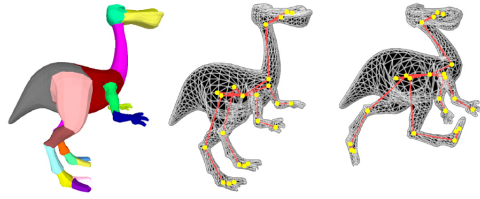
construction is constrained by the manufacturing process. For example, they may need to be assembled from materials which can be bent but not easily stretched, such as cloth or metals. Here, shape segmentation techniques can assist by automatically partitioning the surface into developable regions which can then be cut from the material and assembled to manufacture the object [46].

Similarly, shape segmentation techniques can assist CAD applications in reverse engineering tasks by automatically partitioning digital models into manufacturable parts such as spheres, cylinders, *etc.* [8, 116].

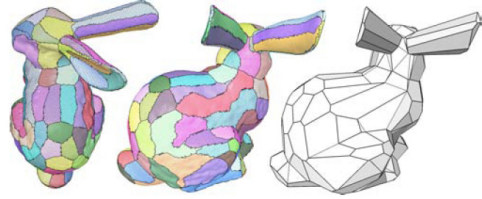
Figure 1.1 illustrates several example segmentations in the context of their application. These approaches will be explored, along with others, in Chapter 2. Furthermore, as we will see in Chapters 3, 4 and 5, we will present novel segmentation approaches with applications to shape approximation and simplification, remeshing, compression, automatic animation skeleton placement, and meaningful decomposition among others, in this last case referring to parts with corresponding semantic labels, *e.g.* limbs and other anatomical parts in animal models.

1.2 Segmentation criteria

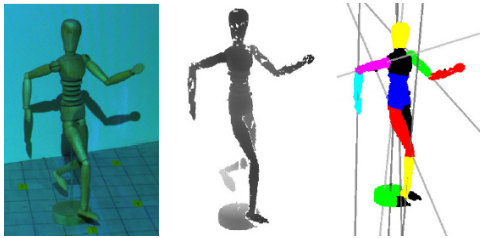
There is no single segmentation criterion by which any surface can be partitioned so as to be useful to all applications. The quality of a given surface segmentation will be determined by the application which requires it, and a partition that is useful in one context, will be inappropriate in another. This can be observed in Figure 1.1 where the bunny and dino models appear differently segmented as dictated by the application's needs. Another example can be seen in Figure 1.2. Here the same model, a bishop chess piece, is shown segmented into the same number of parts but according to two different criteria. On the left, the surface has been segmented into developable patches [46] as would be required if we wished to construct the piece out of a material such as metal or



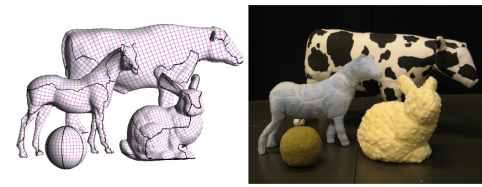
Automatic animation skeleton construction

Katz et al.

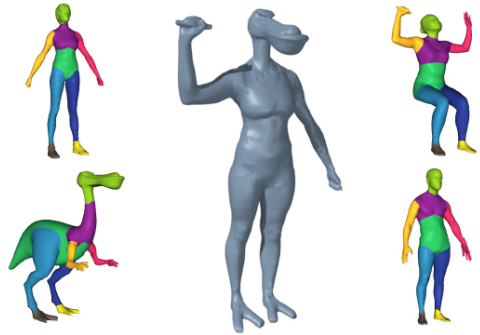
Surface simplification/re-meshing

Cohen-Steiner et al.

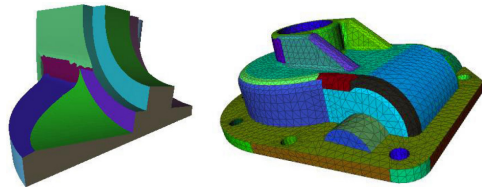
Shape reconstruction

Thrun and Wegbreit

Material patches for construction

Julius et al.

Model creation

Kraevoy et al.

CAD reverse engineering

Attene et al.

Figure 1.1: Applications of mesh segmentation from Katz *et al.* [49], Cohen-Steiner *et al.* [20], Thrun and Wegbreit [109], Julius *et al.* [46], Kraevoy *et al.* [55] and Attene *et al.* [8].

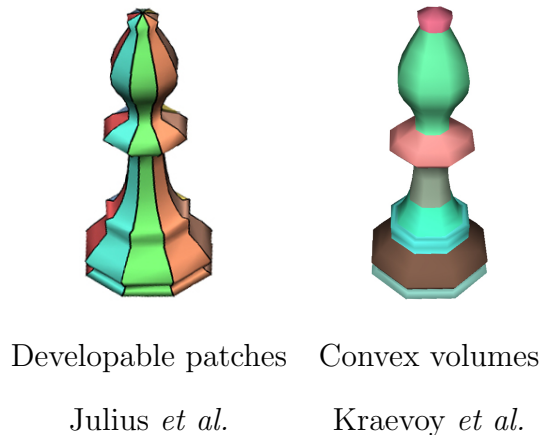


Figure 1.2: The result of segmentating one object with respect to two different criteria: developable patches [46] and convex volumes [54].

cloth. On the right, the surface has been segmented into convex volumes as would be required, for example, for efficient ray tracing of the object.

In Chapter 2 we will explore these and other segmentation criteria along with the algorithms which achieve a segmentation according to them.

1.3 Segmentation parameters

The focus of this thesis will be on automatic segmentation methods. While there has been some work on the interactive specification of surface segmentation [28], and these methods certainly have their use, it would quickly become tedious for a user to have to segment a large set of models. Moreover, depending on the segmentation goal, a manual segmentation could yield suboptimal result if the goal is not a visually intuitive one. This entails that the user will have to specify his segmentation goals in some other manner.

As we will see in Chapter 2, different approaches will require, in addition to the surface to be segmented, different input parameters. Typically, a user will specify the number of segments desired. Conversely, an algorithm may require one or more threshold values indicating a tolerance which will determine the number of segments indirectly. Additionally, iterative approaches may require a maximum number of iterations.

In Chapter 5 we will propose a general segmentation model which decouples the segmentation objective from the segmentation algorithm. This novel approach allows for users to describe their problem-specific knowledge to the algorithm by giving as input one or more segmentation objective functions. Moreover, we will allow each objective to possibly refer to more than one segment, thus novelly allowing users to describe segment interrelationships.

1.4 Part labeling

Current segmentation algorithms apply a single criterion uniformly to each segment. In essence, their aim is to obtain a partition of the surface that optimizes some objective function which is applied to all segments. Segmentations are typically expressed by associating a number to each surface element that indicates the segment to which it belongs. However, given the uniform nature of the objective, any permutation of segment numbers will essentially represent the same segmentation. As a result of our allowing for multiple heterogeneous objective functions to selectively refer to one or more segments, this need not be the case in general.

We introduce the notion of automated segmentation labeling, in contrast to application-dependent heuristic [108] or example-based [96] approaches. Given a surface segmentation and a set of unique labels, a labeling will refer to an assignment of labels to segments. Given a set of segmentation objectives which refer to these part labels, an optimal labeling will be one that optimizes the cumulative objective function value. As we will see in Chapter 5, in the case that all objective functions each refer to only one label, this assignment is optimally solvable in polynomial time. When objective functions are individually allowed to refer to more than one label, however, we will show that this labeling problem becomes NP-complete, and we will present a heuristic algorithm to address labeling.

1.5 Terminology

There are terms which are common within the mesh processing literature and which we will use throughout this thesis. In the following, we discuss some of this terminology. We also refer the interested reader to Shamir’s recent survey [97].

A *mesh* can be thought of as a tuple (V, E, F) , where V is the set of *vertices*, E is the set of *edges* and F is the set of *faces*. A *vertex* is a point in $\mathbb{R}^3$. An *edge* is a tuple $e_{ij} = (v_i, v_j)$ where $v_i, v_j \in V$. A *face* is an n -tuple $f_{i_1, i_2, \dots, i_n} = (e_{i_1, i_2}, e_{i_2, i_3}, \dots, e_{i_n, i_1})$, essentially a closed cycle of edges. For the purposes of this thesis we will assume $n = 3$ for all faces, which is referred to as a triangle mesh. Meshes where $n = 4$, referred to as quad meshes, and mixed cases which also include faces of varying n , are also common.

An edge e_{ij} is said to be *incident* on vertices v_i and v_j . A face $f_{i_1, i_2, \dots, i_n}$ is said to be incident on each edge $e_{i_k, i_{(k \bmod n) + 1}}$ and on each vertex v_{i_k} , $k \in \{1, \dots, n\}$. Two faces are said to be *adjacent* if there is some edge on which they are both incident. Given two adjacent faces, the *dihedral angle* refers to the angle formed between them.

An edge is said to be a *boundary* edge if there is only one face incident on it. An edge is said to be a *manifold edge* if there are two faces incident on it, and *non-manifold* otherwise. A mesh is said to be *watertight* if it contains no boundary edges.

1.6 Contributions of this thesis

This thesis will present a categorization of current surface segmentation algorithms in the geometry processing community, discriminating them into affinity-based, model-fitting, and property-based approaches.

Within this categorization, our first two algorithmic contributions share the goals of automatically finding redundancy in a shape and proposing shape representations that leverage this redundancy to achieve descriptive compactness. Our third and final contribution is a generalized segmentation approach which decouples the algorithm from

the objective function, allowing not only for multiple heterogenous objectives applying to different segments but, also novelly, to groups of segments and their interrelationships.

Ellipsoidal segmentation

Our first contribution is to the area of model-fitting approaches through the introduction of a method for segmenting a surface into piece-wise ellipsoidal parts. We note that curved surface regions can be well approximated by an ellipsoid and the corresponding region boundary. We thus explore the potential of compactly approximating a shape using a relatively small number of such regions. This is motivated by the fact that most organic objects and many manufactured objects have large curved areas. In areas where the surface of interest is flat, a region from an ellipsoid with sufficiently large radius can be used.

Ellipsoids, unlike planes, being closed surfaces, are also capable of providing information about the volume of closed objects. This means that we are also able to represent the volume enclosed by a given surface as the union of a set of ellipsoidal volumes. In this case, we use the entire ellipsoid and not just regions of its surface.

Once a shape has been segmented into an ellipsoidal approximation, it is only necessary to store the ellipsoids and, in the surface case, the segmentation boundaries at which they meet in order to represent the object.

Local approximate symmetry-based segmentation

Our next contribution is to the area of property-based approaches by automating the search for another form of shape redundancy which is ubiquitous in natural and human-made shapes: symmetry.

We present an algorithm for robustly detecting global and local approximate planar-reflective symmetry and show how this algorithm can be used to simplify a shape by removing redundant elements. We store only the non-redundant parts and respective

planes of symmetry in a novel structure we call a “folding tree.” We also show how an approximation of the original shape can be recovered from this representation in a bottom up fashion by reflecting the non-redundant elements and reconnecting segmentation boundaries.

Multi-objective segmentation

As our final contribution, we propose a way to generalize the segmentation problem to heterogenous criteria. We note that criteria such as ellipsoidal shape, symmetry, or indeed any other criterion, need not be applied homogeneously to all segments of a shape. This is to say, a user may wish to assert that a given segment is ellipsoidal, that another is symmetric, and a third respects yet another property. In fact, we note that the users of segmentation algorithms, such as biologists, doctors, mechanical engineers or digital character modelers and animators, possess a wealth of knowledge about the objects they are analyzing and working with. However, current algorithms for shape processing do not allow for the incorporation of detailed domain knowledge and usually limit their input to the number of segments desired and a handful of threshold values. If a user wishes to automate the segmentation of models according to his domain-specific knowledge, the current approach is to implement a new segmentation algorithm from scratch. This is often out of the realm of most users. Even when it is not, such a ground-up implementation requires a considerable investment of time and resources.

We thus introduce a framework and relevant algorithms for multi-objective segmentation of 3D shapes, which allow for the incorporation of domain-specific knowledge through heterogenous objectives, each of which refers to one or more segmentation labels. These objectives can be unary, asserting properties of an individual part associated with a given label (*e.g.* that it should be narrow, compact, flat, symmetric) or they can be n -ary, referring to part interrelations, (*e.g.* a set of parts should be parallel to each other, or have the same proportions, or two parts should be perpendicular to each other.) We thus

cast the segmentation problem as a black-box optimization minimizing a combination of heterogenous objectives.

Summary of contributions

The details of the algorithmic contribution chapters of this thesis can be summarized as follows:

Chapter 3, A model-fitting approach: Ellipsoidal segmentation

- We specifically motivate the problem of approximating shape using piece-wise ellipsoidal regions and in particular describe dual uses of ellipsoids as describing surface patches and also volumes.
- We define error metrics for ellipsoidal surface fitting (combining position, normal, and curvature errors) as well as a volumetric error metric.
- Based on these metrics, we introduce a variational approach to obtaining ellipsoidal segmentations which minimizes according to either metric.
- Given a surface-based segmentation, we show how the original shape can be approximated by an ellipsoidal representation consisting of the union of piece-wise ellipsoidal regions represented by a set of ellipsoid parameters and a region boundary lying on the ellipsoid's surface.
- We introduce algorithms for re-meshing such a representation by means of either regular parametric sampling or iterative optimal vertex insertion.

Chapter 4, A property-based approach: Local approximate symmetry-based segmentation

- We motivate the problem of symmetry detection in shape analysis and describe the notions of approximate vs. perfect symmetry, and local (or partial) vs. global symmetry.
- We cast the approximate global and local symmetry detection problem in the framework of robust estimation and introduce an algorithm for detecting such regions.
- We introduce the concept of a “folding tree” as a hierarchical symmetry-based representation of a shape which can be “un-folded” to recover an approximation of the original shape. We show how this decomposition can be obtained through recursive application of the robust detection algorithm.

Chapter 5, Generalizing to heterogenous criteria: A multi-objective approach to shape segmentation and labeling

- We describe the notion of multi-objective shape segmentation and motivate it as a means for incorporation of user knowledge to the shape segmentation problem.
- We motivate the use of a continuous domain parameterization of shape segmentations and introduce multiplicatively weighted Voronoi space partitioning as a means to do so.
- We show how shapes that are under complex articulation can still be addressed with our approach through the use of multi-dimensional scaling applied to surface samples and pair-wise approximate geodesics to untangle the shape.
- We address initializing a segmentation based on the above-mentioned parametrization using a k -means approach and then describe how the segmentation can be optimized from a coarse initialization, motivating and describing the use of pattern search to do so.

- We define the notions of segmentation labels and objectives and how they relate to segments. We then explore ways in which segmentation labels can be automatically assigned to segments to maximize objectives. In particular, we introduce an efficient approach which is optimal under unary objectives. We also show that, in the presence of binary objectives, the labeling problem is NP-complete by reducing the Traveling Salesman Problem to our labeling problem. Thus motivated, we present an evolutionary algorithm [75] which can optimize with respect to n -ary objectives.
- We motivate the incorporation of symmetry constraints on segmentations as further means by which a user can supply domain knowledge to the segmentation problem and illustrate how our parameterization and optimization approaches can naturally accommodate this.
- We apply our segmentation method to obtain semantically meaningful labeled segmentations, optimized bounding volume hierarchies for path tracing, animation skeleton placement for automatic character rigging, and automatic transfer of clothing for animation characters.

Chapter 2

Related work

2.1 Image segmentation

The notion of segmentation has been around for quite some time as it applies to images within the computer vision and image processing communities. An in-depth study of image segmentation methods is outside the scope of this thesis, but the interested reader is referred to textbooks [26, 88] and surveys [36, 84, 87] of this now mature area of research .

In recent years, the use of affinity-based methods remains an area of active research in image and video segmentation. In one such method, users interactively provide scribbles indicating regions of interest and affinities are computed as geodesic distances to scribbles which are weighted based on spatial and/or temporal gradients [12]. A more efficient alternative to graph cut-based methods casts the segmentation problem as an approximate energy minimization in a Conditional Random Field (CRF) [23]. The CRF approach has also been used recently to jointly address segmentation and 3D pose estimation of a human body from multiple views [53]. In this approach, the cost function incorporates edges, color smoothness and shape priors, combining them in a Bayesian framework.

As with 3D shape segmentation, it also holds within image segmentation that the notion of what is a meaningful or important segment varies from application to application. Common criteria include color, texture, or local symmetry and novel criteria are continually proposed. For example, a recent approach attempts to generalize these by defining a meaningful region as one that can be easily composed using its own pieces but is difficult to compose using pieces from other parts of the image [11].

Comparing the accuracy of different segmentation approaches has been a difficult problem and recent strides have been made toward addressing this matter by showing how a recently proposed similarity metric, the Normalized Probabilistic Rand (NPR) index, can be used to perform a quantitative comparison between image segmentation algorithms using a hand-labeled set of ground truth segmentations [113]. In the context of 3D shape segmentation how to perform such comparisons has remained largely unaddressed and we expect that as the area matures, research attention will be focused on the creation of ground truth benchmarks and comparison metrics for results. At the time of the writing of this thesis, Chen *et al.* have very recently introduced the first of such benchmarks [19]. This work includes a data set of nearly 400 models, each of which is hand segmented by an average of eleven people using a tool created by the authors. In the work, the authors compare these human segmentations to the automatic segmentations produced by several state-of-the-art algorithms using metrics which are analogs of those commonly used in the vision community to compare image segmentations.

Perhaps most closely related to 3D shape segmentation is the area of range image segmentation. In range images the value at each pixel corresponds to a distance to points in the scene so that the image can be thought of as a height field. These images can be obtained by a number of methods including stereo triangulation and structured light approaches such as laser scanners. Recent approaches to segmenting these types of images include using Markov random fields to learn the relative importance of features for the segmentation task [5], and belief propagation to incorporate knowledge of articulated

objects in the scene [93].

The automatic labeling of image regions using descriptive words has also been examined within computer vision and image segmentation. One such approach proposes to learn a mapping from a set of training images labeled with keywords [25]. The images are segmented and classified into types based on various features. A mapping from region types to labels is then learned using an EM-based method. A more recent approach extends the type of labels to include those which describe relationships between objects, *i.e.* prepositions and comparative adjectives [34]. Learning these relations proves useful in resolving labeling ambiguities for objects that frequently appear together (*e.g.* car and road.)

Large sets of carefully and fully labeled image data for this type of learning can be difficult to obtain. To this end, recent efforts explore the use of partially labeled data using a hybrid approach which integrates a discriminative model of region labels with a generative model of region appearance [37]. Another recent approach focuses on automatically discovering the typical location of certain labeled textures in images (*e.g.* sky and tree labels tend to appear in the upper half of images while roads tend to be in the bottom half) and using this information to improve automatic labeling of such regions in novel images.

Image segmentation and 3D shape segmentation algorithms both share the goal of partitioning their input into significant parts, this significance being determined by the driving application.

In the case of image segmentation, the data comes in the form of a uniform grid which stores the intensity values at any given pixel. These pixels are the atomic elements handled by image segmentation algorithms, in contrast to the mesh faces handled within 3D surface segmentation. This pixel data contains elements captured from the object of interest as well as everything else in the scene within view of the capturing device. These are commonly referred to as the foreground and background elements of the image, and

their automatic segmentation and classification is a common goal of vision algorithms.

In contrast, the mesh data handled by 3D shape segmentation methods contains no background elements, *i.e.* every face is considered to be an element of the shape being analyzed. Still, even though the task is different, several approaches which were initially proposed for image segmentation have been carried over into 3D shape segmentation, including region growing, watershed, affinity/graph-based approaches, active contours, model fitting and others.

2.2 Approaches to 3D shape segmentation

As we have mentioned, a mesh segmentation refers to a (usually connected) partitioning of the set of mesh faces that optimizes some objective function. Shape segmentation algorithms can be broadly classified according to the nature of this function and optimization strategy into affinity-based, model-fitting and property-based approaches.

There are other possible classifications, such patch-based versus part-based as used by Shamir, [97] or geometric versus structural versus functional as used by Liu [62]. These classifications focus on the nature of the segmentation. In contrast, this thesis focuses on the algorithmic nature of the approaches and our novel classification reflects this by grouping approaches by their high level algorithmic schemata. While model-fitting approaches could be seen as a kind of property, we treat them as distinct because they are a common enough group often addressed in a very similar algorithmic fashion.

In this section, we give an overview of these approaches, discuss the common algorithms used in each, their advantages and scenarios to which they are best suited as well as their natural limitations. Specific instances of each approach will only be briefly mentioned in this section. A more detailed description of each algorithm can be found in Section 2.3.

2.2.1 Affinity-based approaches

In affinity-based approaches, it is assumed that for every pair of mesh faces one can assign a confidence to their being in the same segment, or conversely, a dissimilarity metric that determines how likely they are to be in different segments. In such approaches the goal is to maximize intra-segment similarity and inter-segment dissimilarity. Approaches to obtaining a segmentation from such metric include graph cut, watershed, and geodesic k -means.

Graph cut

In graph cut algorithms, the face graph of the mesh is considered, defining edge weights between adjacent faces based on some attribute of the adjacent face pair, such as a combination of distance and dihedral angle concavity [49, 63] or a less local metric, for example capturing the change in internal visibility of the shape [65]. Affinities can also be obtained indirectly by, for example, using randomized versions of alternative segmentation approaches and weighting edges by how often they lie on segmentation boundaries [30]. It is also possible to use the pair-wise distances between surface elements to find an embedding of the shape and thus be able to detect elements robustly with respect to articulation [48, 64].

Graph-cut approaches have significant advantages if the problem in question is amenable to an affinity-based formulation. The solutions found by min-cut algorithms are global optima of the objective function and the approach is capable of finding complicated yet clean boundaries. The downside is that these approaches tend to have high memory requirements. However, this can be circumvented by decimating the original surface to a manageable size and then applying the method in a coarse-to-fine manner.

Watershed

In watershed approaches, affinity is encoded indirectly by means of some “height” function defined over surface elements. In the case of 3D shapes this can be defined using local curvature of the surface [70, 118]. Local minima of the height function are initially found and marked as basins using a unique label for each. Then, for each surface element, the algorithm iteratively flows greedily downhill until a basin is reached and the starting face is marked with the corresponding label.

Watershed approaches can be efficient and easy to implement but are also susceptible to noise and over-segmentation due to spurious local minima. The former can be addressed by prior smoothing and the latter by iterative merging of “shallow” minima with neighbouring segments.

Affinity-based k -means

Yet another form of using surface affinities is by computing all-pairs surface affinities which can then replace Euclidean distances in a k -means clustering of surface elements. This type of approach can be very easy to implement and has been successfully used for surface simplification by using Geodesic distances as affinities [57, 115]. However, as with k -means, it will tend to capture compact, similarly shaped clusters.

General limitations

Affinity-based methods require that it be possible to evaluate the function in question at the face level for pairs of faces. This may not be possible for functions that require a larger context to evaluate sensibly, such as developability, tubularity, *etc.* Moreover, this approach is not amenable to optimizing multiple segment-dependent functions (*e.g.* if one part should be convex and another should not.)

2.2.2 Model-fitting approaches

In model-fitting approaches, it is assumed that every segment of the mesh was independently generated by a different instance of a parameterized model. In such a scenario, given a mesh segmentation, it is possible to determine the model types (if more than one is available) and associated parameters that best fit the observations. Conversely, given an arrangement of models, it is possible to classify the mesh faces accordingly and thus determine a segmentation. Within this type of segmentation, known approaches for optimizing the overall segmentation include region growing, variational, and hierarchical approaches.

Region growing

The region growing approach begins from a set of seed patches. The model is locally fit to each of them and then the algorithm proceeds iteratively, at each step adding the un-assigned face adjacent to the patch for which the fitting error is increased the least. Thus, the regions grow until no un-assigned faces remain. Seeds can be selected by hand, randomly or heuristically [69]. Alternatively, a threshold can be provided and whenever no face can be added to a region without violating this threshold, a new segment can be created.

The region growing approach has the advantage of simplicity and efficiency. However, it can be susceptible to noise and over-segmentation, particularly in the threshold case where regions are added as needed. One way to alleviate over-segmentation is by using a region-merging step in which adjacent regions are merged into a single one so long as it is deemed possible to do so, *e.g.* by threshold violation.

Variational

The variational approach consists of the iteration of two steps: a fitting step and a classification step. In the fitting step, model parameters are fit to each of the segments

according to the current segmentation. In the classifications step, each face is assigned to the model to which it is closest. Initialization can be handled heuristically, typically through some form of initial region growing. In this form, the iteration is guaranteed to converge given that each step reduces the overall error. However, there is no guarantee that the segments will remain connected. If this is important, the classification can be done using a priority queue-based flooding which ensures connectivity is preserved. Examples of using this approach include planar primitives for mesh decimation [20], cone primitives for segmenting into developable regions [46], general quadric patches [116], and the approach we introduce in Chapter 3 for segmenting based on ellipsoidal primitives.

The variational approach is relatively easy to implement and converges to a local optimum. However, it can be computationally expensive, depending on the nature of the distance function used, since it will be typically evaluated more times than in the region growing case. Each classification step, if maintaining connecting regions is required, is a region growing instance.

Hierarchical

The hierarchical approach begins by fitting a model to every single mesh face, or to surface regions which are as small as possible if single faces are insufficient to determine model parameters. From there, the approach iteratively merges the two adjacent segments such that their merger increases the overall error the least.

The stopping criteria can be reaching a specified number of segments, violating a specified threshold, or the algorithm can continue until all regions have been merged. An advantage of the hierarchical segmentation approach is that all intermediate segmentations can be stored efficiently in a tree structure called a dendrogram which allows for realtime browsing and user selection from among the segmentations determined by each level.

The disadvantage of hierarchical clustering is that it is a heuristic approach with no

general guarantees on the optimality of the segmentation. It has been shown to work well on reverse engineering of CAD models using spheres, cylinders and planes [8].

General limitations

Approaches based on generative models have the advantage of a continuous domain of optimization and possibility of gradient computation for efficient model fitting. Unfortunately they are also not applicable to some intuitive segmentation objectives that do not lend themselves to generative model formulations, such as convexity, symmetry, *etc.*

2.2.3 Property-based approaches

Finally, there are property-based approaches. This category includes segmentations based on some surface property of interest held to some degree locally by the segments but not by the overall surface. This property is often not possible to indicatively measure at the atomic level of faces, thus precluding affinity-based approaches, and difficult or impossible to capture by a generative model approach.

In some instances, the techniques that we saw for model-fitting approaches can be modified to allow the optimization of non-model based properties. For example, the region-growing approach has been used to segment with respect to local symmetry [109], and a kind of variational approach has been proposed to segment with respect to convexity [54, 55] and tightly bound segments [68].

Alternatively, specialized algorithms have been proposed (see Section 2.3 for details) to segment with respect to local symmetry [100, 78, 89], convexity [18, 61], tubular shape [79], texture [66, 67], star shape [60] and curve skeleton association [91, 4, 110, 9], diffusion over surfaces [24], and modes of vibration [41].

General limitations

Optimization approaches that address property-based segmentations are naturally tailored for, and tied to, the specific property in regards to which they are segmenting. Furthermore, this property is uniformly attributed to all parts in the segmentation.

2.2.4 Other considerations when segmenting shape

When choosing a segmentation method, in addition to deciding the criteria upon which the segmentation is to be based, there are several data attributes which can constrain the choice of segmentation algorithm.

Region-growing and affinity-based approaches will typically require a surface with manifold connectivity given. Additionally, volumetric approaches will typically require a closed, watertight shape. To work around this problem, it might be possible to re-mesh the given data so that it satisfies these properties [98], though this is in itself a difficult problem.

Moreover, if there is noise in the surface positions, some algorithms will exhibit more robustness than others (for instance, Lien *al.* [60] with regards to convexity; see Section 2.3.) Certain algorithms are, or can be made, tolerant of noise, *e.g.* a model-fitting approach using a robust fitting method or property-based approach which takes care to evaluate robustly the property in question. Other times, this problem can be circumvented by pre-applying a smoothing process to the data.

Finally, there can be varying sampling densities/tessellation coarseness across the surface which can affect algorithm performance if explicit care is not taken in the implementation to compensate for this. In cases in which the algorithm remains sensitive to varying sampling/tessellation density, the surface can be uniformly re-meshed [2]. Again however, this can be in itself a difficult problem.

The importance of these issues will vary from algorithm to algorithm, and possibly

from implementation to implementation, but they should be kept in mind when choosing a segmentation approach for any given problem. For example, algorithms which segment based on local concavities as determined by dihedral face angles will fare well on relatively coarse meshes. On the other hand, model-fitting approaches, for example, will fare better with more densely-sampled surfaces where an abundance of data will contribute to the stability of the results.

2.3 Summary of shape segmentation methods and typical results

In this section, we will see the techniques that have been proposed or adapted in order to be applied to 3D shape representations within the geometry processing community. Earlier, in Section 2.2, we discussed how segmentation approaches could be broadly categorized as affinity-based, model-fitting, and property-based methods. Let us now examine each of these categories in detail and present an overview of relevant approaches to shape segmentation in the literature as they fit into each one. Table 2.1 presents a summary of the categorization of the methods to follow.

2.3.1 Affinity-based approaches

Mangan and Whitaker, inspired by watershed methods in image processing, extend the approach to surfaces in 3D [70] by defining the domain as the surface vertices and the height function as the total curvature. Areas of higher or drastically varying curvature are considered to be watershed boundaries. The initial basins are those vertices that are local curvature minima. Over-segmentation is compensated by iterative merging of shallow regions, *i.e.* regions whose depth is above a given threshold.

Zuckerberger *et al.* also present watershed and region flooding approaches [118]. In this case, their domain is the mesh edges and the height field is defined based on the

Affinity		
Criterion	Method	Technique
High curvature boundaries	Mangan and Whitaker [70]	Watershed
High curvature boundaries	Zuckerberger <i>et al.</i> [118].	Watershed
Concave boundaries	Katz <i>et al.</i> [49]	Fuzzy clustering, graph cut
Features vs. Core	Katz <i>et al.</i> [48]	MDS, feature/core detect, graph cut
Small normal variation	Yamauchi <i>et al.</i> [115]	Mean shift
Geod. compactness and unif. texture	Lai <i>et al.</i> [57]	Hierarchical geodesic k -means
Concave boundaries	Liu and Zhang [63]	Spectral clustering
Concave boundaries	Liu and Zhang [64]	Spectral Emb., contour analysis, graph cut
Shape visibility	Liu <i>et al.</i> [65]	Graph cut using shape visibility
Randomized seg. consensus	Gologvin and Funkhouser	Randomized seg. sampling, graph cut
Model-fitting		
Criterion	Method	Technique
Spherical, cylind., conical, toroidal	Lukacs <i>et al.</i> [69, 72]	Region growing
Planar patches	Cohen-Steiner <i>et al.</i> [20]	Variational
Ellipsoidal patches or volumes	Simari and Singh [101]	Variational
Conical patches	Julius <i>et al.</i> [46]	Variational
General quadric patches	Yan <i>et al.</i> [116]	Variational
Planar, spherical and cylindrical	Attene <i>et al.</i> [8]	Hierarchical clustering
Property		
Criterion	Method	Technique
Local symmetry	Simari <i>et al.</i> [100]	Recursive robust estimation
Local symmetry	Mitra <i>et al.</i> [78]	Transf. space clustering
Local symmetry	Podolak <i>et al.</i> [89]	Transf. space binning and graph cuts
Local symmetry	Thrun <i>et al.</i> [109]	Region growing
Convexity	Chazelle <i>et al.</i> [18]	BSP, Space-sweep, Flood, Flood & Retract
Convexity	Lien and Amato [61]	Concave features and cuts
Convexity & Compactness	Kraevoy <i>et al.</i> [54, 55]	Variational
Tightly bound (ellipsoids, boxes)	Lu <i>et al.</i> [68]	Variational
Tubularity	Mortara <i>et al.</i> [79]	Sphere to surface intersection analysis
Uniform texture	Liu <i>et al.</i> [66, 67]	Snakes
Local star-shaped	Lien [60]	Iterative insertion
Unique skeletal segment	Reniers and Telea [91]	Skeletal critical points and geodesics
Unique skeletal segment	Sharf <i>et al.</i> [4]	Surface-Skeletal correspondence
Unique skeletal segment	Au <i>et al.</i> [9]	Surface-Skeletal correspondence
Topology (Core vs. Limbs)	Tierny <i>et al.</i> [110]	Topological skel., hierarchical decomp.

Table 2.1: A summarizing taxonomy of representative segmentation methods. Each method will be discussed in greater detail in Chapter 2.

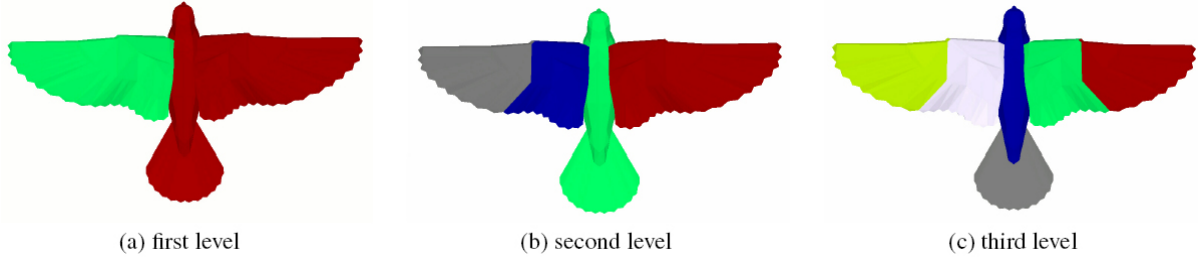


Figure 2.1: Katz and Tal propose an approach consisting of a fuzzy clustering followed by a min cut [49]. Distances between surfaces elements are defined as a combination of geodesic and angular distances and edge capacities for the cut are defined based on angle alone. The result is segments of roughly similar size with boundaries passing through convex regions. To obtain more segments, the algorithm is applied recursively.

cosine of the dihedral angle between the incident faces. As in the previous case, a post-processing merging process is used to avoid over-segmentation. In this case, segment area (not depth) is taken into account by iteratively merging the segment of smallest area to its largest neighbor.

More recently, Katz and Tal [49] propose an approach which also takes dihedral angles into account. Specifically, the approach considers the face graph of the surface and defines a weight function between adjacent faces which is a linear combination of two terms, one based on their dihedral angle and the other based on their geodesic distance. With this weight defined, the authors define a distance function between any two faces which is the sum weight of the minimum weight path connecting them. Fuzzy clustering is carried out using a maximally distant pair of faces as the initial cluster centers. Finally, this binary segmentation is then refined by using a min-cut approach where edge capacity is defined based on dihedral angle alone. The result is a segmentation into two parts of approximately the same size whose boundary tends to pass through convex edges. In order to obtain more segments, the algorithm is applied recursively. A typical result of this approach is shown in Figure 2.1.

In a related work, Katz *et al.* extend their approach to automatically find non-binary

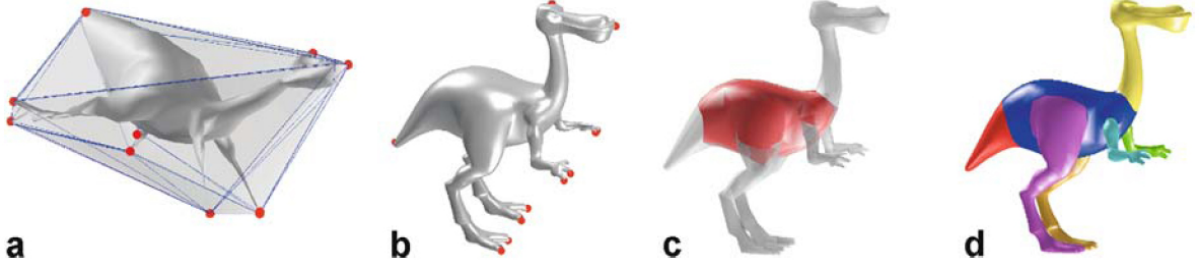


Figure 2.2: Katz *et al* [48] propose an approach which computes an unfurling of the object using MDS in order to ignore articulation (a), finds feature points which lie on the convex hull of the embedded shape (b), computes the core segment using spherical mirroring (c) and then computes the final segmentation using a min-cut approach favoring short convex boundaries (d).

segmentation seeds which correspond to feature points of the surface, such as articulation extremities. The approach consists of applying multi-dimensional scaling (MDS) to find a 3D embedding of the object’s vertices such that Euclidean distances between vertices in the embedding closely approximate geodesic distances over the original surface. The result is an unfurling of the object. The convex hull of the embedded points is computed and its vertices are considered to be the above mentioned features. The “core” (central segment) of the object is found by applying spherical mirroring of the embedded shape, taking the faces that lie on it, and then growing this region until all feature points are separated by the core. Finally, the segmentation boundaries are refined using a min-cut approach where the cost is defined to favor short convex edges. A summary of the steps and a representative result of this approach can be seen in Figure 2.2.

Yamauchi *et al.* [115] propose a method to segment according to regions of small normal variations. Face centroids are considered along with the associated face normals as a 6D set and mean shift clustering [21] is applied as a means to pre-smooth mesh normals by replacing them with the corresponding cluster center. The magnitude of the density kernel will indirectly determine the number of segments obtained. Then, starting



Figure 2.3: Results from the method of Yamauchi *et al.* [115]. Mesh normals are first clustered and filtered using a mean shift. The initial segments are then refined using a variational approach.

from current cluster centers, a variational approach is applied to refine the patches. The result is a set of relatively compact and flat patches as illustrated in Figure 2.3.

Lai *et al.* [57], also propose a method based on a k -means approach. The distance between faces used is a combination of geodesic, curvature, and geometric texture differences. Maintaining all pairs of geodesics in memory can be expensive for large models. In order to be able to handle such models, a hierarchical top-down approach is used combined with feature-sensitive mesh decimation. The result is a coarse-to-fine partitioning with geodesically compact segments with uniform texture and boundaries aligned with surface features. To further ensure this last aspect, a boundary smoothing postprocess is applied. Figure 2.4 summarizes these steps and shows an example result.

Liu and Zhang [63] introduce the technique of spectral clustering to the segmentation problem. The approach considers the affinity matrix, *i.e.* the all-pairs probabilities that two faces will belong to the same segment, based on the distance function of Katz *et al.* [49]. Then, the leading k eigenvectors of this matrix are computed and used to find an embedding of the mesh faces onto the k -dimensional unit sphere. K -means clustering is then applied in this embedding, using furthest point initialization. Results typical of this approach are illustrated in Figure 2.5.

In a subsequent approach, Liu and Zhang [64] propose an iterative binary segmenta-

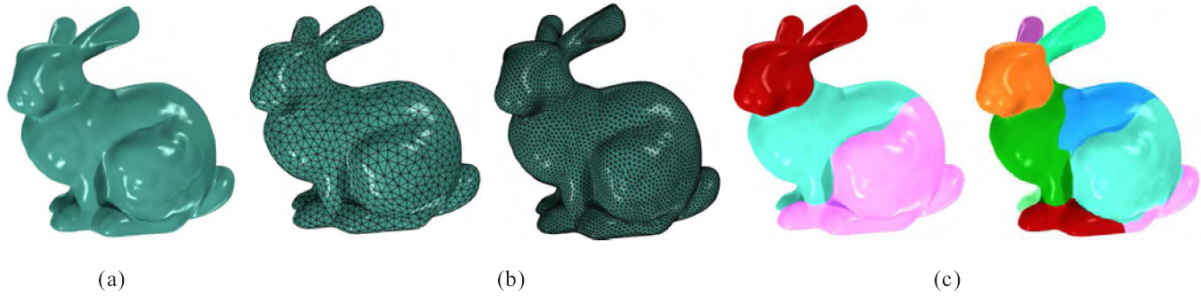


Figure 2.4: The method of Lai *et al.* takes an object (a) and pre-computes a hierarchical coarse-to-fine using feature sensitive decimation (b) and then applies a top down k -means (c). The distance used combines geodesic, curvature and local texture differences.

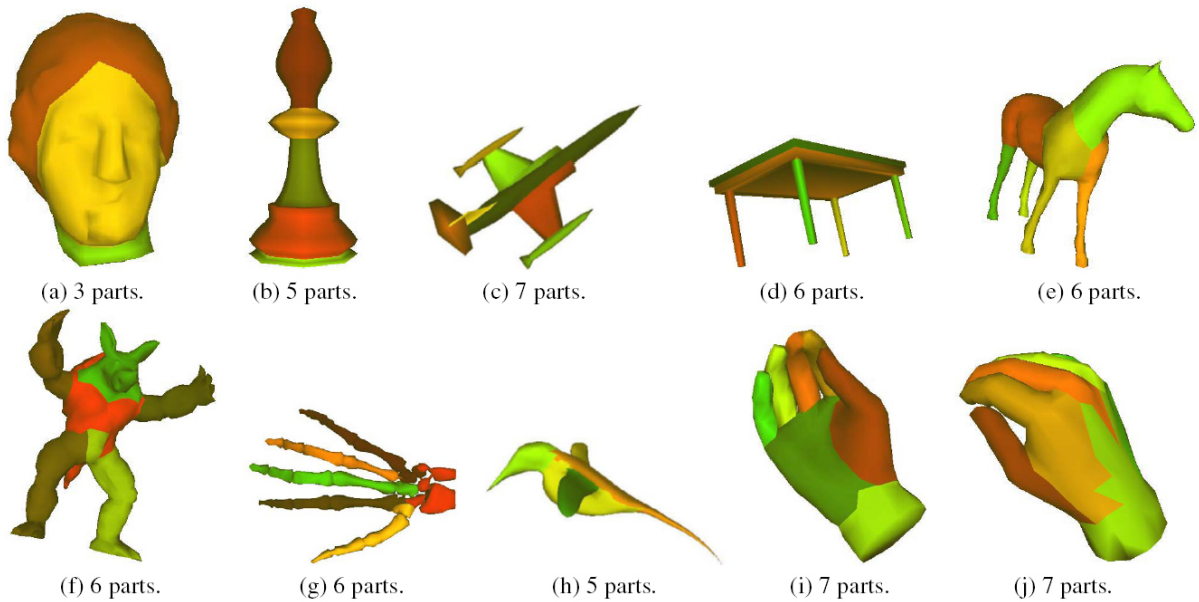


Figure 2.5: Results typical of Liu and Zhang's spectral clustering approach [63].

tion method using a queue. At each iteration, the segment with the currently largest area is segmented and the resulting two parts are placed back into the queue. The problem of achieving a binary segmentation is reduced to contour and sequence analysis by embedding the 3D shape into 2D and 1D by means of spectral embedding. Two operators are alternatively used to compute the projection: one for structural segmentation caused by branching and another for geometrical segmentation caused by convex regions. The authors also introduce a shape segmentability measure, which combines two convexity

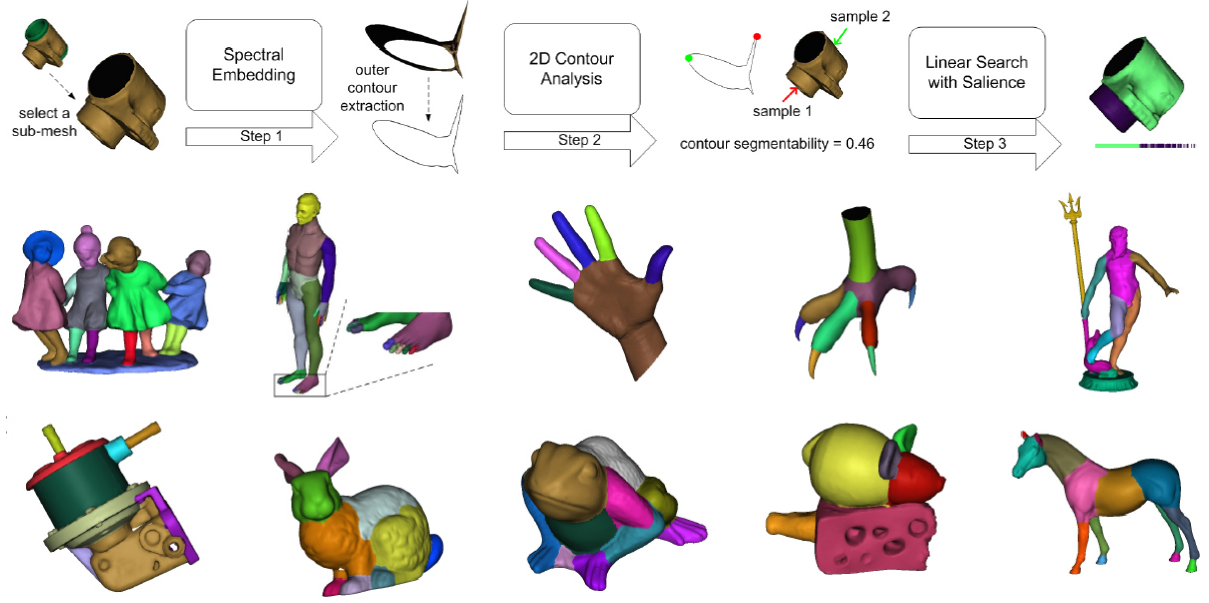


Figure 2.6: **Top:** Summary of Liu and Zhang’s spectral embedding and contour analysis method [64]. **Bottom:** Typical results of applying this method.

measures, and use it as a stopping criterion for the iterative segmentation process. The approach is summarized and characteristic results are presented in Figure 2.6.

More recently, Liu *et al.* propose a new metric which incorporates a novel encoding of volumetric context: the volumetric shape image (VSI) [65]. They augment the geodesic and angular affinities with this function which encodes the change in visibility of the interior of the shape across faces. This helps to better separate parts for segmentation applications. All three metrics (geodesic, affinity, and VSI) are then linearly combined with application dependant, hand-tuned weights.

Recently, Golovinsky and Funkhouser [30] propose a method in which randomized segmentation algorithms can be used to repeatedly sample segmentations using varying initializations; *e.g.* different randomly selected initial locations for k -means clusters, or seed locations for a min-cut algorithm, *etc.* Each such segmentation is accumulated into a “partition function” which keeps count of edges that appeared as boundary edges in each case. This function will then indicate edges that are most likely to be in any given

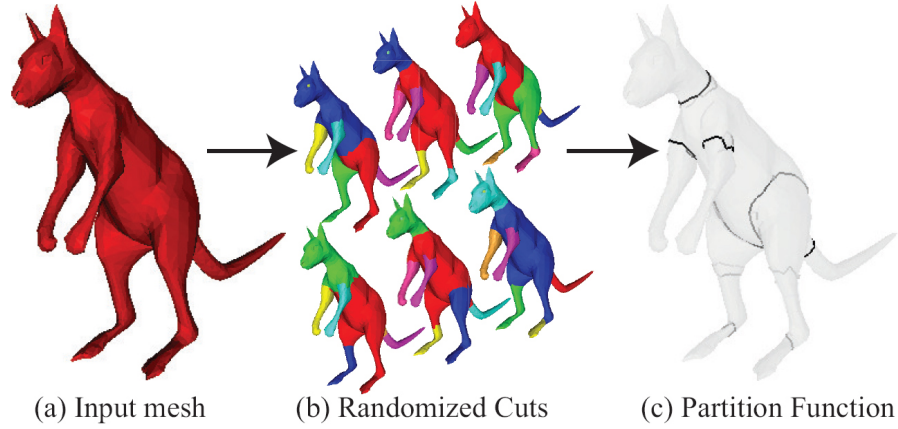


Figure 2.7: The approach of Golovinsky and Funkhouser [30] takes an input object (a) and, given a segmentation method, samples segmentations with randomized parameters (b) in order to estimate a partition function (c).

segmentation. Once this partition function is obtained, the final segmentation is achieved by means of a min-cut. Figure 2.7 summarizes this approach.

2.3.2 Model-fitting approaches

Lukács *et al.* use a model-fitting approach to segment range images allowing for spheres, cylinders, cones and toroids [69, 72]. The algorithm begins by selecting random seed patches to which a minimum-error model is fit. Then, each region is iteratively grown for a fixed number of steps, incorporating surface elements and adapting the model. When regions overlap, one of the two models is chosen based on criteria involving region area, error and quality of fit and simplicity of model (all things being equal, simpler models are preferred.) Models with poor fit are also pruned. Figure 2.8 illustrates a typical result of this method applied to a range image.

An alternative to region growing would be bottom-up hierarchical approaches, as presented, for example, by Attene *et al.* [8]. In this approach, planar, spherical and cylindrical models are used. Initially, each individual surface element is considered to be in its own segment and the best model is fit to each one. Then, the dual graph of

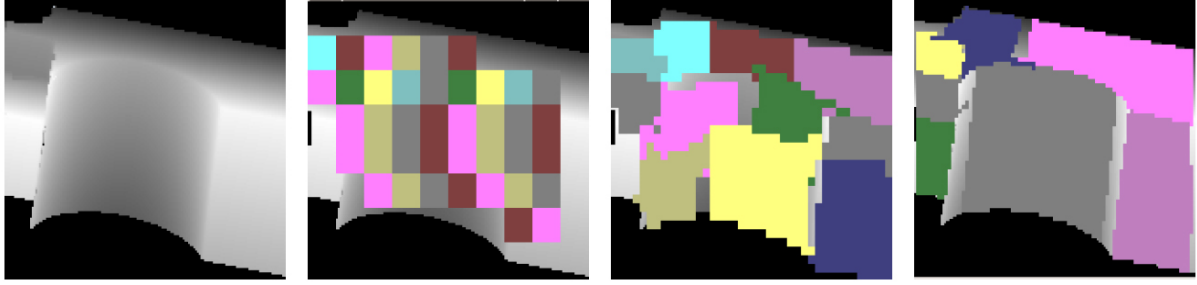


Figure 2.8: A result of applying the region-growing model-fitting method of Lukács *et al.* [69] to range data. From left to right: initial range scan, seed placement, intermediate grown regions and final segmentation.

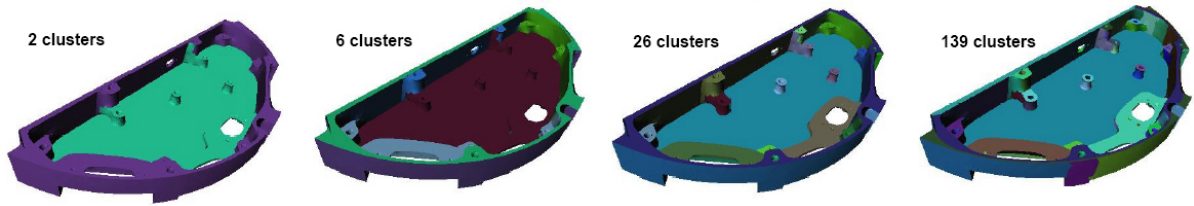


Figure 2.9: Result of applying the hierarchical method of Attene *et al.* [8] which uses planes, spheres and cylinders. Once the hierarchy of segmentations is complete, it is easy to obtain the segmentation corresponding to any number of segments.

the surface is considered, where edges correspond to a pair of adjacent segments. For each such edge, the cost of collapsing that edge is considered. An edge collapse entails removing the two models corresponding to each segment and finding the best model that fits the union of these two segments. The cost of the collapse is the error introduced by replacing the two models by just one. The algorithm then proceeds by iteratively collapsing the edge of lowest cost using a priority queue scheme. Once completed, the entire hierarchy of segmentations is available and the segmentation resulting for any number of segments can be very quickly obtained, allowing for real-time browsing. A typical result of Attene *et al.*'s method is illustrated in Figure 2.9.

An alternative to region-growing and hierarchical approaches are variational approaches. Such approaches consist of a fitting step, in which given a current segmen-

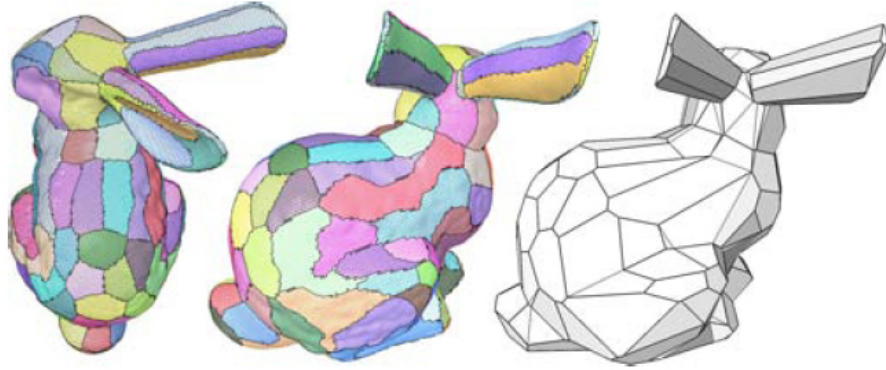


Figure 2.10: Cohen-Steiner *et al.* apply a variational approach using a planar model to segment a surface. The results are used in an adaptive mesh decimation scheme since the resulting flat regions can be well approximated by single polygonal face.

tation the best model is fit to each segment and a classification step, in which given the parameters for each model, the surface elements are classified according to the best (closest) model.

Cohen-Steiner *et al.* [20] introduce a variational approach to shape approximation. Their goal is, essentially, mesh decimation; *i.e.* to approximate the original mesh using a relatively small number of planar primitives. However, this can be viewed as a kind of model-fitting surface segmentation where the models used are planes. A queue-based flooding approach is used in classification step to ensure segment connectivity. A typical segmentation result along with the resulting mesh is illustrated in Figure 2.10.

Julius *al.* apply the variational method using cone primitives as a means to segment a surface into approximately developable patches. A developable patch is one that can be embedded in 2D (laid out flat) with no distortion. Finding developable segmentations of shapes has important applications including construction and surface texturing. The method uses cone primitives based on the observation that, given that a cone is developable, a segment that is well approximated by a cone should, itself, be approximately developable. The algorithm begins by selecting segment seeds using furthest point initialization. Regions are then grown, ensuring that an error tolerance parameter is not



Figure 2.11: Julius *et al.* use a variational method with cone primitives to segment a shape into approximately developable patches.

surpassed by growing new regions as needed. After convergence, a postprocessing step merges redundant patches and straightens boundaries to produce cleaner segmentations and facilitate their use in applications. Results from this approach are shown in Figure 2.11.

Yan *et al.* apply the variational approach using general quadric surface models [116]. The number of segments is chosen by iterative insertion; i.e. starting with one segment, a proxy is inserted at the point of highest error and the algorithm continued until the error is below a given threshold. As a postprocess, boundaries are made smoother by means of a graph cut. Figure 2.12 illustrates results of this approach.

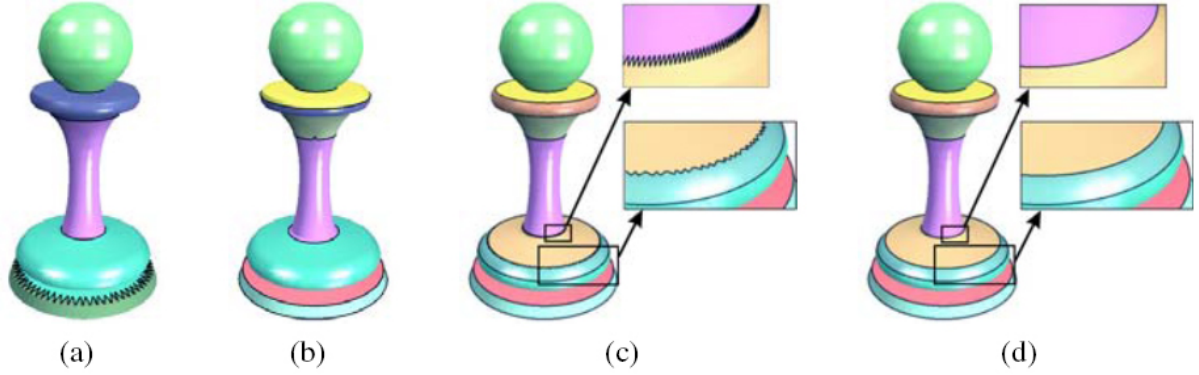


Figure 2.12: Yan *et al.* apply the variational framework using a general quadric model [116]. After each convergence, a new proxy is inserted at the point of highest error until the global error is below a threshold. (a) 5 proxies, (b) 9 proxies, and finally (c) 12 proxies. (d) boundaries are post processed using a graph cut approach.

2.3.3 Property-based approaches

Chazelle *et al.* explore algorithms for the segmentation of mesh surfaces according to convexity [18]. They propose three heuristic methods by which to achieve the segmentation: space partitioning, space sweeping and flooding. In the space partitioning approach, if the current object is not convex, a “reflex edge” is found (one with a convex angle), the shape is split into two along the plane of the edge, and the algorithm proceeds recursively. In the space sweeping approach a plane is moved across the shape and the resulting 2D intersection curve is decomposed into convex pieces. Each curve is maintained as long as possible as the plane moves, resulting in convex patches. Whenever a convexity violation is found, the curve is split and a new patch introduced. Finally in the flooding scheme, a region growing scheme is applied. Starting from a seed face, adjacent faces are added as long as the current region is still convex. As an alternative, a flood-and-retract approach is suggested, in which the original greedy flooding produces overlapping regions which are then retracted. Overall, this is a set of simple, early algorithms meant to handle noiseless, clean meshes where convexity is strictly enforced.

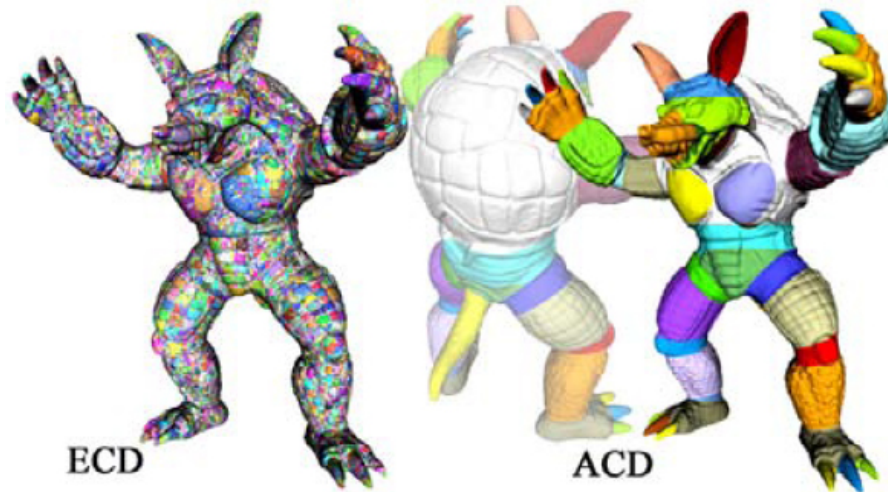


Figure 2.13: The method of Lien *et al.* allows for approximately convex decompositions (ACD) thus making it robust to noise and small features. Contrast their result to an exact convex decomposition (ECD).

More recently, Lien and Amato also address the convex segmentation problem, but in this case, relaxing the convexity property so that each segment need only be approximately convex, thus making it robust to noise and small features [61]. Their method works by identifying the surface points with highest degree of concavity (concave features) and to segment the surface so that the concavity of these points is reduced. The first step involves computing the convex hull of the current segment and finding areas that lay beneath clustered facets of the convex hull. These areas are called “pockets.” The edges of the corresponding convex hull facets are then projected onto the surface. The 1D curve is simplified into piece-wise linear elements and the knots are identified. Then these knots and their connecting curves are studied for a minimum cut local to the pocket, which is then expanded to a global cut of the segment in question. The process continues until the concavity at all points is below a user-given threshold. The method is far from straight-forward or intuitive, but produces high-quality results. Figure 2.13 presents a result from Lien and Amato’s method that also illustrates the difference between an exact and approximate convex segmentation.

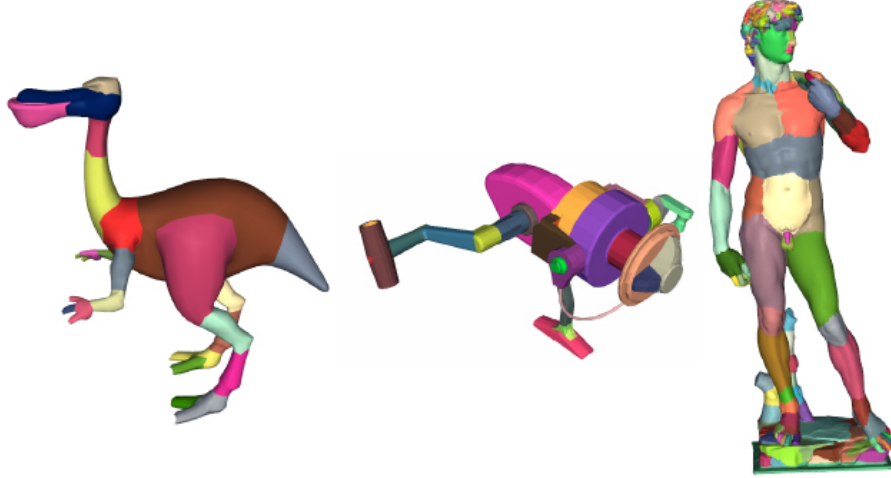


Figure 2.14: Kraevoy *et al.* apply a variational approach to the convexity based segmentation problem.

In a complementary approach, Kraevoy *et al.* propose a variational approach to convexity-based segmentation [54, 55]. This approach proposes a continuous convexity measure which involves the integral distance of a part to its convex hull measured along surface normals. This measure is combined with a measure of compactness defined as a unitless ratio of a segment's surface area to volume. In a selection stage, the surface element closest to the previous convex hull is chosen as a seed for each segment and all other faces are marked unassigned. During a growing stage, unassigned faces are added by region growing in order of least increase in error. The stages are alternated until no changes in classification occur. Figure 2.14 illustrates results typical of this approach.

Lu *et al.* apply a variational approach to segmenting shapes represented as voxel sets with the objective of having tight bounding ellipsoids and oriented bounding boxes. The segment with the highest bounding error is split and the algorithm iterated to convergence until the global error is below a given tolerance. As a postprocess, adjacent segments which can be merged without violating the tolerance are merged. A typical result summarizes the method in Figure 2.15

Mitra *et al.* [78] propose a transformation clustering approach for the detection

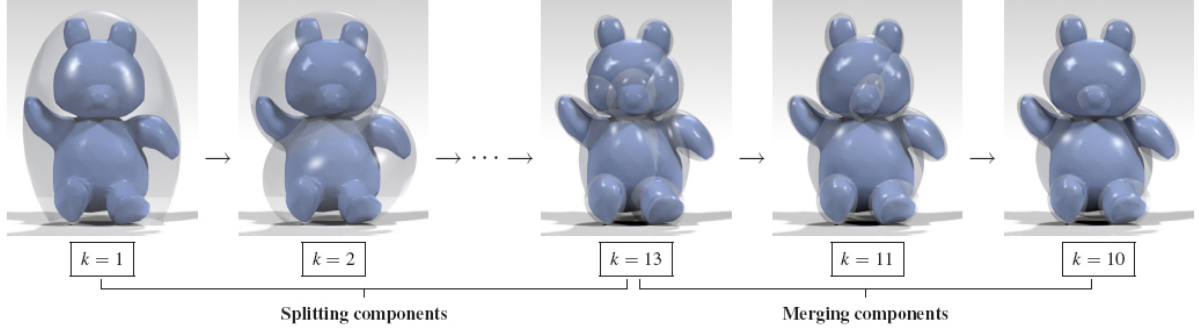


Figure 2.15: Lu *et al.* apply variational approach to find tight bounding volumes for voxelized shapes [68]. Segments are added by splitting the segment with highest error until a given tolerance parameter is not violated. Then, adjacent segments are merged as long as doing so does not violate the tolerance.



Figure 2.16: Local symmetries detected by the method of Mitra *et al.* Left: original sculpted model; center: local symmetries detected; right: color-coded deviations from perfect symmetry as a fraction of the bounding box diagonal [78].

of local approximate symmetries. In this approach, these may include reflective and rotational symmetries, as well as translation and uniform scaling transformations.

The method considers pairs of surface points along with the principal curvature values and directions at said point. Rotation is determined through alignment of the principal curvature directions along with the surface normal, uniform scaling is determined through ratio of principal curvatures, and translation is determined from the surface point coordinates. Reflections are handled by reflecting the model about an arbitrary but fixed plane. Thus, each such pair determines a transformation that can be expressed

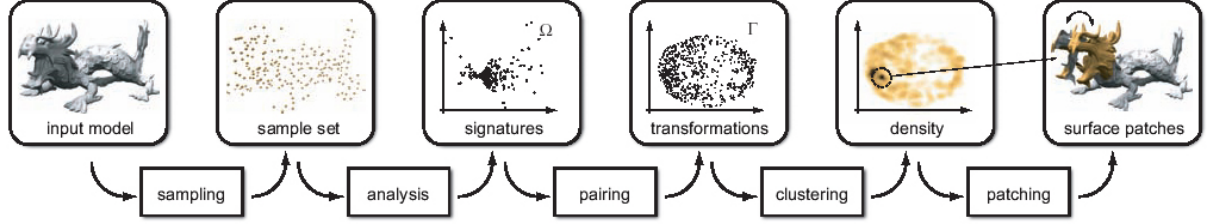


Figure 2.17: Summary of the steps followed by the algorithm of Mitra *et al.* [78].

as a vector of seven scalars (one scalar for uniform scaling, three scalars for rotation, and three scalars for translation.)

Now, pairs of a number of uniform random samples of the surface are considered, however not all such points will determine a unique transformation. Umbilic points are invariant under rotations along the surface normal. Such points will not determine a unique transformation and are therefore pruned as candidates. Only points with distinct principal curvatures are considered. Furthermore, certain pairs may not determine a valid transformation. Only pairs of compatible points are considered, thus further reducing clutter in the transformation space.

Now that the space of possible symmetries has been sampled, representative symmetries are extracted from the transform space through mean shift clustering. This type of clustering has the advantage that the number of symmetries to be extracted as well as the shape of the clusters need not be pre-specified. Each extracted mode represents a symmetry (possibly local and approximate) which can now be verified. This verification step is necessary since it is possible (though unlikely) that many uncorrelated points agree on the same transformation. The steps of the method are summarized in figure 2.17.

Podolak *et al.* [89] also propose a pairwise sampling of symmetries, in their case, limited to the planar reflective variety, in their *planar-reflective symmetry transform* (PRST). For any given plane, the PRST indicates the degree of symmetry which the object exhibits with respect to it. For efficiency of computation, the authors propose a

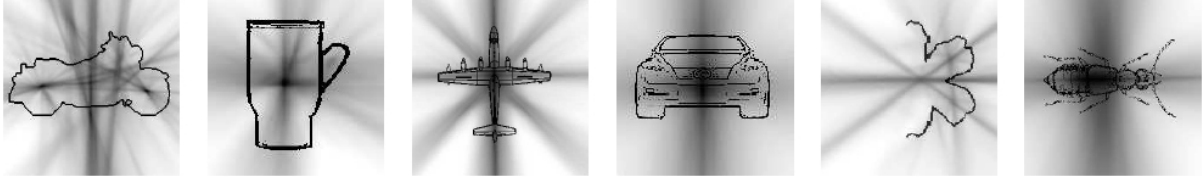


Figure 2.18: Illustration of the planar-reflective symmetry transform (PRST) of Podolak *et al.* The image pixels are colored in proportion to the degree of symmetry of the axis passing through it with the highest degree symmetry with respect to the overlaid silhouette. [89].

Monte Carlo framework, in which pairs of randomly selected points vote for the plane between them. These votes are accumulated in discrete bins over polar parameters in a manner reminiscent of the Hough transform, and local maxima are later refined. These local maxima correspond to planes for which the object exhibits a degree of local or global symmetry. The PRST is illustrated for several 2D shapes in Figure 2.18. The image pixels are colored in proportion to the degree of symmetry of the axis passing through it with the highest degree symmetry with respect to the overlaid silhouette.

The PRST can be used to segment a mesh. The approach uses a k -means clustering approach based on support for local maxima in the PRST. The local maxima of the PRST are found and then for each face and each such plane, the degree to which the face contributes to the symmetry with respect to the plane is computed. For each face, this set of scores is treated as a feature vector for the face clustering, intuitively, grouping faces which support the same subset of planes. The method proceeds hierarchically (with $k = 2$), at each step selecting the two largest connected components and refining the segmentation boundary by computing a weighted min-cut along the boundary edges. This proceeds until a user defined depth or until the planes of local maxima reflect more than 90% or less than 10% of the surface onto itself. The results of this symmetry driven segmentation approach are illustrated 2.19.

Thrun and Wegbreit present an approach taking a single range image as input where

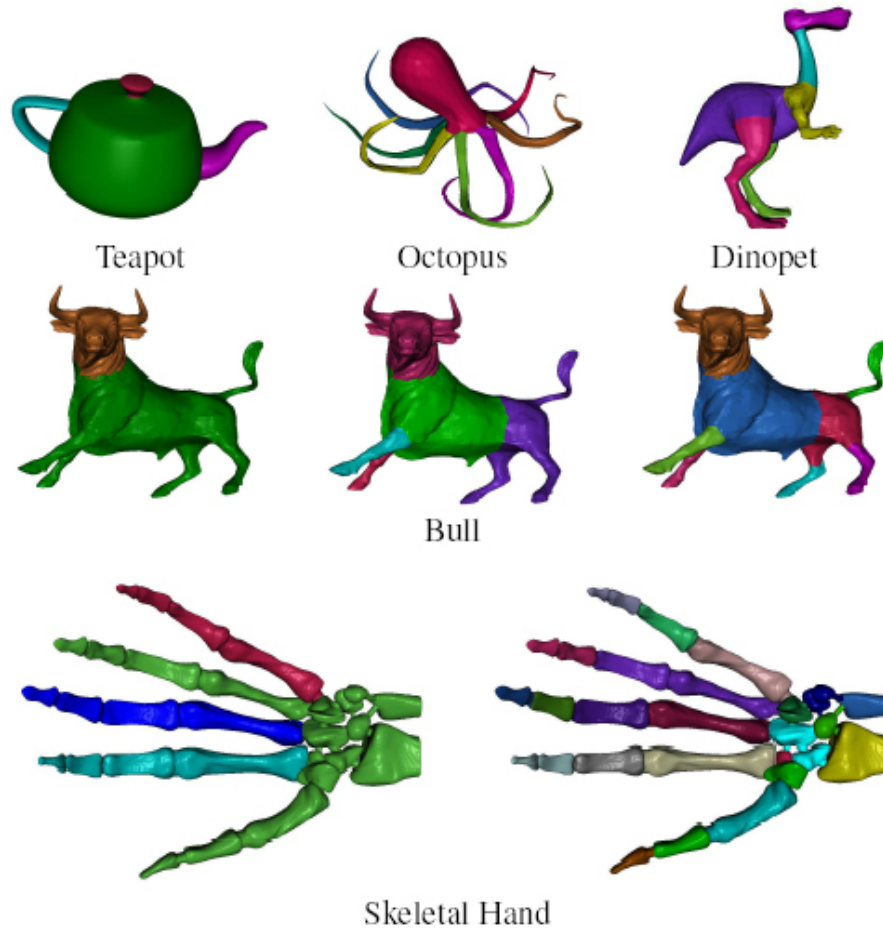


Figure 2.19: Segmentations obtained by the method of Podolak *et al.* based on the PRST. In particular the bull is shown with 2, 4 and 8 segments, and the hand is shown with 4 and 18 segments [89].

the goal is to reconstruct the full model using symmetry to fill in the occluded parts. In contrast with previous approaches, local symmetry is used. Candidate local symmetries are scored using a probabilistic measurement model. The evaluation function includes a Gaussian weighted distance to the nearest visible point and penalizes points that are mapped to visible space but that are not matched by the visible surface. Symmetry types are organized into a hierarchy where more general symmetries (ones that imply others) are searched for first. In order to find local symmetries, the method follows a localized region growing technique starting from a number of points not yet claimed by

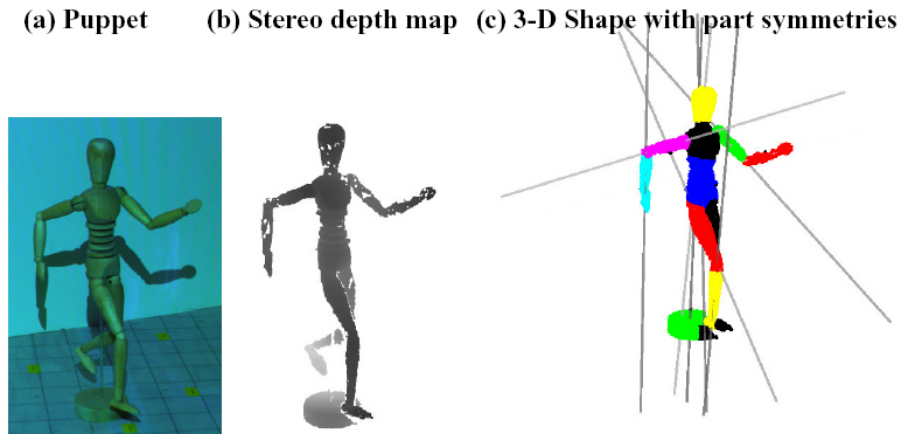


Figure 2.20: The approach of Thrun and Wegbreit reconstructs an articulated figure from a single range scan. The algorithm was pre-constrained to only search for axial symmetry [109].

other symmetries. Figure 2.20 shows a result of the method of Thrun and Wegbreit in which an articulated figure is reconstructed from a single range scan. In this result the algorithm was pre-constrained to only search for axial symmetry.

Mortara *et al.* present a method by which to segment tubular regions from a surface [79]. The method is based on the idea that, when intersected with a sphere of appropriate radius, a tubular segment will result in two closed intersection curves. In contrast, when the intersection produces one closed curve, the region is classified as a topological disc, and a result of three or more curves indicates a branching point. A discrete set of radii is exhaustively explored, it being explicitly given by the user or by found by uniformly sampling the interval ranging from the minimum edge length of the mesh to its bounding box diagonal. Once tube vertices are detected, tube centers are found by iterative refinement using the sphere-intersection curves and their centroids. Figure 2.21 illustrates the sphere-surface intersections and a typical result of this method.

Liu *et al.* segment out geometric reliefs from a surface separating it from the uniform texture of the background. Their approach begins with a loosely-drawn user contour and then uses a snake (active contour) approach to move the contour inward toward the relief

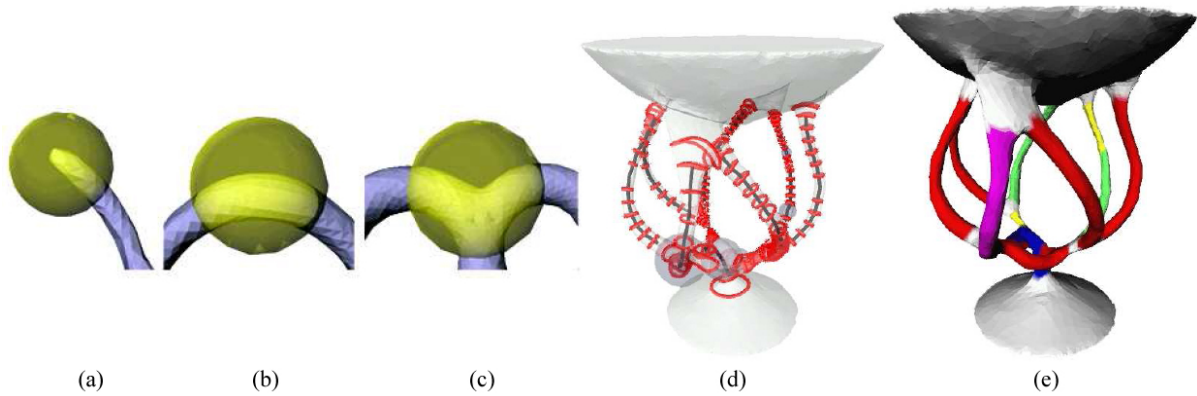


Figure 2.21: Mortara *et al.* segment out tubular regions of a shape by using sphere intersections and analyzing the resulting curves [79]. Parts are classified into disc (a), tube (b), or branching (c) regions according to the number of intersections for a given scale (sphere radius). Tube centers are iteratively refined (d), and a final segmentation of tubular regions is obtained (e).



Figure 2.22: Liu *et al.* present a snake (active contour) approach which segments a relief from a textured background starting from a user-selected, loosely-drawn initialization [66, 67].

boundary. The evolution of the contour is done in two steps, the first proceeding on a smoothed version of the surface to quickly arrive at a coarse placement near the relief, and the second refining the contour on the original, un-smoothed surface. The energy function is a combination of terms including planarity and “relief step”, which attempts to capture the jump in height that occurs when crossing over from the texture into the relief region. Figure 2.22 illustrates a typical result from this approach.



Figure 2.23: Lien presents a method for segmenting surfaces into locally star-shaped segments [60]. Guards are iteratively chosen from the set of yet-unseen points and refined until no unseen points remain.

Lien presents a method to segment surfaces into locally star-shaped segments [60]. These are segments for which there exists an internal point which can “see” all surface points on the segment. This point which can see all others is called a guard. The algorithm works by iterative insertion of guards. At each iteration, an initial guess for a guard is selected from those points which are currently not visible by any other guard. This guess is then refined so that it sees a maximal area and its visible points are marked as such. The algorithm terminates when no surface points remain unseen. Figure 2.23 illustrates a result from this algorithm.

Finally, shape skeletonization can be used as a means to segment shapes by labeling skeleton limbs and labeling each surface element by associating them to a unique limb.

Au *et al.* present an elegant approach to skeletonization based on a constrained smoothing of the mesh. The surface is smoothed until it collapses into a central 1D

curve, and then simplified by iterative edge removal while maintaining the association to the original mesh vertices. This approach assumes a clean, water tight mesh.

Sharf *et al.* present a robust approach to skeletonization which is applicable to point clouds [4] with noise and holes (missing data.) The method is based on the use of a deformable model which is “inflated” inside the shape while tracking its evolving fronts. As these fronts branch out inside the shape, they create the shape’s centerline and here too, surface-skeleton correspondence is available.

Tierny *et al.* extract an enhanced topological skeleton [110]. This is done by, first, finding surface features corresponding to geodesic extrema. Then, a surface function is computed as the geodesic distance to the closest such feature and then used to extract and refine a Reeb graph skeleton.

In these approaches, by labeling the shape skeleton into unique limbs, a segmentation of the surface is immediately obtained through correspondence.

Alternatively, an approach by Reniers and Telea uses voxel thinning [91]. Since voxelization is relatively easy for a large variety of shapes, this imposes less restrictions on shape topology and the original surface’s tessellation quality. However, in this approach, the association between surface points and skeleton points is lost. In order to obtain a surface segmentation, for each skeleton node, a minimum-distance closed surface geodesic is computed which partitions the shape in two. The algorithm proceeds until all nodes are thus exhausted.

Figure 2.24 illustrates results from these approaches.

We’ve discussed how existing segmentation methods can be broadly categorized into affinity-based, model-fitting, and property-based approaches. We explored each of these categories and discussed the common algorithms and initialization approaches that appear in each and illustrated the type of results and applications that each method is suited for. There can clearly be no one “best” segmentation approach since, for a given shape, there are many ways in which a user may wish to segment it depending on the

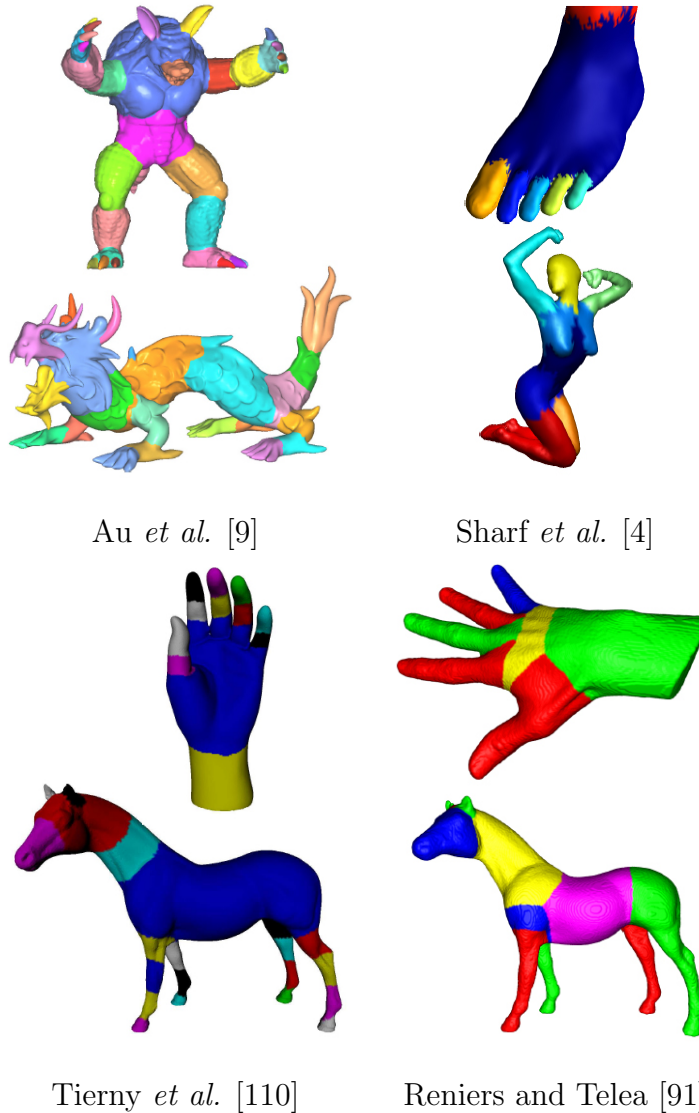


Figure 2.24: Segmentation results from skeletonization approaches.

application. Each individual problem will have its own difficult cases and challenges.

The rest of this thesis will present our contributions to segmentation algorithms. In the Chapter 3 we will discuss our contribution to model-fitting segmentation approaches: a variational approach for segmentation into ellipsoidal regions with applications to surface and volumetric approximations, compact representations, as well as remeshing.

Chapter 4 will present our contribution to property-based segmentation algorithms: a robust-statistical method for segmenting out surface regions which exhibit local approximate symmetry and explore an application to compact representations based on this

form of redundancy.

Finally, Chapter 5 will present a method which generalizes property-based methods so that the segmentation algorithm is decoupled and allows for surface segmentation based on heterogeneous properties, each of which can apply to a single segment or several segments and their interrelations. We will show applications to semantically consistent segmentation, bounding volume hierarchy computation, animation skeleton placement for automatic character rigging, and automated clothes fitting for animation characters.

Chapter 3

A model-fitting approach: Ellipsoidal segmentation

In this chapter we will present a contribution to model-fitting algorithms by means of a variational approach to ellipsoidal segmentation which can be used for surface and volumetric segmentation. Given a surface-based segmentation, we show how the original shape can be approximated by an ellipsoidal representation consisting of the union of piece-wise ellipsoidal regions represented by a set of ellipsoid parameters and a region boundary lying on the ellipsoid's surface. We also introduce algorithms for re-meshing such a representation by means of either regular parametric sampling or iterative optimal vertex insertion.

3.1 Motivation and background

The main goal of this chapter is to explore the potential of approximating the surface of a dense mesh using a relatively small number of ellipsoidal regions. Such an approximation allows us to address a number of issues related to mesh processing including geometric queries, transformations, efficient rendering and compact representation. We are motivated by the fact that most organic objects and many manufactured objects have

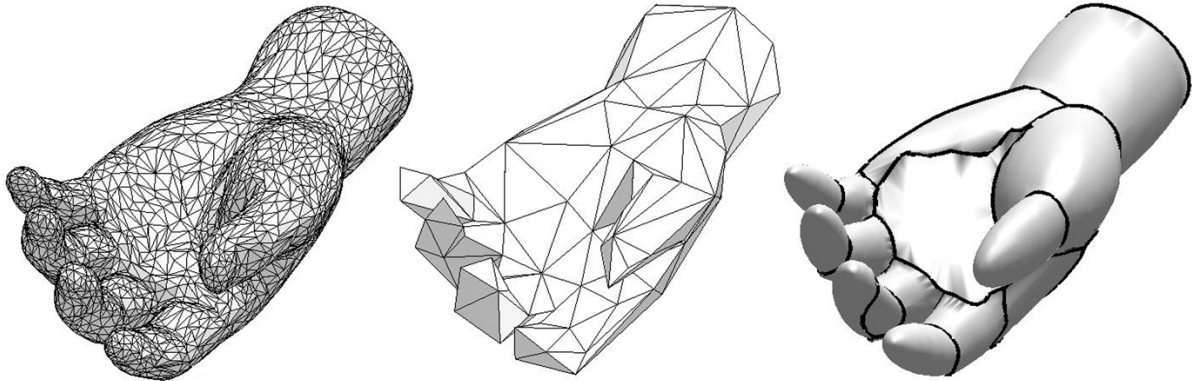


Figure 3.1: Ellipsoidal surface regions are ideal for modeling curved surfaces often found in organic objects. Left: The original triangulated mesh representing a hand using 5k faces. Centre: A decimated version using 180 faces. Right: An ellipsoidal approximation using 18 primitives and their boundaries.

large curved areas. Industrial designers working in physical media often use characteristically shaped tools like French curves [102], sweeps and steels that have traditionally been constructed using conic curve and quadric surface sections.

In this chapter we consider the ellipsoid as a representative quadric with which to approximate 3D objects. Ellipsoids have certain advantages over planes as an approximation primitive. The processing of various common geometric queries such as normal, curvature, proximity and intersection with a ray for an ellipsoid is analytic and equally efficient for ellipsoids as for planes. For curved regions not only are far fewer ellipsoids required for approximation but they have the potential of capturing local shape curvature more accurately. In areas where the surface of interest is flat, a region from an ellipsoid with sufficiently large radii can be used. Figure 3.1 compares the visual results of modeling a hand decimated using planar faces (we apply GSI’s decimation which uses a variant of the MAPS algorithm [58]) and an approximation of superior visual quality using an order of magnitude fewer ellipsoids.

Ellipsoids, unlike planes, being closed surfaces, are also capable of providing informa-

tion about the volume of closed objects. Our algorithm also allows for ellipsoidal fitting that spans from being surface oriented, where a number of ellipsoidal surface regions approximate the surface as best possible, to volume oriented, where the same number of ellipsoids are used to capture the overall volume enclosed by an object. In this case, we use the entire ellipsoid and not just regions of its surface.

The argument for not using primitives constructed with cubic or higher degree polynomials is one of marginal gains in terms of their ability to approximate curved geometry better than quadrics and that evaluation of various geometric queries on higher order primitives is not as efficient. One disadvantage of ellipsoids that we recognize is that they do not capture saddles or regions where the principal curvatures are convex and concave. We note, however, that human perception typically segments shape along such concave boundaries [52, 14]. The inability of the ellipsoid to capture the local curvature of such a shape fragment automatically favours segmentation along such boundaries.

Once a shape has been segmented into an ellipsoidal approximation, it is only necessary to store the ellipsoids and the segmentation boundaries at which they meet in order to represent the object. We will present and contrast two schemes for remeshing from such a representation.

Although there have been many greedy techniques for dense mesh simplification, Hoppe *et al.* [40] present an optimization-oriented approach. They introduce an energy function that models the desire for small meshes as well as fidelity of representation. By adjusting the weight of these two opposing goals, it is possible to choose how much simplification versus fidelity to the original mesh is desired.

The method of Cohen-Steiner *et al.*, as discussed in Chapter 2, can be used for remeshing with the placement of planar primitives optimally with respect to the original mesh [20]. Curved regions, however, still require many such planes in order to be well approximated.

In [15], Bischoff *et al.* propose a representation for 3D models based on ellipsoids.

However, a relatively large number of primitives is used and they retain the entire ellipsoidal surface. Their motivation for their variant of the ellipsoid decomposition is the robust transmission of geometric objects. It represents a coarse approximation which is then increasingly refined by the transmission of the mesh's original vertices.

Pentland, in [86], used superquadrics as geometric primitives for representing 3D objects. The fitting of superquadrics to range data has been well studied [104, 43]. In these cases however, the motivation was not simply that of a compact representation, but the visual recognition of different objects based on this segmentation.

3.2 Error metric for ellipsoidal surface regions

Let us assume we have a triangular mesh surface region R which is a set of mesh faces, that we are approximating with an ellipsoidal primitive P . We wish to know how well P approximates said region. In other words, we would like to define an error function $E(R, P)$.

We begin by considering the distance between a given vertex v_i and an ellipsoid P . There are several distance metrics that one may consider when defining the error function. One such function is Euclidean distance. In the case of ellipsoids, it is simpler to compute the radial distance from a point to its surface.

Euclidean distance: We defined Euclidean distance from a vertex v_i to ellipsoid P , as the Euclidean distance from vertex v_i to its radial projection on ellipsoid P , denoted $\Pi_P(v_i)$.

$$E_{euc}(v_i, P) = \|v_i - \Pi_P(v_i)\|^2$$

The vertex v_i will often have associated a vector $\mathbf{n}_i$ indicating the surface normal at said vertex. It may be desirable for our ellipsoidal primitive to approximate these surface normals as well. Thus, we introduce an angular distance.

Angular distance: The angular distance between vertex v_i and ellipsoid P is a

measure of the angle between the vertex normal at v_i , denoted $\mathbf{n}_i$, and the ellipsoidal surface normal at the radial projection of v_i on ellipsoid P , denoted $\mathbf{n}_P(v_i)$. We define said angular distance as

$$E_{ang}(v_i, P) = \|\mathbf{n}_i - \mathbf{n}_P(v_i)\|^2$$

Finally, given a surface mesh, it is possible to approximate curvature values given for the data. In our case we use a mean curvature estimate for meshes presented in [74]. As in the case of the vertex normals, it may be desirable for the ellipsoidal surfaces to approximate these values as well. For this purpose, we introduce the curvature distance metric.

Curvature distance: The curvature distance between vertex v_i and ellipsoid P measures how well the P follows mesh curvature at said vertex. We define the curvature distance between v_i and ellipsoid P as the magnitude of the difference between the estimated mean curvature at v_i , denoted H_i , and the mean curvature on the ellipsoid at the radial projection of v_i , denoted $H_P(v_i)$.

$$E_{cur}(v_i, P) = (H_i - H_P(v_i))^2$$

An argument against using Gaussian curvature is that it would always be positive, even when considering the inner surface (see negative ellipsoids below.) At a saddle point in the data, the Gaussian curvature is negative, while the mean curvature is zero. Our metric models the fact that a “flat” ellipsoid (with mean curvature close to zero at the point of interest by virtue of very large radii) could approximate this small region.

The three metrics are illustrated schematically in Figure 3.2.

Composite distance: Now, we can compose these error metrics into a single value for v_i and P by taking a weighted sum

$$E(v_i, P) = \alpha E_{euc}(v_i, P) + \beta E_{ang}(v_i, P) + \gamma E_{cur}(v_i, P)$$

The values of α , β , and γ indicate the relative importance that we place on each of the individual error metrics.

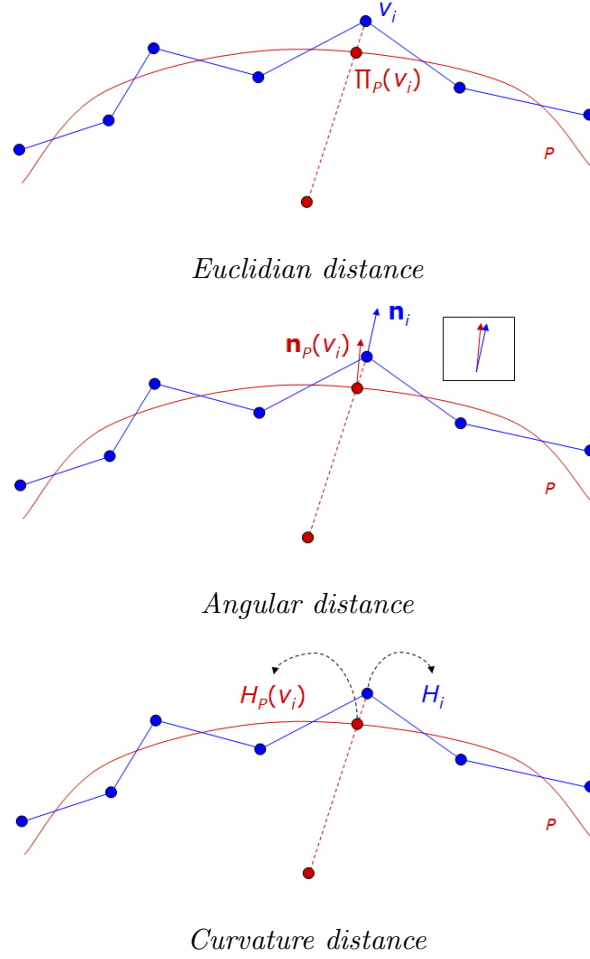


Figure 3.2: Illustration of distance metrics used in defining ellipsoidal approximation error.

Given a triangular mesh face $f_j = (v_1, v_2, v_3)$, we define its distance to the ellipsoid P as follows

$$E(f_j, P) = \sum_i E(v_i, P)$$

Now, given a mesh surface region $R = \{f_1, \dots, f_n\}$, we define the fitting error of P with respect to region R as the sum of errors for each face f_i to P .

$$E(R, P) = \sum_i E(f_i, P)$$

If the mesh is partitioned into n regions $R_1, \dots, R_n$, each approximated by an ellipsoid

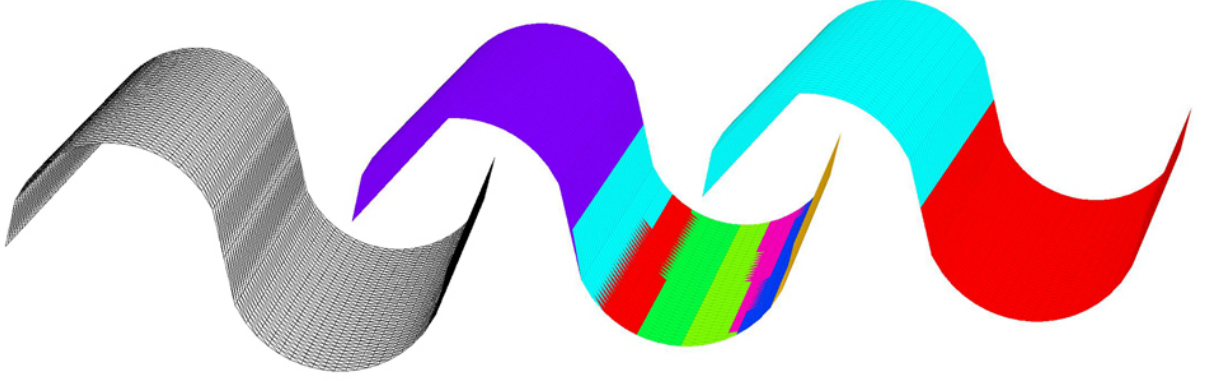


Figure 3.3: Without negative ellipsoids positive curvature is captured by one ellipsoid, but the negative curvature region is approximated by the seven remaining. With negative ellipsoids one ellipsoid is sufficient for each region.

$P_1, \dots, P_n$, the error of the approximation is defined as the sum of errors at each region.

$$E(\{R_1, \dots, R_n\}, \{P_1, \dots, P_n\}) = \sum_i E(R_i, P_i)$$

We should note that there can be additions and variations to these definitions. For example, when calculating the error for a region, each face's error can be weighted by the face's area, thus indicating that larger faces are more important.

A typical value for α in our case is $\alpha = 1/a^2$ where a is the mesh's average edge length. This helps normalize the weight of the Euclidean metric for mesh's sampled at different resolutions. In the case of β and γ , they are not dependent on mesh scale (the curvature estimate we use [74] normalizes for varying face areas.) We tried different weights and found settings of 1.2 and 5×10^{-4} , respectively, to work well.

Negative ellipsoids

When modeling surfaces using ellipsoidal surface sections we are not able to capture bowl-shaped concavities, i.e. regions of negative curvature, if we consider only the ellipsoid's outer surface. In order to be able to capture these types of regions while still using the same type of primitive we introduce the notion of a *negative ellipsoid*.

In essence, a negative ellipsoid has the same shape as a regular ellipsoid, except that we consider its inner surface when measuring angular distance between normals and curvature. Thus, we extend the relevant previous definitions as follows:

$$E_{ang}(v_i, P) = \begin{cases} \|\mathbf{n}_i - \mathbf{n}_P(v_i)\|^2 & \text{if } P \text{ is pos.} \\ \|\mathbf{n}_i + \mathbf{n}_P(v_i)\|^2 & \text{if } P \text{ is neg.} \end{cases}$$

$$E_{cur}(v_i, P) = \begin{cases} (H_i - H_P(v_i))^2, & \text{if } P \text{ is pos.} \\ (H_i + H_P(v_i))^2, & \text{if } P \text{ is neg.} \end{cases}$$

In essence, we are simply considering the negative ellipsoid to have its surface normals pointing inward and its mean curvature to be the compliment of the mean curvature defined on the outer surface. Figure 3.3 illustrates the effect of negative ellipsoids to surface segmentation.

3.3 Error metric for ellipsoid volume

We mentioned earlier that ellipsoids could also be used to represent a region by approximating not only its shape, but also its volume. Here we introduce an error metric to be used when this is intended.

$$E_{vol}(R, P) = \alpha E_{euc}(R, P) + \delta (\hat{\mathcal{V}}(R) - \frac{4}{3}\pi abc)^2$$

where a, b , and c are the radii of P , $\hat{\mathcal{V}}(R)$ is the approximate volume englobed by mesh region R , and δ is a weight determining the relative importance of approximating volume versus fitting the data points to the ellipsoid's surface.

A typical weight for δ in our case is $\delta = 10^5/\hat{\mathcal{V}}^2$, where $\hat{\mathcal{V}}$ is the approximate volume of the entire mesh. The denominator ensures that the volume penalty term is normalized for meshes of different volumes, while the numerator's large magnitude shows a preference for approximating each region's volume over fitting the points individually.

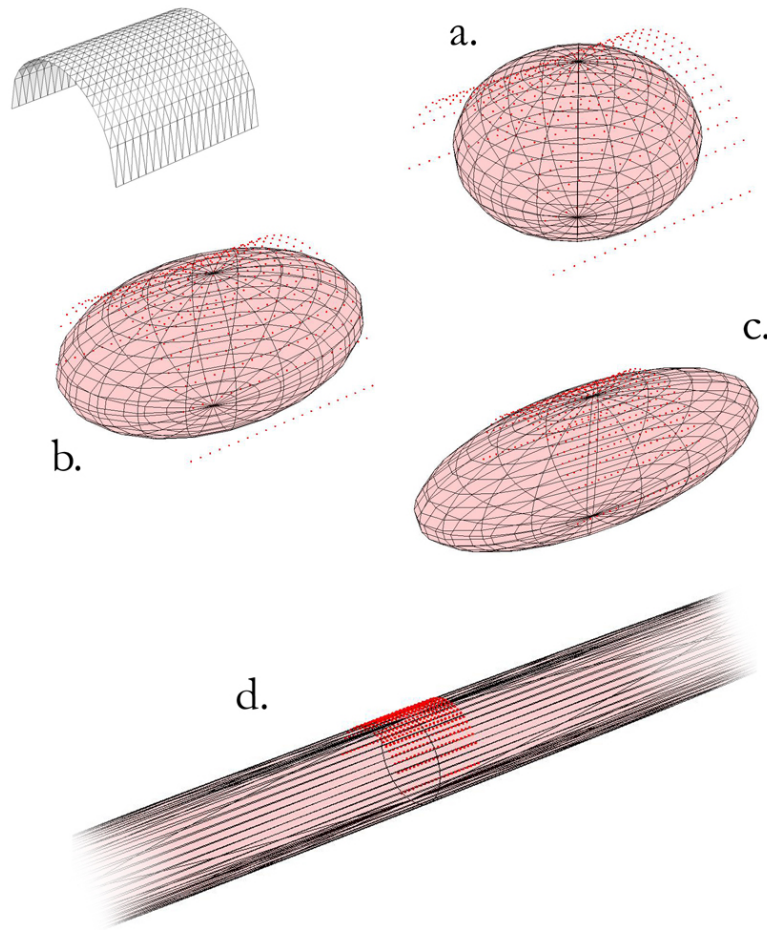


Figure 3.4: As we decrease the relative magnitude of δ in the volume fitting error metric, the ellipsoid fitting transitions from approximating volume, to approximating surface. a. $\delta = 10^{10}/\hat{\mathcal{V}}^2$, b. $\delta = 10^5/\hat{\mathcal{V}}^2$, c. $\delta = 10^2/\hat{\mathcal{V}}^2$, d. $\delta = 0$.

Figure 3.4 illustrates the relevance of the volume penalty weight δ . The mesh is that of a semi-cylindrical section. Initially, the fitting of the ellipsoid using the volume metric with a high δ value approximates the volume enclosed by the convex hull of the section. As we decrease the relative magnitude of δ in the error metric, the ellipsoid fitting transitions from approximating volume, to approximating surface.

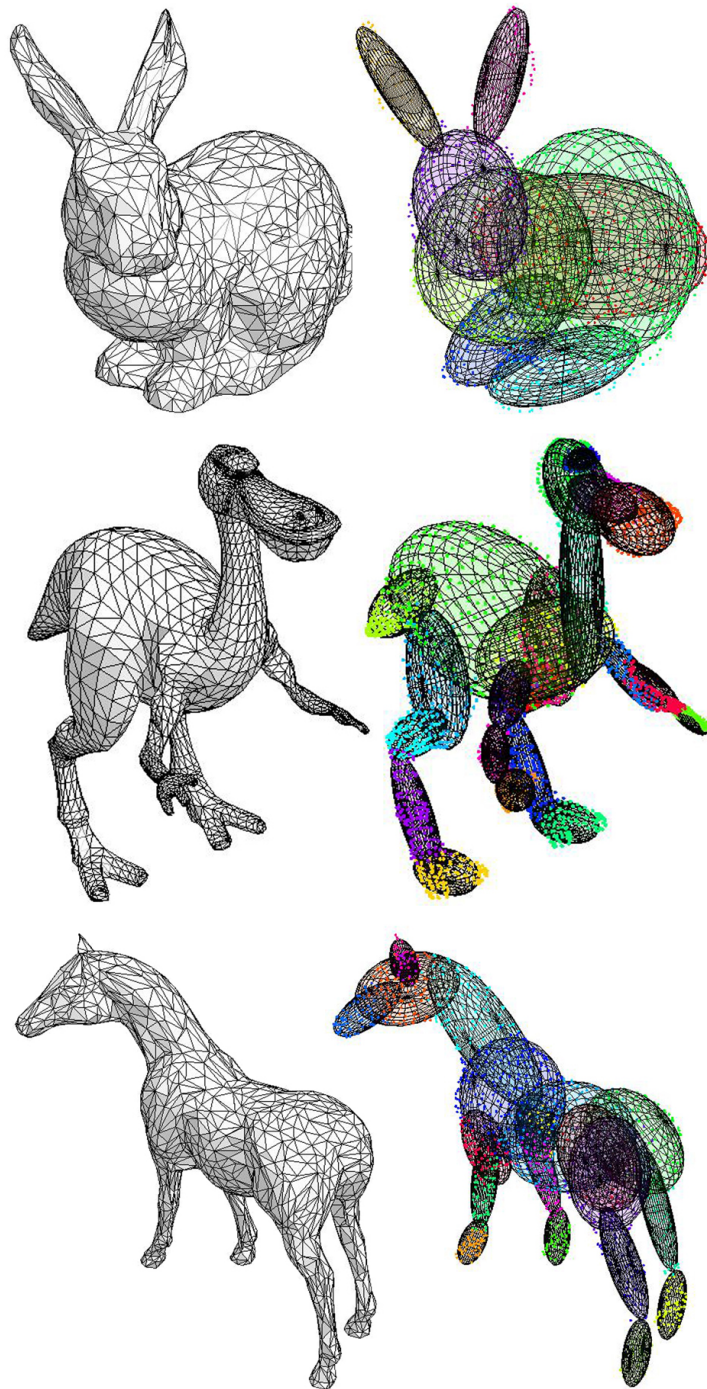


Figure 3.5: Examples of volume segmentation: bunny (8 ellipsoids), dino (22 ellipsoids) and horse (20 ellipsoids).

3.4 Ellipsoidal segmentation

Our segmentation approach is an instance of a variational approach, as discussed in Chapter 2. Starting with an initial classification, the algorithm alternates between a fitting step, and a classification step. In the fitting step, we update each P_i with the ellipsoid that minimizes $E(R_i, P_i)$. In the classification step, the regions R_i are re-computed, assigning each face f_j of the mesh to the region that minimizes $E(f_j, P_i)$ while also under the constraint that the regions must remain connected. To this purpose we implement the priority queue region flooding scheme found in [20], inserting faces to the queue based on their error to their associated ellipsoid. We also implement the region teleportation scheme to avoid local minima. The reader is referred to the publication for greater details.

Ellipsoid fitting We parameterize ellipsoid primitives as tuple of nine scalars, three to represent the ellipsoids centre, three to indicate the length of each radius, and three to indicate its alignment. When considering negative ellipsoids in the case of surface segmentation, we also store if the ellipsoid is positive or negative.

For each mesh region R_i of the current segmentation, we find the P_i that minimizes $E(R_i, P_i)$. Given that the sign value is discrete, we first minimize fixing the sign to be positive, then fixing it to be negative, and keep the result with lowest error.

The minimization requires an initial estimate of the parameters. We estimate the ellipsoid’s centre as the geometric centroid of the vertices of the current region. In order to estimate the alignment and radii, we perform a singular value decomposition of the regions vertices, centering their mean at the origin. The resulting eigenvectors estimate the regions orientation and the eigenvalues allow us to estimate the radii.

Termination: The algorithm terminates when the desired number of iterations have been performed, reporting the segmentation with least error found. Should the algorithm converge to a local minimum, it may continue its search for a better segmentation simply by teleporting a region as mentioned above.

Figure 3.5 shows examples of volume segmentation. In these cases, we set a high

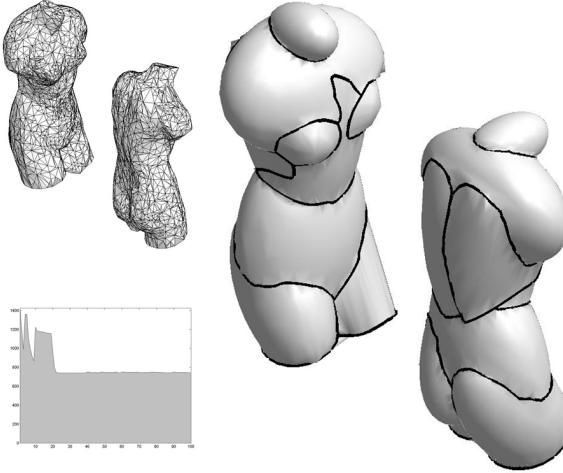


Figure 3.6: Top left: original mesh, front and back views. Right: obtained ellipsoidal representation with smoothed boundaries, 12 ellipsoids. Bottom left: error over iterations. Our reclassification flooding scheme and region teleportation can sometimes increase error. After several iterations however, the algorithm stabilizes on a minimum.

volume error weight δ . This ensures that volume is approximated as closely as possible and only then is the placement of the ellipsoid modified to fit the surface. For our volume approximation $\hat{\mathcal{V}}$ of each mesh segment we use the volume of the convex hull.

The top of figure 3.9 shows the results of surface segmentation. Here the actual ellipsoid's volume is not considered and only the fit to the surface is taken into account. Figures 3.1 and 3.6 show extracted ellipsoidal representations.

3.4.1 Smoothing segmentation boundaries

Once we have obtained an ellipsoidal segmentation, the segmentation boundaries are given by edges whose two adjacent faces have been assigned to different regions. Smoothing of these segmentation boundaries may be desired. To achieve this, we perform a constrained relaxation of the boundary vertices. For each boundary point p , we consider its boundary neighbours as well as its radial projection to all associated ellipsoids. Then

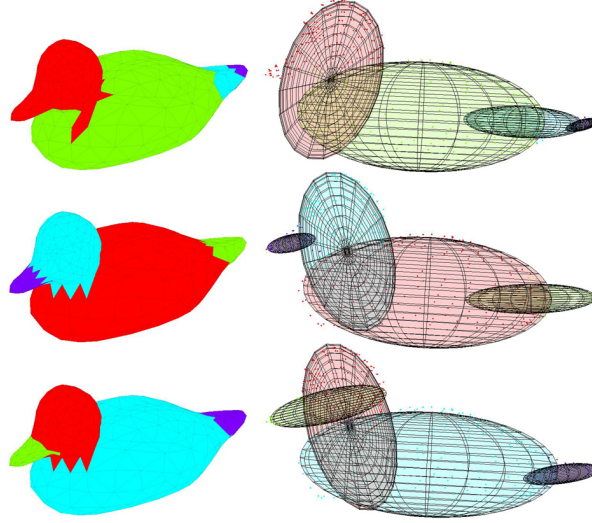


Figure 3.7: Mesh surface segmentations resulting from different settings of weight values for the composite distance metric Top: only Euclidian distance is used, setting other weights to zero. Centre: Euclidean and angular distances are used in error metric. Bottom: mean curvature is incorporated to the error metric along with Euclidean and angular distance.

we take the centroid of these associated points q and update $p \leftarrow (1 - \epsilon)p + \epsilon q$, for some small ϵ . After a number of iterations, the result is points which lie along the appropriate ellipsoidal intersections and tend to be evenly spaced along the boundary.

Figures 3.1 and 3.6 show smoothed segmentation boundaries in the rendered ellipsoidal representations.

3.4.2 Impact of different metrics

Each of the components of the surface fitting metric incrementally allows for a better fit of the ellipsoidal surface to the mesh region. This is illustrated in figure 3.7.

Initially, we start out by using only the Euclidean distance, setting β and γ to 0. Then we run the segmentation for ten iterations and arrive at the segmentation at the top of figure 3.7. To the right, we see the original mesh coloured according to segments,

and to the left we show the resulting ellipsoids. Notice that since we are fitting surfaces, we are only interested in the ellipsoidal surface region near the mesh data. The ellipsoid is free to grow as large as needed to fit the mesh segment since only a region of its surface will be kept for the final representation.

Using only Euclidean distance does not suffice for a good surface fit. This is particularly noticeable in the region of the bill.

Next, we incorporate angular distance into the metric. Now, once again, we run the segmentation for ten iterations and arrive at the segmentation at the centre of figure 3.7. The resulting segmentation follows the surface much more closely. However, we can still see that the fitting is not as good as it could be. The tail is much flatter, and the ellipsoid at the bill is not following closely as it approaches the head.

Finally, we incorporate mean curvature and use all three metrics. This provides more information to disambiguate the ellipsoid fitting and can be appreciated at the bottom of figure 3.7, in which the ellipsoidal surfaces follow the mesh almost perfectly.

3.5 Remeshing of ellipsoidal representations

Given our ellipsoidal representation of a surface, it may be desired to reconstruct a mesh for use in applications that will not handle this representation directly. To this effect we introduce two remeshing schemes. First, we explain how to use the boundary regions of an ellipsoidal surface.

3.5.1 Using ellipsoidal region boundaries

Different applications will require that it be determined if a given point on an ellipsoid lies within its region boundaries. For example, if we are casting a ray at an object represented as a union of ellipsoidal regions, it is easy to determine if said ray intersects an ellipsoid. However, it intersects the object only if the intersection point lies with

the region's boundaries on the ellipsoid. This problem will also arise when we address remeshing below.

If we have a planar polygon, it is easy to determine if a given point lies inside or outside said polygon. In our case, however, the polygon lies on the surface of an ellipsoid. We reduce the problem of determining if a point on an ellipsoid is within the boundaries to that of a planar polygon through the use of stereographic projection.

In our representation, for each ellipsoid we have a set of boundary regions described as a series of cycles of boundary vertices. For every ellipsoid, there is a transformation which maps it to the unit sphere. If we have a point on the ellipsoid's surface, we simply transform the point with said transformation along with all boundaries so the point and boundaries lie on the unit sphere. Now, we take a stereographic projection of the point and boundaries. The original point is within the region of interest if and only if the projected point lies within the polygon determined by the outer boundary's projection and outside that of all hole boundaries.

In our case we use the following stereographic projection, which maps point (x, y, z) on the unit sphere to (x_p, y_p) on the XY plane:

$$\begin{aligned} x_p &= 2x/(1 - z) \\ y_p &= 2y/(1 - z) \end{aligned}$$

Note that the sphere's north pole is mapped to infinity. However, since clearly no region will use the entire sphere, we can simply find a rotation that places the region of interest as close to the south pole of the sphere as possible, while not including the north pole. This rotation is then used after transforming to the unit sphere and prior to projecting.

An illustration of the stereographic projection of a region is shown at the top of figure 3.8. For illustrative purposes, the unit sphere is centred at $(0, 0, 1)$ so that the projection can be better viewed.

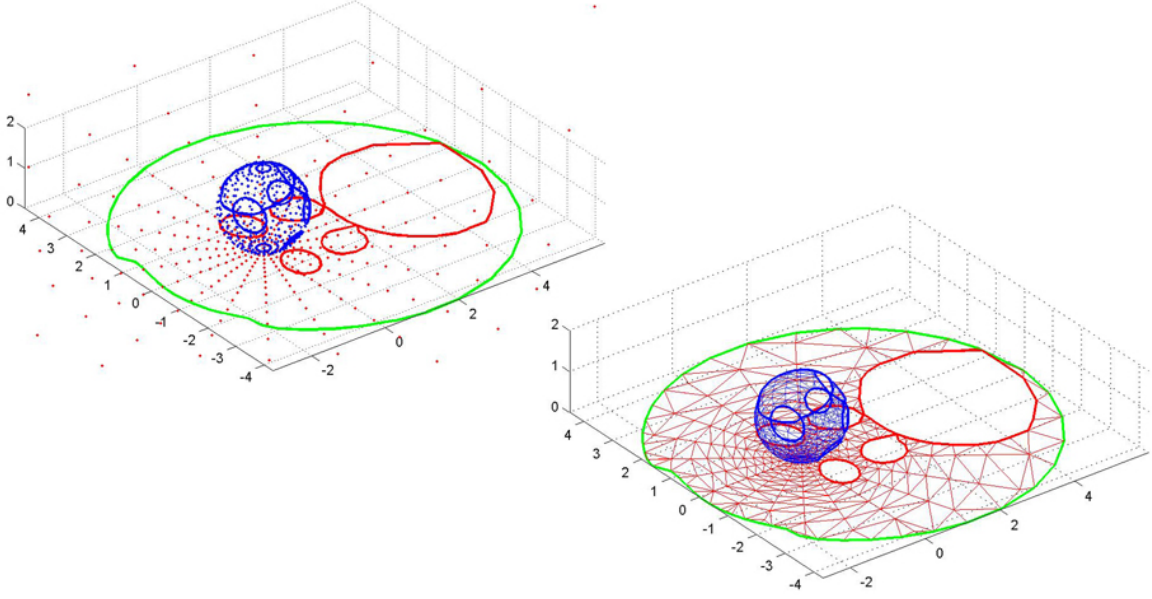


Figure 3.8: Tessellation of an ellipsoidal surface region. Top: Generated points on the unit sphere are stereographically projected along with segmentation boundaries. Bottom: Points outside of boundaries are discarded and the remaining points are triangulated.

3.5.2 Remeshing with parametric tessellation of surfaces

One way in which we may obtain a mesh from our ellipsoid representation is by generating for each region a parametric tessellation. In our implementation we go about this as follows.

For each region, we generate points on the unit sphere. This is easily done by generating points in regular intervals of azimuth and elevation. Next, we consider the stereographic projection of these points along with the relevant boundaries, having mapped these to the unit sphere as described above. This is illustrated at the top of figure 3.8.

We keep the points that respect the boundaries, and then perform a constrained Delaunay triangulation on projected points respective to the boundaries. Note that in stereographic projections, circles on the sphere's surface are mapped to circles on the plane, so a Delaunay triangulation on the projected points corresponds to a Delaunay

triangulation on the sphere’s surface. Then through the transformation to ellipsoid space, the anisotropic nature of the surface is captured in the triangulation.

To achieve the triangulation, we begin with a triangulation of the boundaries, and then iteratively add each point to the triangulation. For each one, we find the face in which it lies, insert it, resulting in the face being split into three triangles, and then perform edge flipping in order to re-establish the Delaunay triangulation. Once this is done, the faces obtained directly translate to faces on the sphere’s surface, which can now be transformed into the ellipsoid region. The bottom of figure 3.8 illustrates the resulting triangulation of the surface region of interest. The centre of figure 3.9 shows meshes resulting from this approach.

This scheme has some drawbacks. The uniform parametric sampling of the sphere to obtain the points results in the poles being much more densely sampled. This is clearly visible at the centre of figure 3.9. Also, it is hard to determine a priori how many points to use for each region. In our approach we used the same amount of points for each ellipsoid. As a result, vertices are not placed optimally.

3.5.3 Remeshing by iterative vertex addition

In order to address the drawbacks of the previous approach we propose a different remeshing scheme which attempts to place vertices where they are most needed.

Initially, we start out with a triangulation of only the region boundaries as described for the previous scheme. Now we iteratively add vertices as follows. Our candidates are the face centres of the current mesh. For each candidate we measure its radial distance to the relevant ellipsoid and choose the one that is farthest but is also far enough from the points already added according to some threshold value. This single point is projected onto the ellipsoid and then added using the same technique described for the previous scheme. Note that edge flipping will be constrained to the relevant ellipsoidal region, thus, boundary edges, which we consider feature edges [16], are not flipped.

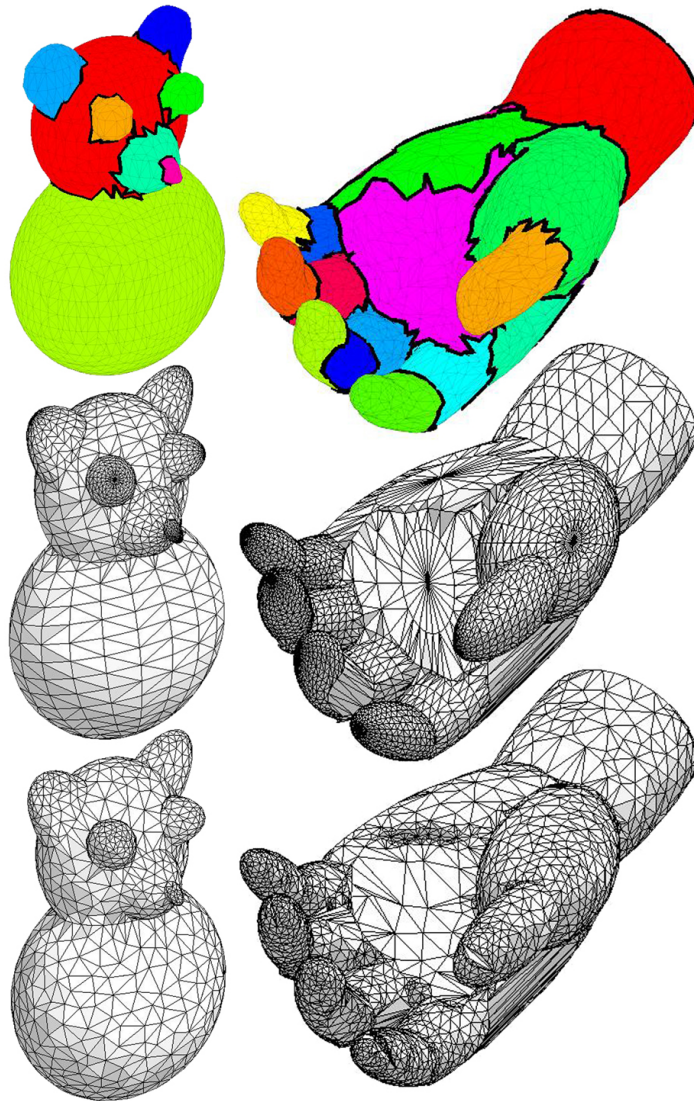


Figure 3.9: Top: Ellipsoidal surface segmentation. Centre: remeshing by parametric tessellation. Bottom: remeshing by iterative vertex addition using same total number of vertices.

The bottom of figure 3.9 shows the results for this algorithm. We use the same total number of vertices as resulted from our other approach. In the results, it can be seen that regions of higher curvature are naturally assigned more vertices while regions of lower curvature imply less vertices, and thus larger faces.

3.6 Limitations

Currently, our approach requires that the number of primitives be an input parameter to the algorithm. There are several potential alternatives to automatically determine the number of ellipsoids to use indirectly based on some other user input.

Error-driven iterative insertion In this case, the user can specify an error tolerance value. The algorithm would then proceed to, starting from some minimum number of ellipsoids, insert a new primitive at the vertex of highest error, iterate till convergence, and check if the maximum error is below the user-given value. If not, another primitive is inserted, and the process is repeated.

Clustering of local fittings For each vertex, consider a small surface area surrounding it and fit an ellipsoid to this patch. Do this for each vertex (or a representative subset if the total number is prohibitive) and consider the distribution of parameter values. High-density areas in the distribution of parameters represent agreement on parameter settings for local ellipsoid fittings. These density modes, and corresponding parameter settings, could be extracted from this set of local parameters using a density-driven clustering algorithm such as mean shift [27, 22, 78].

Medial curve-guided initialization In the case of volume-based segmentation, we may consider using the medial curve of the shape, the intuition being that ellipsoidal volume placement would coincide with locally linear sections of the medial curve, as

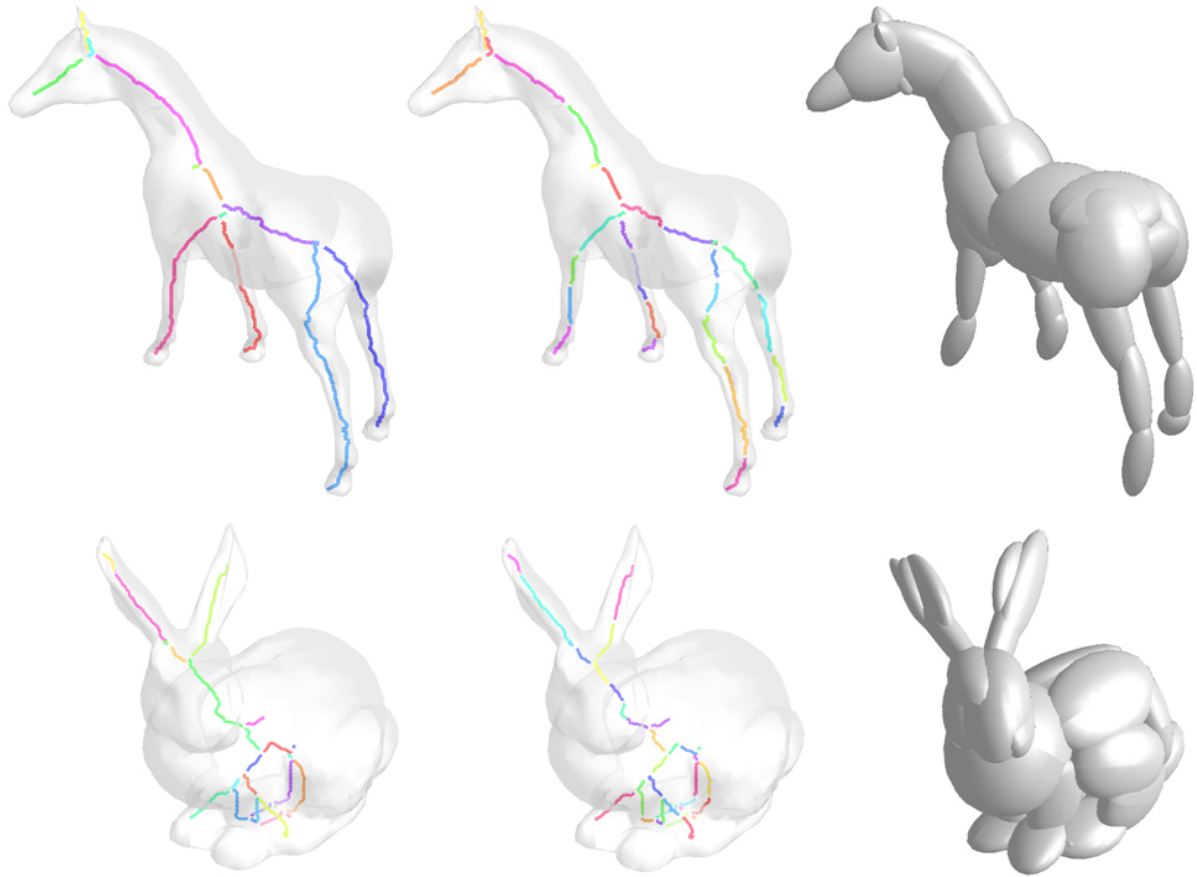


Figure 3.10: Initialization of ellipsoid placement based on medial curves. Left: Approximate medial curves computed through voxel thinning [76, 85]. Centre: Bottom:

illustrated in Figure 3.10. We obtain an approximation of the shape’s medial curve by voxel thinning [76, 85] and segment out the individual curves (Figure 3.10, left.) We then, based on a user specified tolerance, segment each curve into approximately linear parts (Figure 3.10 centre.) Finally, we place initial estimates of ellipsoid positions based on these approximately linear medial curve segments and then iterate until convergence. (Figure 3.10, right.)

As we’ve also mentioned, a natural limitation of this approach is that it cannot capture saddle regions. Subsequent to our work, Yan *et al.* present a variational approach general quadric fitting [57] (see Section 2.3 for a description.)

3.7 Summary

We have presented the idea of modeling a 3D object using a relatively small number of ellipsoidal surface regions, each represented by an ellipsoid and boundaries indicating where adjacent regions meet. This results in a compact representation which supports efficient geometric queries and transformations.

In order to obtain the segmentation, we introduced metrics based on Euclidean radial distance, surface normals and surface curvature; all properties which are easily and efficiently computable for an ellipsoidal surface. Using these metrics we applied a variational approach to achieve the segmentation.

Given that a triangulated mesh is sometimes desirable, we also presented two remeshing schemes which we compared. We showed that in the second scheme, vertices are placed intelligently, resulting in regions of higher curvature being densely sampled, while flatter regions are represented by larger faces in the resulting mesh.

This has been a contribution to the set of model-fitting segmentation approaches with applications in compact shape representation and approximation. In the next chapter, we will present a property-based approach which searches for shape redundancy of a different nature: local symmetry.

Chapter 4

A property-based approach: Local approximate symmetry-based segmentation

This chapter will present a contribution to the category of property-based segmentation approaches. Using a robust-statistical approach, we will show how regions of local approximate planar-reflective symmetry can be found and segmented from the rest of the object. We will show how this segmentation method can be applied hierarchically to construct a novel symmetry-based representation which we call a “folding tree” which can be “un-folded” to recover an approximation of the original shape.

4.1 Motivation and background

Symmetry plays a fundamental role in nature, manifested both in the form and function of living organisms. Visually, symmetry is important to humans, as it influences our perceptual understanding of objects in the world. Symmetric patterns, not surprisingly, are an important design principle in guiding the aesthetic and construction of synthetic

objects [6, 31]. Neuroscience research goes so far as to indicate that aspects of symmetry in humans may be hard-wired into our visual processing system [81]. Symmetries are ubiquitous in humans, our environment and our creations of art and architecture [59].

The classification, understanding and intelligent representation of shape is an active area of research in geometry processing. Recognizing the common presence of symmetries in many real world objects can greatly assist in solutions to various shape representation problems such as simplification, repair, noise removal and skeletal extraction. Of the various types of symmetries found, planar symmetry is perhaps the most commonplace and is thus the focus of this chapter. While planar symmetries have been recognized to be an important feature in shape understanding, there has been little work in shape representations that are defined as a structured assembly of symmetric parts. In this chapter we present the concept of a *folding tree* (see figure 4.4), where an object is defined as a hierarchical union of planar symmetric and asymmetric parts. Each nested detection of a symmetric part reduces the complexity of representation of said part by half, greatly simplifying the overall representation complexity of many objects.

For folding trees to be useful beyond an academic concept, we must be able to automatically construct them from geometric data such as meshes as well as regenerate the object from its folding tree. Central to folding tree construction is the problem of automatically finding a maximally symmetric part of the object. We observe that most organic objects with planar symmetry are articulated figures with coherent symmetric parts that are connected at the joints of an underlying skeletal structure. Most synthetic objects show a construction history involving symmetric primitives, symmetry creating operations such as reflections, planar symmetry preserving operations like revolves and coherent combinations of various symmetric parts often with some asymmetric refinement. Motivated by these observations we additionally constrain our problem to finding a maximal symmetric part that is a single connected surface component of the object. The constraint has several advantages, including simplifying reconstruction. Multiple

surface components with the same planar symmetry are just as easily represented as multiple symmetry nodes that have the same symmetry plane.

Our approach to finding maximally symmetric parts is based on robust M-estimation using an iteratively reweighted least squares (IRLS) algorithm. We simultaneously solve for the reflection plane as well as the region of surface that is symmetric with respect to it. Upon convergence, the symmetric region found is separated from the rest. The algorithm is then applied to the remaining regions to find other local symmetries, as well as recursively to half of the symmetric region to find nested symmetries. This process leads to a hierarchical folding tree representation of the object where geometry is only stored in tree leaves. The original surface can then be reconstructed from its folding tree in a bottom up fashion by reflecting symmetric geometry and reconnecting segmentation boundaries.

This chapter presents two principal contributions. Firstly, we introduce a method capable of detecting *global* as well as *local* approximate planar symmetries in 3D meshes. Our algorithm has solid foundations on statistical methods for robust M-estimation [35] which has also been used in many recent computer vision applications [95, 17, 105]. Applying this approach results in a mesh segmentation into two parts, the symmetric region and the non-symmetric remainder (in the case of global symmetry, only one segment will result.)

Secondly, we apply this local symmetry segmentation hierarchically, exploiting our symmetry detection approach for *mesh folding*: the elimination of planar symmetry redundancy from the mesh data. As we will see, this results in a mesh segmentation at each level of the tree structure. The final result, if desired, could be aggregated into a single segmentation by coloring each face according to its placement in the folding tree, though this may not always yield meaningful components in the perceptual sense.

Our algorithm is orthogonal to other existing methods for mesh compression [39, 32, 47, 42] which do not explicitly take advantage of repeating symmetric areas in 3D meshes.

(For a recent survey, see [1].)

Although the computation of symmetries in shapes has been an intriguing area of research in computer vision and computational geometry literature for the last 30 years at least, to our knowledge, there has been comparatively little research on symmetry detection in 3D meshes mainly due to the increased complexity of the existing algorithms when extended from the 2D to the 3D case. Our work differs from most approaches for 3D symmetry detection in meshes in that we aim at the robust detection of not only global but also local reflection symmetries, i.e. those present only in parts of a 3D mesh, and we exploit these symmetries in order to achieve mesh compression by eliminating faces implied by the discovered symmetries.

The detection of symmetries in 2D and 3D models has mainly been applied in object classification, recognition and reconstruction. Early approaches dealt with symmetries of planar point sets by applying pattern recognition algorithms that search for matches in circular strings representing the graphs of polyhedral objects [7, 114, 38, 44].

Despite the optimality of these algorithms that could also detect all the possible symmetries in a shape, they were only able to recover perfect symmetries in 2D and 3D shapes making them useless in the presence of small perturbations, imprecision and noise which is very common in meshes. The problem of approximate symmetries was addressed in [3] that considered the problem of computing generic geometric transformations between two point sets. A detailed theoretical analysis is given for the developed algorithm, however, it deals with global symmetry and it is unclear if the given algorithm could be extended to three dimensions. Such is also the case of other methods for finding symmetries of symmetric or almost symmetric 2D planar images [71, 29, 99].

An interesting extension of that early work which introduced the notion of symmetry distance, meaning how much of a given symmetry an object possesses, was developed in [117]. The approach can evaluate symmetries in the presence of noise and also find locally symmetric regions in 2D images and 3D range data. The reflection plane of the

image is determined by minimizing the symmetry value over all possible reflection planes using a gradient descent algorithm to locate the plane of maximal symmetry. More recently, an approach also taking 3D range images as input was presented in [109]. A probabilistic measurement model is used to detect symmetries in order to reconstruct partially occluded 3D shape models. Although both methods can find local symmetries, they follow a greedy technique searching in growing localized regions. In contrast, our method is top down, leading to the gradual removal of asymmetric outliers from the region for the robust detection of the maximal symmetric area using an M-estimator approach. In this regard, our approach tends to find the largest areas of symmetry first, avoiding over-segmentation.

Another original approach that detects the dominant hyperplane of bilateral symmetry in range images of 3D objects with a linear time algorithm is presented in [83]. The hyperplane is uniquely defined by the centroid and eigenvectors of the covariance matrix of the object. This method is limited to the detection of the plane of global symmetry and is not robust to outliers or imprecision in the 3D object. Sun *et. al* [106, 107] address the symmetry detection problem by, in the first case, searching for correlations in the 3D object's gaussian image, and in the second, by using the image's orientation histogram in a similar fashion. More recently, Martinet *et al.* [73] recover symmetries by examining the extrema and spherical harmonic coefficients of the object's generalized moments. These approaches, however, also focus strictly on global symmetry detection.

Symmetry shape descriptors are introduced in [50, 51] where a collection of spherical functions are used to describe the measure of rotational and reflective symmetry present in a mesh with respect to every axis passing through its center of mass. The descriptors had several desirable properties such as robustness and stability. However, the approach aimed at using global symmetry information as a shape descriptor and not at extracting local symmetries.

In parallel with our work, two very interesting symmetry detection algorithms have

been developed. These are the approaches of Mitra *et al.* [78] and Podolak *et al.* [89], both of which we describe in detail in Chapter 2. Both proposed methods seem robust and efficient. On the other hand, our approach represents an easy-to-implement alternative symmetry detection method which is also based on robust statistics along with a hierarchical simplification scheme that can be directly applied to mesh compression and reconstruction, in both cases through our folding tree structure.

4.2 Symmetric region detection

Given a mesh, we wish to find a connected region S of faces that exhibit planar symmetry within a tolerance parameter ϵ . In the case of global symmetry, this region should be the entire mesh. We approach the problem as a model fitting scenario, in which the model consists of the sought plane, and the connected region of symmetry.

Error metric: given a plane p and mesh M , we define the symmetry error at vertex v_i of M with respect to p as $d_i^p = \text{dist}(r_i^p, M)$, where point r_i^p is the reflection of v_i with respect to p , and dist denotes point-to-mesh distance. For notational simplicity we will avoid including the p superscript in the following.

We compute the distance function dist from a reflected vertex to the mesh by taking the minimum point-to-triangle distance from the point to the closest compatible face on mesh M [94]. We consider a face to be compatible with a given query vertex if the angle between the interpolated normal at the closest point on the face and the vertex’s reflected normal is less than 45 degrees.

Given the presence of structured outliers in the form of the non symmetric regions of the mesh, we interleave solving for the symmetric region S and the plane p based on an iteratively reweighted least squares (IRLS) approach, using the Geman-McClure (GM) robust M-estimator [35, 26]. In essence, the GM estimator maps error values to an associated cost. This cost approaches constant (and the associated weight approaches zero)

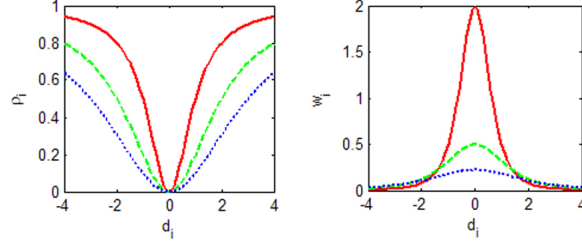


Figure 4.1: GM estimator cost ρ (left) and associated weight w (right) as a function of distance for scale parameter $\sigma = 1$ (red solid), $\sigma = 2$ (green dashed) and $\sigma = 3$ (blue dotted).

as error values approach infinity, thus mitigating the influence of outliers on the minimization process. The GM estimator exhibits excellent behavior in rejecting structured outliers with the appropriate choice of the scale factor σ [95]. This parameter essentially controls the rate with which weight decreases as error increases. (See figure 4.1.)

Solving for the plane: Given the current distances d_i , the GM cost estimator ρ_i and associated weight w_i for each vertex are given by

$$\rho_i = \frac{d_i^2}{\sigma^2 + d_i^2} \quad w_i = \frac{1}{d_i} \frac{\partial \rho_i}{\partial d_i} = \frac{2\sigma^2}{(\sigma^2 + d_i^2)^2}$$

In addition, in order to be robust in the presence of tessellations with varying face sizes, we multiply the obtained weights by their associated vertex weights, i.e. $w_i \leftarrow w_i \frac{1}{3} \sum_{j=1}^k \text{area}(f_j)$, where $f_1, \dots, f_k$ are the faces incident on vertex v_i .

For a body which exhibits planar symmetry it is known that its plane of symmetry is perpendicular to a principal axis and contains the object's center of mass [77]. This lets us solve for the current plane of maximum symmetry in a closed form manner by considering the center of mass m and weighted covariance matrix C relative to the weights w_i .

$$m = \frac{1}{s} \sum_{i=1}^n w_i v_i \quad C = \frac{1}{s} \sum_{i=1}^n w_i (v_i - m)(v_i - m)^T$$

where $s = \sum_i w_i$.

We compute the eigenvectors of C and consider the three planes determined by these

vectors and m . For each of these planes we compute the distances d_i and associated costs ρ_i retaining the one of minimum sum cost. This now lets us solve for the support region.

Support region: Given the current ρ values and a candidate face $f = (v_1, v_2, v_3)$ we consider it to be a *support* face if it holds that $\forall_{i \in \{1,2,3\}} d_i \leq 2\sigma$ [35], a value which we choose experimentally. We then find the largest connected region of support faces, taking this as our new estimate of the support region and set the weights for all vertices outside this region to be 0, thus controlling leverage.

The estimation and region finding steps are iterated until convergence obtaining a local optimum with respect to the previously defined robust symmetry cost ρ . In theory, the support region selection step makes it so there is no guarantee of convergence. In practice, however, we do not observe convergence problems when there is a unique and strong local symmetry. In instances when there is more than one equally strong symmetry (such as the chess piece example of Figure 4.6) the procedure may toggle back and forth between them, but this is not a problem since choosing either allows the algorithm to proceed recursively to the next.

Initialization and details: Initially we simply define w_i to be the mesh area associated with vertex v_i as defined above, and the initial support region contains all faces.

In order to accelerate and improve convergence we initially use a discrete approximation of the above distance function. For a given vertex v_i and face f_j this distance function $dist_{discrete}(v_i, f_j)$ is defined as the Euclidean distance from the reflected vertex r_i to the face plane of f_j if the angle between the reflected normal and f_j is less than 90 degrees, and infinite otherwise. The distance $dist_{discrete}(v_i, M)$ is defined as the distance to the f_j whose centroid is closest to v_i . During these initial iterations we set $\sigma = 1.4826 \cdot median(d_i)$, which is a popular estimate of scale [26], not letting it fall below 2ϵ to avoid instability.

This discrete distance function is first used until convergence, after which the more precise distance function $dist$ is used. At this point we set $\sigma = 2\epsilon$. This setting allows for

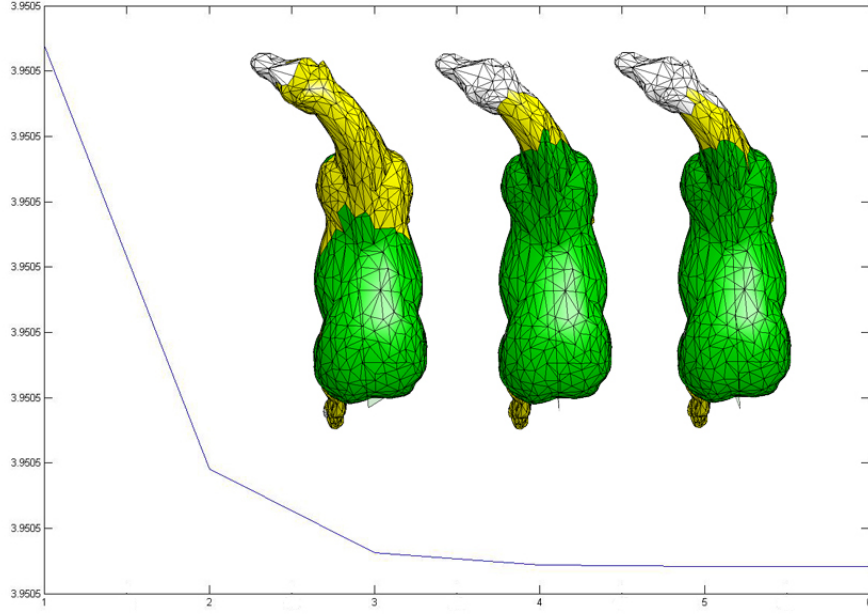


Figure 4.2: Illustration of algorithm convergence. The plot shows the objective function $\sum_i \rho_i$ for support vertices v_i . The placement of the estimate of the symmetry plane along with support region (yellow) and symmetric region (green) are shown for the horse model. Left to right, iterations 1, 5 and 10 respectively, using the precise distance metric.

near-symmetric vertices to be included in the support region albeit with lower weight.

Upon convergence, the symmetric region S is extracted as the largest connected region of faces whose vertex distance values are all below ϵ . We detect convergence by comparing the current plane estimate with that of the previous iteration checking for a sufficiently small difference.

Convergence: In our experiments both distance functions exhibit very good convergence behavior (to their respective minima). Figure 4.2 illustrates an example of the convergence properties of our approach.

4.3 Folding trees

In this section we will describe a structure, which we call a *folding tree*, that encodes a 3D shape by storing its planes of local reflective symmetry as well as its non-redundant geometry. Essentially, the structure is constructed by the recursive application of our approximate local symmetry detection method. Given an input shape, we apply our detection method and, if a plane of local symmetry is found, the symmetric region is segmented from the rest. We simplify the symmetric region by discarding a redundant half, store the plane of symmetry, and proceed recursively on the remaining half in search of nested symmetries. We also recurse on the remaining surface, from which the locally symmetric segment was removed, in search of other local symmetries.

The folding tree structure as well as the algorithmic steps are formalized in the following subsections.

4.3.1 Definitions

We consider a region R of a mesh M to be a connected subset of the faces of M .

A segmentation $\{R_1, R_2, \dots, R_n\}$ of a mesh M is a set of mutually exclusive regions whose union results in M .

A folding tree T representing a mesh M is inductively defined as one of the following:

- a *leaf* node, which contains mesh data for M .
- a *folding* node, with one subtree S , and a plane of symmetry p .
- a *segmentation* node, with n subtrees $T_1, T_2, \dots, T_n$, where T_i is a folding tree for region R_i such that regions $R_1, R_2, \dots, R_n$ are a segmentation of the mesh represented by T .

The *unfold* operation can now be defined on a folding tree T as follows.

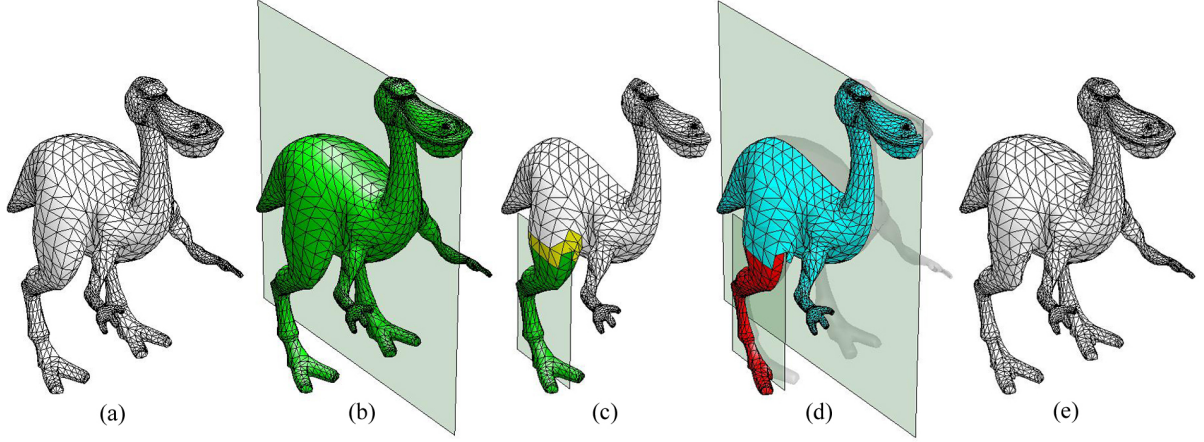


Figure 4.3: **(a)** Original dino mesh. **(b)** Detected global symmetry plane. **(c)** Remaining half dino mesh with a detected local symmetry. Green indicates the region of symmetry and yellow indicates faces still in the support region but not included in the symmetric region. **(d)** Remaining leaf geometry (the two leaves shaded a different color). **(e)** Dino reconstructed from the leaf geometry and symmetry planes.

- The mesh data M if T is a leaf.
- $unfold(S) \cup reflect(unfold(S), p)$, where S is the unique subtree of T , p is the reflection plane,
- $\cup_{i=1}^n unfold(T_i)$ where $T_1, T_2, \dots, T_n$ are the subtrees of T .

Here, *reflect* indicates the mesh resulting from planar reflection of the argument mesh's vertices with respect to the argument plane.

4.3.2 Folding tree construction and unfolding

Given a mesh M , a folding tree T that represents it can be constructed in preorder through repeated application of the segmentation method of section 4.2. First, we apply the segmentation algorithm to M to find a subregion of planar symmetry, S , also obtain-

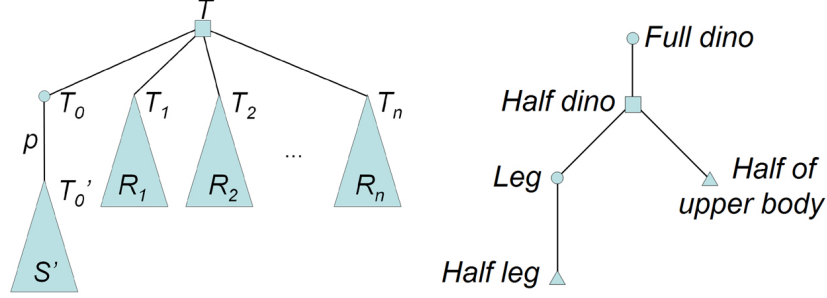


Figure 4.4: **Left:** Folding tree construction structure when a local symmetry is detected. **Right:** Example of a complete folding tree for the dino mesh decomposition of figure 4.3.2. Circles represent folding nodes, squares segmentation nodes, small triangles are leaf geometry and large triangles represent (recursive) folding trees.

ing the plane of symmetry p . We remove S from M and consider the set of remaining connected components $\{R_1, R_2, \dots, R_n\}$. Note that in the case of a global symmetry, this set will be empty. We now construct a folding tree T with $n + 1$ children $T_0, T_1, \dots, T_n$, each representing $S, R_1, R_2, \dots, R_n$ respectively. We know S to be symmetric with respect to plane p , so we can now *fold* S , retaining half of its surface S' . In particular, T_0 will be a folding node, labeled with p , and its child T_0' will represent S' . The resulting structure is illustrated in figure 4.4. The subtrees $T_0', T_1, \dots, T_n$ can now be created recursively with regions $S', R_1, \dots, R_n$ respectively as inputs.

When discarding half of the faces of a particular region, it must be decided which half to keep, which to discard, and which to modify, if any. Because of varying tessellation and the provided tolerance, both sides need not be identical. In our implementation we keep the side with the most faces in order to preserve detail. Alternatively, we could keep the half with the fewest faces in order to minimize storage. Faces with all vertices on the discarded side of the plane are removed. In addition, faces that straddle the symmetry plane are clipped.

The recursive construction of the tree may be stopped, triggering the creation of a leaf node, by using one or more criteria: when the number of faces in the mesh is below

Mesh	# Orig. f's	#Nonred. f's	Mesh bytes	Tree bytes	Comp.	Tree height	Rec. error	Time
Dino	6638	3142	265864	130916	50.76%	3	0.0358%	152 sec
Horse	3306	2672	120312	93932	21.93%	3	0.0004%	24 sec
Chair	5736	2460	229880	104352	54.61%	4	0.1600%	58 sec
Hammer	4360	677	174712	53120	69.60%	7	0.1764%	174 sec
Triceratops	5660	2447	226712	123696	45.44%	7	0.0147%	202 sec
Eagle	33072	15808	160440	96516	39.84%	5	0.0067%	936 sec
Queen	3360	600	134712	31952	76.28%	4	0.0001%	120 sec

Table 4.1: Results for seven characteristic meshes. Columns from left to right: mesh name, number of faces in the original mesh, number of non redundant faces stored in folding tree leaves, original mesh storage in bytes, tree storage in bytes, compression achieved according to one minus the ratio of the previous two columns, height of the folding tree, reconstruction error as a percentage of the bounding box diagonal and running time, as reported by our Matlab 7 implementation.

a given threshold, when the area of the mesh is below a certain percentage of that of the original mesh, or when the number of recursive folds exceeds a given maximum. Our implementation allows for any or all three.

The procedure for unfolding a tree consists of a postorder traversal according to the definition of subsection 4.3.1. Upon reconstruction, because of the tolerance parameter of the region-finding algorithm, as well as differences in tessellation, the resulting mesh may have vertex misalignments. These can be corrected automatically using known techniques [33, 112, 45] or software. The unfolding drives this repair, the union of the definition of subsection 4.3.1 between unfolded parts indicating that these should be processed in this manner. Figures 4.3.2, 4.4 and 4.5 illustrate the concepts.

4.4 Results

The implementation of the symmetry detection algorithm and the folding tree representation of meshes, as described in sections 3 and 4, has been developed in Matlab 7. The user defines the tolerance parameter of the algorithm and the criteria for stopping the

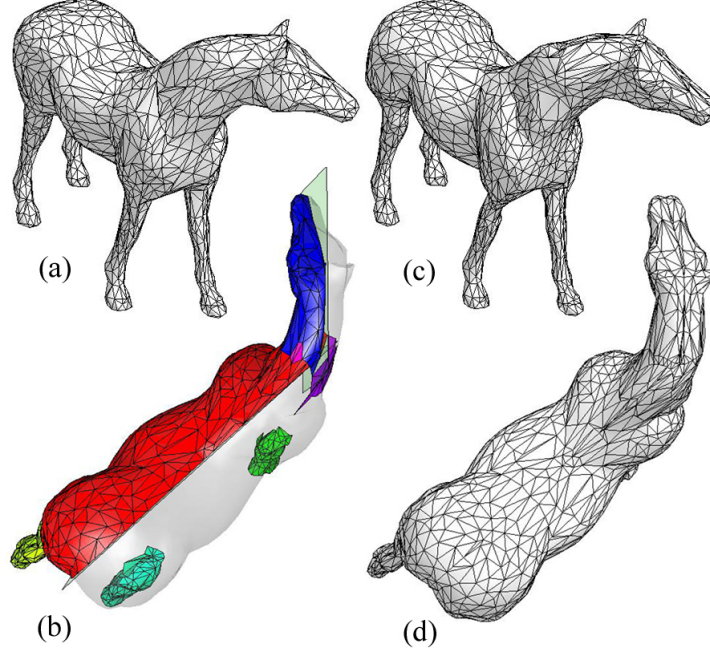


Figure 4.5: (a) Original horse model. (b) Resulting tree leaves after folding and symmetry planes. Note that the local symmetries of the body and the articulated head were detected. (c) and (d) The reconstructed horse model.

hierarchical segmentation of the mesh. The default tolerance value is 2% of the bounding box diagonal of the mesh. The default criterium for terminating the recursion is that the area of the current region is less than 5% of the total mesh area.

We present characteristic results, concerning mesh compression, the depth of the hierarchical segmentation, average reconstruction error as a percentage of the bounding box diagonal, as well as running times in table 4.1. Figure 4.6 shows initial and reconstructed meshes, complementing the results of figures 4.3.2 and 4.5, as well as illustrating all folding plane positions and the folding trees. In the chair model, we find the vertical plane of global symmetry then each cushion, which was a separate connected component, was folded through three perpendicular planes. The hammer is firstly folded in half through a vertical global plane of symmetry. Then, the handle is divided twice more. The cylindrical portion of the head is also subdivided twice more recursively. In the case of the

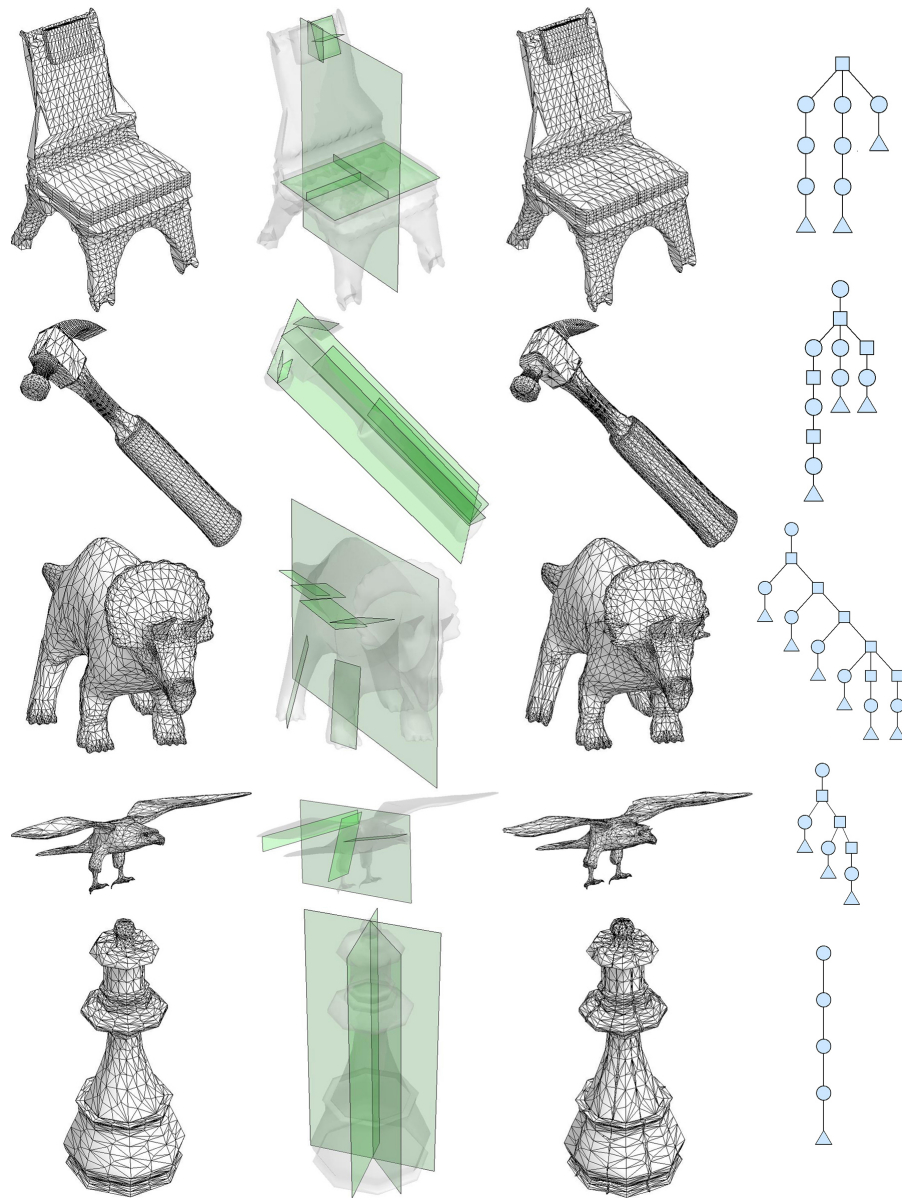


Figure 4.6: **1st column:** original mesh data. **2nd column:** All folding planes present in the constructed tree. **3rd column:** model resulting from unfolding of folding-tree representation. **4th column:** Folding tree. Circles represent folding nodes, squares segmentation nodes, and triangles are leaf geometry. Leaf nodes corresponding to asymmetric regions are not shown for clarity.

triceratops and eagle models we find a global plane of symmetry and then local symmetries in the legs, tail and body for the triceratops model, and in the wings and upper legs of the eagle. Finally, in the case of the octagonal queen chess piece, all planar symmetries are recursively discovered resulting in one eighth of the original surface being stored.

Figure 4.7 further illustrates the symmetry detection approach. In the woman, we firstly find the dominant partial symmetry that includes her body and legs. Searching for nested symmetry, the algorithm detects the symmetry of the leg. Proceeding to the remaining regions, we find the local symmetries of the head and arms. Analogously, in the dragon, we find the symmetry of its body and then legs, head and arms. In the bull, we detect the local symmetry of its body including the back left leg, then symmetries in the other three legs and the head and finally another weak nested symmetry found in the middle of its body. We detect symmetries on the body, and separately in head and the ears of the bunny and also find two other weak nested symmetries in its body. Lastly, in the octopus, its head is found to be symmetric, also containing a nested symmetry, and multiple local symmetries are found in different parts of its tentacles. We would like to note that all meshes with the exception of the chair are originally a single connected component.

Our tests were run on an Intel Pentium M 2.13GHz processor under Matlab 7.

4.5 Limitations

Our approach to symmetry plane initialization assumes that, in the given shape, the locally symmetric segment is sufficiently large to dominate the shape's moments of inertia, and thus, the initial guess will be sufficiently close to the actual symmetry present. This, however, need not be the case. There may be small local symmetries missed by our initialization method which a user might still consider significant. Figure 4.8 illustrates such an example. Here, our approach terminated on what remains of the dino because

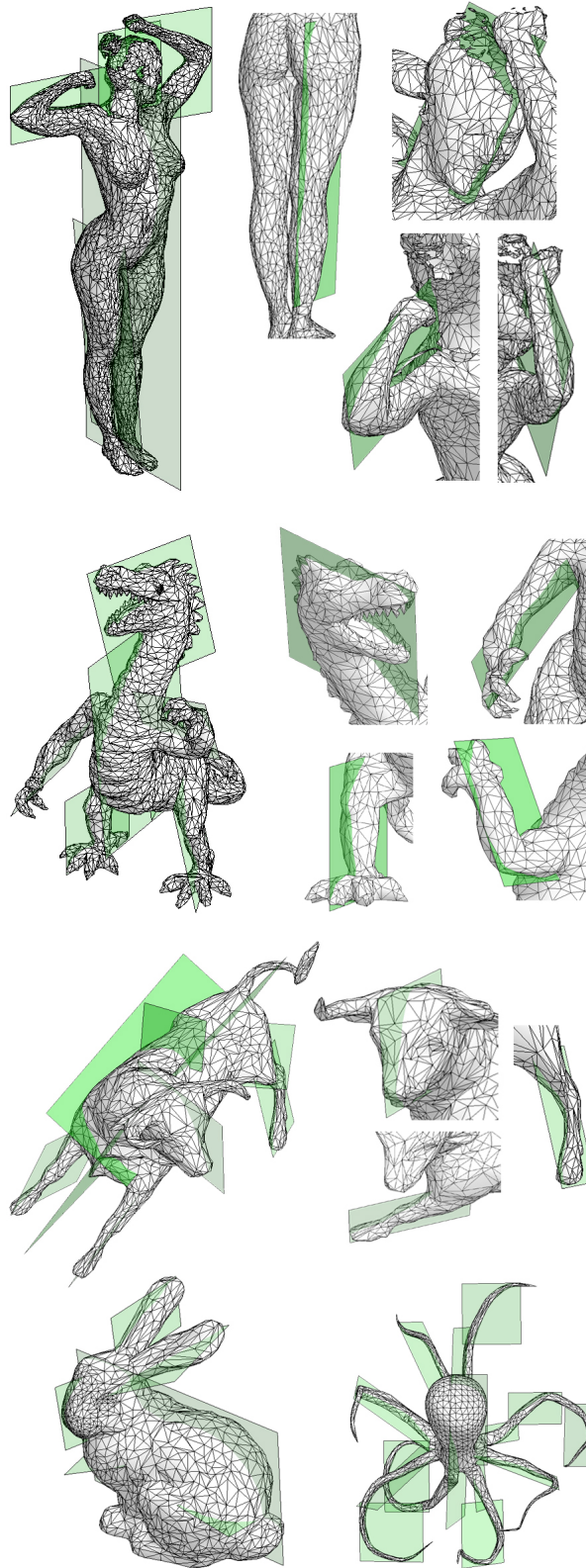


Figure 4.7: Local symmetry planes identified by the method.

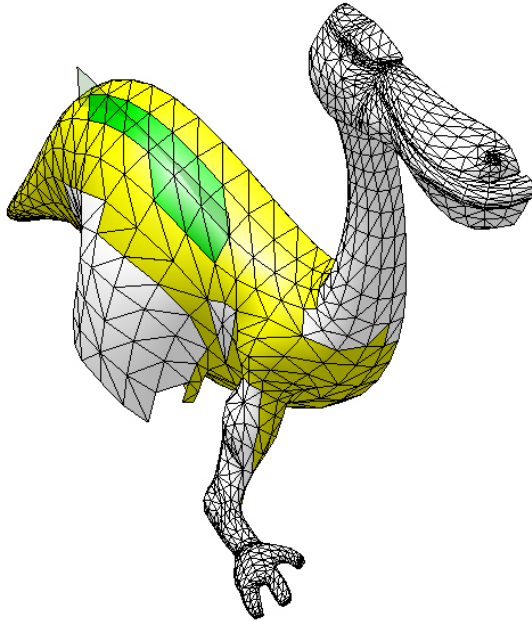


Figure 4.8: Our approach misses discovering local symmetries in the hand, arm, neck and tail of the remaining dino because these local symmetries are too far from the shape’s principal directions.

it failed to find any further significant local symmetries. However, the dino’s hand, arm, and parts of its neck and tail do exhibit a degree of local symmetry. The problem is that they are too far from the shape’s moments of inertia for the initial guess to discover them.

One way to address this problem would be to use a voting scheme similar to that of Mitra *et al.* [78] or Podolak *et al.* [89]. Alternatively, it is possible to use a RANSAC approach which attempts plane candidates based on randomly chosen pairs of points until a candidate with sufficiently strong local symmetry to be used as an initial guess is discovered.

Another limitation of our approach is that we assume that the local symmetry to be detected is in a single connected component. This is due to the local flooding nature of our approach to discovering support regions. This flooding also assumes that face adjacency is available. In the case of non-manifold meshes, our approach could potentially be

adapted by using k -nearest neighbours for substitute connectivity.

4.6 Summary

We have proposed a novel property-based approach for segmenting out local planar symmetries in 3D meshes based on robust M-estimation. The approach is applied hierarchically in order to construct a new compact representation of meshes, called folding trees, which represent the original mesh by only encoding the non redundant regions as well as the planes of symmetry and can be used to recover the original object through unfolding.

Chapter 5

Generalizing to heterogenous criteria: A multi-objective approach to shape segmentation and labeling

In this chapter, we will introduce the notion of multi-objective shape segmentation and motivate it as a means for the incorporation of user knowledge to the shape segmentation problem. We will motivate the need for a continuous domain parameterization of shape segmentations and introduce multiplicatively weighted Voronoi space partitioning as a means to do so. We will show how shapes that are under complex articulation can still be addressed with our approach through the use of multi-dimensional scaling applied to surface samples and pair-wise affinities (in particular we use approximate geodesics) to untangle the shape. We will take up once more the notion of segmentation labeling as defined in section 5.5 and explore ways in which segmentation labels can be automatically assigned to segments to maximize objectives. In particular, we introduce an efficient approach which is optimal under unary objectives. We also show that, in the presence of binary objectives, the labeling problem is NP-complete by reducing the Traveling Salesman Problem to our labeling problem. Thus motivated, we present an

evolutionary approach which can handle n -ary objectives. We motivate the incorporation of symmetry constraints on segmentations as further means by which a user can supply domain knowledge to the segmentation problem and illustrate how our parameterization and optimization approaches can naturally accommodate this. Finally, we apply our segmentation method to obtain semantically meaningful labeled segmentations, optimized bounding volume hierarchies for path tracing, animation skeleton placement for automatic character rigging, and automatic transfer of clothing for animation characters.

5.1 Motivation and background

Most object classes and segmentation applications can be associated with domain-specific knowledge that can help define object parts as well as provide them with semantically meaningful labels. Humans for example [92] have well-documented relationships between different body parts. The humanoids in Figure 5.8, after being consistently segmented and labeled to have a head, torso, arms and legs, can be automatically rigged, clothed and even morphed into each other. While it is not reasonable for users to manually segment and label even tens of shapes, one can imagine that a user-specified description of segments and labels can suffice to automatically segment and label an entire class of objects consistently.

Current segmentation approaches, which we presented in Chapter 2, limit user input to the number of desired segments or a handful of threshold values. The current approach to incorporate domain knowledge is to design and implement a new segmentation algorithm, which is infeasible for most users (artists, animators, medical experts.) Moreover, current approaches have intrinsic limitations corresponding to their category. Affinity-based methods require that it be possible to evaluate the objective function for pairs of faces. This may be difficult for functions that require a larger shape context to sensibly evaluate, such as being developable or co-axial. Moreover, these approaches are

not amenable to optimizing heterogeneous objectives, *e.g.* one part being convex and the other concave. Approaches based on generative models have the advantage of a continuous domain of optimization and possibility of gradient computation. Unfortunately they are also not applicable to some common segmentation objectives such as convexity or symmetry that are hard to formulate generatively. Finally, property-based segmentation approaches tend to be specifically tailored to homogeneous segmentations, meaning all parts share the desired property, and this property is fixed.

Here, we present a framework for shape segmentation that allows for the incorporation of domain-specific knowledge through arbitrary objective functions that geometrically describe each segment (with an optional label). These objectives can be unary, asserting properties of an individual part (*e.g.* that it should be narrow, compact, developable, flat) or they can be n -ary, referring to part inter-relations (*e.g.* a set of parts should be parallel to each other, or have the same approximate size, or two parts should be perpendicular to each other.) The segmentation problem is thus cast as an optimization problem, minimizing the aggregate objective function over all possible segmentations.

5.2 Multi-objective shape segmentation

While current approaches work well for their intended applications, the incorporation of object and application specific knowledge can result in better quality and consistently labeled segmentations (see Figures 5.3, 5.5 and 5.8).

Implementing a segmentation algorithm from scratch is a non-trivial task beyond the capability of most users. Users can thus convey domain knowledge to a general segmentation framework either by providing example labeled segmentations [96], or by encoding this knowledge using a set of geometric objective functions. In this chapter we take the latter, novel approach by providing a general multi-objective segmentation and labeling framework and a working set of useful geometric objectives (see Figure 5.9) that

users can combine or augment to suit their application. Each part of a segmentation description is thus uniquely identified by a label and a multi-objective function that geometrically describes the part and (possibly) its relation to other parts.

As a simple example, consider a hammer. This object is defined by Merriam-Webster as “a hand tool consisting of a solid head set crosswise on a handle and used for pounding.” This naturally suggests two segmentation labels: *handle* and *head*. Moreover, it suggests the segmentation objective that the part labeled *head* be perpendicular to the part labeled *handle*. Our knowledge might further suggest that the parts should be approximately convex and that the part labeled *handle* should be narrow. The effects of incorporating multiple objectives into the segmentation of a hammer shape can be seen in Figure 5.1. For comparison purposes, we also show in the figure the results of applying the approaches of Liu and Zhang [64], Kraevoy and Sheffer [54] and our ellipsoidal segmentation. Note that in even a simple case such as this, none of these approaches achieve the desired, semantically meaningful segmentation.

Objectives for a labeled part may thus be unary (*eg.* narrow, flat, symmetric convex, compact, developable), binary (*eg.* perpendicular to, bigger than, adjacent to), or n -ary (*eg.* parallel, coaxial, similar in size).

Given a segmentation description, shape segmentation can be cast as an optimization that minimizes an aggregate objective function which combines, as the sum of squares, the objective functions describing each part label. The segmentation optimization algorithm thus decoupled from its labeled description has the following characteristics: 1) Heterogeneous objectives for different part labels allows these parts to be distinguished from each other and overcomes the drawback of a homogenous segmentation criterion assumed by most existing segmentation techniques. 2) The objective functions are evaluated on fully instantiated segmentations, avoiding the need of affinity-based approaches, to evaluate objectives sensibly on single face pairs. 3) As long as objective function evaluation is possible, a generative model is not strictly necessary (though certainly possible.)

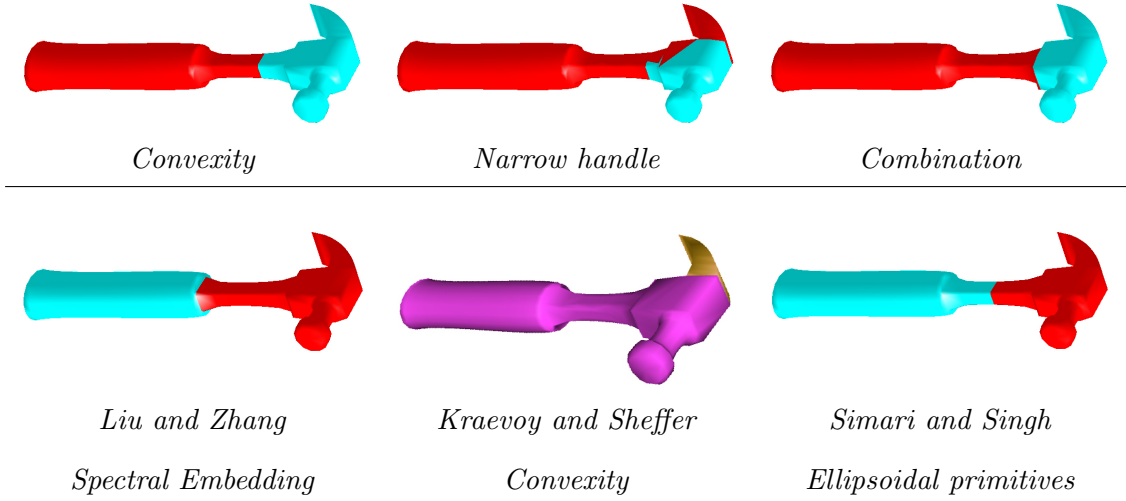


Figure 5.1: Influence of multiple objectives. **Above:** Our multi-objective approach. On the left we see the result of optimizing for minimum squared difference of convex hull volume to mesh volume. In the center is the result of optimizing only for the narrowness of the handle. On the right is the result of optimizing a combination of these two objectives and perpendicularity between head and handle. (See Section 5.7 for details.) **Below:** For comparison purposes, the result of applying the approaches of Liu and Zhang [64], Kraevoy and Sheffer [54] and our ellipsoidal segmentation.

4) The framework easily allows for assertions not just about parts but of inter-part relations using n -ary objectives. 5) Users wishing to define a new segmentation description can select from a set of already implemented objective functions (see Section 5.7) and new objective functions seamlessly integrate into the framework.

5.3 Parameterizing segmentations

In order to optimize a segmentation with respect to an aggregate objective function as described above, we need to be able to describe segmentations by means of some set of parameters. The set of all possible segmentations can be parameterized directly by a vector indicating the segment label of each face explicitly, but such a parameterization

Algorithm 1 Connected partitioning

```

1: Let  $\mathbf{C}$  be the given set of weighted Voronoi centers.
2: for all  $f \in \text{surface}, i \in [1, n]$  do
3:      $\mathbf{D}[f, i] \leftarrow$  weighted Voronoi distance from  $f$  to  $\mathbf{C}[i]$ 
4: Initialize an empty priority queue  $\mathbf{Q}$ 
5: Initialize segmentation indices  $\mathbf{Seg}$  to 0
6: for  $i = 1$  to  $n$  do
7:      $f \leftarrow \underset{f}{\operatorname{argmin}} \mathbf{D}[f, i]$ 
8:      $\mathbf{Q}.push(f, priority = \mathbf{D}[f, i], centerid = i)$ 
9: while  $\neg \mathbf{Q}.isempty()$  do
10:     $(f, priority, i) \leftarrow \mathbf{Q}.pop()$ 
11:    if  $\mathbf{Seg}[f] = 0$  then
12:         $\mathbf{Seg}[f] \leftarrow i$ 
13:        for all  $f'$  adjacent to  $f$  do
14:             $\mathbf{Q}.push(f', priority = \mathbf{D}[f', i], centerid = i)$ 
15: return  $\mathbf{Seg}$ 

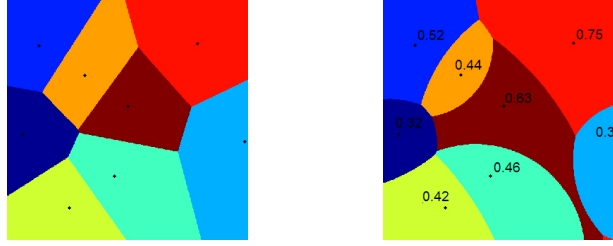
```

would be cumbersome, since the vast majority of the segmentations would be undesirable, being unconnected, having complicated boundaries, *etc.* Note the difference between using this as a representation and using it as a parameterization meant to be directly manipulated in order to search the segmentation space.

We would also like to avoid a surface parameterization, since this is its own difficult problem, and places topological constraints on the shape representation as well as introducing distortion.

We thus propose the use of a Voronoi space partition to define the surface segmentation of an embedded shape (see Figure 5.3). A set of 3D points, each corresponding to a part label, naturally induces a segmentation on the shape by classifying each surface element according to the Voronoi region it occupies. We classify polygon faces based on their centroid but a segmentation at sub-face resolution can also be defined.

A natural limitation of this representation is that Voronoi regions are only capable of describing planar boundaries. Augmenting the centers with a distance scaling weight further allows curved boundaries and non-convex regions. Formally, a point p 's distance to a center c with weight w is given by $\|p - c\|/w$. The result is known as a multiplicatively weighted Voronoi partitioning [82]. The following illustration compares the standard Voronoi partition induced by a random set of points and the multiplicatively weighted Voronoi partition induced by points with the same locations but varying associated weights.



Standard Voronoi partition Weighted Voronoi partition

The set of Voronoi points and weights thus defines a continuous parameterization over a subset of all possible surface segmentations. In practice, the compact shape of Voronoi cells naturally tends to induce connected and compact surface segmentations. It is, however, possible for the segment induced by a Voronoi region on an embedded manifold to be disconnected.

Ensuring segment connectedness: To achieve connectivity we use a priority queue flooding scheme as introduced by Cohen-Steiner *et al.* [20] and successfully used in at least one other segmentation approach [101]. This process is detailed in Algorithm 1.

Segmenting shapes under complex articulation: As expected, surface elements within a Voronoi region will tend to be proximal in Euclidean space. For many desired surface segmentations, however, geodesic proximity is more important than Euclidean proximity. While ensuring segment connectivity alleviates the problem of geodesically

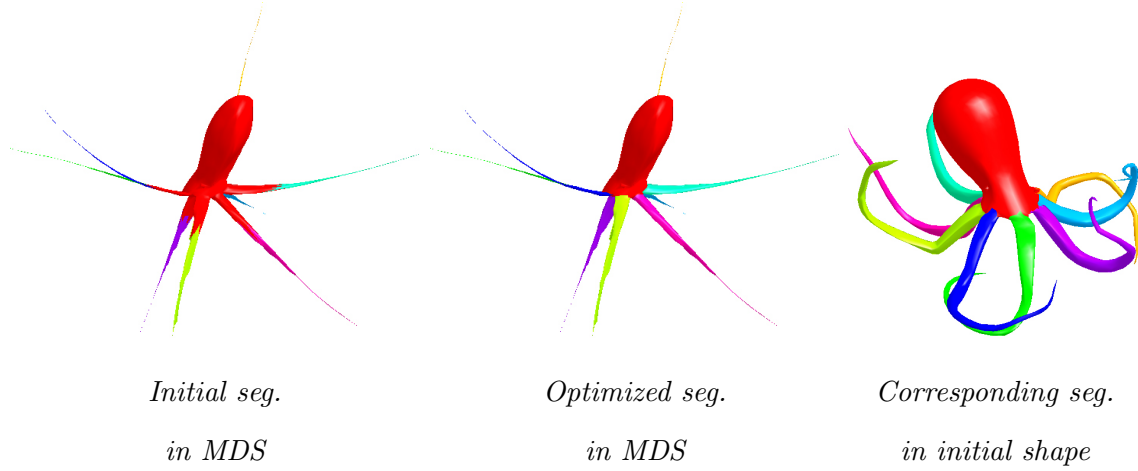


Figure 5.2: Our approach can deal with shapes with convoluted articulations through the use of multi-dimensional scaling. The segmentation induced by the partition in MDS space is then easily mapped through correspondence with the original shape. Here we simultaneously optimize for narrow arms with similar proportions and an ellipsoidal head. See Section 5.7 and Figure 5.9 for details.

distant surface elements belonging to the same segment, it does not address the fact that a part with complex articulation may be hard to capture using a single Voronoi cell. We address this concern using multi-dimensional scaling (MDS) as a pre-processing step, which has proven useful in other segmentation approaches [48, 64]. We pre-compute all pairs of geodesic distances between surface elements and then use MDS to find a 3D Euclidean embedding of the shape with these distances. The result is an unfurling of articulation such that Euclidean distances in the embedded shape approximate geodesic distances in the original shape. One may now find a weighted Voronoi partition in the MDS space and optionally evaluate objective functions in this space or the original undistorted one as is appropriate for the given objective. The resulting segmentation is trivially mapped by element correspondence to the original surface. Figure 5.2 illustrates this approach by cleanly segmenting a highly articulated octopus model. The results of Figures 5.5 and 5.8 were also obtained using this technique.

5.4 Initial center placement

The first step in our optimization framework is to automatically find a coarse initial placement of partition centers which will serve as the initial guess for the optimization. We achieve this by way of k -means clustering where k is the known number of labels. This approach has the advantages of simplicity and the fact that it produces a Voronoi partition by construction, given that it is based on distance to cluster centers. It also naturally produces compact, similarly sized segments.

The centers are initialized using furthest point initialization: choose the farthest pair of surface points and then, while centers remain to be initialized, select the point with maximum closest distance to the points chosen thus far. If adjacency information is available and connectivity of segments is desired, the classification step of the k -means algorithm can be done using the queue approach described in Algorithm 1. Should a segment become empty, we select as its new center the surface element furthest from the current non-empty centers. Initial segmentations resulting from this approach are illustrated in Figures 5.2 (left), and 5.3 (left.)

5.5 Assigning labels to segments

A set of partition centers induces a shape segmentation. In many, though not all (see applications section) cases however, the segmentation will refer to objectives of a heterogeneous nature. For instance, in the hammer example of Figure 5.1 the *narrow* objective term refers only to the segment labeled *handle*. This means that labels must be assigned to each center prior to objective evaluation, as different labelings produce different objective function values. When there are relatively few labels and the objective is not computationally expensive, an exhaustive approach of all permutations may be feasible. As the number of segments grows, this approach quickly becomes intractable. In the following, we present two alternatives to address this labeling problem.

5.5.1 Optimal unary labeling

If the segmentation makes use of only unary objectives, or if we consider only the set of unary objective terms for the purposes of addressing the assignment problem, it is possible to efficiently find an optimal labeling.

Assume that for each label j we have a set of k unary objective terms $\mu_{j,1}, \mu_{j,2}, \dots, \mu_{j,k}$. We can build an $n \times n$ cost matrix C such that

$$C_{i,j} = \sum_k \mu_{j,k}(i)$$

which represents the cost of assigning label j to the part associated with center i for all i, j pairs.

Given this cost matrix, we can now cast part labeling as an assignment problem which can be optimally solved in $O(n^3)$ by the Hungarian algorithm [56, 80]. Given C the algorithm will return an assignment vector $\mathbf{a}$ such that $\mathbf{a}[i] = j$ indicates that the part associated with segment i should receive label j to minimize the sum cost.

It should be noted that one may choose to use only unary objectives to address the assignment problem if they are descriptive enough. Once the assignment is obtained, all objectives should be evaluated to determine fitness during the optimization described in Section 5.6.

5.5.2 Complexity of n -ary labeling

When binary or higher order objectives become involved in the segmentation description, the problem grows in complexity. In fact, when binary objectives are present, we may perform a straightforward reduction to show that our labeling problem includes the Traveling Salesman Problem (TSP).

In effect, map cities to segments consisting of single points, and assume there are n such cities. Further assume we have labels $l_1, l_2, \dots, l_n$ indicating that the city labeled l_i should be the i -th visited on the tour. Assuming we have the binary objective *dist*,

where $dist(l_i, l_j)$ denotes the distance between the cities labeled l_i and l_j , then the labeling which minimizes the aggregate objective

$$\sum_{i=1}^n dist(l_i, l_{(i \bmod n)+1})$$

is the solution to the given TSP instance.

For any given k and label assignment, it follows that we may check that the sum objective is lesser or equal to k in linear time, and the reduction is also linear in the number of cities. This shows the problem to be NP-complete.

Thus motivated, we address the n -ary objective labeling problem with an evolutionary approach. Such approaches have had success in the past with problems such as TSP and other ordering problems [75].

5.5.3 Evolutionary labeling

An individual in our population is naturally represented using permutation encoding: a vector $\mathbf{x}$ of length n where $\mathbf{x}[i] = j$ indicates segment i will receive label j . The fitness of an individual will be determined by the value the aggregate objective function takes when using the labeling specified by the individual. We use tournament selection as our selection strategy, in which a small random subset of the previous population is considered and the individual with highest fitness is selected. An individual can be mutated by uniform randomly selecting two of its entries and interchanging them. Finally, given two individuals $\mathbf{x}_1$ and $\mathbf{x}_2$, we can produce an offspring candidate $\mathbf{y}$ using uniform crossover. In particular, we generate a uniformly random bit vector $\mathbf{b}$ and let $\mathbf{y}[i] := \mathbf{x}_1[i]$ if $\mathbf{b}[i] = 0$ and $\mathbf{y}[i] := \mathbf{x}_2[i]$ if $\mathbf{b}[i] = 1$ unless $\mathbf{x}_2[i]$ already appears in $\mathbf{y}$ from $\mathbf{x}_1$. After crossover, any such conflicting entries are repaired, assigning them randomly from the set of remaining (unassigned) labels.

This evolutionary labeling approach is specified in Algorithm 2. In particular we use a population size of 20, a crossover fraction of 80%, a tournament size of 4, and typically

Algorithm 2 Evolutionary optimal objective labeling

```

1: // Initialize population  $\mathbf{P}$ 
2: for  $i = 1$  to populationsize do
3:      $\mathbf{P}[i] \leftarrow$  random permutation of the  $n$  labels
4: loop
5:     // Evaluate fitness
6:     for  $i = 1$  to populationsize do
7:          $\mathbf{F}[i] \leftarrow$  aggregate obj. function value with assignment  $\mathbf{P}[i]$ 
8:         Remember best fitness and individual so far
9:      $i \leftarrow 1$ 
10:    // Generate crossover individuals
11:    while  $i \leq \text{crossoverfraction} * \text{populationsize}$  do
12:         $\mathbf{x}_1 \leftarrow \text{tournamentselection}(\mathbf{P}, \mathbf{F})$ 
13:         $\mathbf{x}_2 \leftarrow \text{tournamentselection}(\mathbf{P}, \mathbf{F})$ 
14:         $\mathbf{P}'[i++] \leftarrow \text{uniformcrossover}(\mathbf{x}_1, \mathbf{x}_2)$  // w/repair
15:    // Generate mutation individuals
16:    while  $i \leq \text{populationsize}$  do
17:         $\mathbf{x} \leftarrow \text{tournamentselection}(\mathbf{P}, \mathbf{F})$ 
18:         $\mathbf{P}'[i++] \leftarrow \text{mutate}(\mathbf{x})$ 
19:     $\mathbf{P} \leftarrow \mathbf{P}'$ 
20: return individual with best fitness observed

```

achieve an optimal labeling in under 10 iterations.

It should be noted that the labeling objective is free to differ from the segmentation objective. The labeling objective may describe part adjacencies and similarities, while the segmentation objective describes the desired final configuration of parts.

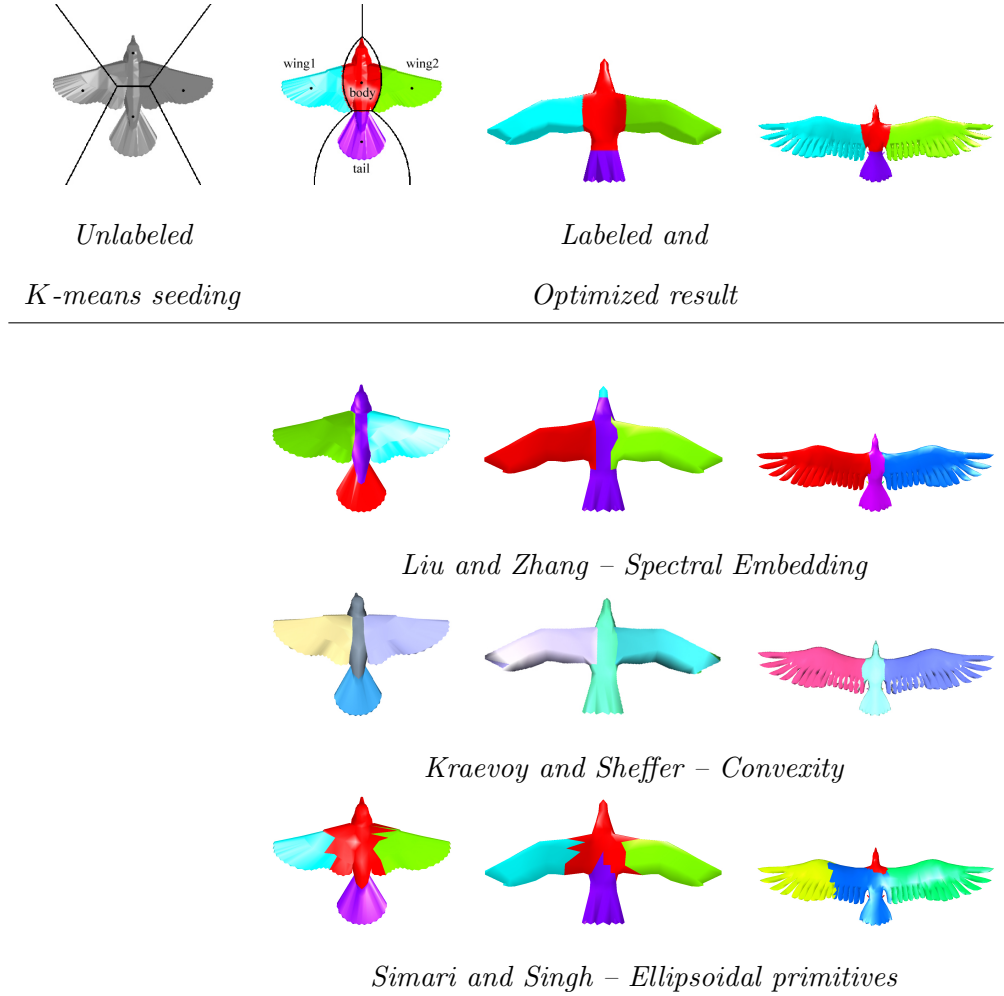


Figure 5.3: Segmentation resulting from simultaneously optimizing for flatness of wings and tail, narrowness of body and compactness of tail, while incorporating the constraint information that the body and tail lie on the plane of symmetry and that the wings are symmetric to each other. (See Section 5.7 and Figure 5.9 for details.) **Upper left:** Result of k -means approach to initialization. **Upper right:** Labeled and optimized result along with another two bird models consistently segmented and labeled. **Below:** For comparison purposes, the result of applying the approaches of Liu and Zhang [64], Kraevoy and Sheffer [54] and our ellipsoidal segmentation. Note that only our approach achieves semantic consistency across shapes.

5.6 Segmentation optimization

Having addressed the matters of center initialization and label assignment, we now focus on segmentation optimization.

If we wish to obtain a segmentation consisting of n segments, as previously described, we can consider the segmentation parameterized by a real vector $\mathbf{x}$ of dimensionality $m = 4n$ of the form

$$\mathbf{x} = (x_1, y_1, z_1, w_1, x_2, y_2, z_2, w_2, \dots, x_n, y_n, z_n, w_n)$$

where (x_i, y_i, z_i) are the 3D coordinates of the i -th Voronoi center and w_i is its associated weight. The task is now to search for a value of $\mathbf{x}$ that minimizes the aggregate objective function when evaluated on the segmentation induced by $\mathbf{x}$.

Gradient-descent type approaches are not suitable given the discrete nature of the representation: a center must be altered sufficiently to induce a change in the labeling of at least one surface primitive. We instead choose generalized pattern search (GPS), which is a derivative-free, direct search method [111, 10].

Given a pattern size Δ , at each iteration the method evaluates the aggregate objective function at all $2m$ neighbors formed by adding and subtracting Δ to each coordinate of the current $\mathbf{x}$. If any such neighbor produces a lower objective value, the iteration is considered successful, $\mathbf{x}$ is updated and Δ is multiplied by an expansion factor. Otherwise, the iteration is considered unsuccessful, no update of $\mathbf{x}$ occurs, and Δ is multiplied by a contraction factor. The algorithm terminates when Δ falls below a given threshold. In particular, we use an initial Δ of $.2d$ where d is the shape’s bounding box diagonal, an expansion factor of 1.2, a contraction factor of .5 and a Δ threshold of $10^{-4}d$. We typically observe convergence in fewer than 50 optimization iterations.

Symmetry constraints: Recent methods allow for the robust automatic detection of symmetries in 3D shape [51, 109, 100, 78, 89] and, whenever possible, shape processing

algorithms should leverage this redundancy. An advantageous property of our framework for parameterizing mesh segmentations is that it is easy to apply symmetry constraints to optimization parameters. For instance, if one segment is known to be symmetric to another, then the latter’s Voronoi center position and weight can be generated by symmetry from the first. If a segment is known to lie on a plane of global symmetry, then its Voronoi center can be constrained to lie on the plane. The advantages of exploiting this known redundancy are two-fold: first, if a shape is known to have a certain global symmetry, then the segmentation is assured to have the same symmetry by construction. Second, the dimensionality of the optimization domain is reduced by removing parameters which can effectively be generated from others. The user need only provide a constraint function which takes in the non-redundant parameters and produces the others from known symmetry. Figure 5.3 illustrates this in our results on bird models by constraining one wing to be symmetric to the other and the body and tail to lie on the plane of global symmetry.

Hierarchical segmentation: Our approach can easily be used to segment by multiple objectives hierarchically. Figure 5.4 illustrates a case where we first segment to maximize planar symmetry of parts, then by convex components, and then integrate the result. In the applications section we will also describe how our approach can be used to obtain hierarchical bounding volumes for path tracing acceleration (Figure 5.7) as well as a semantically meaningful hierarchical segmentation of humanoid characters (Figure 5.8.)

5.7 Applications

Our framework and the following applications were implemented in Matlab. Labeling plus optimization time is ~ 3 min. per segmentation on average, but of course this will depend on mesh density, the objective functions used, and the efficiency of their implementation. All experiments were run on a Pentium M 2.13Ghz processor with 2Gb RAM.

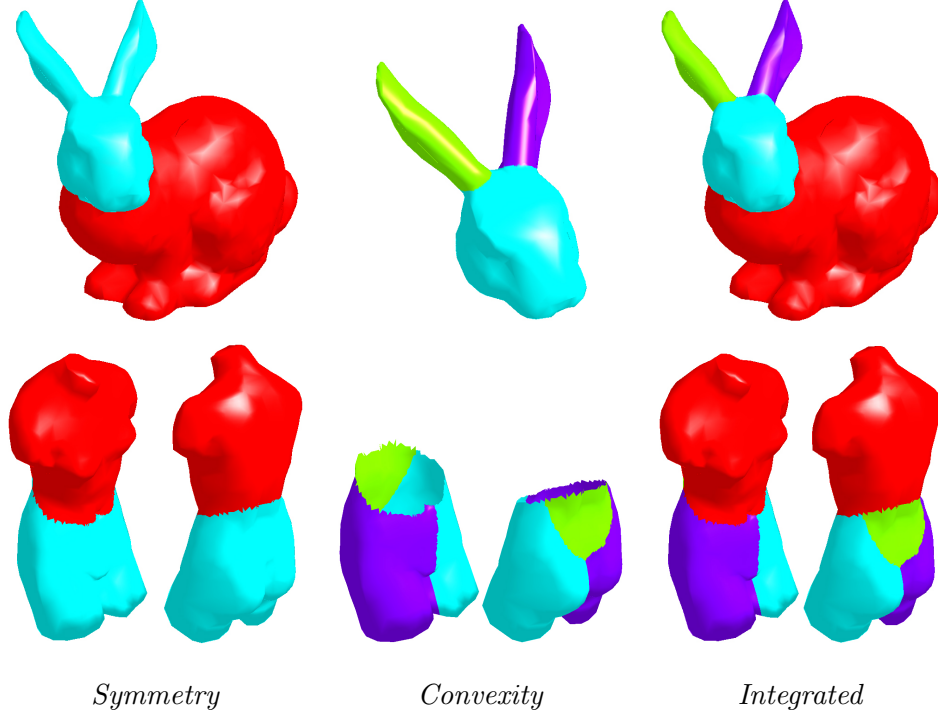


Figure 5.4: Our approach can easily be used to segment by multiple objectives hierarchically. Here we first segment to maximize planar symmetry of parts, then by convex components, and then integrate the result.

5.7.1 Semantically meaningful segmentations

Figures 5.1 through 5.6 show examples of semantically meaningful segmentations and associated labelings obtained using our method. The decompositions are determined by user-specified objectives which encode a description of the parts of the model and their interrelations (see table in Figure 5.9). We now define the objective function terms used to describe these segmentations.

Given a segmented surface, let P be the surface segment associated with a given label. Define $P.scale_i$, $i \in \{1, 2, 3\}$ as the part’s scales resulting from PCA, *i.e.* as the square root of the eigenvalues of the surface’s 3×3 covariance matrix, sorted in descending magnitude. We may define the following unary objectives named with the adjective to which they intuitively correspond

$$\text{narrow}(P) := \frac{\frac{1}{2}(P.scale_1 + P.scale_2)}{P.scale_3}$$

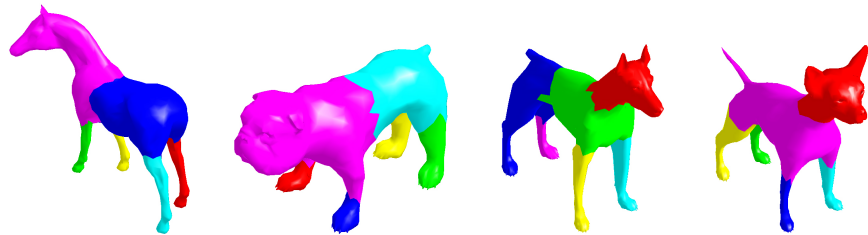
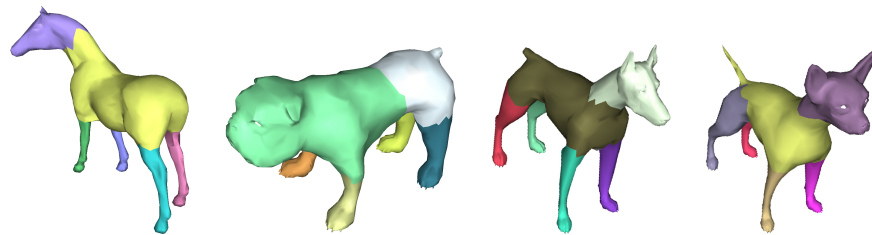
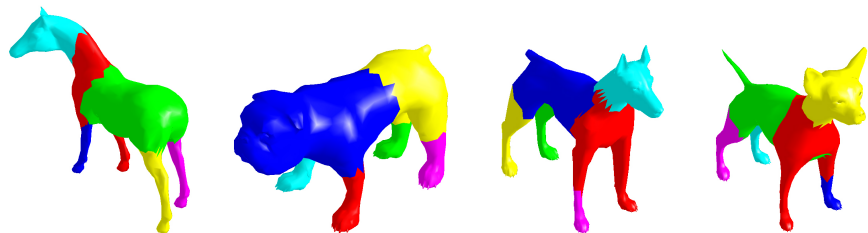
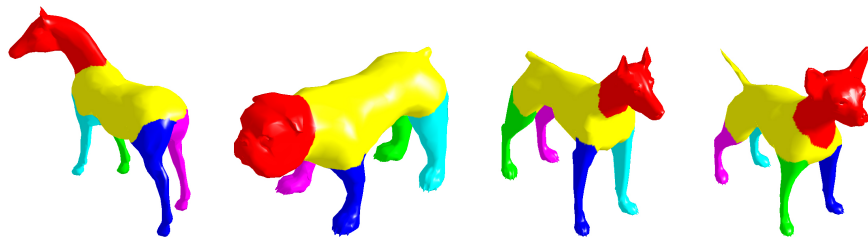
*Liu and Zhang – Spectral Embedding**Kraevoy and Sheffer – Convexity**Simari and Singh – Ellipsoidal primitives**Multi-objective – Labeled and optimized*

Figure 5.5: Comparison of our approach to the result of applying Liu and Zhang [64], Kraevoy and Sheffer [54] and our ellipsoidal segmentation. Note that only in our results are the head, body and legs consistently segmented and labeled across models (leg labels are considered interchangeable.) Also, given the nature of our approach, we are able to clip boundaries below mesh resolution.



Figure 5.6: In these cases, our multi-objective approach (**above**) can obtain results that correspond with those obtained from human hand-done segmentations [19] (**below**).

$$\text{flat}(P) := \frac{1}{2} \left(\frac{P.\text{scale}_3}{P.\text{scale}_1} + \frac{P.\text{scale}_3}{P.\text{scale}_2} \right)$$

$$\text{compact}(P) := 1 - \text{narrow}(P)$$

Let us further define $P.c$ as part P 's centroid and $P.\text{axis}_i$, $i \in \{1, 2, 3\}$ as the part P 's i -th normalized eigenvector also resulting from PCA. We may then define the objectives

$$\text{planarsymmetric}(P) := \min_i \text{surfdist}(P, \text{reflect}(P, P.c, P.\text{axis}_i))$$

$$\text{ellipsoidal}(p) := \text{surfdist}(P, \text{covarellipsoid}(P))$$

We define $\text{surfdist}(S_1, S_2)$ as the sum of area-weighted squared distance of points from S_1 to S_2 normalized by the total area of S_1 and by its bounding box diagonal length. In turn $\text{reflect}(S, p, \vec{n})$ represents the planar reflection of surface S about the plane determined by point p and normal $\vec{n}$. This is similar to the symmetry metric used by Simari *et al.* [100] but is non-iterative. Naturally, we define the $\text{covarellipsoid}(P)$ as the covariance ellipsoid of part P determined by its centroid, and PCA scales and axes.

Let us also define the following n -ary objectives that refer to label interrelations. Specifically

$$\text{perpendicular}(P_1, P_2) := |P_1.\text{axis}_1 \cdot P_2.\text{axis}_1|$$

$$\text{similarsize}(P_1, P_2, \dots, P_k) := \frac{1}{d} \sum_i ||P_i.\text{scale} - \hat{s}||$$

where d is the global surface's bounding box diagonal, $P_i.scale$ refers to the part's entire 3×1 scale vector and $\hat{s} = \frac{1}{k} \sum_i P_i.scale$.

Finally we define the global objective

$$\text{convexparts}(Seg) := \left(\frac{1}{V} \left| \left(\sum_{P \in Seg} H(P) \right) - V \right| \right)^{\frac{1}{3}}$$

where Seg refers to the segmentation, $H(P)$ is part P 's convex hull volume and V is the volume enclosed by the total original surface.

The table in Figure 5.9 describes the specific objectives and constraints (where applicable) used in each of the results of Figures 5.1, 5.2, 5.3, 5.5 and 5.8. The first three were automatically labeled using the optimal unary approach while the last two were done using the evolutionary approach.

5.7.2 Bounding volume hierarchies for path tracing

Bounding volume hierarchies are often used to accelerate path tracing. A common way to obtain such hierarchies is through the use of an octree which at each level divides the space into axis-aligned octants through the object's centroid.

We use our segmentation approach to obtain axis-aligned bounding box hierarchies by segmenting a model using an objective which optimizes the tightness of the bounding volumes:

$$\text{bbox}_1(Seg) := \sum_{P \in Seg} AAB B(P)$$

where $AAB B(P)$ is part P 's axis-aligned bounding box volume.

Alternatively, our method allows us to optimize bounding volume tightness simultaneously with distribution of faces:

$$\text{bbox}_2(Seg) := \left(\frac{1}{H(P)} \sum_{P \in Seg} AAB B(P) \right)^{\frac{1}{3}} + cv(\{|P| : P \in Seg\})$$

where $H(P)$ is P 's convex hull volume and cv is the coefficient of variation defined as the ratio of the standard deviation to the mean of a set. This term favors segmentations

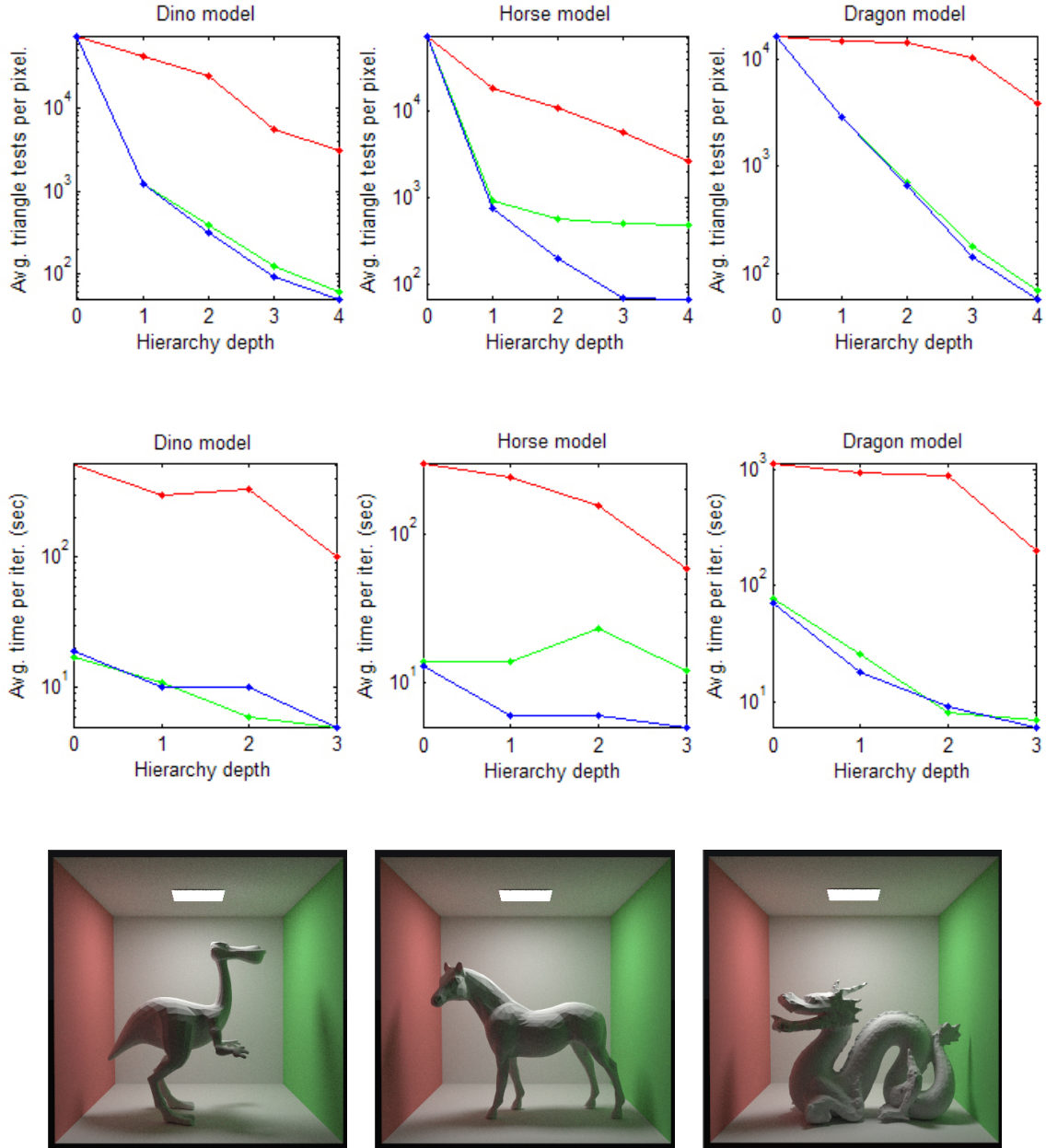


Figure 5.7: Comparison of average number of triangle intersections per pixel (top) and average time per iteration in seconds (centre) computed (on $\log_{10}$ scale) as a function of axis-aligned bounding box hierarchy depth. An iteration involves launching primary and recursive secondary rays from the eye through every pixel in the image, terminated using a Russian Roulette scheme. Curves correspond to octree (red) and our method’s results when optimizing volume tightness (green) and a combined objective of volume tightness and uniform partitioning (blue.)

where all segments have a similar number of faces.

Figure 5.7 shows a comparison of the bounding boxes obtained with octree partitioning and those obtained by our method using either objective on three scenes. In each case, we show an order of magnitude improvement in triangle intersection tests per pixel rendered, with the combined objective function showing further improvement over the single objective. Note that in this application labeling is unnecessary since the objectives are homogenous to all parts.

5.7.3 Automatic character rigging

Another useful application is that of correctly placing a given animation skeleton, such as one available from motion capture, on a given mesh of roughly the same class, *e.g.* bipedal humanoid.

Baran and Popović [13] recently address this problem in their Pinocchio framework. Pinocchio discretizes the problem of embedding the skeleton by constructing a graph whose vertices represent potential joint positions and whose edges are potential bone segments. These vertices are found by packing spheres computed from the medial surface. Then, the method learns a good embedding of the skeleton into the graph using a maximum margin method based on a manually chosen set of selected training examples with associated labels of “good” or “bad” embeddings and a set of features that penalize short bones, improper orientation between joints, length differences in bones marked symmetric, *etc.* Finally, a gradient descent method is used to refine the skeleton by minimizing a cost function that penalizes bones that do not fit inside the surface nicely, bones that are too short, and bones that are oriented differently from the given skeleton.

Here, we propose an alternative approach which uses our multi-objective segmentation framework. Given a humanoid mesh, we:

1. Perform an MDS embedding as described in Section 5.3 to undo any articulation and perform a few steps of smoothing to remove high frequency detail that is not

relevant to the coarse-level segmentation.

2. Segment the result into six parts according to a descriptive set of objectives, the part labels being *head*, *torso*, *arm_left*, *arm_right*, *leg_left*, and *leg_right*. The labeling objective asserts part adjacencies as well as similarity between the two arms and similarity between the two legs. The segmentation objectives then assert part properties and interrelationships as detailed in Figure 5.9. Left and right are disambiguated assuming a close-to-canonical orientation (as Baran and Popović also do.) For the dragon model, which has a large tail, an extra part is added without any objective information.
3. Each part is then decomposed hierarchically (see Section 5.6) in the original space into regions by means of convexity. Arms are segmented into upper arm, forearm, and hand. Legs are segmented into upper leg, lower leg and foot. The torso is segmented into chest and abdomen. These subparts are easily and unambiguously labeled based on geodesic proximity to the torso, in the case of the arms and legs, and to the head in the case of the chest and abdomen.

These objectives and hierarchy result in the segmentations shown in the top row of Figure 5.8. Note the consistency achieved across a range of varied models using a general set of objectives which describe what we know about humanoids and the animation skeleton.

The animation skeleton can now be easily placed according to this segmentation. The joints corresponding to the head, chest, abdomen, hands and feet are placed at the corresponding part centroid. Joints corresponding to shoulders, elbows, hips, knees, ankles, and mid-body are placed at the centroid of the corresponding part boundaries. The second row of Figure 5.8 shows in blue the skeleton placement that results from our segmentations.

For comparison, the third row of Figure 5.8 shows in red the skeleton placement

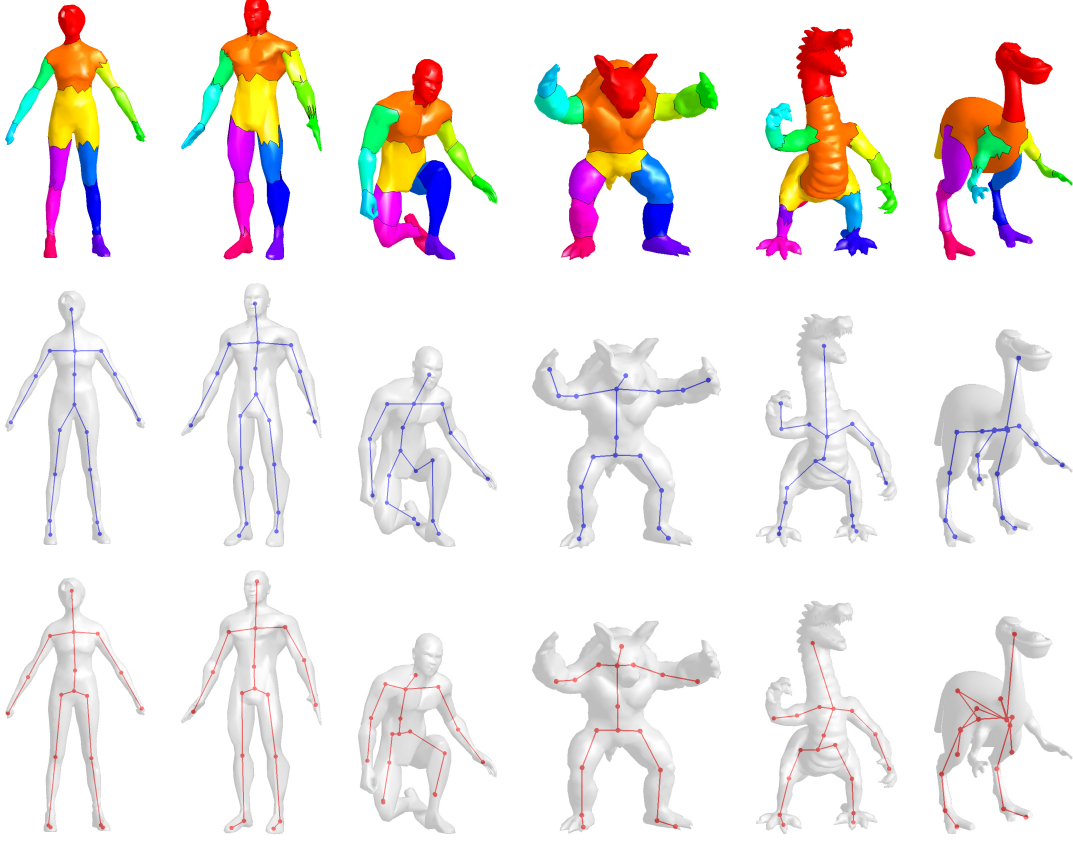


Figure 5.8: **Top row:** Our multi-objective segmentation consistent across models. **Center row (blue):** Resulting skeleton placement using part and part-boundary centroids. **Bottom row (red):** Skeleton placement of Baran and Popović.

produced by the Pinocchio system of Baran and Popović. Our approach results in similar placements for well posed characters, such as the first two. However, our approach needed no training since it directly received the user knowledge in the form of the segmentation objectives. Moreover, as can be observed, our approach much better handles characters under articulation as well as those with large variations in part proportions and locations. This is the case of the dino model, in which the Pinocchio’s skeleton embedding failed to find the dino model’s small arms. On the other hand, our approach will naturally fail when expected parts are not present, whereas Pinocchio can still produce a result in these situations. We thus see the two approaches as complementary.

<table border="1"> <tr> <th colspan="3">Summary of objectives used</th></tr> <tr> <td colspan="3"> narrow($\cdot$), flat($\cdot$), planarsymmetric($\cdot$), ellipsoidal($\cdot$), perpendicular($\cdot, \cdot$), similarsize($\cdot, \cdot$), convexparts($\cdot$) </td></tr> </table>			Summary of objectives used			narrow($\cdot$), flat($\cdot$), planarsymmetric($\cdot$), ellipsoidal($\cdot$), perpendicular($\cdot, \cdot$), similarsize($\cdot, \cdot$), convexparts($\cdot$)		
Summary of objectives used								
narrow($\cdot$), flat($\cdot$), planarsymmetric($\cdot$), ellipsoidal($\cdot$), perpendicular($\cdot, \cdot$), similarsize($\cdot, \cdot$), convexparts($\cdot$)								
Model	Labels	Segmentation objectives						
Hammer	$handle, head$	$5 * \text{narrow}(handle)$, $\text{perpendicular}(handle, head)$, $\text{convexparts}(Seg)$						
Quadruped	$head, body, leg_1, \dots, leg_4$	$\forall_i \text{narrow}(leg_i)$, $\text{similarsize}(leg_1, \dots, leg_4)$, $\text{compact}(head)$, $10 * \text{convexparts}(Seg)$ In dogs, compactness of head is emphasized with $8 * \text{compact}(head)$						
Bird	$body, wing_1, wing_2, tail$	$\text{narrow}(body)$, $\forall_i \text{flat}(wing_i)$, $\text{flat}(tail)$, $\text{compact}(tail)$, $10 * \text{convexparts}(Seg)$ Constraints: body and tail lie on plane of global symmetry and parameters for $wing_2$ are reflected from $wing_1$.						
Octopus	$head, arm_1, \dots, arm_8$	$\text{ellipsoidal}(head)$, $\forall_i \text{narrow}(arm_i)$, $\text{similarsize}(arm_1, \dots, arm_8)$, $10 * \text{convexparts}(Seg)$						
Humanoid	$head, torso, arm_left$ $arm_right, leg_left, leg_right$	$\text{narrow}(arm_left)$, $\text{narrow}(arm_right)$, $\text{narrow}(leg_left)$, $\text{narrow}(leg_right), \dots$ $\text{compact}(head)$, $\text{similarsize}(arm1, arm2)$, $\text{similarsize}(leg1, leg2)$, $10 * \text{convexparts}(Seg)$ Subparts (upper arm, forearm, hand, etc.) are obtained hierarchically using convexity.						

Figure 5.9: Objectives used to obtain segmentations of each model. Notice that weights remain constant within each class of object.

Automatic character dressing

Another compelling application of the semantically-labeled humanoid characters above is the ability to automatically clothe such characters. Clothes and accessories parametrically defined on a segmented and labeled mannequin can be easily transferred to dress a new character.

Accessories such as the hat and shoes (see Figure 5.10) are defined with respect to parts such as the head and feet on the mannequin. Dressing a given semantically-segmented character with a desired accessory requires transforming it to the correspondingly labeled part. We define this as the affine transform that maps the oriented bounding box (OBB) of the mannequin’s part to the OBB of the corresponding part on the given character.

Clothing such as the shirt and shorts (Figure 5.10) and accessories that drape across parts are first coarsely deformed to fit the new character and then relaxed by simulation to drape naturally. We automatically fit spline curves to the boundaries of the parts that the item of clothing drapes over. These corresponding curves (left of Figure 5.10) on the mannequin and target character define a wire deformation [103], that coarsely reshapes the clothing around the new character. To ensure that the entire item of clothing is deformed, the radius of influence of the wire curves is set conservatively large to 1.5 times the largest bounding box dimension of its adjacent parts on the mannequin. Finally, a cloth simulation drapes the coarsely fitted clothing over the target character.

The results on three characters, female, armadillo and muscleman, are illustrated at the right of Figure 5.10. As can be seen, in the case of the armadillo the transfer only works well for the shoes and not the shirt and hat, understandably, because of the oddly shaped torso and protruding ears. The coarse fitting of the shorts cause the tail to protrude resulting in the shorts relaxing over the body with a protruding tail.



Figure 5.10: **Above:** A character with deformers associated to its clothing. **Below:** Clothing scaled and transferred automatically onto the semantically correspondent places. Clothing such as the shirt and shorts that drape across parts are first coarsely deformed using a wire deformation [103] to fit the new character and then relaxed by simulation to drape naturally. Notice that in the case of the armadillo, its torso and head prevent the adequate placement of a shirt or hat, but the shorts and shoes are still transferable.

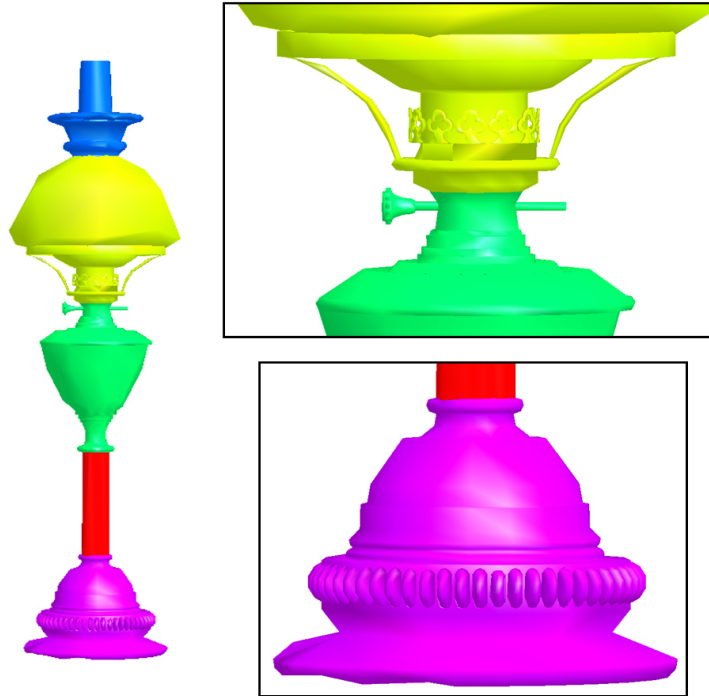


Figure 5.11: Our spatial parametrization can easily handle polygon-soup models by ignoring mesh connectivity. Here we segment such a model into convex parts. Closeups show non-manifold regions with interpenetrating faces and small disconnected components. Such a model could not be handled by current methods that require manifold geometry.

5.8 Limitations

We recognize that there are classes of objects which our parametrization might have difficulty with as well as our parametrization’s limitations with respect to complex boundaries (Figure 5.12). Our approach captures simple boundaries and more complex boundaries could be addressed in a refinement pass using min-cuts or active contours, for example.

We can perhaps look at this parametrization with the analogy of spatial volume-based vs. surface-based deformation techniques. Both have pros and cons and have been the subject of ongoing research for over 25 years. This is the first volume-based segmentation parametrization approach and is likely to stimulate further research in this direction. It

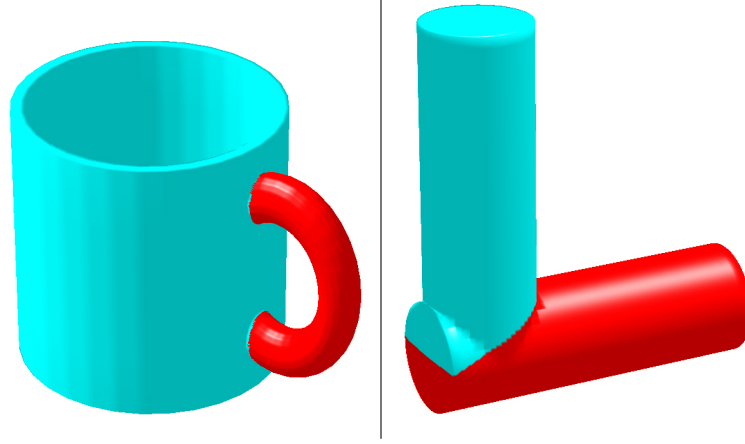


Figure 5.12: **Left:** Our spatial parametrization can segment shapes of non-zero genus. **Right:** Our approach captures simple boundaries. More complex boundaries could be addressed in a refinement pass using, *e.g.* min-cuts or active contours. Both models are segmented using our minimum convex hull volume criterion.

also shares pros and cons analogous to those of volume-based deformations. It is simple and compact (a vector of $4k$ scalars, where k is the number of segments) and decouples the segmentation search from the surface resolution-dependent objective function evaluation (most objectives we use are $O(n)$, where n is the number of surface elements).

Our approach is surface representation-agnostic, works with non-manifold or disconnected objects (Figure 5.11), can address non-zero genus shapes (Figure 5.12), and we show it to work well on examples typical to segmentation literature. As with the many extensions on the original box-shaped FFDs we expect work subsequent to this paper to be able to capture richer volume shapes with more complex boundaries.

We show our results on what could be considered a relatively low number of parts. However, in many real-world applications this is more than sufficient, especially (as we show) when the method is applied hierarchically. We present practical results across several such applications and show them to compare favorably to state-of-the-art methods.

With respect to initialization, our k -means approach has limitations as to the kinds of segmentation to which it is suitable. Figure 5.13 shows examples in which the nature



Figure 5.13: Our k -means approach has limitations as to the kinds of segmentation to which it is suitable. In these examples, the nature of the objects is such that the placement of parts by k -means is very distant from a desirable segmentation, making it an inadequate initialization. **Above:** Human segmentation [19]. **Below:** k -means initialization.

of the objects is such that the placement of parts by k -means is very distant from a desirable segmentation, making it an inadequate initialization.

Manual intervention, allowing for user placement of initial segmentation parameters, can be added to our system with no conceptual difficulty. In our work, user input can help with both the segmentation and labeling. Our goal, however, is to be able to consistently label and segment not one, but entire classes of objects, and our approach is the first to automate this process based on multiple heterogeneous black-box objectives.

The formulation of the objective functions is not an automated process. However, a single weighted combination of simple objectives, once defined, can suffice to segment and label an entire class of objects, as illustrated by Figure 5.3, 5.5, and 5.8. While

the objective functions we provide have intuitive meaning and are easy to reuse, we acknowledge that the choosing of these objectives and the fine tuning of their weights can be made more user friendly, or perhaps be learned using statistical methods and we hope to address this in the future.

5.9 Summary

We motivated the need for a segmentation framework which allows the incorporation of application and model specific knowledge to the segmentation algorithm. As we have shown, multiple heterogeneous objectives for different parts allows both a more robust segmentation and a consistent labeling of parts across a class of objects. To accomplish this we have introduced the notion of multi-objective shape segmentation and labeling and the use of multiplicatively weighted Voronoi space partitioning as an approach to parameterizing segmentations. Our optimization framework is general, comprising algorithms for coarse center placement, segmentation labeling as a function of objectives, and center refinement. We showed how our approach can accommodate symmetry constraints, incorporate segmentation hierarchies, and how shapes under complicated articulation can be handled by using multi-dimensional scaling.

We validated our approach by demonstrating its use on several applications. We applied it to obtain semantically meaningful labeled segmentations, optimized bounding volume hierarchies for path tracing, animation skeleton placement for automatic character rigging, and automatic transfer of clothing for animation characters.

Chapter 6

Conclusion

6.1 Summary

In this thesis we have motivated and studied the problem of 3D shape segmentation. We began by providing an overview and discussion of the previous work in the area, as well as a classification which grouped the approaches into the three major categories of affinity-based, model-fitting and property-based methods.

We then presented the first two algorithmic contributions of this thesis, both of which had applications in automatically finding shape redundancy and which proposed shape representations that leverage this redundancy to achieve descriptive compactness.

The first was to the area of model-fitting approaches through the introduction of a method for segmenting a surface into piece-wise ellipsoidal parts. We explored the potential of compactly approximating a shape using a relatively small number of ellipsoidal regions motivated by the fact that most organic objects and many manufactured objects have large curved regions.

We further noted that ellipsoids, unlike planes, being closed surfaces, are capable of providing information about the volume of closed objects. This means that we are also able to represent the volume enclosed by a given surface as the union of a set of ellipsoidal

volumes. In this case, we used the entire ellipsoid and not just regions of its surface.

Once a shape has been segmented into an ellipsoidal approximation, it is only necessary to store the ellipsoids and, in the surface case, the segmentation boundaries at which they meet in order to represent the object.

Our second contribution was to the area of property-based approaches by automating the search for symmetry, another form of shape redundancy which is ubiquitous in natural and human-made shapes. We presented an algorithm for robustly detecting global and local approximate planar-reflective symmetry and a hierarchical segmentation approach based on this detection method. We then showed how this algorithm could be used to simplify a shape by removing redundant elements. We stored only the non-redundant parts and respective planes of symmetry in a novel structure we call a “folding tree.” We also showed how an approximation of the original shape could be recovered from this representation in a bottom up fashion by reflecting the non-redundant elements and reconnecting segmentation boundaries.

As our final contribution, we proposed a way to generalize the segmentation problem to heterogeneous criteria. We noted that criteria such as ellipsoidal shape, symmetry, or indeed any other criteria, need not be applied homogeneously to all segments of a shape. This is to say, a user may wish to assert that a given segment is ellipsoidal, that another is symmetric, and a third respects yet another property.

In fact, the users of segmentation algorithms may possess a wealth of knowledge about the objects they are analyzing and working with. However, current algorithms for shape processing did not allow for the incorporation of detailed domain knowledge and usually limited their input to the number of segments desired and a handful of threshold values. If a user wished to automate the segmentation of models according to his domain-specific knowledge, the current approach would be to implement a new segmentation algorithm from scratch. This is often out of the realm of most users. Even when it is not, such a ground-up implementation would require a considerable investment of time and resources.

We introduced a framework and relevant algorithms for multi-objective segmentation of 3D shapes, which allow for the incorporation of domain-specific knowledge through heterogeneous objectives, each of which refers to one or more segmentation labels. These objectives can be unary, asserting properties of an individual part associated with a given label or they can be n -ary, referring to part interrelations. We thus cast the segmentation problem as an optimization minimizing an aggregate objective function which combines all objectives as a weighted sum.

6.2 Discussion

Our ellipsoidal segmentation and local symmetry based approaches have in common the fact that they can be applied as surface simplification approaches. The ellipsoidal approach uses an external model of a smooth surface, while the symmetry based approach leverages the shape’s own redundancy in the form of self-similarity.

The ellipsoidal segmentation algorithm is based on a variational approach and our implementation requires that the number of ellipsoidal primitives be pre-specified. In contrast, our symmetry-based segmentation uses a tolerance parameter and proceeds hierarchically in its segmentation until no further symmetries can be found without violating the given error threshold. These limitations are not intrinsic to the approaches, however, and one can imagine dual implementations. For example, in the case of the ellipsoidal segmentation, instead of specifying the number of primitives, we can easily specify an error threshold. Beginning with one primitive, upon convergence of the fitting/classification steps, we can insert a primitive at the point of highest error and re-start the iterations. This would proceed until the error threshold is not violated.

Our multi-objective segmentation approach decouples the segmentation algorithm from the objective, and one could imagine using it with ellipsoidal or symmetry objectives as opposed to the dedicated algorithms we propose. This would require the

implementation of an ellipsoidal and symmetry objective functions for each case. This would, however, defeat the purpose. We would lose the efficiency and robustness of these dedicated approaches which can quickly converge to the desired segments having their search especially tailored to their particular objective. More importantly, the use of our multi-objective framework would not be warranted if we wished to optimize a single objective for which dedicated algorithms already exist. The strength of our multi-objective framework is in its ability to assign multiple objectives, each referring to its own segment or to multiple segments simultaneously, according to the user's domain knowledge, without having to implement a dedicated algorithm from scratch.

As we have seen, there is a plethora of segmentation algorithms to choose from depending on the nature of the application at hand and the nature of the data. When faced with a segmentation problem, these two factors will play a large role when choosing an appropriate algorithm or designing a new one.

What is the nature of the segmentation sought? What is the criterion which is desired in a segment? If this criterion can be evaluated at the lowest level between atomic surface elements (*i.e.* faces or points) then an affinity based approach should likely be used. If on the other hand, there is some continuous parametric model which describes the parts (*e.g.* planes, cylinders, cones, spheres, *etc.*) then one of the model-fitting approaches will be best.

If the criterion is expressed as a property which cannot be captured at the atomic level or described by a continuous primitive, then a property-based approach may be in order. As we have seen, there currently exist algorithms for finding parts exhibiting convexity, symmetry, tubularity, and more. If none of these will do, or if no single criterion describes all parts and it would be desirable to also make assertions about inter-part relations, then our multi-objective approach provides a unique possibility.

6.3 Future directions

With respect to our approach to ellipsoidal segmentation, it could be pointed out that for highly dense meshes, the resulting segmentation boundaries may include many vertices. These boundaries are simply a sequence of points and can easily be simplified for a more compact representation. Also we could add another parameter to the ellipsoid representation in order to introduce bending in the style of superquadrics, allowing for banana-shaped surfaces. This could be useful both in volume and surface segmentations, in the latter case allowing the possibility of modeling saddle regions, with the trade off of decreasing the efficiency of geometric queries.

Regarding our approach to symmetry-based segmentation, given the fact that real objects, both organic and synthetic, often exhibit this type of data redundancy and human perception is strongly related to the notion of symmetry, a significant number of applications based on our methodology can further be developed. The elimination of faces which are repeated in redundant areas of global and local symmetries leads to new mesh compression schemes that can be used for mesh storage, processing, and transmission. Automatic segmentation, reconstruction and repairing of the meshes, driven by the extracted symmetries, is also another interesting field of application of our method. The folding tree representation could also facilitate skeleton extraction and advanced editing operations which preserve symmetries. Future research could also focus on the exploitation of other types of symmetries in 3D meshes that can open up new implementations and extensions of our proposed methodology.

Regarding improvements to our approach, in cases where there is no strong symmetry in the neighborhood of the principal axes, we may fail to find existing local symmetries in small regions. In cases such as these in which our initial guess does not provide a large enough support, an alternative initialization can be sought through random sampling or perhaps through a voting scheme like the ones described by Mitra *et al.* [78] or Podolak *et al.* [89].

Finally, there is our generalized multi-objective approach to segmentation. Specialized and efficient segmentation algorithms tailored to specific objective functions will always have their place. Our general multi-objective framework, however, allows for quick experimentation with various objective functions for existing and new segmentation applications.

We believe the area of general purpose segmentation and labeling algorithms is a useful and fertile area of study. One avenue of future research lies in the investigation of alternate schemes for parameterizing segmentations, as well as approaches to segment initialization and optimization.

During our research, before reaching the Voronoi space partition parameterization and generalized pattern search approach to optimization, we attempted other approaches. Our initial approach involved a direct parameterization of the segmentation by means of a vector which contained, for each face, the corresponding segment number. We then attempted to use simulated annealing and genetic algorithms which used random flooding to generate new segmentations and alter existing ones in ways that kept each segment connected. This parameterization was extremely flexible, allowing for the possibility of any segmentation, and the meta-heuristic optimization approaches made minimal assumptions on the nature of the objectives. However, even though we obtained some initial results for simple, homogenous objectives, the search space quickly became too vast for this unconstrained method to yield reasonable results.

We believe that perhaps a hybrid approach that reduces the number of elements by pre-clustering to a manageable number might make the direct-parameterization approach more viable. Such an approach could leverage existing segmentation methods to detect basic components of shape much in the same way as low-level computer vision approaches find edges, corners, regions of uniform texture, *etc.*. This type of approach has had success within image segmentation where the over-segmented clusters of uniform texture are called *superpixels* [90].

Another direction is that of automatically discovering the applicable objective functions and their weights for different parts from user-segmented and labeled example data such as that presented in the new Princeton segmentation benchmark [19].

Finally, there is also the potential of using segmentation as a means of shape description and retrieval. Given an objective function especially designed for a class of objects, one would expect that low error values would only be achievable for instances of objects in this class. Thus, the residual segmentation objective value might be useable as a discriminator.

The problem of what constitutes a meaningful segmentation remains, and likely will remain, an open question. As we have seen, surface segmentation as an area of study is directly dependant on and driven by application. We believe, however, that the area of meta-heuristic methods is ripe for exploration in a manner analogous to the meta-heuristic search methods of Artificial Intelligence. With the final contribution of this thesis, we have taken a first step, but there are many avenues open for continued research in approaches to parameterization, initialization, search, and specification and evaluation of segmentation objectives.

Bibliography

- [1] P. Alliez and C. Gotsman. *Recent Advances in Compression of 3D Meshes*. Springer-Verlag, 2005. ISBN 978-3-540-21462-5.
- [2] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. Recent advances in remeshing of surfaces. Research report, AIM@SHAPE Network of Excellence, 2005.
- [3] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete Computational Geometry*, 3(3):237–256, 1988.
- [4] A. S. L. K. Andrei Sharf, Thomas Lewiner. On-the-fly curve-skeleton computation for 3d shapes. *Eurographics*, 2007.
- [5] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of markov random fields for segmentation of 3D range data. In *IEEE CVPR*. 2005.
- [6] R. Arnheim. *Art and Visual Perception: A Psychology of the Creative Eye*. University of California Press, Berkeley, 1954.
- [7] M. Atallah. On symmetry detection. *IEEE Transactions on Computers*, 34(7):663–666, 1985.
- [8] M. Attene, B. Falcidieno, and M. Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *Visual Computer*, 22(3):181–193, 2006.

- [9] O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee. Skeleton extraction by mesh contraction. *ACM SIGGRAPH*, 2008.
- [10] C. Audet and J. E. D. Jr. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, 2003.
- [11] S. Bagon, O. Boiman, and M. Irani. What is a good image segment? a unified approach to segment extraction. In *ECCV*. 2008.
- [12] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. 2007.
- [13] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. *ACM SIGGRAPH*, 2007.
- [14] I. Biederman. Recognition-by-components: a theory of human image understanding. *Psychological Review*, 94(2), 1987.
- [15] S. Bischoff and L. Kobbelt. Ellipsoid decomposition of 3d-models. *3DPVT Proceedings*, pages 480–488, 2002.
- [16] S. Bischoff and L. Kobbelt. Towards robust transmission of geometric data. *Computers & Graphics*, 26(5):665–675, 2002.
- [17] M. J. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.
- [18] B. Chazelle, D. P. Dobkin, N. Shouraboura, and A. Tal. Strategies for polyhedral surface decomposition: an experimental study. *Computational Geometry: Theory and Applications.*, 7(5-6):327–342, 1997.
- [19] X. Chen, A. Golovinskiy, , and T. Funkhouser. A benchmark for 3D mesh segmentation. *ACM SIGGRAPH*, 2009.

- [20] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM SIGGRAPH*, 2004.
- [21] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE PAMI*, 24(5):603–619, 2002.
- [22] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE PAMI*, 24:603–619, 2002.
- [23] A. Criminisi, T. Sharp, and A. Blake. Geos: Geodesic image segmentation. *ECCV*, 2008.
- [24] F. de Goes, S. Goldenstein, and L. Velho. A hierarchical segmentation of articulated bodies. *Eurographics Symposium on Geometry Processing*, 2008.
- [25] P. Duygulu, K. Barnard, J. F. G. de Freitas, and D. A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV*. 2002.
- [26] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, first edition, 2002. ISBN 0130851981.
- [27] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975.
- [28] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. *ACM SIGGRAPH*, 2004.
- [29] Y. Gofman and N. Kiryati. Detecting symmetry in grey level images: The global optimization approach. *ICPR*, 1996.
- [30] A. Golovinskiy and T. Funkhouser. Randomized cuts for 3D mesh analysis. *ACM SIGGRAPH Asia*, 2008.

- [31] E. H. Gombrich. *Art and Illusion: A Study in the Psychology of Pictorial Representation*. Princeton University Press, Princeton, 1969.
- [32] A. Gueziec, F. Bossen, G. Taubin, and C. Silva. Efficient compression of non-manifold polygonal meshes. *IEEE Visualization*, 1999.
- [33] A. Gueziec, G. Taubin, F. Lazarus, and W. Horn. Converting sets of polygons to manifold surfaces by cutting and stitching. *IEEE Visualization*, 1998.
- [34] A. Gupta and L. S. Davis. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *ECCV*. 2008.
- [35] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley-Interscience, New York, USA, 1986. ISBN 0471735779.
- [36] R. M. Haralick and L. G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29:100–132, 1985.
- [37] X. He and R. Zemel. Learning hybrid models for image annotation with partially labeled data. In *NIPS*. 2008.
- [38] P. T. Highnam. Optimal algorithms for finding the symmetries of a planar point set. *Information Processing Letters*, 22(5):219–222, 1986.
- [39] H. Hoppe. Progressive meshes. *Computer Graphics*, 30(Annual Conference Series):99–108, 1996.
- [40] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. *SIGGRAPH*, 1993.
- [41] Q. Huang, M. Wicke, B. Adams, and L. Guibas. Shape decomposition using modal analysis. *Eurographics*, 2009.

- [42] M. Isenburg and P. Alliez. Compressing polygon mesh geometry with parallelogram prediction. *IEEE Visualization*, 2002.
- [43] A. Jaklic, A. Leonardis, and F. Solina. *Segmentation and recovery of superquadrics*. Kluwer Academic Publishers, 2000.
- [44] X. Jiang, K. Yu, and H. Bunke. Detection of rotational and involutorial symmetries and congruity of polyhedra. *Visual Computing*, 12(4):193–201, 1996.
- [45] T. Ju. Robust repair of polygonal models. *ACM Transactions on Graphics*, 23(3):888–895, 2004.
- [46] D. Julius, V. Kraevoy, and A. Sheffer. D-charts: Quasi-developable mesh segmentation. *Eurographics*, 2005.
- [47] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. *ACM SIGGRAPH*, 2000.
- [48] S. Katz, G. Leifman, and A. Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8-10):649–658, 2005.
- [49] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM SIGGRAPH*, 2003.
- [50] M. Kazhdan, B. Chazelle, D. Dobkin, T. Funkhouser, and S. Rusinkiewicz. A reflective symmetry descriptor for 3d models. *Algorithmica*, 38(1):201–225, 2004.
- [51] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Symmetry descriptors and 3d shape matching. *Eurographics Symposium on Geometry Processing*, 2004.
- [52] J. J. Koenderink and A. J. van Doorn. The shape of smooth objects and the way contours end. *Perception*, 11(2):129–137, 1982.

- [53] P. Kohli, J. Rihan, M. Bray, and P. Torr. Simultaneous segmentation and pose estimation of humans using dynamic graph cuts. *International Journal of Computer Vision*, 79(3):285–298, 2008.
- [54] V. Krayevoy and A. Sheffer. Variational, meaningful shape decomposition. *ACM SIGGRAPH Sketches*, 2006.
- [55] V. Krayevoy, D. Julius, and A. Sheffer. Model composition from interchangeable components. *Pacific Graphics*, 2007.
- [56] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.
- [57] Y.-K. Lai, Q.-Y. Zhou, S.-M. Hu, and R. R. Martin. Feature sensitive mesh segmentation. *ACM Symposium on Solid and Physical Modeling*, 2006.
- [58] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. Maps: multiresolution adaptive parameterization of surfaces. In *ACM SIGGRAPH*. 1998.
- [59] M. Leyton. *A Generative Theory of Shape*. Springer, 1 edition, 2001. ISBN 3540427171.
- [60] J.-M. Lien. Approximate star-shaped decomposition of point set data. *Eurographics Symposium on Point-Based Graphics*, 2007.
- [61] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polyhedra. *ACM Symposium on Solid and Physical Modeling*, 2005.
- [62] R. Liu. *Spectral Mesh Segmentation*. Ph.D. thesis, Simon Fraser University, 2009.
- [63] R. Liu and H. Zhang. Segmentation of 3d meshes through spectral clustering. *IEEE Pacific Graphics*, 2004.

- [64] R. Liu and H. Zhang. Mesh segmentation via spectral embedding and contour analysis. *Eurographics*, 2007.
- [65] R. Liu, H. Zhang, A. Shamir, and D. Cohen-Or. A part-aware surface metric for shape processing. *Eurographics*, 2009.
- [66] S. Liu, R. R. Martin, F. C. Langbein, and P. L. Rosin. Segmenting reliefs on triangle meshes. *ACM Symposium on Solid and Physical Modeling*, 2006.
- [67] S. Liu, R. R. Martin, F. C. Langbein, and P. L. Rosin. Segmenting geometric reliefs from textured background surfaces. *Computer-Aided Design and Applications*, 4(5), 2007.
- [68] L. Lu, Y.-K. Choi, W. Wang, and M.-S. Kim. Variational 3d shape segmentation for bounding volume computation. *Eurographics*, 2007.
- [69] G. Lukács, R. Martin, and D. Marshall. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. *ECCV*, 1998.
- [70] A. P. Mangan and R. T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.
- [71] G. Marola. On the detection of the axes of symmetry of symmetric and almost symmetric planar images. *IEEE PAMI*, 11(1):104–108, 1989.
- [72] D. Marshall, G. Lukacs, and R. Martin. Robust segmentation of primitives from range data in the presence of geometric degeneracy. *IEEE PAMI*, 23(3):304–314, 2001.
- [73] A. Martinet, C. Soler, N. Holzschuch, and F. Sillion. Accurately detecting symmetries of 3d shapes. Technical Report RR-5692, INRIA, September 2005.

- [74] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 35–57. Springer-Verlag, Heidelberg, 2003.
- [75] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 3 edition, 1998. ISBN 3540606769.
- [76] P. Min. Thinvx: 3d voxel model thinning.
- [77] P. Minovic, S. Ishikawa, and K. Kato. Three dimensional symmetry identification, part i: Theory. Technical Report 21, Kyushu Institute of Technology, Japan, 1992.
- [78] N. J. Mitra, L. J. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM SIGGRAPH*, 2006.
- [79] M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Plumber: a method for a multi-scale decomposition of 3d shapes into tubular primitives and bodies. *ACM Symposium on Solid Modeling and Applications*, 2004.
- [80] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, 1957.
- [81] A. Norcia, R. Candy, M. Pettet, V. Vildavski, and C. Tyler. Temporal dynamics of the human response to symmetry. *Journal of Vision*, 2(2):132–139, 3 2002.
- [82] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, 2 edition, 2000. ISBN 0471986356.
- [83] D. O’Mara and R. Owens. Measuring bilateral symmetry in digital images. *IEEE TENCON*, 1996.
- [84] N. R. Pal and S. K. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294, 1993.

- [85] K. Palágyi and A. Kuba. Directional 3d thinning using 8 subiterations. *Lecture Notes in Computer Science*, 1568:325–336, 1999.
- [86] A. P. Pentland. Perceptual organization and the representation of natural form. *Artificial Intelligence*, 28(3):293–331, 1986.
- [87] D. L. Pham, C. Xu, and J. L. Prince. Current methods in medical image segmentation. *Annual Review of Biomedical Engineering*, 2(1):315–337, 2000.
- [88] I. Pitas. *Digital Image Processing Algorithms and Applications*. Wiley-Interscience, 2000. ISBN 0471377392.
- [89] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz, and T. Funkhouser. A planar-reflective symmetry transform for 3d shapes. *ACM SIGGRAPH*, 2006.
- [90] X. Ren and J. Malik. Learning a classification model for segmentation. In *IEEE ICCV*. 2003.
- [91] D. Reniers and A. Telea. Skeleton-based hierarchical shape segmentation. *IEEE Shape Modeling International*, 2007.
- [92] T. Richardson. *The Art Student's Guide to the Proportions of the Human Form*. Tom Richardson, 2008. ISBN 0982167806.
- [93] J. Rodgers, D. Anguelov, H.-C. Pang, and D. Koller. Object pose detection in range scan data. In *IEEE CVPR*. 2006.
- [94] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. *International Conference on 3-D Digital Imaging and Modeling*, 2001.
- [95] H. S. Sawhney, S. Ayer, and M. Gorkani. Model-based 2d&3d dominant motion estimation for mosaicing and video representation. *IEEE ICCV*, 1995.

- [96] S. Shalom, L. Shapira, A. Shamir, and D. Cohen-Or. Part analogies in sets of objects. *Eurographics Workshop on 3D Object Retrieval*, 2008.
- [97] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.
- [98] C. Shen, J. F. O’Brien, and J. R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. *ACM SIGGRAPH*, 2004.
- [99] D. Shen, H. S. Ip, K. T. Cheung, and E. K. Teoh. Symmetry detection by generalized complex moments: A close-form solution. *IEEE PAMI*, 21(5):466–476, 1999.
- [100] P. Simari, V. Kalogerakis, and K. Singh. Folding meshes: Hierarchical mesh segmentation based on planar symmetry. *Eurographics Symposium on Geometry Processing*, 2006.
- [101] P. Simari and K. Singh. Extraction and remeshing of ellipsoidal representations from mesh data. *Graphics Interface*, 2005.
- [102] K. Singh. Interactive curve design using digital french curves. *ACM Symposium on Interactive 3D graphics*, 1999.
- [103] K. Singh and E. Fiume. Wires: a geometric deformation technique. In *ACM SIGGRAPH*. 1998.
- [104] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE PAMI*, 12(2):131–147, 1990.
- [105] C. V. Stewart. Robust parameter estimation in computer vision. *SIAM Rev.*, 41(3):513–537, 1999.
- [106] C. Sun and J. Sherrah. 3d symmetry detection using the extended gaussian image. *IEEE PAMI*, 19(2):164–168, 1997.

- [107] C. Sun and D. Si. Fast reflectional symmetry detection using orientation histograms. *Real-Time Imaging*, 5(1):63–74, 1999.
- [108] M. Thorne, D. Burke, and M. van de Panne. Motion doodles: an interface for sketching character motion. *ACM SIGGRAPH*, 2004.
- [109] S. Thrun and B. Wegbreit. Shape from symmetry. In *IEEE ICCV*. 2005.
- [110] J. Tierny, J.-P. Vandeborre, and M. Daoudi. Topology driven 3d mesh hierarchical segmentation. *IEEE Shape Modeling International*, 2007.
- [111] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.
- [112] G. Turk. and M. Levoy. Zippered polygon meshes from range images. *ACM SIGGRAPH*, 1994.
- [113] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward objective evaluation of image segmentation algorithms. *IEEE PAMI*, 29(6):929–944, 2007.
- [114] J. D. Wolter, T. C. Woo, and R. A. Volz. Optimal algorithms for symmetry detection in two and three dimensions. *Visual Computer*, 1:37–48, 1985.
- [115] H. Yamauchi, S. Lee, Y. Lee, Y. Ohtake, A. Belyaev, and H.-P. Seidel. Feature sensitive mesh segmentation with mean shift. 2005.
- [116] D.-M. Yan, Y. Liu, and W. Wang:. Quadric surface extraction by variational shape approximation. 2006.
- [117] H. Zabrodsky, S. Peleg, and D. Avnir. Symmetry as a continuous feature. *IEEE PAMI*, 17:1154–1166, 1995.
- [118] E. Zuckerberger, A. Tal, and S. Shlafman. Polyhedral surface decomposition with applications. *Computers and Graphics*, 26(5):733–743, 2002.