Volume Graphics (2006)
T. Möller, R. Machiraju, T. Ertl, M. Chen (Editors)

Live Surface

Christopher J. Armstrong, William A. Barrett and Brian Price

Brigham Young University

Abstract

Live Surface allows users to segment and render complex surfaces from 3D image volumes at interactive (sub-second) rates using a novel, cascading graph cut (CGC). The process consists of four steps. (1) Preprocessing for generation of a complete 3D watershed hierarchy followed by tracking of all catchment basin surfaces. (2) User selection of foreground and background seeds from two-dimensional, image cross-sections. (3) Segmentation of the volume by cascading through the 3D watershed hierarchy from the top, applying graph cut successively, at each level, only to catchment basins bordering the segmented surface from the previous level. (4) OpenGL rendering for display and update of the segmented surface at interactive rates. CGC allows the entire image volume to be segmented an order of magnitude faster than existing techniques that make use of graph cut. Segmentation and rendering, combined, is accomplished in about 0.5 seconds, allowing 3D surfaces to be displayed and updated dynamically as each additional foreground seed is deposited. CGC allows the user to control and steer the segmentation with immediate user feedback, providing a Live Surface tool for 3D image volumes similar to the Live Wire (Intelligent Scissors) tool used in 2D images.

Categories and Subject Descriptors (according to ACM CCS): I.2.10 [Artificial Intelligence]: Vision and Scene Understanding I.3.3 [Computer Graphics]: Picture/Image Generation I.4.6 [Image Processing and Computer Vision]: Segmentation J.3 [Life and Medical Sciences]:

1. Introduction

The Live Wire interface of Intelligent Scissors [MB95] is an effective means for performing 2D segmentation by providing immediate feedback for boundary selection as the mouse moves. This gives the user constant awareness of what belongs to the current selection. The introduction of Intelligent Scissors fundamentally changed the paradigm for interactive boundary detection. For 3D volume segmentation, the analogous feedback model would be a "live surface." The ability to update surfaces incrementally and interactively, within an optimal framework, would provide a user with the same freedom of interaction in extracting surfaces from image volumes as Intelligent Scissors provides in the extraction of boundaries from 2D images. Unfortunately, there is no straightforward extension of Live Wire to surfaces.

Interactive 3D segmentation has wide application medically in the analysis of CT and MRI imagery. Accurate segmentation of anatomy facilitates measurement and diagnosis and is critical in areas such as surgical planning, prosthesis fitting, and diagnosis. Interactivity allows the medical professional to guide the segmentation, preventing errors.

The major concern with 3D segmentation techniques to date is that they are unable to combine optimality with sufficient interactivity. Optimal techniques, though successful in 2D datasets, have difficulty coping with the massive amounts of data encountered in 3D sets. Faster methods such as thresholding, levelsets, or region growing algorithms, have problems with object under/over-estimation and have difficulty distinguishing between neighboring objects with similar characteristics, making it difficult to isolate an object from its neighbors.

We introduce Live Surface for interactive selection and display of objects in 3D image volumes. We claim that Live Surface does for 3D image volumes what Intelligent Scissors did for 2D images; specifically, it brings existing optimal 3D segmentation techniques into an interactive framework by providing immediate user feedback to verify the selection.

2. Related Work

Intelligent Scissors [MB95, MB99] introduced an efficient means of interactive 2D segmentation through the use of

Live Wire which adheres to edges within the image. Intelligent Paint [RB02], a region-based segmentation tool, provides interactive segmentation through a painting metaphor. Graph cut [BJ01, BK01] has been used to extend these concepts to images within an optimal framework in Lazy Snapping [LSTS04] and GrabCut [RKB04].

The use of graph cut for optimal and accurate segmentation of 3D surfaces has been extensively validated for medical image volumes [LWCS06]; however, execution times can last up to 10s of minutes to cut volumes of 2–8 megavoxels. In order to accelerate the process, a single layer of watershed regions [VS91] has been used in the place of voxels for medical volumes [YZNC05], and video [LSS05]. Interactive Video Cutout [WBC*05] goes a step further, generating two layers of oversegmented image regions. Lombaert uses a resolution pyramid to perform coarse-to-fine refinement in Banded Graph Cuts [LSGX05]. However, each of these accelerated approaches still typically requires from 10 seconds to one minute.

The Voxel-Man project, based on the work of Karl Heinz Höhne [HPP*01, HH92, HBP*90], extends segmentation to 3D for visualization of medical volumes. Voxel-Man is a visually impressive tool for exploring human anatomy. These tools produce a visualization of objects within a volume very quickly, but require extensive user interaction and are based on nonoptimal interactive thresholding.

Fuzzy connectedness [US96, USdAL02, US03] seeks to produce a more optimal segmentation than techniques based on simple homogeneity criteria such as thresholding or region-growing. Fuzzy techniques measure the "affinity" between image elements and then group them accordingly.

The graph cut algorithm popularized by Boykov and Jolly [BJ01] has found widespread use in advancing the work of optimal 2D and 3D segmentation. Efficiencies in interaction have been gained through the use of oversegmentaion, but so far these have been limited to 1 or 2 levels [WXSC04, LSTS04, YZNC05, LSS05]. The complete watershed hierarchy as described by Reese and Barrett [RB02] now provides a means for near real-time segmentation in Live Surface.

3. System Overview

The interface to Live Surface is shown in Figure 1. The user is presented with the raw data which is displayed as a parallelepiped volume. The user may rotate the volume and advance a cross section through it. The user may also lock the cross section to a specific voxel that will remain in the cross section as the volume is rotated. This allows the user to more easily find a cross section that will minimize interaction.

Live Surface allows the user to extract the surface of objects using simple mouse clicks and brush strokes. The voxels specified in this manner will be used as foreground and background seeds for the graph cut segmentation. The surface is displayed in the left panel and updated in real time.

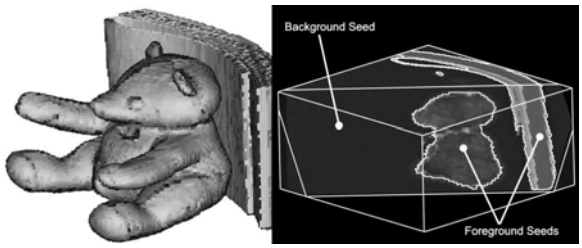

Figure 1: *Visual Interface for Live Surface. User interacts with cross sections of the image volume (right). The Resulting surface is rendered with OpenGL (left).*

Graph cut, as it is typically used, is unable to handle the massive number of voxels in image volumes at interactive speeds. We generate a tobogganed watershed hierarchy as a preprocessing step in order to present graph cut with a simpler graph to work with. Each catchment basin's visible surface is also defined during preprocessing so that during interaction the surfaces are simply selected from a table. These important preprocessing steps make sub-second surface extraction possible.

As the user interacts with the volume it is segmented using the novel cascading graph cut (CGC) algorithm described in 5.3, which allows for sub-second segmentation, taking advantage of the watershed hierarchy structure. Finally the predefined surfaces that form the surface of the segmentation are then rendered using OpenGL.

4. Preprocessing

The goal of the preprocessing phase of Live Surface is to compute everything possible *a priori* so as not to slow the interaction. The principal computations of the preprocessing phase are the tobogganed watershed hierarchy and the surface of each catchment basin. On average, the preprocessing phase lasts a little over one minute for 1–18 megavoxel volumes.

4.1. Watershed Hierarchy

Voxels are grouped into catchment basins using the tobogganing watershed algorithm [Fai90] (using the keep sliding method [YH91] to handle plateaus). Catchment basins are grouped into larger basins using a similar algorithm [Ree99]. The process is repeated until all basins are grouped into a single basin, thus generating a complete hierarchy. Each level of the hierarchy forms an adjacency graph for the graph cut algorithm.

3D tobogganed watersheds are a straightforward extension of their 2D counterpart. First we compute the gradient

magnitude of each voxel. Voxels are grouped into regions called catchment basins. In row major order each voxel is recursively grouped with ("slides" to) its 6-connected neighbor whose gradient magnitude is the smallest, with the condition that the neighbors gradient magnitude is less than or equal to that of the voxel itself, where $\leq$ handles the plateau problem. A catchment basin is thus composed of all voxels that slide to the same local minimum.

A similar method is used to group catchment basins for the next level of the hierarchy. We use the t-score difference between adjacent regions in place of the gradient magnitude to recursively group each basin with the basin most similar to it [Ree99]. The t-score models the distributions of colors in neighboring basins parametrically (equation 1), allowing us to quickly compare neighboring basins.

$$t^2 = \frac{N_m N_n \|\overline{C_m} - \overline{C_n}\|^2}{N_m V_n + N_n V_m} \tag{1}$$

where N is the number of voxels in the basin (the subscripts m and n indicate which basin), $\overline{C}$ is the average color vector of the basin, and V is a measurement of the color variance of the basin. V is computed as follows (where C_k is the color of some voxel k in the basin):

$$V = \frac{1}{N} \sum_{k \in n} \|\overline{C_n} - C_k\|^2 \tag{2}$$

This process is repeated until the image volume is reduced to a single basin. This is what is meant by a *complete* watershed hierarchy. Each basin corresponds to a node in the hierarchy, with its children nodes corresponding to the basins from which it is composed. Each level of the hierarchy represents a level of image granularity. The hierarchy forms a complete partition of the image volume. The basins at each level of the hierarchy are mutually exclusive and exhaust the space. The average branching factor of the hierarchy is 5. Hierarchy depths range from 9–12 levels. The watershed hierarchy preserves the natural edge information in the image volume, thus preventing the loss of detail resulting from a traditional resolution pyramid.

Each level is also used to generate a weighted graph for graph cut. Each graph represents the 3D adjacency of all basins at the same level of the hierarchy. Each node represents a basin. Each edge represents the adjacency of a pair of basins. We are currently using the metric described in Lazy Snapping [LSTS04] to weight the edges in the graph:

$$w(m,n) = \frac{1}{1 + \|\overline{C_m} - \overline{C_n}\|^2} \tag{3}$$

where $w(m,n)$ is the weight of the edge between m and n. $\overline{C_m}$ and $\overline{C_n}$ are the average color vectors of basins m and n.

4.2. Predefining Surfaces

In order to speed rendering we predefine the visible surface of each basin at the lowest level of the hierarchy. The surface of any selection at any level of the hierarchy is composed of these surfaces. We store a surface patch for each edge in the graph at the lowest level of the hierarchy in a lookup table. Each patch can take on the color of either of the two basins which it separates depending on which is classified as foreground. Each patch is composed of actual voxel faces which may be rendered with one of two surface normals computed from the x, y, and z gradients of the two voxels which the face separates. The surfaces are generated by sweeping through the volume and finding neighboring voxels belonging to different catchment basins.

5. Live Surface Extraction

The goal of Live Surface is to extract and render a surface which evolves as the user moves the mouse. The mouse input generates the foreground and background seeds which are applied to the watershed hierarchy. The volume is then segmented using CGC. Finally the result of the segmentation is rendered using the predefined surfaces.

5.1. Graph Cut

A key algorithm for extraction of a live surface is graph cut. We use Kolmogorov's implementation [BK01] to perform the graph cut. Graph cut segments a graph into two mutually exclusive subgraphs (foreground and background) by cutting the edges between them such that the sum of the weights of the cut edges is minimized. Seeds (placed by the user) assign nodes to belong to one of the two subgraphs. Our graph is formed as described in 4.1.

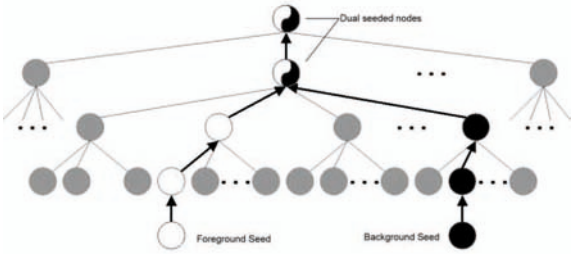

Figure 2: *Propagation of seeds and dual-seeded basins.*

5.2. Seeding

The user extracts a surface by specifying seed voxels within the volume. Each seed the user specifies in the volume must be represented at each level of the hierarchy (Figure 2). When a foreground (white) or background (black) seed is placed, it is propagated up the hierarchy from child to parent until it reaches the top. Some basins then contain both foreground and background seeds (shown as a mix of white and black). We call such basins dual-seeded basins. Such basins

indicate that the user disagrees with the hierarchy's aggregation of certain basins and thus signals the CGC algorithm that further refinement is needed in that area.

5.3. Cascading Graph Cut

The intent of cascading graph cut is to provide hierarchical speedup by reducing the number of nodes in the graph to be cut. This is done by computing a graph cut for a coarse approximation of the surface and including only basins adjacent to that surface to refine the surface at the next level of the hierarchy with another graph cut.

CGC computes a coarse graph cut segmentation at the top of the hierarchy based on the user specified seeds. If a basin is not dual seeded and does not lie on the surface of the cut it is ignored (Figure 3 a). The children of the remaining basins (Figure 3 b) are used to compute a refined cut at the next level of the hierarchy (Figure 3 c). Once again the basins not on the surface of the cut are ignored. Notice the striped basin that was previously ignored is now included because it lies on the surface of the cut (Figure 3 d). The process repeats until the bottom of the hierarchy is reached. As we cascade down the hierarchy the vast majority of basins are ignored, providing a hierarchical speed-up. The number of nodes in the graph to be cut no longer depends on the size of the volume but on the surface area of the selected object. Since the watershed hierarchy is a proper tree structure, determining which basin to include and which to ignore is done by simply traversing the tree. This allows for sub-second segmentation results. The resulting cut is the selected surface.

The recursive **CascadingGraphCut** algorithm in Appendix A details this process. It first computes a graph cut on the input graph G, classifying each basin as foreground or background. It then generates the next graph to be cut H. The children of basins along the surface of the cut are included in H (white in Figure 3 a). Some of these basins may not be in G (striped in Figure 3 d). The children of dual seeded basins are also included in H. Each edge in H corresponds to an edge in the existing graph in the hierarchy. Any edge connecting two basins in H is included as is. All edges from a basin in H to a basin not in H are represented in the cascaded graph by an edge from the basin in H to a generic seed basin. The classification of the basin not in H determines which of the two generic seed basins to use. This has the effect of lumping each ignored basin into one of the generic seed basins.

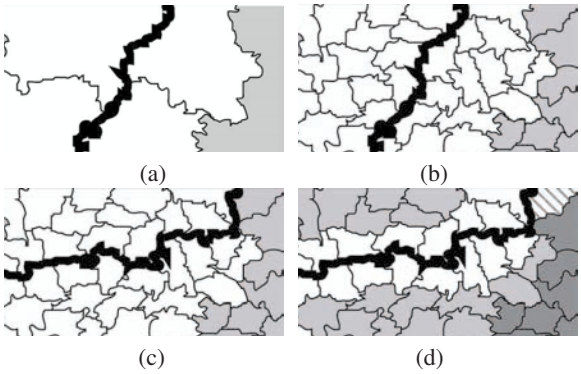

Figure 3: *Cross section of a volume to which CGC is applied. (a) An initial graph cut is computed and basins not on the surface of the cut are ignored (gray). (b) The children of the remaining basins are collected into a cascaded graph (white). (c) The cascaded graph is segmented, using graph cut to refine the surface. (d) Once again basins not on the surface are ignored. The process repeats.*

5.4. Cascading Graph Cut Refinement

It is inefficient to compute a segmentation of the entire volume as the user refines the selection. We assume that once the user has a general selection, further interaction will be aimed at refining local discrepancies. In this case local refinement would be preferable to a global segmentation. In order to accomplish this we provide a variation of CGC, cascading graph cut refinement (CGCR).

When the user places a foreground (or background) seed in a basin that already contains a foreground seed the graph cut performed on levels of the hierarchy at or above that basin give no new information and so is not performed. Instead we simply form a graph from the children of the previously seeded basin and start a CGC. This localizes the bulk of the CGC to the region near the new seed. However, as CGC cascades, basins that do not descend from the preseeded basin (striped in Figure 3 d) may be included in the CGC as they become a part of the surface. The effect is that cuts made by local refinements are integrated seamlessly into the globally optimal surface.

Appendix A contains the detailed algorithm for **CascadingGraphCutRefine**. When a user deposits a seed, it is propagated up the hierarchy until either the top of the hierarchy is reached or a basin is reached that already contains the same kind of seed. If the top of the hierarchy is reached we begin a CGC with the graph at the top of the hierarchy as input. If we find a basin that already contains the same kind of seed, we generate a graph with only the children of that basin, formed in the same manner as in CGC.

Figure 4: *Selecting the bones in the arm and hand by dragging the mouse over them in the voxel display. The result is an evolving surface that appears to be growing.*

The effect of this update appears as a growing surface like that shown in Figure 4. As the user adds new seeds, the selection is updated on average in less than a second.

5.5. Rendering

A critical feature of live surface is the rendered surface of the selection. This allows for inspection of the results which often leads to recognition of discrepancies. In order to render the selection we scan through the edges of the graph at the lowest level of the hierarchy. Each edge we find that separates foreground from background is used to look up a surface patch in the table of predefined surfaces. The color and normals of the foreground basin are applied to the surface patch. OpenGL allows the evolving surface to be rendered at interactive rates.

5.6. Application to video

Live surface has been designed for (x,y,z) volumes such as CT and MRI. However, video (x,y,t) can also be segmented with live surface. The video surface in Figure 12 is extracted from the volume represented by the frames shown to the left of the renderings. Each rendering shows an evolving video surface viewed from different projections.

Video demonstrates more significant changes in the t dimension than volumes in the z dimension, requiring slightly different treatment. One simple change to allow for this difference is to first group regions within frames before grouping them between frames. For video we compute the first level of catchment basins within frames before grouping basins between frames. Although Live Surface currently requires a little more interaction to segment video volumes than (x,y,z) volumes, the sub-second update time still holds, making the process considerably less tedious than other systems.

6. Results

Performance results for Live Surface applied to a variety of medical image volumes are summarized in Table 1. These results were gathered using a machine with a 3.6 Ghz Intel® Xeon™ processor with 3 GB RAM. For image volumes ranging from 1 to 18 megavoxels, preprocessing required from 2 seconds to 2 minutes depending linearly on the size of the volume and the total number of basins at all levels of the hierarchy. User interaction required about 5 minutes on average. User interaction time is indicative of the number of seeds specified by the user. The average time to compute CGC for all volumes was .42 seconds, with an average of .05 seconds to render the result. The number of 3D catchment basins in the initial hierarchy was 199,239 on average. The number of levels in the complete watershed hierarchy ranged from 9 to 12 (Table 2). The number of levels for each volume corresponds to the size of the volume and the complexity of its content.

	Preprocess	User Interaction	Average Cut	Cut and Render
knee	147	120	0.36	0.38
hand	51	300	0.40	0.45
pollen	16	20	0.058	0.090
skull	80	1200	1.15	1.19
foot	34	300	0.51	0.56
teddy	2	20	0.12	0.13
VHP	35	180	0.37	0.39

Table 1: *Performance results for 7 different image volumes. Times are recorded in seconds*

Segmentation and rendering results for Live Surface applied to the Visible Human Project are shown in Figures 8 and 9. The coloration in the rendering comes from sampling the color in the data itself. The color chosen for the heart is the average color of the basins forming its surface. Soft tissue is typically much more challenging to segment with conventional tools such as thresholding and region growing because of the similarity in color and density with surrounding organs and structures. However, CGC performs very well, allowing the user to extract surfaces interactively and incrementally, with anatomically relevant surfaces (heart chambers, ascending or descending aorta, carotid arteries, ribs, etc.) added to the cumulative surface with each user interaction.

Application of CGC to skeletal anatomy in CT volumes is shown in Figure 10. Segmentation of the bones in the foot is challenging for several reasons. Because of the articulation of the joints, the bones lie in close proximity to one another and therefore, due to aliasing, the intensity of the vox-

	knee	hand	pollen	skull	foot	teddy	VHP
voxel	8M	18M	6M	13M	13M	1M	7M
L 1	.5M	91k	10k	.4M	.1M	32k	.3M
L 2	.1M	24k	4k	97k	9k	8k	64k
L 3	36k	63k	772	25k	2k	2k	5k
L 4	10k	2k	201	6k	613	583	916
L 5	3k	433	50	2k	155	156	207
L 6	933	114	13	416	45	40	52
L 7	223	32	3	102	11	13	15
L 8	53	9	2	27	4	4	5
L 9	13	4	1	9	2	2	3
L 10	4	2		3	1	1	2
L 11	2	1		2			1
L 12	1			1			

Table 2: *Numbers of basins at each level of the complete watershed hierarchy for each of the 7 image volumes.*

Figure 5: *Segmenting "Teddy van Gough" from the CT volume is accomplished with just a few mouse clicks.*

els between the bones is increased making it difficult to separate them using standard interactive segmentation techniques such as thresholding or region growing. However, CGC allows individual bones to be segmented and separated quite easily. In fact, separation of adjoining structures is something at which graph cut, in general, excels. The bones also vary in density through the interior. This too causes techniques such as thresholding or levelsets to sometimes fail by segmenting out the interior and/or overestimating the surface. Finally, because the foot bones consist of fine, thin bones of varying thickness and density, many techniques are not adaptive or robust enough to capture them. Once again Live Surface has no trouble including them in the selection.

MRI images are inherently more noisy than CT images but this is another area where the globally optimal properties of graph cut excel. Figure 11 illustrates segmentation of the femur, tibia, fibula, and muscles of the knee from an MRI volume. In this example, CGC is still extremely efficient, completing each cut in an average of .38 seconds, well within the user's interactive time constant. The muscles are an example of the ability of CGC to segment not only the high contrast skeletal anatomy, but also the low contrast, soft tissue.

The teddy bear data set is an excellent example of how Live Surface is frequently able to extract well defined surfaces with a single foreground and background seed. In Figure 5 the initial surface is segmented with only one of each seed. Each update to the surface is achieved with a single foreground seed. Updates for the teddy bear are computed and rendered in .13 seconds average.

The hand in Figure 6 shows the versatility of CGC in its ability to adapt to both skeletal and soft tissue anatomy in the same segmentation. This allows the anatomy to be explored in context, and because of the segmentation and rendering speed, anatomical layers can be added or deleted interactively.

Finally the skull in Figure 7, while exceeding one second in average update time when selecting the entire skull, updates much more quickly when selecting just the teeth. Traditionally segmenting individual teeth would be considerably difficult using techniques such as thresholding or levelsets. Live Surface is not just able to select individual teeth, but each tooth selected on the left of the figure has a distinct surface and is not fused with any of the neighboring teeth.

The focus of Live Surface is to provide an interactive technique for optimally segmenting image volumes. At this point we rely on the body of existing literature, as well as visual inspection, to validate the correctness and accuracy of the extracted surfaces. For example, graph cut has been shown to produce accurate results in medical imaging [LWCS06] and desirable results for video [WBC*05, LSS05]. However, an important part of future work will be to quantitatively compare surfaces obtained from our Cascading Graph Cut (CGC) with the extensively validated voxel-level graph cut. Lombaert [LSGX05], shows that a coarse-to-fine approach results in a good approximation, lending great confidence to the quality of our results. Furthermore, our interactive speed allows the user to quickly adjust the segmentation to the desired result if it has not yet been achieved.

7. Conclusion and Future Work

Live Surface does for 3D volumes what Intelligent Scissors did for 2D images. Intelligent Scissors was not the first technique to formulate optimal boundary detection as a minimum cost path search in a graph, but it brought it into an interactive framework. In the same way, Live Surface provides the user with immediate (~ 0.5 sec) feedback about the selected surface. This immediate feedback helps the user decide what to sample next. This is a significant improvement upon existing graph cut techniques which typically require 5–10 seconds (often much more) to make the cut. (Imagine trying to steer a car remotely with a snapshot of the car's position on the road updated only every 5–10 seconds.) Live Surface allows the user to segment volumes continuously with immediate visual feedback in the refinement of the selected surface. Hence Live Surface fundamentally changes the paradigm for interactive 3D segmentation and surface extraction.

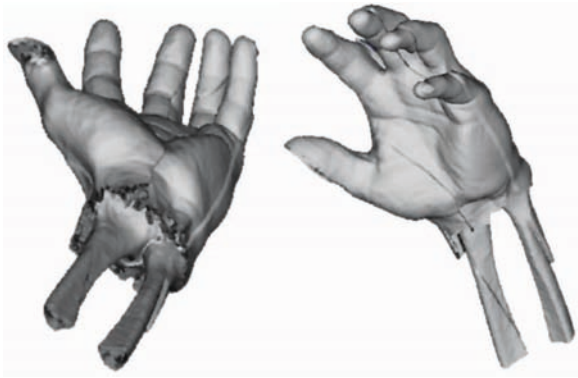

Figure 6: *Skeletal and soft tissue anatomy are segmented adaptively and simultaneously, allowing the user to explore both in context.*

Figure 7: *Segmenting individual teeth from the skull CT volume (13 Mega-voxels) is accomplished in about two minutes. Segmenting the entire skull takes a little over twenty minutes.*

None of the core components of Live Surface limit it to 3D data. Thus Live Surface could be applied to n-dimensional data sets as well [Bar86, SUR*05]. We expect that such applications may exceed the sub-second update speeds; however, the speedup should still scale and demonstrate significant improvement over non-hierarchical approaches.

Most formulations of graph cut make use of a second term or second kind of edge other than that described by equation 3. The second term is a measurement of how each node conforms to some foreground or background model and may provide greater adaptivity to the current Live Surface formulation.

User interaction requires by far the greatest amount of time in the process, especially when extracting fine, complex structures. Much of user interaction results from exploration of the cross-

Live Surface allows objects in 3D image volumes to be extracted and visualized interactively using a cascaded formulation of optimal graph cut. By identifying object surfaces in a preprocessing stage, data structures can be generated that allow for sub-second object segmentation. Optimal graph cut coupled with immediate user feedback makes Live Surface an efficient tool for segmentation, visualization, and exploration of object surfaces in 3D image volumes.

Appendix A: Algorithms

Pseudo-code for Cascading Graph Cut and Cascading Graph Cut Refinement is given here.

Global Variables and Data Structures

Globals:

BASIN *fSeed*, *bSeed* {generic seed BASINs}
GRAPH *top* {the top GRAPH in the hierarchy}

Data Structures:

GRAPH {BASIN[] *B*; EDGE[] *E*; **Int** *level*;}
BASIN {
Bool *isFSeed*, *isBSeed*, *isF*, *isB*;
EDGE[] *E*; BASIN *parent*; Basin[] *children*;
}
EDGE {BASIN *from*, *to*; **Float** *weight*;}

Cascading Graph Cut Algorithm

CascadingGraphCut

Input:
GRAPH *G*
Algorithm:
1. **GraphCut**(*G*)
2. **if** *G.level* = 0 **return**
3. Initialize GRAPH *H* to empty
4. *H.level* ← *G.level*-1
5. *H.B* ← the *children* of all BASINs in *G* that border the cut
6. *H.B* ← the *children* of all BASINs in *G* that are dual seeded
7. *H.E* ← all existing EDGEs *to* and *from* a BASIN in *H*
8. **for each** EDGE *E from* each BASIN in *H* **begin**
9. **if** *E.to* is **not** in *H* **begin**
10. Initialize EDGE *F* to be a copy of *E*
11. *F.to* ← **ForeOrBack**(*E.to*)
12. *H.E* ← *F*
13. **end** {this loop generates edges from BASINs
14. **end** in *H* to *fSeed* and *bSeed*}
15. **CascadingGraphCut**(*H*)

GraphCut

Input: GRAPH *G*
Algorithm: run graph cut, set *isF* & *isB* for BASINS in *G*

ForeOrBack

Input: BASIN *B*
Output: *fSeed* **or** *bSeed*
Algorithm:
16. **if** *B.isF* **return** *fSeed*
17. **else if** *B.isB* **return** *bSeed*
18. **else return ForeOrBack**(*B.parent*)

Cascading Graph Cut Refinement Algorithm

CascadingGraphCutRefine

Input:
BASIN *B*
Bool *isFore*
Algorithm:
1. **if hasSeed**(*B*, *isFore*) **then return**
2. initialize **Int** *i* ← 0
3. initialize BASIN *C* ← *B*
4. **while not hasSeed**(*C*, *isFore*) **begin**
5. **setSeed**(*C*, *isFore*)
6. **if** *i* = *top.level* **then return CascadingGraphCut**(*top*)
7. *C* ← *C.parent*; *i* ← *i* + 1

```
 8. end
 9. initialize GRAPH H to empty
10. H.B ← C.children
11. H.E ← existing EDGEs from and to a BASIN in G
12. for each EDGE E from each BASIN in H begin
13.   if E.to is not in H begin
14.     Initialize EDGE F to be a copy of E
15.     F.to ← ForeOrBack(E.to)
16.     H.E ← F
17.   end                {this loop generates edges from BASINs
18. end                                in H to fSeed and bSeed}
19. CascadingGraphCut(H)
hasSeed
  Input: BASIN B, Bool isFore
  Output: Bool
  Algorithm:
20. if isFore then begin
21.   if B.isFSeed then return true;   else return false
22. else
23.   if B.isBSeed then return true;   else return false
setSeed
  Input: BASIN B, Bool isFore
  Algorithm:
24. if isFore then B.isFSeed ← true
25. else B.isBSeed ← true
```

References

[Bar86] BARRETT W. A.: Dynamic three-dimensional shaded surface display of the rotating, beating left ventricle, atrium and aorta from cine ct. In *IEEE Proceedings of Computers in Cardiology* (Washington, DC, USA, 1986), IEEE Computer Society, pp. 491–494.

[BJ01] BOYKOV Y., JOLLY M.-P.: Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *ICCV* (2001), pp. 105–112.

[BK01] BOYKOV Y., KOLMOGOROV V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *Energy Minimization Methods in Computer Vision and Pattern Recognition* (2001), pp. 359–374.

[Fai90] FAIRFIELD J.: Toboggan contrast enhancement for contrast segmentation. In *Proceedings of the 10th International Conference on Pattern Recognition (ICPR '90)* (June 1990).

[HBP*90] HÖHNE K. H., BOMANS M., POMMERT A., RIEMER M., SCHIERS C., TIEDE U., WIEBECKE G.: 3d visualization of tomographic volume data using the generalized voxel model. *The Visual Computer 6*, 1 (1990), 28–36.

[HH92] HÖHNE K.-H., HANSON W. A.: Interactive 3d segmentation of mri and ct volumes using morphological operations. *JCAT 16*, 2 (Mar.–Apr. 1992), 285–294.

[HPP*01] HÖHNE K. H., PFLESSER B., POMMERT A., RIEMER M., SCHUBERT R., SCHIEMANN T., TIEDE U., SCHUMACHER U.: A realistic model of human structure from the visible human data. *Methods Of Information In Medicine 40* (2001), 83–89.

[LSGX05] LOMBAERT H., SUN Y., GRADY L., XU C.: A multilevel banded graph cuts method for fast image segmentation. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 259–265.

[LSS05] LI Y., SUN J., SHUM H.-Y.: Video object cut and paste. *ACM Trans. Graph. 24*, 3 (2005), 595–600.

[LSTS04] LI Y., SUN J., TANG C.-K., SHUM H.-Y.: Lazy snapping. *ACM Trans. Graph. 23*, 3 (2004), 303–308.

[LWCS06] LI K., WU X., CHEN D. Z., SONKA M.: Optimal surface segmentation in volumetric images-a graph-theoretic approach. *IEEE Trans. Pattern Anal. Mach. Intell. 28*, 1 (2006), 119–134.

[MB95] MORTENSEN E. N., BARRETT W. A.: Intelligent scissors for image composition. *ACM Trans. Graph.* (1995), 191–198.

[MB99] MORTENSEN E. N., BARRETT W. A.: Toboggan-based intelligent scissors with a four-parameter edge model. In *CVPR* (1999), pp. 2452–2458.

[RB02] REESE L. J., BARRETT W. A.: Image editing with intelligent paint. In *Proceedings of Eurographics 2002* (2002), vol. 21, pp. 714–724.

[Ree99] REESE L.: *Intelligent paint: Region-based interactive image segmentation*. Master's thesis, Brigham Yound University, 1999.

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: Grabcut: interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph. 23*, 3 (2004), 309–314.

[SUR*05] SIEGLER S., UDUPA J., RINGLEB S., IMHAUSER C., HIRSCH B., ODHNER D., SAHA P., OKEREKE E., ROACH N.: Mechanics of the ankle and subtalar joints revealed through a 3D quasi-static stress MRI technique. *J Biomech 38*, 3 (2005), 567–78.

[US96] UDUPA J. K., SAMARASEKERA S.: Fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation. *CVGIP: Graphical Model and Image Processing 58*, 3 (1996), 246–261.

[US03] UDUPA J. K., SAHA P. K.: Fuzzy connectedness and image segmentation. *Proceedings of the IEEE 91*, 10 (2003), 1649–1669.

[USdAL02] UDUPA J. K., SAHA P. K., DE ALENCAR LOTUFO R.: Relative fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 11 (2002), 1485–1500.

[VS91] VINCENT L., SOILLE P.: Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell. 13*, 6 (1991), 583–598.

[WBC*05] WANG J., BHAT P., COLBURN R. A., AGRAWALA M., COHEN M. F.: Interactive video cutout. *ACM Trans. Graph. 24*, 3 (2005), 585–594.

[WXSC04] WANG J., XU Y., SHUM H.-Y., COHEN M. F.: Video tooning. *ACM Trans. Graph. 23*, 3 (2004), 574–583.

[YH91] YAO X., HUNG Y. P.: Fast image segmentation by sliding in the derivative terrain. *Proceedings of SPIE Intelligent Robots and Computer Vision 1991 1607* (1991).

[YZNC05] YUAN X., ZHANG N., NGUYEN M. X., CHEN B.: Volume cutout. *The Visual Computer (Special Issue of Pacific Graphics 2005) 21*, 8–10 (2005), 745–754.