

# Interactive Part Selection for Mesh and Point Models using Hierarchical Graph-Cut Partitioning

Steven Brown

Bryan Morse

William Barrett

Department of Computer Science  
Brigham Young University

## 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 speed on large models and to provide connectivity when extending the technique to point set models.

**Keywords:** model partitioning, interactive modeling tools, scribble interface, graph cut, mesh, point set

**Index Terms:** I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Modeling packages; H.5.m [Information Interfaces and Presentation]: Miscellaneous

## 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 [18, 21, 25, 31, 34] (see also [32] for an excellent recent survey) 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 [10, 33] 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 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 may already be difficult to render interactively. 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 spatial manipulation



Figure 1: User session with the Lucy point set model. To select the wings of the 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 one view, the wings are completely selected (c,d). The entire interactive process for this initial partitioning takes less than 15 seconds even when using this 14 million point model.

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) or when the geometry is complex (Figure 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. It is well suited to such models, which usually consist of large collections of laser-scanned points, because it can provide 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.

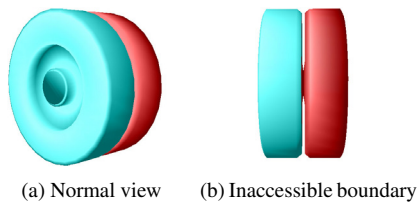


Figure 2: A selection boundary in an occluded area of a double tire (1K vertices). The selection was made with one scribble on each side of the object (a), even though the boundary separating the parts lies inaccessible on the axle connecting them (b).

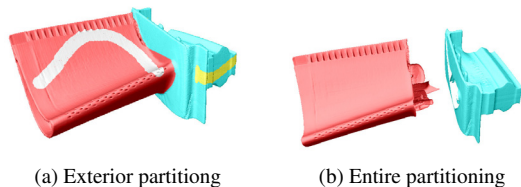


Figure 3: Selecting the blade from the turbine blade mesh model (882K vertices). The selection and resulting partitioning for the exterior of the mesh model can be seen in (a) and the full partitioning, including the occluded inner areas, can be seen in (b).

## 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 primitives include cuboids and spheres [36], 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 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 [9] or by having them place boundary-defining points along the desired cut and connecting those points with least-cost paths [13, 37, 39]. 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 [10, 19, 20, 33]. Such user-drawn boundaries may then be refined using active contours [19, 20], least-cost paths [10], or local graph-cut approaches [33]. 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. [10] has inherent problems with selections near silhouette edges (Figure 4) because users must stroke across desired areas instead of inside them.

It is worth noting the analogous boundary definition methods found in image and video region selection. There are semi-automated methods for assisting the user in selecting object con-

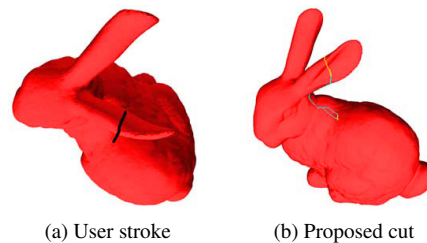


Figure 4: Unintended effects of surface boundary marking near silhouette edges (from [10]).

tours in images [11, 17, 26] and video [1], or assisting them in selecting object regions [24, 29].

Scribble-based interfaces are also popular for segmenting images or video [4, 8, 22, 35, among many others] and more recently for 3D meshes [16, 38]. 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 (included) and background (excluded) areas [6, 8, 38], however other approaches have also been successful, such as geodesic [5, 28] and region growing approaches [16]. Graph-cut segmentation can also be used locally to refine selections made by other methods [33].

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 3D volumes has been achieved at interactive rates using a multiband or hierarchical approach [4, 23] to accelerate the computation, and [38] uses a similar approach with multiresolution meshes.

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 leverages scribble-based selection techniques and contributes a method for hierarchically accelerated graph-cut segmentation that allows part selection for large (millions of vertices/points) 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.

## 3 INTERACTION AND SYSTEM OVERVIEW

Figure 5 demonstrates a typical user session, including selecting disjoint parts and refining the selections. Starting with the initial model (5a), the user marks (5b) 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 (5c). Additional scribbles are used to expand (5d,e) and refine (5f) the selection. The result selects along natural boundaries on the model (5g,h). From here, the selection can then be used for a number of purposes, including editing (5i).

The resulting selection boundaries are calculated from rough initial scribbles using graph-cut methods as described further in Section 4. Smaller mesh models can be formulated directly as graphs with connectivity defined by polygon edges and edge weights determined by differences in vertex normal angles (or alternatively through dual representations and dihedral angles). For larger mesh

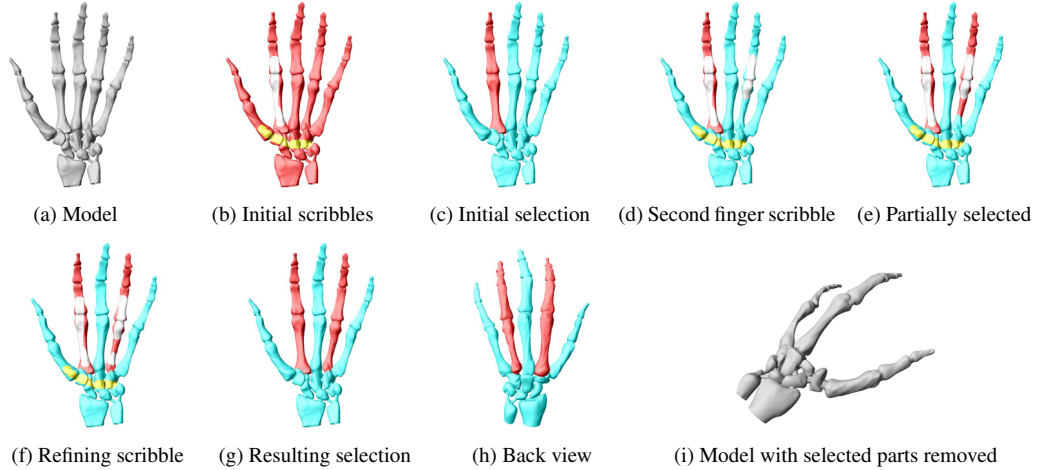


Figure 5: Screenshots from a typical user session using the hand bone mesh model (327K vertices). (a) The original model. (b) The initial scribbles: the white scribble is to be included, the yellow scribble is to be excluded. (c) The resulting selection, where red indicates the selected areas. (d) An 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.

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 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 6 using interpolated surface normals.

#### 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 an implementation of the algorithm described in [7].

Polygonal mesh models naturally lend themselves well to graph-cut segmentation because their inherent connectivity can be easily formulated as a graph, with mesh vertices as the graph vertices and mesh edges as the corresponding graph edges. Alternatively, one can formulate the graph based on the dual representation, with polygon faces as the graph vertices and their adjacency determining their graph connectivity. However, calculating adjacency between faces is either computationally slower or more of a strain on memory when given unorganized faces. Wedge-edge or similar data structures may be used for computational efficiency, but the additional memory costs become troublesome for very large models. The point-as-graph-node method is also more easily extensible to point set representations as discussed in Section 6. The drawback to the point-as-graph-node technique is that cuts do not occur on natural polygon edges but instead leave a single-polygon wide strip separating the included and excluded vertices. For our purposes we have chosen to group these polygons with the excluded region. We have implemented both approaches, and the dual-representation approach may be preferable for modest-sized models because it separates along model edges, but we have found the vertex-representation more (memory) efficient for large models.

During user interaction, vertices are selected and marked 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.

Using the common graph-cut notation of [8], given the set of vertices  $\mathcal{P}$ , the unordered set of vertex pairs  $\mathcal{N}$  representing mesh edges, and the binary partitioning vector

$A = (A_1, \dots, A_p, \dots, A_{|\mathcal{P}|})$  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  $A$  is defined as

$$E(A) = \lambda R(A) + B(A) \quad (1)$$

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  $\lambda$  represents the relative importance of these two terms. Since the work presented here focuses on geometric processing only, we limit our use of the regional properties term to enforcing correct labeling based on the user’s specifications. Should vertex/face color or other model information be available, one can also incorporate these properties into the region term as is commonly done with image segmentation.

The selection is calculated by finding the minimum cost partitioning as defined by Equation 1 over all possibilities of  $A$ , for which polynomial-time algorithms exist [7].

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

$$R(A) = \sum_{p \in \mathcal{P}} R_p(A_p) \quad (2)$$

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

$$R_p(A_p) = \begin{cases} 1 & \text{if } A_p \text{ conflicts with the user's input} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

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

$$B(A) = \sum_{\{p,q\} \in \mathcal{N}} B_{\{p,q\}} \delta(A_p, A_q) \quad (4)$$

and

$$\delta(A_p, A_q) = \begin{cases} 1 & \text{if } A_p \neq A_q \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

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).

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 [14, 19, 20], which states that people perceive part boundaries along concave creases, or negative minima of principle curvature. To achieve this effect, we assign edge weights  $B_{\{p,q\}}$  based on the cosine of the angle between the surface normals  $n_p$  and  $n_q$  of adjacent vertices  $p$  and  $q$  respectively. Vertex normals are sometimes provided with the model but can also be interpolated from the mesh data. However, because normal estimation can sometimes give inverted normals (see Section 6), and because it can be reasonably assumed that a surface will not have adjacent normals at more than  $90^\circ$  angles, the absolute value of the dot product between the normals can be used:

$$B_{\{p,q\}} = |n_p \cdot n_q| \quad (6)$$

and hence  $0 \leq B_{\{p,q\}} \leq 1$ . This formulation does not strictly distinguish between negative minima (concave creases) and positive maxima (convex ridges) due to its accounting for the potential lack of consistency between inward- and outward-pointing normals, but we have not found this to be a problem. Since we want to minimize costs in areas of concavity or sharp angles, values of  $B_{\{p,q\}}$  approaching 1 ( $0^\circ$  difference) indicate a flat and unlikely cut location, while values approaching 0 ( $90^\circ$  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  $\lambda$ , 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. For our implementation, we set  $\lambda = \infty$  to strictly enforce the user's marking of the surface.

## 5 HIERARCHICAL ACCELERATION

While the use of graph-cut segmentation directly works well for smaller models, it does not scale well for larger ones.<sup>1</sup> Because of this, hierarchical 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 6.

The adjacency of octree cells is used as graph adjacency. The edge weights are calculated as in Section 4 using the superpoint normals. The connectivity at the leaf level is determined as follows:

```

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

```

<sup>1</sup>The worst-case computational complexity of the implementation we use is approximately  $O(n^3)$ , but it technically sacrifices a strict polynomial bound for increased general performance. The typical running time for this implementation for most problems is between linear and quadratic.

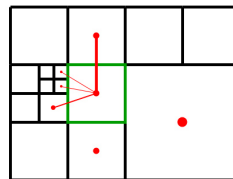


Figure 6: A two-dimensional example of the octree connectivity algorithm. The leaf being considered is marked in green. The coarser leaf to the right is ignored since an edge to it will be created when that leaf is considered, the finer leaves to the left are connected with  $\frac{1}{2}$  and  $\frac{1}{4}$  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.

This algorithm is illustrated in 2D in Figure 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 hierarchical technique is also applicable to point sets, which will be discussed in Section 6. While leaf adjacency 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 [4]. This process is repeated until the finest level of the hierarchy 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.

## 6 POINT SET MODELS

The methods presented here may also be applied to purely point-based representations (e.g., [2, 27, 40]) without having to explicitly reconstruct the surface mesh (e.g. [3, 15]). Extending our technique to point set models (Figure 7) is fairly straightforward but requires changes to some of the key elements to obtain 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 7.3).

While the hierarchy provides connectivity for the graph, the surface normals needed for edge weight calculation are still lacking. We use the method described in [12] to compute surface normals for each leaf cell, or superpoint. A single point  $p$  is found for each

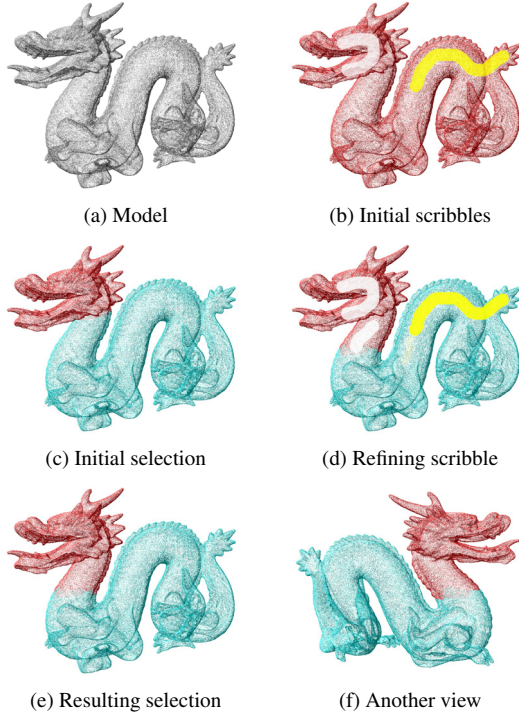


Figure 7: User session with a dragon point set model. (a) The original model. (b) The initial scribbles; the white scribble is to be included, the yellow scribble is to be excluded. (c) The resulting selection. (d) An additional scribble to include the neck. (e) The resulting selection. (f) View of the back.

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  $n_p$  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  $V_i = q_i - p$ , where  $1 \leq i \leq k$ , then we need to find  $n_p$  such that it minimizes

$$\sum_{i=1}^k \left( n_p \cdot V_i - \frac{\sum_{i=1}^k n_p \cdot V_i}{k} \right)^2 \quad (7)$$

or equivalently

$$\sum_{i=1}^k \left( \left( V_i - \frac{\sum_{i=1}^k V_i}{k} \right) \cdot n_p \right)^2 \quad (8)$$

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

$$n_p = \operatorname{argmin}_n \sum_{i=1}^k ((V_i - C) \cdot n)^2 \quad (9)$$

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 9 can be rewritten as

$$n_p = \operatorname{argmin}_n \|An\|_2 \quad (10)$$

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 10. Additionally, the smallest singular value can function as a measure of fit (see Section 8).

Surface normals for the parent cells are calculated using the same method, considering the child cell superpoints as neighboring vertices of the centroid vertex of the parent. The resulting normals are not necessarily aligned inwardly or outwardly and cannot be easily aligned because of the lack of a true surface representation. However, because inverted normals have no effect on our edge weighting (Equation 6) this step was ignored. Our interpolated normals were used for all selection calculations, though we imported normals from corresponding mesh models for rendered display purposes only.

## 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.

### 7.1 General Mesh Model Results

Figure 5 (shown earlier on page 3) 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 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 1, and examples can be found in Figures 8 and 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 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 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 inter-

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 1: Selection times in seconds for mesh models with and without the hierarchy. These models, while not all shown here, were included to quantify the performance of the proposed method on models of varying types and sizes. Regular selection times include user time. Hierarchical selections include hierarchy preprocessing (shown in parentheses).

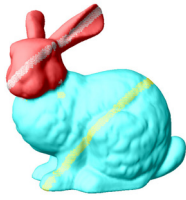


Figure 8: Selecting the head from the Stanford bunny (35K vertices). This selection was obtained in 9 seconds without hierarchical acceleration.

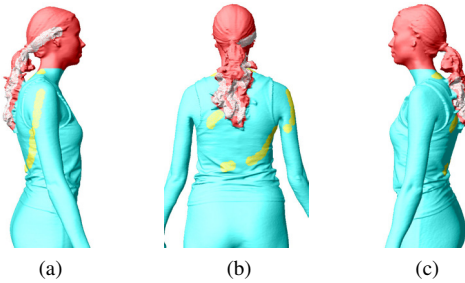


Figure 9: Selecting the head and hair from the female01 model (184K vertices). This selection was obtained in 28 seconds without hierarchical acceleration.

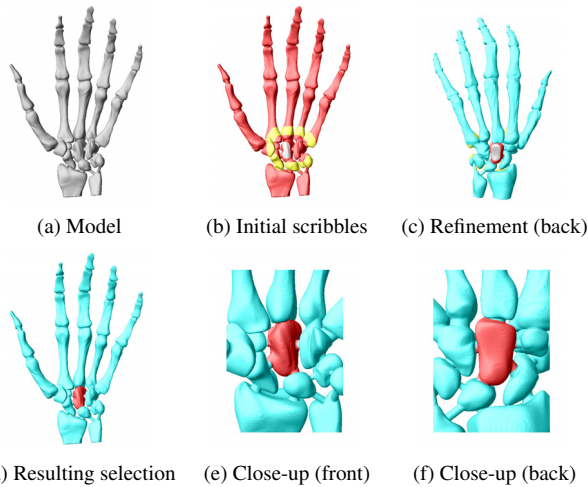


Figure 10: Selecting the center wrist bone of the hand mesh model (327K vertices). Note that the selection requires five separate selection boundaries. (a) The original model. (b) The initial scribbles: the white scribble is to be included; the yellow scribble is to be excluded. (c) An additional scribble 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.

action required. The majority of the overall selection time of the larger examples in Table 1 was spent on the initial cut.

Figure 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 [10] took 13 minutes of user time, while the automatic segmentation in [18] took 28 minutes of computer time.

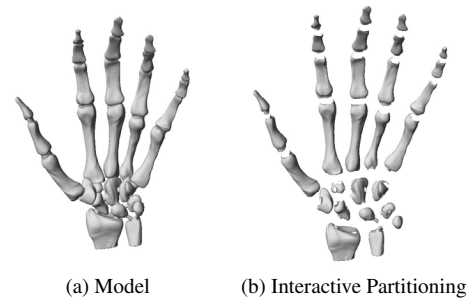


Figure 11: A mesh model of hand bones interactively segmented in under 2 minutes.

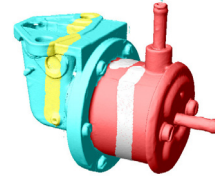


Figure 12: Selecting a cylinder from an oil pump model (542K vertices). This selection was obtained in 61 seconds with hierarchical acceleration.

## 7.2 Hierarchical Acceleration Results

An example of partitioning using hierarchical acceleration can be seen in Figure 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 13. The hierarchical method cuts the processing time in half with nearly identical results.

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 million vertices require 1 second or less additional time to create the hierarchy and normals. For larger mesh models the overall preprocessing time is still only a matter of seconds, with file read time being a significant portion.

For larger models, the increase in preprocessing time and potential user intervention is more than made up for by the speedup in selection times. As shown in Table 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 time spent waiting, 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.

## 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) because of the necessity of interpolating surface normals. However, due to the size of some point set models, and depending on the user's intent, this may be preferable to first meshing the model.

Simple selections on point set models remain virtually unchanged as can be seen in Figure 7, in which the user wishes to

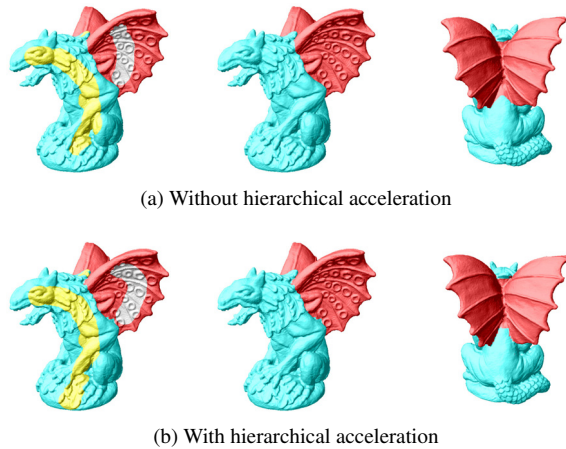


Figure 13: Selecting the wings from a gargoyle (863K vertices). The regular mesh selection (a) took 45 seconds. The hierarchical selection (b) took 20 seconds, including preprocessing, for an almost identical result.

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: 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.

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 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.

## 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 types of media.

Hierarchical acceleration helped to handle 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 option 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.

We have chosen to use a volumetric (octree) hierarchy because its simplicity gives it significant computational efficiency advantages for very large models. However, alternative forms of hierarchies using surface patch hierarchies rather than by volumetric means should be explored. For example, a hierarchical form of [25] would align more closely to the cascading watershed regions used to accelerate volume segmentation in [4]. These may produce more intuitive behavior, though for large models this would have to be weighed against any resulting increase in associated computational or memory costs. Improved multilevel point-set representations [30] could also be explored.

Point set selections can be made rapidly using the hierarchy, but the preprocessing time to interpolate surface normals 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. Faster normal interpolation would reduce this time. Alternately, the interpolation method we use [12] 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 and better handle complex surfaces.

Basing the weighting function on the difference in surface normals between two points generally provides 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 boundary cost term should improve selection. If additional vertex information (color, etc.) was available, this could likewise be incorporated into the region cost term, providing segmentation based on both geometry and coloring.

We also note that in areas where the surface of the model is flat or nearly flat in one or more directions, small perturbations in the vertex angles can cause cuts to occur in non-intuitive places. The same effect is also common when using similar methods for other forms of media, where noise in otherwise nearly homogeneous regions can cause similar arbitrary cuts. Smoothing or other forms of regularization may help alleviate this effect.

## 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 large 3D mesh and point set models. Our technique gives good results for 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 also be applied to large, unconnected point set models.

## ACKNOWLEDGEMENTS

We appreciate those who graciously provided the 3D models used in this work: 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); OpenScene-Graph.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.

## REFERENCES

- [1] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 584–591, New York, NY, USA, 2004. ACM.
- [2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.
- [3] N. Amenta, S. Choi, and R. K. Kolluri. The power crust. In *SMA '01: Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*, pages 249–266, New York, NY, USA, 2001. ACM.
- [4] C. J. Armstrong, B. L. Price, and W. A. Barrett. Interactive segmentation of image volumes with live surface. *Computers & Graphics*, 31(2):212–229, 2007.
- [5] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007.
- [6] Y. Boykov and G. Funka-Lea. Graph cuts and efficient n-d image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006.
- [7] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *EMM-CVPR '01: Proceedings of the Third International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 359–374, London, UK, 2001. Springer-Verlag.
- [8] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *Proceedings Eighth International Conference on Computer Vision (ICCV 2001)*, volume 1, pages 105–112, 2001.
- [9] C. D. Bruyns and S. Senger. Interactive cutting of 3D surface meshes. *Computers & Graphics*, 25(4):635–642, August 2001.
- [10] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. *ACM Transactions on Graphics*, 23(3):652–663, Aug. 2004.
- [11] M. Gleicher. Image snapping. In *Proceedings of SIGGRAPH '95*, Computer Graphics Annual Conference Series, August 1995.
- [12] M. Gopi, S. Krishnan, and C. T. Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. In M. Gross and F. R. A. Hopgood, editors, *Computer Graphics Forum (Eurographics 2000)*, volume 19(3), 2000.
- [13] A. Gregory, A. State, M. C. Lin, D. Manocha, and M. A. Livingston. Interactive surface decomposition for polyhedral morphing. *The Visual Computer*, 15(9):453–470, 1999.
- [14] D. D. Hoffman, W. Richards, A. Pentl, J. Rubin, and J. Scheuhammer. Parts of recognition. *Cognition*, 18:65–96, 1984.
- [15] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of ACM SIGGRAPH*, pages 71–78, New York, NY, USA, 1992. ACM.
- [16] Z. Ji, L. Liu, Z. Chen, and G. Wang. Easy mesh cutting. In E. Groller and L. Szirmay-Kalos, editors, *Computer Graphics Forum (Eurographics 2006)*, volume 25(3), 2006.
- [17] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.
- [18] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graphics*, 22(3):954–961, 2003.
- [19] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H. P. Seidel. Intelligent mesh scissoring using 3D snakes. In *Proceedings Pacific Conference on Computer Graphics and Applications*, pages 279–287, 2004.
- [20] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel. Mesh scissoring with minima rule and part salience. *Computer Aided Geometric Design*, 22(5):444–465, 2005.
- [21] X. Li, T. W. Toon, and Z. Huang. Decomposing polygon meshes for interactive applications. In *13D '01: Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 35–42, New York, NY, USA, 2001. ACM.
- [22] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Transactions on Graphics*, 23(3):303–308, 2004.
- [23] H. Lombaert, Y. Sun, L. Grady, and C. Xu. A multilevel banded graph cuts method for fast image segmentation. In *IEEE International Conference on Computer Vision*, volume 1, pages 259–265 Vol. 1, 2005.
- [24] F. Maes, D. Vandermeulen, P. Suetens, and G. Marchal. Computer-aided interactive object delineation using an intelligent paintbrush technique. In N. Ayache, editor, *International Conference on Computer Vision, Virtual Reality, and Robotics in Medicine*, volume 905 of *Lecture Notes in Computer Science*, pages 77–81. Springer, 1995.
- [25] A. P. Mangan and R. T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.
- [26] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 191–198, New York, NY, USA, 1995. ACM.
- [27] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 641–650, New York, NY, USA, 2003. ACM.
- [28] A. Protiere and G. Sapiro. Interactive image segmentation via adaptive weighted distances. *Image Processing, IEEE Transactions on*, 16(4):1046–1057, 2007.
- [29] L. J. Reese and W. A. Barrett. Image editing with intelligent paint. In *Computer Graphics Forum (Eurographics 2002)*, volume 21, pages 714–724, 2002.
- [30] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proceedings of ACM SIGGRAPH 2000*, pages 343–352, July 2000.
- [31] A. Shamir. A formulation of boundary mesh segmentation. In *Proceedings 2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, pages 82–89, 2004.
- [32] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, January 2008. Appeared previously as a State-of-the-Art Report in Eurographics 2006.
- [33] A. Sharf, M. Blumenkrantz, A. Shamir, and D. Cohen-Or. SnapPaste: an interactive technique for easy mesh composition. *Vis. Comput.*, 22(9):835–844, 2006.
- [34] P. D. Simari and K. Singh. Extraction and remeshing of ellipsoidal representations from mesh data. In *Proceedings of Graphics Interface*, pages 161–168, 2005.
- [35] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 585–594, New York, NY, USA, 2005. ACM.
- [36] T. Weyrich, M. Pauly, R. Keiser, S. Heinzele, S. Scandella, and M. Gross. Post-processing of scanned 3D surface data. In *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, pages 85–94, 2004.
- [37] K. C.-H. Wong, T. Y.-H. Siu, P.-A. Heng, and H. Sun. Interactive volume cutting. In *Graphics Interface*, 1998.
- [38] S. Zelinka and M. Garland. Surfacing by numbers. In *Proceedings Graphics Interface*, 2006.
- [39] M. Zöckler, D. Stalling, and H.-C. Hege. Fast and intuitive generation of geometric shape transitions. *The Visual Computer*, 16(5):241–253, 2000.
- [40] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3d: an interactive system for point-based surface editing. In *Proceedings of ACM SIGGRAPH*, pages 322–329, New York, NY, USA, 2002. ACM.