2008-06-16

Interactive Part Selection for Mesh and Point Models Using Hierarchical Graph-cut Partitioning

Steven W. Brown
Brigham Young University - Provo

INTERACTIVE PART SELECTION FOR MESH AND POINT MODELS USING HIERARCHICAL GRAPH-CUT PARTITIONING

by

Steven W. Brown

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

August 2008

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Steven W. Brown

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Bryan S. Morse, Chair
Date	Dan R. Olsen
Date	Cristophe Giraud-Carrier

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Steven W. Brown in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Bryan S. Morse Chair, Graduate Committee

Accepted for the Department

Date	Kent E. Seamons Graduate Coordinator

Accepted for the College

Date	Thomas W. Sederberg Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

INTERACTIVE PART SELECTION FOR MESH AND POINT MODELS USING HIERARCHICAL GRAPH-CUT PARTITIONING

Steven W. Brown

Department of Computer Science

Master of Science

This thesis presents a method for interactive part selection for mesh and point set surface models that combines scribble-based selection methods with hierarchically accelerated graph-cut segmentation. Using graph-cut segmentation to determine optimal intuitive part boundaries enables easy part selection on complex geometries and allows for a simple, scribble-based interface that focuses on selecting within visible parts instead of precisely defining part boundaries that may be in difficult or occluded regions. Hierarchical acceleration is used to maintain interactive speeds with large models and to determine connectivity when extending the technique to point set models.

ACKNOWLEDGMENTS

My thanks go to those who graciously provided the 3D models: The Stanford Computer Graphics Laboratory for the dragon, bunny, buddha, and lucy models; The Stereolithography Archive at Clemson University for the hand model; XYZ RGB Inc. for the Thai statuette and dragon (called "dragon 2" in this work) model; Georgia Institute of Technology for hosting the blade model originally included in the Visualization Toolkit (VTK); OpenSceneGraph.org for the tire model (a part of the dumptruck model); Cyberware Inc. for the female01 and horse models; Visual Computing Lab of the Institute of Information Science and Technologies (ISTI) for the gargoyle model, found on AIM@SHAPE Shape Repository; The French National Institute for Research in Computer Science and Control (INRIA) and ISTI for the oil pump and elephant models, found on AIM@SHAPE Shape Repository.

I would also like to thank Dr. William Barrett for inspiring me, Dr. Bryan Morse for supporting me, my friends at the lab for distracting and entertaining me, and most of all, my wife Amber for putting up with me.

Contents

List of Figures

List of Tables

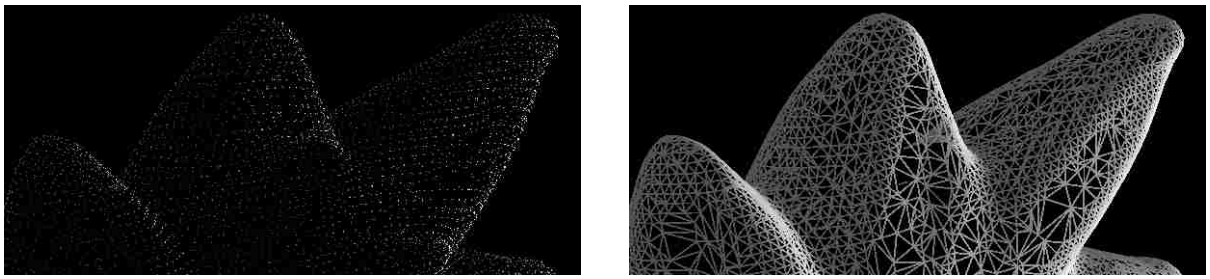
Chapter 1

Introduction

1.1 Problem Description and Motivation

3D computer graphics have become ubiquitous in a wide range of fields. Whether for entertainment in games and movies or for engineering tasks such as simulations and visual design, the need for computer graphics has been quickly growing. At the heart of computer graphics are 3-dimensional models that represent the objects in the rendered scene. As the need for and ability to render more realistic objects increases, the models representing these objects become increasingly large and complex.

Large, complex models today are rarely created by hand due to the intensive effort required. Instead, physical objects are scanned using laser scanners, which record the surface of the object as a dense set of 3D points called a point set (Figure 1.1a). Point sets are often thinned and connected together to form a polygonal mesh (Figure 1.1b). Polygonal meshes are the de facto standard in computer graphics because they can be efficiently rendered. Point sets themselves can also be used as a model type. They require less post-processing since they are not meshed and have the advantage of being smaller in physical (disk) size due to the lack of connectivity information, but are more difficult to render and manipulate.



(a) Point set model (b) Mesh model

Figure 1.1: A close view of a dragon's tail as a point set model (a) and mesh model (b).

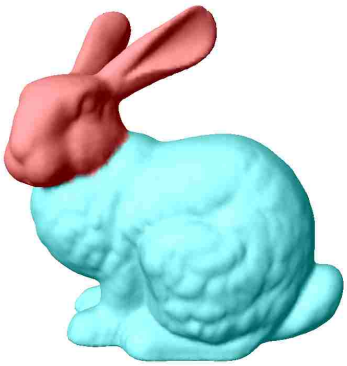


Figure 1.2: An example of part selection. The head is perceived as a part because of the natural boundary between it and the body.

As model sizes increase due to the desire for increased detail, the ability to manipulate these large models with current methods decreases. Manipulations are important to many aspects of computer graphics and include techniques such as deformation, colorization, and segmentation. These manipulations generally require the selection of a subset of the model. On complex models, this can be a difficult due to size, occlusions and the inherent difficulty of manipulating a 3D object using 2D interfaces.

One type of selection that is of particular use is part selection, in which the selection is comprised of a logical part of the model. An example of this would be selecting the head from a bunny model (Figure 1.2), with a logical boundary being the sharp angle between the head and the body. Part selection is challenging because

part boundaries are subjective. While the head may be considered a part in the bunny model in this instance, each ear may be a separate part in another. In other cases, the boundaries between parts are not well defined. Part selection that can be guided by simple user interaction lessens these ambiguities and facilitates manipulations where logical parts are essential, such as creating a new model from parts of existing models.

1.2 Related Work

While much work has been done on automated part selection to assist in tasks such as collision detection, skeletonization, and geometric searches, individual part selection that can be guided by the user has received less attention. Current methods include direct selection of points, selection by geometric primitive, selection by surface scoring to indicate the selection boundary, and scribble-based selection with region growing. These techniques, described in greater detail in Chapter 2, are all effective in certain situations but have difficulty with unsimplified large models and complex geometry.

1.3 Contributions

1.3.1 Graph-Cut Partitioning of Mesh Models

Graph-cut segmentation, a powerful segmentation method that calculates optimal partitioning between marked foreground and background areas, has already been used successfully in the segmentation of 2D images, video, and 3D volumes. We extend this method to 3D mesh models, which pose unique challenges because of their size and difference in connectivity. By combining a scribble-based interface with graph-cut segmentation, we are able to make selections of parts that would be difficult or impossible with other methods such as selections that have multiple boundaries (see

Figure 1.3: Selecting a part with multiple boundaries. The white and yellow areas are scribbles representing desired and undesired areas. Using graph-cut segmentation, five separate selection boundaries are determined.

Figure 1.3), or selections for which the selection boundary is occluded by other parts of the model.

1.3.2 Hierarchical Acceleration

Implementing graph-cut segmentation on large models can cause the selections to be made at less than interactive speed. To provide acceleration, we represent the model as an octree hierarchy composed of the 3D points of the model and perform a series of localized coarse-to-fine cuts. An initial cut is made at a coarse hierarchical level. The areas surrounding the initial cut are changed to represent finer levels of the hierarchy and another cut is made that gives more accuracy in the region of interest. This is continued until the finest level has been reached.

1.3.3 Graph-Cut Partitioning of Point Sets

Our method of hierarchical acceleration provides an additional benefit not found in other techniques. Because the octree hierarchy containing the 3D model points gives rough spatial connectivity, the technique can be directly applied to point sets, which have no inherent connectivity. Each leaf cell in the octree hierarchy, which contains 10 to 15 points, is considered one node in the graph used for graph cut and its neighbors in the graph are determined by the neighboring leaf cells in the hierarchy. The acceleration benefits of the hierarchy remain useful as well, since point sets can often be quite large.

1.4 Thesis Outline

Chapter 2 contains the entirety of a paper as revised after submission to and feedback from ACM SIGGRAPH 2008, with the exception of the References section, which appears at the end of this thesis. Section 2.1 introduces the subject and our approach to solving the part selection problem. In Section 2.2 there is a discussion of related work in the topics of user-guided part selection, scribble-based interfaces, and graph-cut segmentation in 2D and 3D and how our methods fits into the body of that work. Afterwards, in Section 2.3, we describe our interface and a typical user experience.

In Sections 2.4 through 2.6 we detail the methods of our part selection technique. Applying graph-cut segmentation to mesh models is described in Section 2.4. Our process for achieving hierarchical acceleration of mesh models is detailed in Section 2.5, and application of the hierarchical method to point set models is found in Section 2.6. Section 2.7 contains the results for mesh model segmentation (Section 2.7.1), accelerated mesh model segmentation (Section 2.7.2), and point set model segmentation (Section 2.7.3), while Sections 2.8 and 2.9 conclude the paper with a

discussion of our results, comments on the future of this area of study and a conclusion.

Chapter 3 of this thesis contains information not included in the conference submission, either because of space considerations or nonnecessity due to the common knowledge of conference attendees. The additional methods are listed under the section headings to which they apply in the conference submission. In Section 3.1 we describe vertex selection for mesh models, describe in greater detail the reasons for vertex-based graphs over face-based graphs, and explain the physical segmentation of mesh models. We discuss the implementation and searching of the octree hierarchy in Section 3.2. Adjusting vertex selection for point set models and normal interpolation are the subject of Section 3.3.

Chapter 4 contributes additional ideas for future work, such as expanding this technique to other model types, and discusses the overall effectiveness of this technique and how it complements existing selection techniques.

Chapter 2

Research Paper

This chapter contains the research paper in its entirety as revised after submission to and feedback from ACM SIGGRAPH 2008.

Abstract

This paper presents a method for interactive part selection for mesh and point set surface models that combines scribble-based selection methods with hierarchically accelerated graph-cut segmentation. Using graph-cut segmentation to determine optimal intuitive part boundaries enables easy part selection on complex geometries and allows for a simple, scribble-based interface that focuses on selecting within visible parts instead of precisely defining part boundaries that may be in difficult or occluded regions. Hierarchical acceleration is used to maintain interactive speeds on large models and to provide connectivity when extending the technique to point set models.

(a) Original Model (b) Scribbles and resulting selection (c) Side view (d) Front view

Figure 2.1: User session with the Lucy point set model. To select the wings of the original model (a), the user draws a white scribble to select each wing and a yellow scribble to exclude the body (b). Even though the user marked from only from one view, the wings are completely selected (c,d). The entire process for this initial partitioning takes less than 15 seconds even when using this 14 million point model.

2.1 Introduction

As 3D surface models increase in size, they become increasingly difficult to manipulate. One problem is quickly and efficiently selecting parts, or subsections, of large 3D models interactively using a 2D interface. While there are many methods for automatic model partitioning such as [Mangan et al. 1999; Li et al. 2001; Katz and Tal 2003; Shamir 2004; Simari and Singh 2005] that can be applied to tasks such as shape matching, skeleton extraction for deformation and animation, and collision detection, these methods partition the entire model and do not always give the user control over the definition of individual parts. Since part boundaries are ultimately a subjective human decision, this control is essential to accurate part selection.

In this paper we focus on methods for assisting the user in interactively selecting parts, even for very large models. User-guided part selection gives modelers a tool with which they can partition and reassemble models [Funkhouser et al. 2004; Sharf et al. 2006] or simply modify parts of models with color, texture, or deformations.

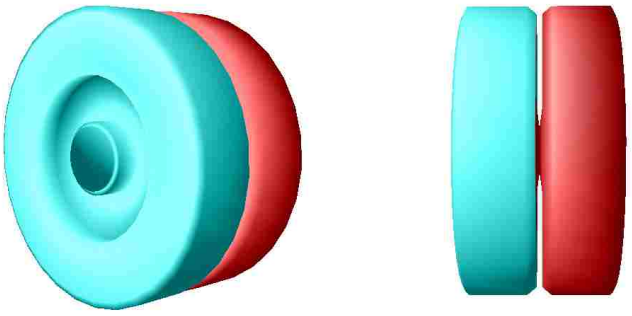
Our goal is to make part selection simple, interactive, and accurate for models of all sizes.

We use a simple tool for interactive part selection for both mesh-based and point set surface models. As shown in Figure 2.1, the user loosely scribbles on parts they want to select and makes other scribbles on the parts they want to exclude. Graph-cut segmentation of the 3D model is then used to automatically calculate the selected part(s). Refining the selection interactively requires only the drawing of additional scribbles.

These rough scribbles are ideal for interaction on large models, which are already difficult to render interactively. Additionally, by drawing on the visible surfaces of the parts themselves rather than tracing or otherwise defining the boundary separating the parts, the user's attention can be focused on those parts and less rotation is required. This makes it easier for a user to select parts even when the boundary separating those parts might be occluded or hard to reach (Figure 2.2) or when the geometry is complex (Figure 2.3). The selection can also be used in conjunction with other selection techniques to give the user further control over the selection.

For an interactive approach to be effective, the tool must maintain rapid feedback in response to the user's input, but this becomes a problem as the models become increasingly large. We introduce a hierarchical approach that allows us to accelerate the partitioning of the model for an interactive response even on large models. The model is initially represented by a high-level octree hierarchy. When a scribble is made the rough model is refined in the areas of interest using lower levels of the hierarchy in a series of coarse-to-fine cuts. This allows areas not affecting the cut to be ignored while maintaining accuracy in the desired areas.

This hierarchical method can also be used directly on point set models. The hierarchy is well-suited to point set models, which are usually very large collections of



(a) Normal view (b) Inaccessible boundary

Figure 2.2: A selection boundary in an occluded area of a double tire (1K vertices). The selection was made with one scribble on each side of the object (a), even though the boundary separating the parts lies inaccessible on the axle connecting them (b).

(a) Exterior partitiong (b) Entire partitioning

Figure 2.3: Selecting the blade from the turbine blade mesh model (882K vertices). The selection and resulting partitioning for the exterior of the mesh model can be seen in (a) and the full partioning, including the occluded inner areas, can be seen in (b).

laser-scanned points, because it can provide some spatial connectivity to the otherwise unassociated points and eases computation. By grouping small numbers of points together into hierarchy leaf nodes and connecting neighboring leaf nodes, a rough connectivity graph can be quickly formed. As with mesh models, the graph is then used to perform coarse-to-fine cuts on the model.

2.2 Related Work

User-guided part selection for 3D surface models can be performed in a number of ways. The simplest approach is to allow the user to select vertices, polygons, or

unorganized points manually, using a brush to mark the surface of the model. While this is very accurate and makes for easy, intuitive selection refinement, it is extremely tedious on large models, is prone to incomplete selections, and fails completely in cases of occlusion where portions of the desired selection are not visible or easily accessible to the user.

Another approach is to use geometric primitives to define a selection volume. The simplest of these is the cutting plane, drawn in screen space or positioned in 3D, which is used to divide the model into two regions. Other geometric primitives include cuboids and spheres [Weyrich et al. 2004], which can be intersected with the model to define selected regions. These selections can be refined by including additional geometric primitives. This method is appropriate for very simple selections, but the primitives can be difficult to place correctly, and they quickly become infeasible for models with complex geometry.

An alternate approach to defining the selected area is to allow the user to define arbitrarily complex part boundaries directly on the surface. This may be done by having the user specify the cut manually [Bruyns and Senger 2001] or by having them place boundary-defining points along the desired cut and connecting those points with least-cost paths [Gregory et al. 1999; Wong et al. 1998; Zöckler et al. 2000]. However, these methods require precise placement of the boundary or boundary-defining points, and drawing or placing points on the boundary often involves rotating the model to see or mark the full boundary. One may also have the user provide only approximate or incomplete strokes, then automatically complete the contour [Funkhouser et al. 2004; Lee et al. 2004; Lee et al. 2005; Sharf et al. 2006]. Such user-drawn boundaries may then be refined using active contours [Lee et al. 2004; Lee et al. 2005], least-cost paths [Funkhouser et al. 2004], or local graph-cut approaches [Sharf et al. 2006]. This is an effective method for many selections, but it can be difficult to refine, can fail in

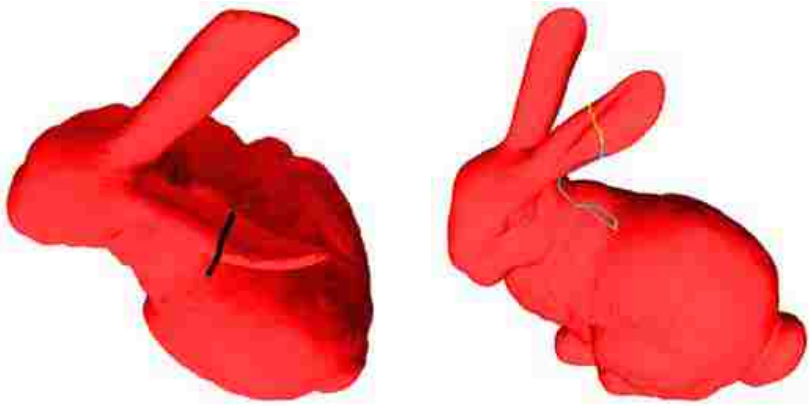
cases of occlusion, and as noted by Funkhouser, et al. [2004] has inherent problems with selections near silhouette edges (Figure 2.4) because users must stroke across desired areas instead of inside them.

It is worth noting the analagous boundary definition methods found in image and video region selection. There are semi-automated methods for assisting the user in selecting object contours in images [Kass et al. 1987; Mortensen and Barrett 1995; Gleicher 1995] and video [Agarwala et al. 2004b], or assisting them in selecting object regions [Maes et al. 1995; Reese and Barrett 2002].

Scribble-based interfaces are also popular for segmenting images or video [Boykov and Jolly 2001; Agarwala et al. 2004a; Li et al. 2004; Wang et al. 2005; Armstrong et al. 2007, among many others] and more recently for 3D meshes [Ji et al. 2006]. Scribbling on a surface requires less fine motor control than boundary tracing, which makes the user training and experience easier. Once initial selections are made, refinement of the selection is also more intuitive, requiring only additional scribbles in incorrectly included or excluded areas.

Many of these assisted selection methods are implemented using graph-cut segmentation, a powerful segmentation method that calculates optimal partitioning between marked foreground and background areas [Boykov and Jolly 2001; Boykov and Funka-Lea 2006], however other approaches have also been successful, such as geodesic [Protiere et al. 2007; Bai and Sapiro 2007] and region growing approaches [Ji et al. 2006]. Graph-cut segmentation can also be used locally to refine selections made by other methods [Sharf et al. 2006].

While graph-cut segmentation can be applied directly to many image and video problems, the computation required prohibits interactive segmentation of larger data sets, such as large 3D volumes and mesh models. Graph-cut segmentation of



(a) User stroke (b) Proposed cut

Figure 2.4: Unintended effects of surface boundary marking near silhouette edges (from [Funkhouser et al. 2004]).

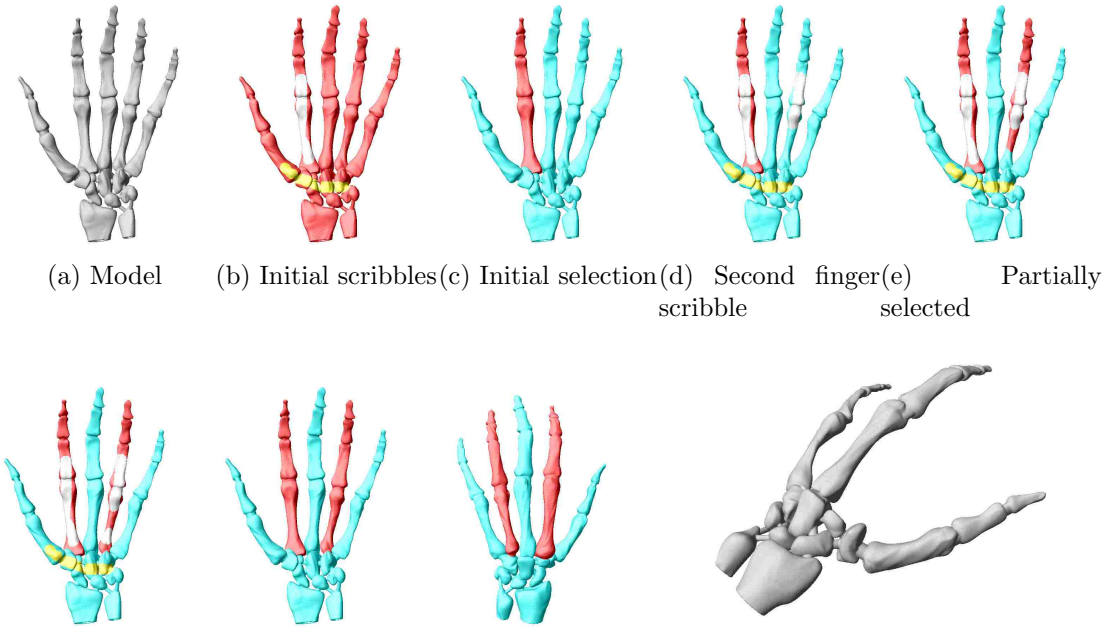
3D volumes has been achieved at interactive rates using a multiband or hierarchical approach [Lombaert et al. 2005; Armstrong et al. 2007] to accelerate the computation.

These assisted selection methods have been designed for meshes or volumes that have inherent connections between data points. Point set models must be meshed and possibly simplified before selection can begin using these methods.

Our approach combines the scribble-based selection techniques appearing in 3D interfaces with hierarchically accelerated graph-cut segmentation to do part selection for large mesh and point set models. The scribble-based interface gives the user simple, intuitive interaction, the graph-cut segmentation allows for selections on complex models and the hierarchical acceleration makes graph-cut segmentation possible for large mesh and point set models.

2.3 Interaction and System Overview

Figure 2.5 demonstrates a typical user session, including selecting disjoint parts and refining the selections. Starting with the initial model (2.5a), the user marks (2.5b)



(a) Model (b) Initial scribbles (c) Initial selection (d) Second finger scribble (e) Partially selected

(f) Refining scribble (g) Resulting selection (h) Back view (i) Model with selected parts removed

Figure 2.5: Screenshots from a typical user session using the hand bone mesh model (327K vertices). (a) The original model. (b) The initial scribbles: the white scribble is to be included, the yellow scribble is to be excluded. (c) The resulting selection, where red indicates the selected areas. (d) An additional inclusion scribble on the ring finger. (e) The resulting selection. (f) A refining scribble to complete the selection of the ring finger. (g) The resulting selection. (h) The obverse view. (i) Result of partitioning and deleting the selected fingers. The entire process takes about 9 seconds.

one or more areas as included (the white vertical scribble along the index finger) and other areas as excluded (the yellow horizontal scribble along the thumb and palm) to obtain an initial selection (2.5c). Additional scribbles are then used to expand (2.5d,e) and refine (2.5f) the selection. The resulting selection is made along natural boundaries on the model (2.5g,h). From here, the selection can then be used for a number of purposes, including editing (2.5i).

The resulting selection boundaries are calculated from rough initial scribbles using graph-cut methods as described further in Section 2.4. Smaller mesh models can be formulated as graphs with connectivity defined by polygon edges and edge weights determined by differences in vertex normal angles. For larger mesh models, we use an octree hierarchy to accelerate the interaction by performing a series of cuts from a coarse to fine level as described in Section 2.5. Edge weights are determined by the difference of the normals of averaged vertex normals at different levels in the hierarchy. The hierarchical method is also applied directly to point set models for part selection as described in Section 2.6 using interpolated surface normals.

2.4 Mesh Model Partitioning with Graph Cut

The heart of our part-selection technique is minimum graph-cut segmentation, in which a weighted graph is partitioned along edges of minimum cost. We use a version of the algorithm described in [Boykov and Kolmogorov 2001].

Polygonal mesh models naturally lend themselves well to graph-cut segmentation because their inherent connectivity can be easily formulated as a graph, with the mesh vertices as the graph vertices and the mesh edges as the corresponding graph edges. Vertices are selected as part of the user's scribbles by projecting the mouse

point onto the surface of the mesh and including all vertices within a certain distance from that point.

One could also formulate the graph based on mesh faces and their connectivity rather than mesh vertices, which would produce clean cuts along polygon edges. However, calculating adjacency between faces is either computationally slower or more of a strain on memory when given mesh data defining unorganized faces only by their vertices. This becomes especially troublesome for very large models. Basing the graph on vertices provides adjacency in linear time with no added memory costs and has the added advantage of mirroring common mesh data types, where surface normals and other information are commonly associated with vertices. It is also more easily extensible to point set representations, as discussed in Section 2.6. The drawback to the point-as-graph-node technique is that cuts do not occur on natural polygon edges. This leaves a single-polygon wide strip separating the included and excluded areas of selection. For our purposes we have chosen to group these polygons with the excluded region.

Using the notation of [Boykov and Jolly 2001], given the set of vertices $\mathcal{P}$, the unordered set of vertex pairs $\mathcal{N}$ representing mesh edges, and the binary partitioning vector $A = (A_1, \ldots, A_p, \ldots, A_{|\mathcal{P}|})$ where each element A_p represents the inclusion or exclusion of each vertex p in $\mathcal{P}$ from the selection, the cost function $E(A)$ for a particular partitioning A can be defined as

$$E(A) = \lambda\ R(A) + B(A) \tag{2.1}$$

where $R(A)$ denotes the penalty cost for incorrectly labeling a vertex specifically marked by the user as included or excluded from the selection ("regional" properties), $B(A)$ denotes the sum of the costs of each edge along the partition boundary in $\mathcal{N}$ ("boundary" properties), and λ represents the relative importance of these two terms.

The selection is calculated by finding the minimum cost partitioning as defined by Equation 2.1 over all possibilites of A.

The region-labeling term $R(A)$ is defined as

$$R(A) = \sum_{p \in \mathcal{P}} R_p(A_p) \tag{2.2}$$

where (R_p) is based on the inclusion or exclusion of each point p in a given partitioning A:

$$R_p(A_p) = \begin{cases} 1 & \text{if } A_p \text{ conflicts with the user's input} \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

The boundary term $B(A)$ is defined as

$$B(A) = \sum_{\{p,q\} \in \mathcal{N}} B_{\{p,q\}}\ \delta(A_p, A_q) \tag{2.4}$$

and

$$\delta(A_p, A_q) = \begin{cases} 1 & \text{if } A_p \neq A_q \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where $B_{\{p,q\}}$ is the cost assumed by each mesh edge between any points p and q not in the same region (i.e., across the cut).

In order for graph-cut segmentation to be effective for part selection, the weights of the edges in $\mathcal{N}$ must be formulated to minimize $B_{\{p,q\}}$ where natural boundaries exist between parts. We do this by calculating weights according to the minima rule [Hoffman and Richards 1983; Lee et al. 2004; Lee et al. 2005], which states that the human vision system perceives part boundaries along concave creases, or negative minima of principle curvature. To achieve this effect, we assign edge weights based on the relative difference of the surface normals of adjacent vertices. Vertex normals are sometimes provided with the model but can also be easily inter-

polated from the mesh data. Since we are interested only in the relative difference between normals and not the actual angle, only the dot product of the normals is needed. However, because normal estimation can sometimes give inverted normals (see Section 2.6) and because it can be reasonably assumed that a surface will not have adjacent normals at more than 90° angles, the absolute value of the dot product can be used. The final edge weighting function $B_{\{p,q\}}$ between two vertices p and q with unit normals N_p and N_q becomes

$$B_{\{p,q\}} = |N_p \cdot N_q| \tag{2.6}$$

where N_p and N_q are of unit length and $0 \leq B_{\{p,q\}} \leq 1$. Since we want to minimize costs in areas of concavity or sharp angles, values of $B_{\{p,q\}}$ approaching 1 (0° difference) indicate a flat and unlikely cut location, while values approaching 0 (90° difference) indicate a sharper angle and more likely cut location.

Since $B(A)$ remains fixed for a static model and given partitioning A, modification of $R(A)$ through specific inclusion and exclusion of vertices provides the user-guided selection. Vertices may be marked as included or excluded from the selection, but not both. (Should the user re-mark an area, we use the most recent labeling.) With large values for λ, the user is effectively able to select directly along borders where graph-cut is not performing well or can't be expected to perform well. Since the relative cost of $R(A)$ will outweigh $B(A)$, the optimal cut will be between marked included and excluded regions.

2.5 Hierarchical Acceleration

While the use of graph-cut segmentation directly works well for smaller models, it does not scale well. Because of the complexity of graph-cut segmentation, hierarchical

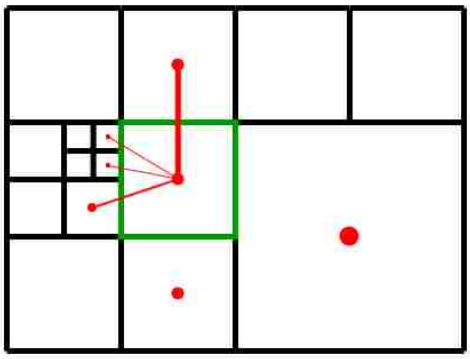


Figure 2.6: A two-dimensional example of the octree connectivity algorithm. The leaf being considered is marked in green. The coarser leaf to the right is ignored since an edge to it will be created when that leaf is considered, the finer leaves to the left are connected with $\frac{1}{2}$ and $\frac{1}{4}$ weight ($\frac{1}{4}$ and $\frac{1}{8}$ for 3D), the leaf above at the same octree level is connected with regular weight, and the leaf below is ignored because it is in a negative cardinal direction.

acceleration is needed in order to achieve interactive speeds using larger mesh models. We use a self-building octree hierarchy to represent the vertices of the model, with leaf nodes generally set to a maximum of 10 vertices per leaf cell. The vertices in each leaf combine to form a representative superpoint with a surface normal that is the average of the vertices' normals. Parent cells are represented by a weighted average of child cell normals. An alternate method that approximates a surface normal over all points in the cell can also be used, as described in Section 2.6.

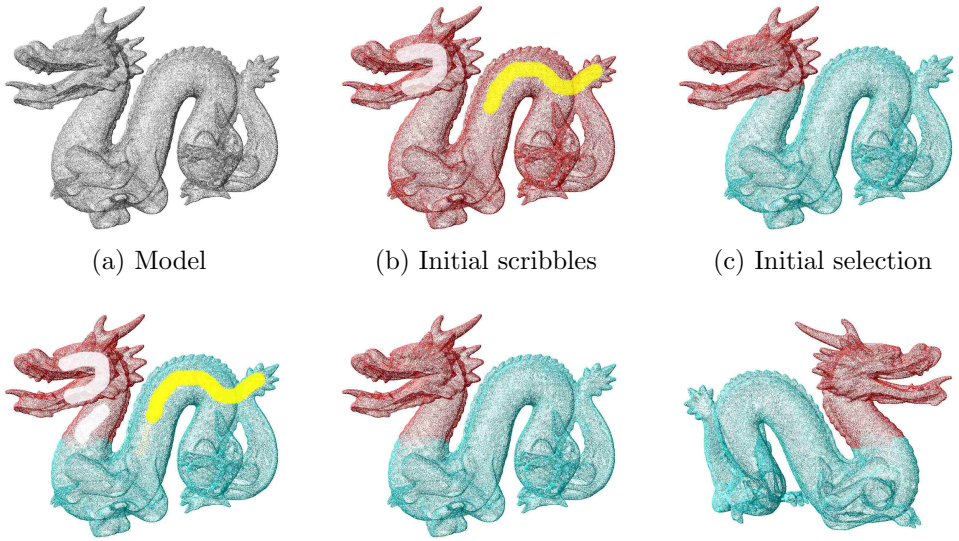
The physical adjacency of octree cells is used as graph adjacency. The edge weights are calculated as described in Section 2.4, using the superpoint normals. The connectivity at the leaf level is determined by this algorithm:

```
for each leaf
  for each of the 6 cardinal directions
    if the neighbor is a coarser leaf
      ignore
    if the neighbor is made of finer leaves
      make fractionally weighted edges
```

```
if the neighbor is a leaf (same level)
  make edge in positive direction
```

This algorithm is illustrated in 2D in Figure 2.6. Adjacent leaves at the same hierarchical level are connected only if the neighbor is in a positive cardinal direction, ensuring there is only one edge between each adjacent pair. A leaf is also connected to neighboring leaves at a finer hierarchical level, but not coarser, again ensuring only one edge between neighboring leaves at different levels. Because the connectivity is limited to the leaf level, this hierachical technique is applicable to point sets as well, which will be discussed in Section 2.6. While leaf adjency could be stored during the creation of the octree hierarchy, for memory reasons we chose to find adjacent leaf nodes as needed during execution by traversing the hierarchy.

For acceleration, parent cells at a coarser hierarchical level are treated as leaf nodes. The graph is formed with the connections and weights at this level and the segmentation boundary found. The cells bordering the segmentation boundary are unmarked as leaf cells and the graph is recreated to include the border cells' children on the next level [Armstrong et al. 2007]. This process is repeated until the finest level of the hiearchy has been reached. Because hierarchical techniques are subject to overcommitting early based on inaccurate aggregate data, we found that starting somewhere between 4 and 6 levels down the tree provides accurate surface normal information to perform the cut without unreasonable performance strains from initial graph size. All models demonstrated here, including the 14 million point Lucy model, had 12 or fewer hierarchy levels.



(a) Model (b) Initial scribbles (c) Initial selection

(d) Refining scribble (e) Resulting selection (f) Another view

Figure 2.7: User session with a dragon point set model. (a) The original model. (b) The initial scribbles; the white scribble is to be included, the yellow scribble is to be excluded. (c) The resulting selection. (d) An additional scribble to include the neck. (e) The resulting selection. (f) View of the back.

2.6 Point Set Models

The methods presented here may also be applied to purely point-based representations (e.g., [Alexa et al. 2001; Alexa et al. 2003; Pauly et al. 2003; Zwicker et al. 2002]) without having to explicitly reconstruct the surface mesh [Hoppe et al. 1992; Curless and Levoy 1996; Amenta et al. 1998; Amenta et al. 2001]. Extending our technique to point set models (Figure 2.7) is fairly straightforward but requires changes to some of the key elements to get similar results.

Since there is no longer a surface from which to find an intersection point, vertex selection during scribbling is accomplished by casting a ray through the object and finding all vertices within a small threshold distance from that ray. From these vertices, the vertex closest to the near cutting plane is selected as the surface intersection point and all points within a threshold determined by the brush size are included

in the selection. By taking advantage of the hierarchy, this can be accomplished in interactive time.

The hierarchical acceleration provides an inherent approximate connectivity, precluding the need for direct meshing. While individual points are not considered, due to the size of most point set models the differences between these vertices is often very small, making the superpoints more receptive to changes in angle. They also retain the acceleration benefits of the hierarchy, making interactive speeds possible on very large models (Section 2.7.3).

While the hierarchy provides connectivity for the graph, the surface normals are still lacking for edge weight calculation. We use the method described in [Gopi et al. 2000] to compute surface normals for each leaf cell, or superpoint. A single point p is found for each leaf cell. It is the point closest to the centroid of all points in the leaf cell. The remaining k points in the leaf cell become the neighbors q_1 to q_k. The vector $\vec{n}_p$ that estimates a normal for p is the vector that minimizes the variance of the dot product between itself and the vectors from p to the neighboring points. If we define these vectors as $\vec{V}_i = q_i - p$, where $1 \leq i \leq k$, then we need to find $\vec{n}_p$ such that it minimizes

$$\sum_{i=1}^{k}(\vec{n}_p \cdot \vec{V}_i - \frac{\sum_{i=1}^{k} \vec{n}_p \cdot \vec{V}_i}{k})^2, \text{ or} \tag{2.7}$$

$$\sum_{i=1}^{k}((\vec{V}_i - \frac{\sum_{i=1}^{k} \vec{V}_i}{k}) \cdot \vec{n}_p)^2 \tag{2.8}$$

If p is at the origin, the centroid of the k neighbors can be defined as $C = \frac{\sum_{i=1}^{k} \vec{V}_i}{k}$. We can then rewrite the equation as

$$\min(\sum_{i=1}^{k}((\vec{V}_i - C) \cdot \vec{n}_p)^2) \tag{2.9}$$

In this form, we can see that if we define a $k \times 3$ matrix A with row vectors defined as $V_i - C$, Equation 2.9 can be rewritten as

$$\min(\|A\vec{n}_p\|_2) \tag{2.10}$$

This minimization problem can then be solved by treating it as a singular value decomposition problem. The vector that corresponds to the smallest singular value of A is the normal vector that minimizes Equation 2.10. Additionally, the smallest singular value can function as a measure of fit (see Section 2.8).

Surface normals for the parent cells are calculated using the same method, considering the children cell superpoints as neighboring vertices of the centroid vertex of the parent. The resulting normals are not necessarily aligned inwardly or outwardly and can't be easily aligned because of the lack of a true surface representation. However, inverted normals have no effect on our edge weighting function (Equation 2.6), therefore for performance this step was ignored. Our interpolated normals were used for all selection calculations, though we imported normals from corresponding mesh models for display purposes only.

2.7 Results

This section demonstrates the results of our technique on various model types and sizes. All computation was done on an Intel Xeon 5160 (3 GHz) with 3GB of RAM and a NVIDIA Quadro FX 4500 (512 MB) graphics card.

2.7.1 Mesh Model Results

Figure 2.5 (shown earlier on page 14) shows a typical session in which the user wishes to select the first and fourth fingers of a model of the bones of the hand. Note

File	Vertices	Part	Select	Hier.	Sel.
Bunny	35,947	Head	9	17	(<1)
Female01	184,196	Head & Hair	28	40	(<1)
Oil Pump	542,199	Cylinder	72	61	(1)
Gargoyle	863,210	Wings	45	20	(1)
Blade	882,954	Blade	44	10	(1)
Elephant	1,537,283	Ears	88	32	(2)
Dragon 2	3,609,600	Front Leg	N/A	20	(3)

Table 2.1: Selection times in seconds for mesh models with and without the hierarchy[1]. Regular selection times include user time. Hierarchical selections include hierarchy preprocessing (shown in parentheses).

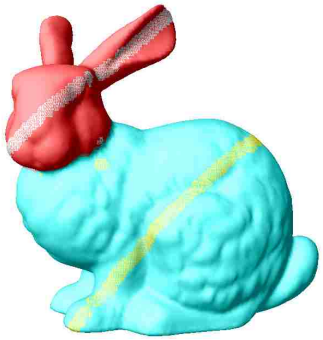
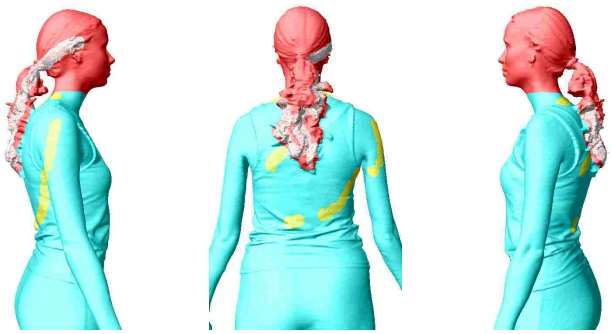


Figure 2.8: Selecting the head from the Stanford bunny (35K vertices). This selection was obtained in 9 seconds without hierarchical acceleration.

that the scribbles are only rough indications of desired and undesired areas, yet the selection border lies along natural concave borders. This selection took approximately 9 seconds of user time and required no rotation of the model. Selection times for other mesh models of varying sizes can be found in Table 2.1, and examples can be found in Figures 2.8 and 2.9.

The advantages of a scribble-based approach can best be seen with parts that have complex geometry, such as the center wrist bone from the hand model (Figure 2.10). In approximately 7 seconds, the user was able to select the desired part with three scribbles. Note that because this model is contiguous, the bone is attached

[1]These models, while not all shown here, were included to quantify the performance of the proposed method on models of varying types and sizes.



(a) (b) (c)

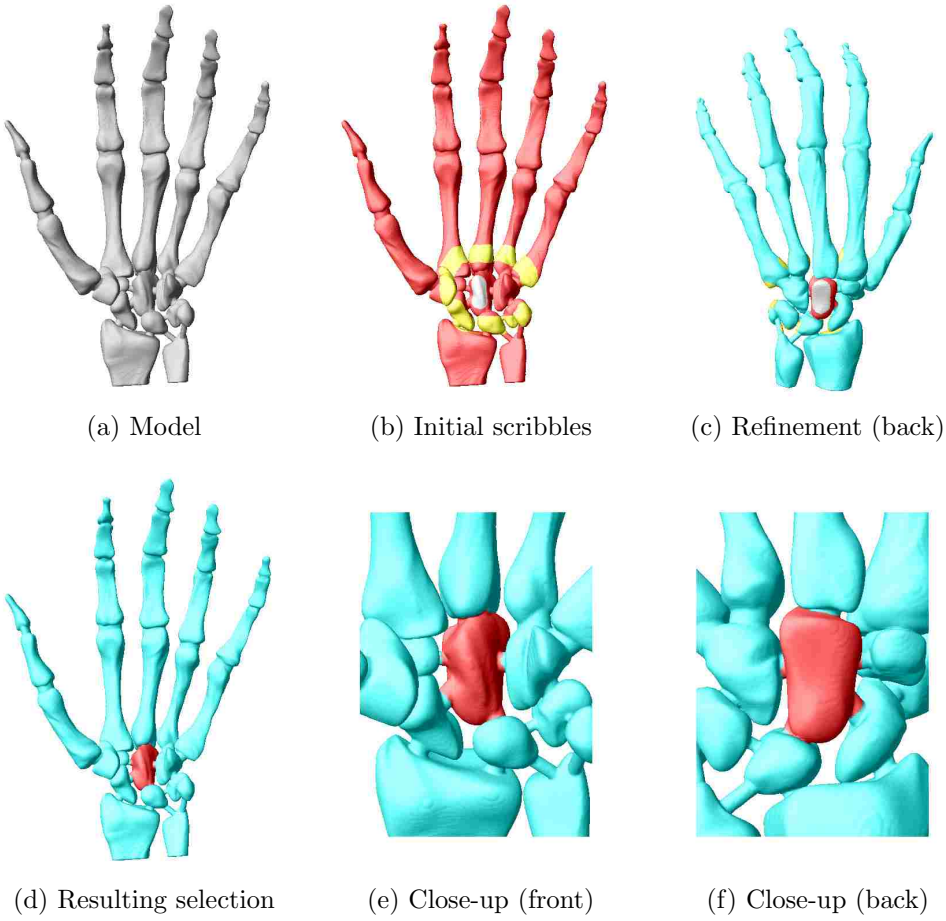
Figure 2.9: Selecting the head and hair from the female01 model (184K vertices). This selection was obtained in 28 seconds without hierarchical acceleration.

in five separate locations, making selection of this part using a single cutting plane impossible and selection by boundary tracing or scissoring laborious.

Preprocessing time for mesh models without hierarchical acceleration is negligible. None of the mesh models tested, including the Stanford Thai statuette with 5 million vertices and 10 million triangles, took more than 1 second to create the cost-weighted graph after loading the model.

The selection calculation time varies depending on several factors. In general, initial cuts are slower than refining cuts because they have no previous information to work from, but initial selection times can be improved by placing more initial seeds. Calculating selections predictably slows as the size of the model, and thus the corresponding graph, increases. Selection time can also vary greatly depending on the complexity of the model and the interaction required. The majority of the overall selection time of the larger examples in Table 2.1 was spent on the initial cut.

Figure 2.11 shows the result of less than two minutes of user time to segment the hand model into individual bones. In comparison, the semi-assisted segmentation done by [Funkhouser et al. 2004] took 13 minutes of user time, while the automatic segmentation done by [Katz and Tal 2003] took 28 minutes of computer time.



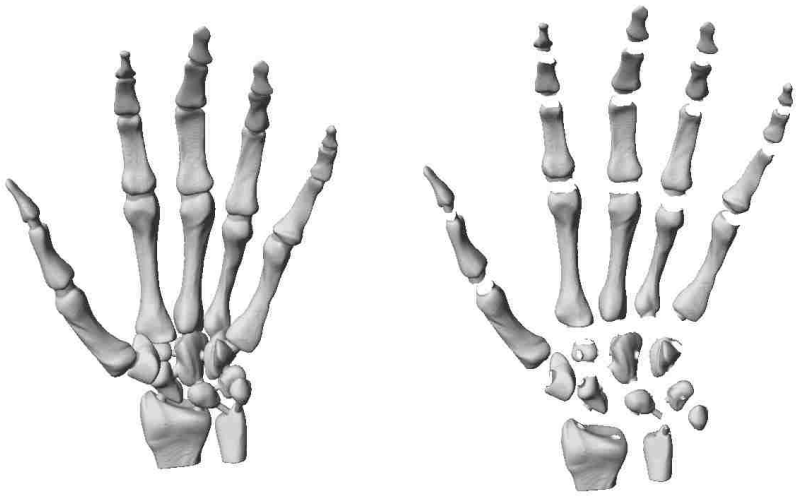
(a) Model (b) Initial scribbles (c) Refinement (back)

(d) Resulting selection (e) Close-up (front) (f) Close-up (back)

Figure 2.10: Selecting the center wrist bone of the hand mesh model (327K vertices). Note that the selection requires five separate selection boundaries. (a) The original model. (b) The initial scribbles; the white scribble is to be included, the yellow scribble is to be excluded. (c) An additional scribble on the back of the bone to complete the selection. (d) The resulting selection. (e) Close-up of the resulting selection (front). (f) Close-up of the resulting selection (back). This selection took about 7 seconds.

2.7.2 Hierarchical Acceleration Results

An example of partitioning using hierarchical acceleration can be seen in Figure 2.12, in which a model with more than a half-million vertices was interactively segmented in approximately one minute. Another example can be seen for the larger gargoyle model in Figure 2.13. The hierarchical method cuts the processing time in half with nearly identical results.



(a) Model (b) Interactive Partitioning

Figure 2.11: A mesh model of hand bones interactively segmented in under 2 minutes.

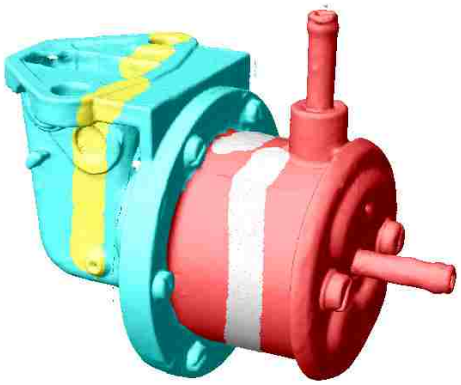


Figure 2.12: Selecting a cylinder from an oil pump model (542K vertices). This selection was obtained in 61 seconds with hierarchical acceleration.

A coarse-to-fine selection using an octree hierarchy greatly reduced the computation time required to partition larger models, but due to the characteristic of all hierarchical approaches to sometimes overcommit with insufficient data, more user interaction is sometimes required. Additional time is also required to preprocess the hierarchy, but it is very short because the representative surface normals for the hierarchy levels are quickly calculated from given normals or from easily interpolated normals. In general, mesh models with fewer than 1,000,000 vertices require 1 second or less additional time to create the hierarchy and normals. For larger mesh models

(a) Without hierarchical acceleration

(b) With hierarchical acceleration

Figure 2.13: Selecting the wings from a gargoyle (863K vertices). The regular mesh selection (a) took 45 seconds. The hierarchical selection (b) took 20 seconds, including preprocessing, for an almost identical result.

the overall preprocessing time is still only a matter of seconds, with file read time being a significant portion.

For larger mesh models, the increase in preprocessing time and user interaction is more than made up for by the speedup in selection times. As shown in Table 2.1, hierarchical selections are faster in overall user time for large models despite the increase in user interaction. The continuous feedback provided by quicker interaction also causes less down time, providing a more responsive user experience. We found the point at which hierarchical acceleration became faster overall than selecting without acceleration to be around 500,000 vertices, although this will vary depending on the model, user preferences, and the computational resources used.

File	Vertices	Read File	Create Hier.	Interp. Norms	Total
Bunny	35,947	<1	<1	<1	<1
Horse	48,485	<1	<1	<1	<1
Hand	327,323	<1	<1	1	1
Dragon	437,645	<1	1	1	2
Buddha	543,652	<1	1	1	2
Blade	882,954	<1	1	2	3
Elephant	1,537,283	<1	1	4	5
Dragon 2	3,609,600	<1	3	8	11
Statuette	4,999,996	<1	4	13	17
Lucy	14,027,872	1	6	20	27*

*Leaf cell max at 25 instead of 10

Table 2.2: Preprocessing times in seconds for point set models. Reading the file includes creating all necessary data structures for display. Creating the hierarchy is the organizing of vertices only. Interpolating normals is creating the representative normals for the hierarchy cells. Total time includes all loading time, including display setup and creating the graph.

2.7.3 Point Set Model Results

Applying the hierarchical acceleration method to point set data gives rough selections similar to those for accelerated mesh models but increases the preprocessing time (Table 2.2) compared to mesh models because of the necessity of interpolating surface normals. Due to the size of some point set models, however, and depending on the intent of the user, this may be preferable to first meshing the model.

Simple selections on point set models remain virtually unchanged, as can be seen in Figure 2.7 in which the user wishes to select the head of the dragon. With a few rough scribbles, the user is able to select the head in under 5 seconds. Refining strokes allow the user to extend the selection to include more of the neck.

The advantages are especially apparent with large models, such as the Lucy point set model with 14 million vertices. Figure 2.1 shows screenshots from a point set selection of the wings of the Lucy model. Within 15 seconds, an initial rough selection has been made. Because of the size of the model, the maximum number of

vertices per octree leaf cell had to be increased to 25 to fit in memory. This was the only model that required this.

2.8 Discussion and Future Work

Our technique excels at giving very quick general selections, but as with any automatic or semi-automatic parts-selection method it cannot always find subjective part boundaries. In some cases of detailed refinement it can be more effective to directly select the boundary. Our tool would ideally be used in combination with other tools such as direct selection and boundary-based methods to give the user other forms of control over the selection, as is often done with other media.

Hierarchical acceleration proved invaluable to point set models for speed and connectivity, and it is essential for interactive selection on large mesh models. Since model size and computation vary greatly, we believe combining the hierarchical and non-hierarchical methods could be an effective solution. One option is to make the initial selection with the hierarchy, then calculate refinements of the initial selection, which are generally much faster, without the hierarchy. Another is to continue the coarse-to-fine cuts past the leaf level and incorporate the connectivity of the mesh in the final cut. Currently the maximum number of vertices per leaf node, the starting level of the first cut, and whether to use the hierarchy are set manually. Automatic techniques for determining these values based on model size and available computation could make the tool more robust for differently-sized models.

Point set selections can be made rapidly using the hierarchy, but because of the need to interpolate surface normals, the preprocessing time was significant. This trading of preprocessing time for increased responsiveness during interaction is reasonable, and even for a model with 14 million points took only 27 seconds.

Using a faster normal interpolation method could reduce this time. Alternately, the interpolation method we use [Gopi et al. 2000] solves for the normal as the minimum of a local fitting function, the value of which can be used as a goodness-of-fit measure to apply the hierarchy normals only at cells where they accurately represent the surface of the contained vertices. This could reduce the overcommitting due to inaccurate normals on coarse levels as seen on some models.

Basing the weighting function on the difference in surface normals between two points provided pleasing results. However, as with all graph-cut methods, we observed a bias towards cuts with fewer, longer mesh edges than cuts of equal length with more, shorter edges. Edge weighting being equal, denser areas will have higher costs than sparse areas. Incorporating vertex separation (mesh edge length) into the weighting along with any available vertex information, such as color, could improve selections and reduce interaction.

2.9 Conclusion

We have presented an interactive technique that combines a simple, scribble-based interface with hierarchically-accelerated graph-cut segmentation to perform part selection on 3D mesh and point set models. Our technique gives good results on both mesh and point set models, and works at interactive speeds even for very large models. The drawing of scribbles on visible surfaces reduces interaction on the model and allows the user to select parts in complex models where the boundary may be occluded or difficult to reach. While the user focuses on selecting within parts, the graph-cut segmentation determines optimal boundary placement, using the minima rule to select visibly intuitive part boundaries. The hierarchical acceleration allows this technique to be applied to large, unconnected point set models.

Chapter 3

Additional Methods

This chapter contains additional details of our methods that we were unable to include in the research paper (Chapter 2) either due to paper length restrictions or lack of need due to shared knowledge of conference attendees. These details are arranged in the same order and under the same headings as those in the research paper.

3.1 Mesh Model Partitioning

3.1.1 Vertex Selection

Vertex selection during scribbling for mesh models is done through OpenGL's UnProject function. The function reverses the series of transformation matrix calculations used to project the object onto the screen in order to calculate the object space 3D position of the screen coordinate indicated by the mouse. This 3D object space coordinate is calculated for the center and cardinal radii of the selection brush circle. All points within the distance defined by the smallest radius are included in the scribble. For display purposes, only triangles that have all three vertices included in the scribble selection are rendered in the scribble color.

3.1.2 Graph Creation

Mesh data is normally stored by defining triangle faces. Some data types store a list of vertices, each defined as three floating-point numbers, along with triangle faces, each defined by three indices into the vertex list. This allows a vertex to be used multiple times, which it invariably is in a triangle mesh, with the added cost of only 1 index value (typically a 4-byte integer) instead of redefining the vertex (typically three 8-byte floats). Other data types forgo the vertex list, repeating vertex values to define each triangle face. In each of these cases, determining face adjacency, that is, which triangles share two common vertices, requires an exhaustive search of all triangle definitions to find a match for each triangle edge. While some speed-ups are possible, such as eliminating edges already found from further searches, the complexity of calculating adjacency still prohibits processing times of under a minute to all but the smallest models. Another speed-up is to store a mapping for each edge, which would require only two passes through the data, but this becomes prohibitively memory-intensive for large models.

A third data type defines triangles in strips. Each strip consists of an initial triangle defined by three vertices and additional triangles defined by an additional vertex combined with the previous two vertices. While this method does give some adjacency, it is partial adjacency at best, with two edges defined for each inner triangle and 1 for triangles at each end of the strip. Completing the connectivity still requires expensive computation, and since this data type is less common, an alternate solution would still have to be made to process models based on the more common data types.

Determining the adjacency of vertices, however, is relatively simple. Since the data types are fundamentally lists of triangles as defined by three vertices, the adjacency between vertices can be derived in linear time with no extra memory costs

by pairing each possible combination of two vertices from the list of three for each triangle.

3.1.3 Model Segmentation

To show the utility of our part selection technique, we implemented a segmentation tool. Because the data are stored as triangles defined by indices into a vertex array, segmenting a mesh model requires the creation of new vertex arrays for each segment and a redefinition of each triangle to reflect this new array. To do this, a mapping is created for each segment to map each vertex in that segment from the original vertex array to a new vertex array. Each segment's new triangle definitions use this mapping to correctly index into its new vertex array. Once the reassignment is complete, each segment is finalized as a new object. The file format we created allows multiple objects to be saved in a single scene.

3.2 Hierarchical Acceleration

3.2.1 Building the Hierarchy

The octree hierarchy used for acceleration is self-building. An initial octree cell is created as defined by the maximum and minimum coordinates over all vertices. As vertices are added to the octree, the initial cell is divided into eight child cells when the number of vertices reaches the limit for vertices in a leaf node. The vertices contained in the initial cell are distributed to the child cells, which are marked as leaf nodes, and the initial cell becomes a parent cell. This process continues as added vertices trickle down the hierarchy and child cells that exceed the limit divide and become parent cells.

3.2.2 Searching the Hierarchy

The octree hierarchy is used to determine physical adjacency of groups of points as defined by the leaf cells. Because some cells within the octree are broken down into finer cells while others are not, there are three possible outcomes when searching for physical neighbors of a given cell face. The neighbor could be a leaf cell of the same hierarchic level, a leaf cell from a coarser hierarchic level, or a parent cell containing children leaf cells at finer hierarchic levels. The basic algorithm for determining which connections to make is outlined in Section 2.5 and illustrated in Figure 2.6.

To implement the algorithm described in Section 2.5, each octree cell stores its own relative position in relation to the parent cell as an index 0 through 7. When finding a neighbor leaf cell, the index is referenced. If the direction of the desired neighbor indicates the neighbor is within the same parent cell, finding the neighbor is trivial. If this neighbor is a parent cell, all child leaf cells facing the original cell are included as neighbors. If the direction of the desired neighbor indicates the neighbor is outside the parent cell, the hierarchy is traversed upwards until a cell is found that has a known neighbor in that direction, i.e., they are within the same parent cell (or the root is reached indicating this cell is on an outside edge). The hierarchy is then traversed downwards through cells that face the direction of the original cell. A coarse neighbor can be found this way, as can a neighbor of equal hierarchical level or multiple neighbors of finer hierarchical levels.

3.2.3 Implementing Graph Cut

When performing the coarse-to-fine cuts necessary for acceleration, it is essential to know along which edges the cut lies in order to refine the graph along those edges. Since the graph-cut package used had no mechanism for querying edges, or node pairs, it was necessary to keep a separate data structure of edges consisting of pointers to

graph node objects. After the cut, each edge can then easily be evaluated with the included foreground/background test by testing each node in the pair to see if the edge crossed from a foreground to a background node.

3.3 Point Set Models

3.3.1 Vertex Selection

Due to the lack of connectivity in point set models, the point selection method for the scribble-based interface used for mesh models could not be used. Points are selected by casting a ray through the point set and finding all vertices within a small distance from that ray. As before, the initial ray can be found easily in OpenGL by using the UnProject function to find the object coordinates of the mouse point on the near and far cutting planes. It can then be used to translate the brush radius in screen pixels to an object space radius on the near and far cutting planes. Given the ray, we take advantage of the hierarchy to limit the search to leaf cells intersected by the ray. From the points contained in those leaf cells, the point closest to the viewing plane is interpreted as the surface. The distance from this point to the points earlier determined on the near and far cutting planes is determined and used to interpolate the brush size, or selection threshold, at that point. All points within the threshold distance from this point are included in the scribble.

3.3.2 Surface Normal Approximation

To implement the singular value decomposition we used the JAMA package.

Chapter 4

Conclusion

While the problem of part selection of 3D models is not completely solved, we feel our technique has made a significant step in that direction. By combining the interaction and segmentation techniques used successfully in other media, we were able to provide an alternate method for part selection that is responsive enough for easy interaction on all models, including very large models.

In addition to the future work discussed in Section 2.8, we also feel that our technique has great potential for other 3D model types that use sparse surface-point representations, such as certain implicit representations. While there would be differences in finding connectivity, forming the graph, and weighting the graph, as there were between mesh and point set models, the underlying construction remains compatible.

On mesh and point set models, our technique works very well. The scribble-based interface allows users to focus on the parts they want to keep instead of meticulously defining borders. This lack of necessary precision increases user interaction speeds and makes selections that require borders in difficult or occluded areas possible. By combining the interface with graph-cut segmentation, it also simplifies and makes possible selections of parts that have multiple connections to the remainder of

the model by requiring the user to focus only on the desired part, instead of defining multiple selection boundaries.

The hierarchical acceleration was successful in maintaining interactive speeds on large mesh models, something not possible with other methods. It allows selections that might otherwise have taken minutes to be done in seconds. While work can be done to improve the performance and automation of this technique (see Section 2.8), its effectiveness even in this initial work shows it to be a success.

Perhaps the most exciting of all our results was the quickness in which we could select parts from large point set models. With other methods, point set models require meshing and possibly thinning. Our hierarchical method provides both the essential connectivity and acceleration to give very similar results to the mesh models in what was an otherwise unexplored problem of direct, point-set part selection. With the increased role point set models are playing in the graphics community, this could be an important basis for future work in this area.

In comparison with other techniques such as direct and geometric selection, direct boundary definition, and region-growing, our technique proved to be versatile and fast, especially on large, complex models. Because of the arbitrary nature of part definitions, the sometimes unpredictable results from graph-cut selections, and the imperfect connectivity provided by the hierarchy, however, some selections become inefficient using only our technique. A simple cutting plane may be the most efficient in one case, while a combination of our technique with a direct selection tool for refinement may be better for another. As with selection in other media, such as 2D selections in image editing, our tool is best used in conjunction with other techniques.

Bibliography

Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., and Cohen, M. 2004. Interactive digital photomontage. *ACM Transactions on Graphics 23*, 3, 294–302.

Agarwala, A., Hertzmann, A., Salesin, D. H., and Seitz, S. M. 2004. Keyframe-based tracking for rotoscoping and animation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 584–591.

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., and Silva, C. T. 2001. Point set surfaces. In *Proceedings IEEE Visualization*, IEEE Computer Society, Washington, DC, USA, 21–28.

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., and Silva, C. T. 2003. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1, 3–15.

Amenta, N., Bern, M., and Kamvysselis, M. 1998. A new voronoi-based surface reconstruction algorithm. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 415–421.

Amenta, N., Choi, S., and Kolluri, R. K. 2001. The power crust. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, ACM, New York, NY, USA, 249–266.

Armstrong, C. J., Price, B. L., and Barrett, W. A. 2007. Interactive segmentation of image volumes with live surface. *Computers & Graphics 31*, 2, 212–229.

Bai, X., and Sapiro, G. 2007. A geodesic framework for fast interactive image and video segmentation and matting. In *IEEE 11th International Conference on Computer Vision*, 1–8.

Boykov, Y., and Funka-Lea, G. 2006. Graph cuts and efficient n-d image segmentation. *International Journal of Computer Vision 70*, 2, 109–131.

Boykov, Y. Y., and Jolly, M.-P. 2001. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *Proceedings Eighth International Conference on Computer Vision (ICCV 2001)*, vol. 1, 105–112.

Boykov, Y., and Kolmogorov, V. 2001. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *EMMCVPR '01: Proceedings of the Third International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, Springer-Verlag, London, UK, 359–374.

Bruyns, C. D., and Senger, S. 2001. Interactive cutting of 3D surface meshes. *Computers & Graphics 25*, 4 (August), 635–642.

Curless, B., and Levoy, M. 1996. A volumetric method for building complex models from range images. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 303–312.

Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., and Dobkin, D. 2004. Modeling by example. *ACM Transactions on Graphics 23*, 3 (Aug.), 652–663.

Gleicher, M. 1995. Image snapping. In *Proceedings of SIGGRAPH '95*, Computer Graphics Annual Conference Series.

Gopi, M., Krishnan, S., and Silva, C. T. 2000. Surface reconstruction based on lower dimensional localized delaunay triangulation. In *Computer Graphics Forum (Eurographics 2000)*, M. Gross and F. R. A. Hopgood, Eds., vol. 19(3).

Gregory, A., State, A., Lin, M. C., Manocha, D., and Livington, M. A. 1999. Interactive surface decomposition for polyhedral morphing. *The Visual Computer 15*, 9, 453–470.

Hoffman, D. D., and Richards, W. 1983. Parts of recognition. Tech. Rep. AIM-732.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. 1992. Surface reconstruction from unorganized points. In *SIGGRAPH '92:*

Proceedings of the 19th annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, 71–78.

Ji, Z., Liu, L., Chen, Z., and Wang, G. 2006. Easy mesh cutting. In *Computer Graphics Forum (Eurographics 2006)*, E. Groller and L. Szirmay-Kalos, Eds., vol. 25(3).

Kass, M., Witkin, A., and Terzopoulos, D. 1987. Snakes: active contour models. *International Journal of Computer Vision 1*, 4, 321–331.

Katz, S., and Tal, A. 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics 22*, 3, 954–961.

Lee, Y., Lee, S., Shamir, A., Shamir, A., Cohen-Or, D., and Seidel, H. P. 2004. Intelligent mesh scissoring using 3D snakes. In *Proceedings Pacific Conference on Computer Graphics and Applications*, 279–287.

Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., and Seidel, H.-P. 2005. Mesh scissoring with minima rule and part salience. *Computer Aided Geometric Design 22*, 5, 444–465.

Li, X., Toon, T. W., and Huang, Z. 2001. Decomposing polygon meshes for interactive applications. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, ACM, New York, NY, USA, 35–42.

Li, Y., Sun, J., Tang, C.-K., and Shum, H.-Y. 2004. Lazy snapping. *ACM Transactions on Graphics 23*, 3, 303–308.

Lombaert, H., Sun, Y., Grady, L., Grady, L., and Xu, C. 2005. A multilevel banded graph cuts method for fast image segmentation. In *IEEE International Conference on Computer Vision*, vol. 1, 259–265 Vol. 1.

Maes, F., Vandermeulen, D., Suetens, P., and Marchal, G. 1995. Computer-aided interactive object delineation using an intelligent paintbrush technique. In *First International Conference on Computer Vision, Virtual Reality, and Robotics in Medicine*, Springer, N. Ayache, Ed., vol. 905 of *Lecture Notes in Computer Science*, 77–81.

Mangan, A. P., Whitaker, R. T., and Whitaker, R. T. 1999. Partitioning 3d surface meshes using watershed segmentation. *Transactions on Visualization and Computer Graphics 5*, 4, 308–321.

Mortensen, E. N., and Barrett, W. A. 1995. Intelligent scissors for image composition. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 191–198.

Pauly, M., Keiser, R., Kobbelt, L. P., and Gross, M. 2003. Shape modeling with point-sampled geometry. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, ACM, New York, NY, USA, 641–650.

Protiere, A., Sapiro, G., and Sapiro, G. 2007. Interactive image segmentation via adaptive weighted distances. *Image Processing, IEEE Transactions on 16*, 4, 1046–1057.

Reese, L. J., and Barrett, W. A. 2002. Image editing with intelligent paint. In *Computer Graphics Forum (Eurographics 2002)*, vol. 21, 714–724.

Shamir, A. 2004. A formulation of boundary mesh segmentation. In *Proceedings 2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, 82–89.

Sharf, A., Blumenkrants, M., Shamir, A., and Cohen-Or, D. 2006. Snappaste: an interactive technique for easy mesh composition. *Vis. Comput. 22*, 9, 835–844.

Simari, P. D., and Singh, K. 2005. Extraction and remeshing of ellipsoidal representations from mesh data. In *GI '05: Proceedings of Graphics Interface 2005*, Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 161–168.

Wang, J., Bhat, P., Colburn, R. A., Agrawala, M., and Cohen, M. F. 2005. Interactive video cutout. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM, New York, NY, USA, 585–594.

Weyrich, T., Pauly, M., Keiser, R., Heinzle, S., Scandella, S., and Gross, M. 2004. Post-processing of scanned 3D surface data. In *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 85–94.

Wong, K. C.-H., Siu, T. Y.-H., Heng, P.-A., and Sun, H. 1998. Interactive volume cutting. In *Graphics Interface*.

Zöckler, M., Stalling, D., and Hege, H.-C. 2000. Fast and intuitive generation of geometric shape transitions. *The Visual Computer 16*, 5, 241–253.

Zwicker, M., Pauly, M., Knoll, O., and Gross, M. 2002. Pointshop 3d: an interactive system for point-based surface editing. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 322–329.