Eurographics Workshop on Sketch-Based Interfaces and Modeling (2005)
Takeo Igarashi, Joaquim A. Jorge (Editors)

Sketch-based Segmentation of Scanned Outdoor Environment Models

Xiaoru Yuan Hui Xu Minh X. Nguyen Amit Shesh Baoquan Chen †

University of Minnesota at Twin Cities, MN, USA

Abstract

When modeling with scanned outdoor models, being able to select a subset of the points efficiently that collectively represent an object is an important and fundamental operation. Such segmentation problems have been extensively studied, and simple and efficient solutions exist in two dimensions. However, 3D segmentation, especially that of sparse point models obtained by scanning, remains a challenge because of inherent incompleteness and noise. We present a sketched-based interface that allows segmentation of general 3D point-based models. The user marks object and background regions by placing strokes using a stylus, and the tool segments out the marked object(s). To refine the results, the user simply moves the camera to a different location and repeats the process. Our method is based on graph cuts, a popular and well-tested paradigm for segmentation problems. We employ a two-pass process: we use the strokes to perform 2D image segmentation in the projection plane of the camera and use its results for the 3D scanned data segmentation. The advantages of our method are ease of use, speed and robustness. Our method works for general 3D point models and not just range images. Important applications include selection of objects when dealing with large, unorganized point models for refinement, remodeling, meshing, etc.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques I.4.6 [Image Processing and Computer Vision]: Segmentation H.5.2 [User Interfaces]: Theory and methods

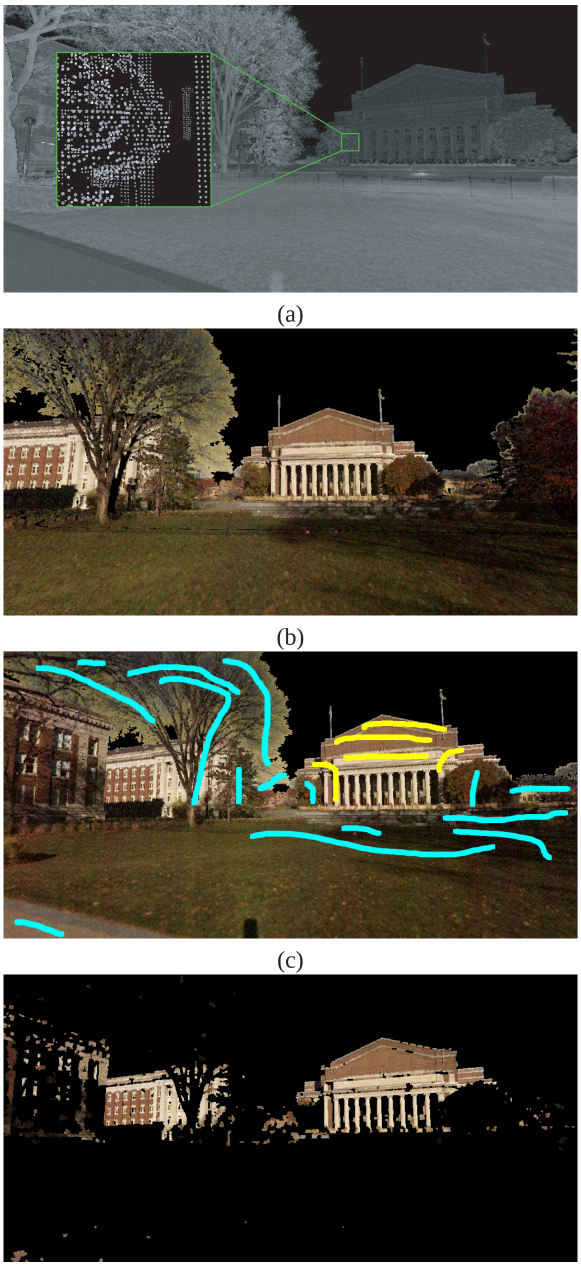
1. Introduction

With the advent of practical scanning technologies and commercial scanning equipment, it has become easier to capture indoor and outdoor environments by scanning them. Processing and rendering such scanned data is an active area of research in computer graphics and finds diverse applications in modeling, architectural design, urban planning, etc. Some examples of processing these unorganized point models are meshing (converting into a triangle mesh), refinement of structure (constructing precise geometry), transformation in representation (hybrid models for rendering), etc., each of which opens up new avenues of research.

All of the above operations assume that a subset of points representing a semantic object can be extracted from the point model. When an outdoor environment is scanned, this is a non-trivial operation as many objects exist in the resulting point model. Moreover, unlike scanning of small models, outdoor environment scanning is often imprecise and sparse. After scanning the whole environment, many operations often require the user to focus on one object in the environment, and hence the ability to select points forming a semantic object is very important. Providing a sketch-based interface for such segmentation of point models and realizing the segmentation efficiently is the subject of this paper.

Segmentation, in general, refers to the problem of extracting a desired subset of some set of n-dimensional primitives. In two dimensions, this translates to extracting particular regions in an image. Several diverse approaches to image segmentation [LSTS04, RKB04, MB98, MB95] have been proposed and implemented. These approaches can be easily extended to a special case of 3D point models: range images. Range images are 2D images where every pixel has a depth value. Scanning devices usually produce range images. However, when multiple scans are registered together,

† Email: {xyuan,hxu,mnguyen,ashesh,baoquan}@cs.umn.edu


(a)

(b)

(c)

(d)

Figure 1: Segmenting out an object using out tool. (a) the input point model containing 2,234,945 points. (b) the user interactively navigates the scene rendered with point sprites. (c) the user selects a camera location and places yellow strokes for object and cyan strokes for background. (d) the result of this segmentation.

a general point model is obtained. Our method works for general 3D point models.

The nature of scanned data from outdoor environments presents several new challenges for segmentation. Being points, there is no connectivity information to leverage for object selection. Secondly, unlike indoor environments that can be scanned multiple times to acquire precise models, outdoor environments are dynamic. Changing positions of the sun, trees moving due to wind, etc. are examples of challenges that are not presented by indoor environments. Moreover, as outdoor environments are more complex, it is not feasible to sample them in detail by scanning it several times. Therefore, a precise model cannot be obtained by simply scanning the environment several times from different view points.

The main advantage of our method is that it is easy to use, is efficient and involves little skill. The user navigates in scenes rendered photorealistically and determines a suitable camera position, and then merely places strokes to hint foreground and background regions on a rendered image of the model. The system segments out (possibly part of) the desired foreground object. The user can then change the view and place more strokes similarly to refine the selection. The whole process is thus, very similar to normal image segmentation. The advantages of our method are simplicity, speed and robustness. Our method can be embedded in larger point-based modeling systems like PointWorks [XGC04].

2. Related Work

The method for segmentation of point models that we propose in this paper augments state-of-the-art development in image processing and vision with added interaction. Our method performs both 2D and 3D segmentations in two consecutive stages. We now review several 2D and 3D segmentation methods, emphasizing on the user interface techniques they support.

The widely used Magic Wand [Inc02] collects color statistics from a user-specified image region and segments a sub-image region, where colors fall within some specified thresholds. Suitable threshold values depend on individual image context. This method is prone to error when the foreground and background pixels overlap in their color distributions. Intelligent paint [Ree99], which is a region-based interactive segmentation technique based on *Tobogganing* [Fai90], extracts regions of interest from the image background starting from input paint strokes. Intelligent scissors [MB95, MB98, MB99], a boundary-based method, computes a minimum-cost path via a shortest-path graph algorithm between user-specified boundary points. Intelligent scissors and paint allow the user to quickly and accurately select objects of interest [MR99,Mor99]. A tool for selecting objects developed by Tan and Ahuja [TA01] uses freehand sketches based on the decomposition of the image segmentation into a triangulation and captures the adjacency information of the segments as well as the shape of the segment boundaries. Sketched-based input has been used for various segmentation problems. *Lazy Snapping* [LSTS04] uses foreground and background strokes for image segmentation. Our interface is inspired from this work. More recently, video segmentation has been realized [WBC*05] by extending the idea of sketch-based image segmentation over time.

Graph cuts [GPS89, BJ01, BK04] is a combinatorial optimization technique using global optimal pixel labeling of an object and its background, which is computed by the max-flow/min-cut algorithm. A binary image segmentation problem can be posed as a 2D graph cuts problem. Several interactive methods for 2D image segmentation with user interaction have been developed [AHSS04, LSTS04, RKB04] based on the graph cuts algorithm. *Lazy Snapping* [LSTS04] combines graph cuts with pre-computed over-segmentation and provides users a set of intuitive interface tools for flexible control and editing. *Grabcut* [RKB04] extends graph cuts by introducing an iterative scheme using graph cuts for optimization of intermediate steps which also simplifies the user interaction. Each iteration estimates color statistics of the object and the background and applies graph cuts to compute a refined segmentation. More recently, the graph cuts formulation has been applied to interactively extract foreground objects from a video [WBC*05]. Boykov and Jolly [BJ01] extended the graph cuts algorithm to volume data sets by marking foreground and background seeds on volume slices. We have also developed a graph cuts-based method for volume data set segmentation with flexible user sketch input [YZNC05].

The output of most scanning devices is a range image. Most 2D image segmentation methods can be extended to work on such range images by considering these depth values in the segmentation process. Djebali *et al.* [DMS02] and Heisele *et al.* [HR99] employ an automatic divide and conquer strategy to fit several simple 3D patches to the point model and merge them into bigger surfaces. Yu *et al.* [YFM01] propose segmentation of point models obtained by registering several range images. However, the environments they deal with are indoor rooms which can be scanned several times to obtain a fairly precise model. Outdoor environments are dynamic in nature and so scanning them repeatedly may not result in a precise model. Moreover, objects like trees are much more difficult to segment automatically because of the inherent fuzziness in their structure. Therefore, some user interaction is needed. Usually, the user marks out the object in a range image or a rendered image of a point model by a closed region and an automatic segmentation algorithm is run within this closed space [XC02]. Another approach is to reconstruct surfaces out of point models and use mesh segmentation algorithms like Shock graphs [SKK01], medial axis [LK01] and Reeb graphs [HSKK01]. Besides being expensive, surface reconstruction is not robust for noisy or sparse data like that obtained by scanning outdoor environments.

The rest of the paper is organized as follows: we provide a user walk-through and overview of our pipeline in Section 3. We then summarize the graph cuts problem and its formulation for the segmentation problem in Section 4. We then explain our pipeline in detail in Sections 5 and 6. In Section 5, we explain our over-segmentation procedure for point models for efficiency. In Sections 6.1 and 6.2 we discuss the formulation and solution of the segmentation problem for our application. In Section 7, we describe a tool for correcting misclassified objects. Finally, we show and discuss some results in Section 8 and conclude our paper in Section 9.

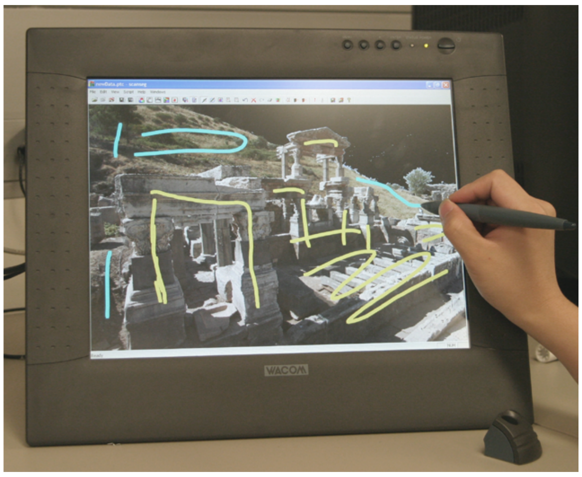


Figure 3: User interface. User direct draws strokes on a tablet display.

3. User Walk-through and Overview

Our input data sets are point models that result from a single or multiple scans of some outdoor environment. The data set contains the position, color (Figure 2(a)) and intensity (Figure 2(b)) at every point. Normals (Figure 2(c)) are computed by least-squares plane fitting. Since laser scanners often do not capture color well, the color information is less reliable in the data sets unless photos have been registered with the scanned data.

The user loads a point model into our system and navigates it freely. The model may result from one scan, in which case it is simply a range image, or it may be the result of multiple registered scans. The user selects an appropriate camera position and then specifies foreground and background regions with yellow and cyan strokes respectively (Figure 1(c)). As a result of this, the desired foreground object is fully or partially selected (Figure 1(d)). To refine the selection (possibly to select points hidden in this view), the user simply moves the camera to a different position and repeats the process. If the user is not satisfied with the results of the segmentation, he/she can undo the operation. Figure 3 demonstrates the user interface of our system.

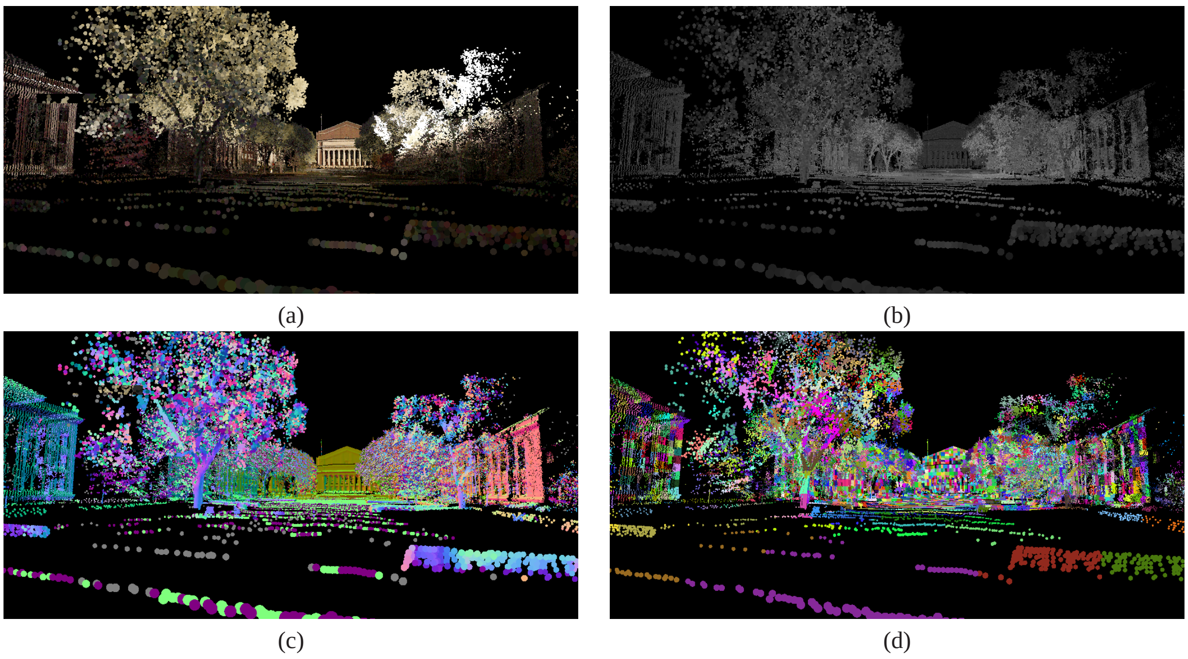


Figure 2: Input model and results of over-segmentation. This is a single scan of the Northrop Mall at the University of Minnesota. (a) the input colors. Note how the input colors are not captured faithfully by the scanner device. (b) the input intensity (c) the input normals encoded as colors that are obtained by least-squares plane fitting around every point. (d) the result of over-segmentation. All points in a cluster are given the same color.

When a point model is loaded, an over-segmentation procedure is carried out on it to make the segmentation process more efficient, as done in *Lazy Snapping* [LSTS04] (Section 5). When the user fixes a camera position to sketch, the system performs image over-segmentation on the rendered 2D image of the point model from that camera position. After the user hints some foreground and background regions in the image, our system separates background and foreground regions using graph cuts on a graph of the over-segmented image. Results from this are used to achieve the final segmentation (Section 6.2). The final result can be further refined according to user freehand drawing (Section 7).

4. Segmentation and Graph Cuts

We now summarize the formulation of segmentation as a graph cuts problem for completeness. A more detailed explanation can be found in [BJ01].

Our system constructs a weighted graph G_I with two special nodes and applies the graph cuts algorithm to it for the 3D segmentation process. Construction of G_I is explained in Section 6.1. Let us assume in this section that G_I exists with a weight function $w_I(e)$ on its edges.

For $G_I = (V,E)$, let N be the set of $\langle p,q\rangle$ pairs, where $p,q \in V$, p,q are adjacent in G_I. The segmentation problem can be formulated as a graph cuts problem. Every vertex v of G_I is labeled as "F"(foreground) and is part of a set V_F or "B"(background) and is part of a set V_B in a valid segmentation. Given some vertices that are labeled either "F" or "B", all other vertices are labeled "U"(unknown). The aim is then to determine V_B and V_F such that all vertices in V_B and V_F are labeled "B" and "F" respectively, and the following energy function is minimized:

$$E(V) = \lambda \sum_{p_i \in V} R(p_i) + (1-\lambda) \sum_{p_i,p_j \in N} B(p_i,p_j) \qquad (1)$$

where

$$R(p_i) = \begin{cases} 0 & : \quad p_i = \text{“}F\text{”} \\ +\infty & : \quad p_i = \text{“}B\text{”} \\ -\ln \frac{d_F^{p_i}}{d_B^{p_i}+d_F^{p_i}} & : \quad p_i = \text{“}U\text{”} \end{cases} \qquad (2)$$

if $p_i \in V_F$, and

$$R(p_i) = \begin{cases} +\infty & : \quad p_i = \text{“}F\text{”} \\ 0 & : \quad p_i = \text{“}B\text{”} \\ -\ln \frac{d_B^{p_i}}{d_B^{p_i}+d_F^{p_i}} & : \quad p_i = \text{“}U\text{”} \end{cases} \qquad (3)$$

if $p_i \in V_B$, and

$$B(p_i,p_j) = \alpha e^{\frac{d_{p_i,p_j}}{2\sigma^2}} \qquad (4)$$

This is achieved by assigning appropriate weights to the edges of the graph and finding a minimum-weight edge cut of the graph such that the two special nodes are in separate components. The edge weights are assigned as enumerated in Table 1. Our algorithm performs segmentation in two passes: in the first pass it performs image segmentation using G_I and in the second pass it performs 3D segmentation by projecting the clusters created by the over-segmentation (Section 5). The values of $d_B^{p_i}$, $d_F^{p_i}$ and d_{p_i,p_j} for G_I are defined in subsequent sections.

The following notation is used henceforth:

C_{p_i}	:	Average color of points/pixels in vertex p_i
$\hat{N}_i$	:	Average color of points in vertex p_i
C_B	:	Initial avg. background color (Section 6.1)
C_F	:	Initial avg. foreground color (Section 6.1)
N_B	:	Initial avg. background normal (Section 6.1)
N_F	:	Initial avg. foreground normal (Section 6.1)

5. Preprocessing: Over-segmentation

In order to make the overall segmentation process faster, we perform two types of over-segmentation on the point model. First, we perform over-segmentation on the input 3D point model to accelerate subsequent operations in the pipeline. The clusters formed as a result of this are used for 3D segmentation. This operation is done only once per model, before the user begins to navigate it and place strokes.

First, we compile a kD-tree for the given points in the model, based on their 3D positions. For every node v in the tree, let T_v be the subtree rooted at v. We compute a vector W_v for every node in the tree.

$$W_v = [n_v, d_{v(max)}, \sigma_v, \theta_v] \tag{5}$$

where n_v is the number of points in T_v. $d_{v(max)}$ is the maximum Euclidean distance between any two points in T_v. σ_v is the standard deviation of the intensities of all points in T_v, and θ_v is the apex angle of the smallest cone that encloses all normals of points in T_v.

We define a global threshold vector $W_{threshold} = [n_t, d_t, \sigma_t, \theta_t]$. During a top-down traversal of the tree, if $W_v < W_{threshold}$ (where $<$ is the element-wise comparison of two vectors) at a node v, we cluster all the points in T_v, else we descend into the child nodes of v. The selection of $W_{threshold}$ value depends on the properties of the scan data. We use $W_{threshold} = [256, 0.25, 32, 0.2]$ (obtained empirically) for the results in this paper. In this way, we obtain a tree whose leaves are the desired clusters. Figure 2(d) shows an example of such over-segmentation.

When the user fixes a camera position with the intention to place strokes, we perform 2D image over-segmentation on the rendered image to cluster pixels into regions. We use the watershed algorithm to accomplish this, similar to [LSTS04]. We form a graph G_I from these 2D clusters. The clusters form vertices of G_I and edges are introduced between adjacent clusters. Two special vertices s_I and t_I are introduced and edges are added from s_I to all vertices, and from all vertices to t_I. Every edge e has a weight $w_I(e)$ as summarized in Table 1.

edge	weight(cost)	for
$\{p_i, p_j\}$	$B(p_i, p_j)$	$p_i, p_j \notin s, t$
$\{s, p_i\}$	$\lambda \cdot R(p = \text{“}B\text{”})$	$p = \text{“}U\text{”}$
	∞	$p = \text{“}F\text{”}$
	0	$p = \text{“}B\text{”}$
$\{p_i, t\}$	$\lambda \cdot R(p = \text{“}F\text{”})$	$p = \text{“}U\text{”}$
	0	$p = \text{“}F\text{”}$
	∞	$p = \text{“}B\text{”}$

Table 1: Assigning edge weights for the graph cuts problem. Notation is as enumerated in Section 4 and equations 2,3,4. This table is taken from [BJ01] and is mentioned here for completeness.

6. Two-step Segmentation

In this section we explain how segmentation of an object in a point model is achieved in two steps. As stated before, we first perform 2D image segmentation on a rendered image of the scene and use its results for 3D segmentation.

6.1. Image Segmentation

As explained in Section 5, when the user fixes a camera position with the intention to place strokes, 2D over-segmentation is performed and a graph G_I is compiled. When the user marks foreground and background regions using sketched strokes, all vertices of G_I that overlap those strokes are marked “F”(foreground) or “B”(background) respectively, while all others are marked “U”(unknown).

We calculate $R(p_i)$ as

$$R(p_i) = w_1 * R_1(p_i) + w_2 * R_2(p_i) \tag{6}$$

where $R_j(p_i)$ are calculated as per equations 2 and 3. For $R_1(p_i)$, $d_B^{p_i} = \|C_{p_i} - C_F\|$ and $d_F^{p_i} = \|C_{p_i} - C_B\|$, and for $R_2(p_i)$, $d_B^{p_i} = \|1 - (\hat{N}_i \cdot \hat{N}_F)\|$ and $d_F^{p_i} = \|1 - (\hat{N}_i \cdot \hat{N}_B)\|$. During initialization, all the vertices marked “F” are further clustered on their colors (and normals), the shortest distance (or deviation in case of normals) between any two such clusters is C_F (or N_F). C_B and N_B are obtained similarly.

$$B(p_i, p_j) = \sum_{k=1}^{3} w_k * B_k(p_i, p_j) \tag{7}$$

where $B_k(p_i, p_j)$ are calculated as per equation 4. Here, the d_{p_i,p_j} for all $B_k(p_i, p_j)$ $(k = 1,2,3)$ measure differences between colors, normals and projected depths of p_i and p_j respectively.

When the max-flow min-cut algorithm is run on G_I in the above setting, a minimum weight edge cut is produced, in which every vertex in G_I is labeled "F" or "B". This labeling is used for the 3D segmentation.

6.2. 3D Segmentation

The results of 2D segmentation represent a subset of all points that the user desired to select. All 3D clusters obtained earlier (Section 5) are now projected to the image plane. Points belonging to clusters that project to the foreground regions are selected. This set of points may represent a superset of the set of points that the user desired to select, in the case where the point model is not a simple range scan. In these cases, our interface allows the user to navigate to another camera position and de-select the undesired points. In any case, if an object is not fully visible in any camera position, the user has to select it in multiple passes of deciding a camera position and placing strokes.

7. Segmentation Refinement

As our segmentation procedure may introduce unwanted points, it is desirable to have an interactive tool for the user to edit incorrectly classified points. The tool we have designed for user-guided segmentation refinement is similar to the lasso and has been applied to segment volume data [YZNC05]. The user draws freehand curves or rectangles to selectively remove or add unwanted objects from the segmentation result. Our system achieves this refinement by projecting scanned point data objects onto the current viewing plane and checking whether their projections fall into the region specified by the user. The user can also adjust viewing angles to get the best editing position for a particular semantic object.

8. Results

The scanning device used for obtaining the point models shown in this paper is the Riegl LMS-Z360 3D Imaging Sensor, which has 12mm precision up to 200m, and outputs range, intensity and RGB color as images. Our *PointWorks* system [XGC04] has been used for rendering. We have performed all our experiments on a Dell Precision 530 workstation with single Intel Xeon 2.20G Hz CPU, 1GB RAM, 4X AGP motherboard and a Nvidia GeForce 6800 Ultra graphics card with 256MB memory. Our system is implemented using MFC, OpenGL, and Cg.

Figure 4 shows our tool working on a scan data set of the Ephesos excavation site in Turkey. This point model has $2,005,430$ points. The 3D oversegmentation takes about 2 minutes. 2D image oversegmentation for each view takes around 0.3 seconds which depends on the actual size of the rendering window. The running time for 3D segmentation is around $0.5-2$ seconds. Figure 4(b) shows the foreground and background sketched strokes that the user draws to hint at segmenting out the stones in the model.

Figure 4(c) shows the result of this operation. After rotating the camera once more and placing some strokes, the object is fully segmented out, as shown in Figure 4(d).

Figure 5 shows how our segmentation can be used to emphasize (or de-emphasize) some objects in a scanned environment by rendering them non-photorealistically. Once points representing an object are identified through our segmentation process, the object can be subjected to any modeling operations like remeshing, refining, etc.

9. Discussion and Future Work

Most work on processing point-based models today is on models of small scanned objects that can be scanned individually with fairly highly precision. Those scanning processes are performed indoors and the environments are well controlled. When large outdoor environments are scanned, such objects have to be segmented out before any processing. Noise and imprecision make such segmentation even more challenging. There are also important user interface considerations in such operations, as this operation is usually embedded as a tool in larger systems that process point models comprehensively. Our method is both easy to use and implement, and works for general point models.

Noise in data tends to deteriorate results of any processing done on it, and our system is not immune to it. Noise occurs in our point models due to three reasons: imprecision in the range of a scanner, inadequate sampling of the environment and dynamism of the environment (wind, people walking, etc.). Noise generally occurs at micro-levels, and hence for selecting "rough" objects which are macro-entities, it often tends to play only a minor role. When noise is excessive, a smoothing filter (e.g. [SBS05]) can be used to tone it down so that results of segmentation are better. However segmentation of trees is a bigger problem since this noise is created by their movement. In such cases, the user can start by segmenting out stable parts like tree trunks and main branches, and then resort to manual point-grouping to segment out the noisy and ambiguous tree crown. Such tree segmentation is very useful in tree-modeling systems based on scanned data [XGC05].

There is some work remaining to be done in the segmentation of general point models. Though the graph cuts method is effective for such segmentation, there is some "parameter-tweaking" required to get desired results. A method that adapts based on the level of noise in the model, the precision of the model, etc. is highly desirable to free the user from counter-intuitive changing of parameters. The user can

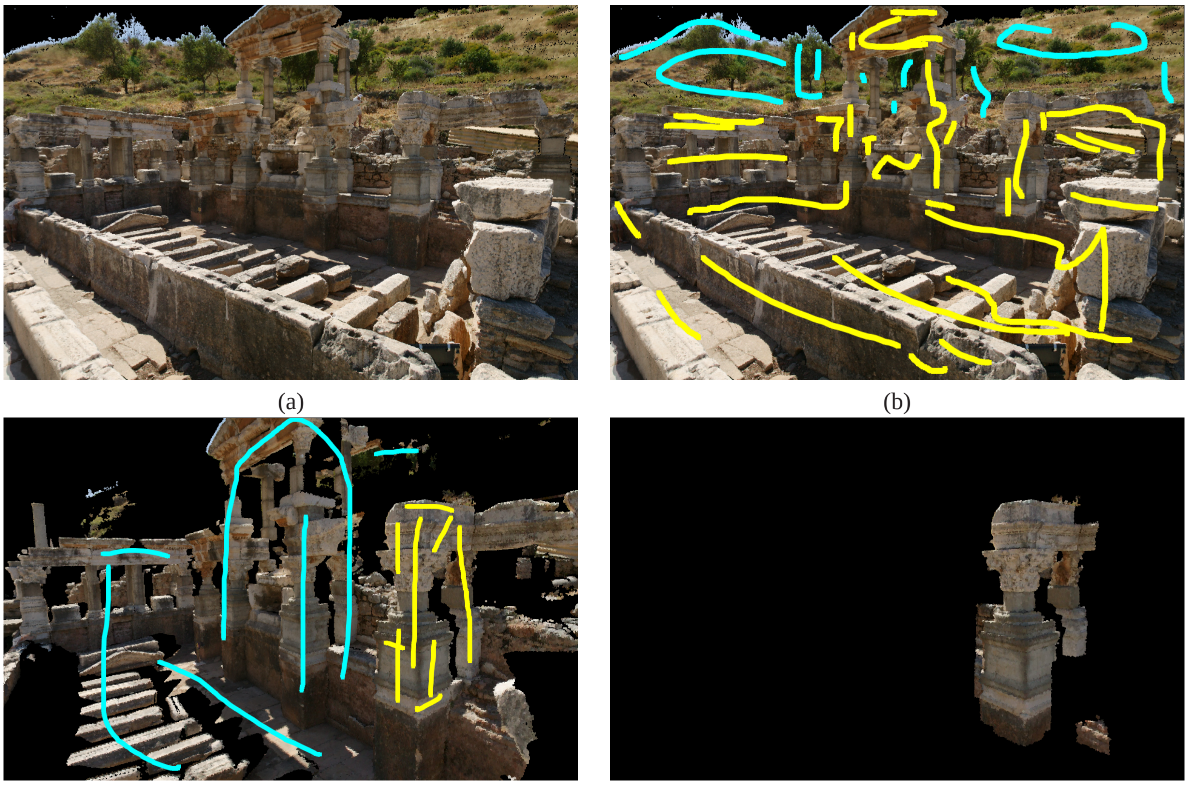


Figure 4: Segmenting out an object using our tool. (a) an input scan model. (b) user input strokes. (c) user works on the segmentation result of (b). (d) after second segmentation.

even avail of possible gestures to signal the fuzzier parts of a model (like roughly lassoing them) instead of moving swatches, so that the segmentation automatically adapts itself to be more robust in those parts.

The pen device can be used more effectively in such segmentation tasks. Instead of drawing strokes in background and foreground regions, the user could start tracing an actual object boundary, and the system could segment out the object just by such partial tracing of its boundary. This input is more intuitive for users that have some experience in lassoing in images.

Acknowledgments: Support for this work includes University of Minnesota Computer Science Department Start-up funds, University of Minnesota Digital Technology Center Seed Grants 2002-4, a Microsoft Gift, a Ted & Linda Johnson Donation, NSF ACI-0238486 (CAREER), and NSF EIA-0324864 (ITR). The dataset of the Ephesos excavation site is provided by Michael Wimmer, the Vienna University of Technology. We especially thank Yuri Boykov and Vladimir Kolmogoro for their help on graph cuts. We also thank the anonymous reviewers for helpful suggestions.

References

[AHSS04] Agarwala A., Hertzmann A., Salesin D. H., Seitz S. M.: Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graph. 23*, 3 (2004), 584–591.

[BJ01] Boykov Y., Jolly M.-P.: Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *International Conference on Computer Vision (ICCV)* (2001), vol. I, pp. 105–112.

[BK04] Boykov Y., Kolmogorov V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Tran. on Pattern Analysis and Machine Interlligence 26*, 9 (2004), 1124–1137.

[DMS02] Djebali M., Melkemi M., Sapidis N.: Range-image segmentation and model reconstruction based on a fit-and-merge strategy. In *SMA '02: Proceedings of the seventh ACM symposium on Solid modeling and applications* (New York, NY, USA, 2002), ACM Press, pp. 127–138.

[Fai90] Fairfield J.: Toboggan contrast enhancement for contrast segmentation. In *10th IEEE Interna-*

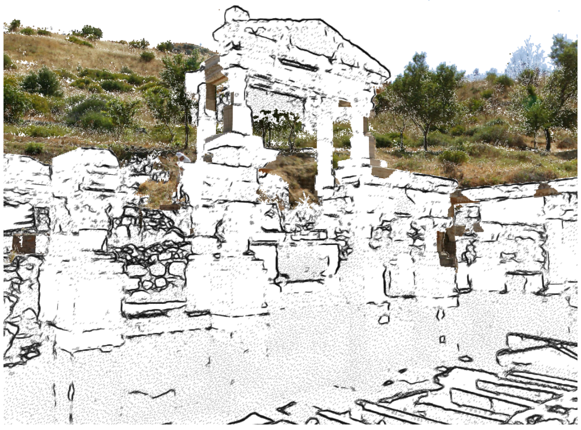


Figure 5: Illustration of use of segmentation. In this example, the user segments out a building and the PointWorks system is used to render it in a sketchy style. The background is rendered photorealistically. Segmentation of objects by our tool can be used to emphasize them in the final rendering as shown in this image.

tional Conference on Pattern Recognition (1990), vol. I, pp. 712–716.

[GPS89] GREIG D., PORTEOUS B., SEHEULT A.: Exact map estimation for binary images. *J. Roy. Stat. Soc. B. 51* (1989), 271–279.

[HR99] HEISELE B., RITTER W.: Segmentation of range and intensity image sequences by clustering. In *Proc. IEEE Conference on Information Intelligence and Systems* (1999), pp. 223–225.

[HSKK01] HILAGA M., SHINAGAWA Y., KOHMURA T., KUNII T. L.: Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 203–212.

[Inc02] INCORP. A. S.: Adobe photoshop user guide, 2002.

[LK01] LEYMARIE F. F., KIMIA B. B.: The shock scaffold for representing 3d shape. In *IWVF-4: Proceedings of the 4th International Workshop on Visual Form* (London, UK, 2001), Springer-Verlag, pp. 216–228.

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

[MB95] MORTENSEN E. N., BARRETT W. A.: Intelligent scissors for image composition. In *Computer Graphics (SIGGRAPH '95)* (Aug. 1995), pp. 191–198.

[MB98] MORTENSEN E. N., BARRETT W. A.: Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing 60*, 5 (Sept. 1998), 349–384.

[MB99] MORTENSEN E. N., BARRETT W. A.: Toboggan-based intelligent scissors with a four parameter edge model. In *in Proc. IEEE: Computer Vision and Pattern Recognition (CVPR '99), Vol. II* (June 1999), pp. 452–458.

[Mor99] MORTENSEN E. N.: Vision-assisted image editing. *Computer Graphics 33*, 4 (Nov. 1999), 55–57.

[MR99] MORTENSEN E. N., REESE L. J.: Intelligent selection tools. In *Computer Science Dept. Colloquium Brigham Young Univ.* (March 1999).

[Ree99] REESE L.: *Intelligent Paint: Region-Based Interactive Image Segmentation*. Master thesis, Department of Computer Science, Brigham Young University, Provo, UT, 1999.

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

[SBS05] SCHALL O., BELYAEV A., SEIDEL H.-P.: Robust filtering of noisy scattered point data.

[SKK01] SEBASTIAN T., KLEIN P., KIMIA B.: Recognition of shapes by editing shock graphs. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01)* (Los Alamitos, CA, July 9–12 2001), IEEE Computer Society, pp. 755–762.

[TA01] TAN K.-H., AHUJA N.: Selecting objects with freehand sketches. In *In Proceedings IEEE International Conference on Computer Vision* (2001), vol. 1, pp. 337–344.

[WBC*05] WANG J., BHAT P., COLBURN A., AGRAWALA M., COHEN M. F.: Interactive video cutout. *to appear in ACM SIGGRAPH* (2005).

[XC02] XU H., CHEN B.: Activepoints:acquisition, processing and navigation of large outdoor environments. *Technical Report, Computer Science Department, University of Minnesota TR.03.02* (2002).

[XGC04] XU H., GOSSETT N., CHEN B.: Pointworks: Abstraction and rendering of sparsely scanned outdoor environments. In *Rendering Techniques* (2004), pp. 45–52.

[XGC05] XU H., GOSSETT N., CHEN B.: Knowledge-based modeling of laser-scanned trees. *To appear in SIGGRAPH SKETCHES* (2005).

[YFM01] YU Y., FERENCZ A., MALIK J.: Extracting objects from range and radiance images. *IEEE Transactions on Visualization and Computer Graphics 7*, 4 (2001), 351–364.

[YZNC05] YUAN X., ZHANG N., NGUYEN M. X., CHEN B.: Volume cutout. *The Visual Computer (Special Issue for Pacific Graphics 2005)* (2005).