Scanline Surfacing: Building Separating Surfaces from Planar Contours

David Weinstein

Scientific Computing and Imaging Institute
School of Computing
University of Utah

Abstract

A standard way to segment medical imaging datasets is by tracing contours around regions of interest in parallel planar slices. Unfortunately, the standard methods for reconstructing three dimensional surfaces from those planar contours tend to be either complicated or not very robust. Furthermore, they fail to consistently mesh abutting structures which share portions of contours. In this paper we present a novel, straight-forward algorithm for accurately and automatically reconstructing surfaces from planar contours. Our algorithm is based on scanline rendering and *separating surface* extraction. By rendering the contours as distinctly colored polygons and reading back each rendered slice into a segmented volume, we reduce the complex problem of building a surface from planar contours to the much simpler problem of extracting separating surfaces from a classified volume. Our *scanline surfacing* algorithm robustly handles complex surface topologies such as bifurcations, embedded features, and abutting surfaces.

CR Categories: I.3.5 [Computer Graphics] Computational Geometry and Object Modeling—Curve, surface, solid and object representations

Keywords: Separating Surfaces, Planar Contours, Surface Construction, Scanline

1 Introduction

Computational models of biological structures are generally built from CT or MRI scans. The scan measurements are acquired on a regularly gridded volume of hexahedral voxels, and the data are stored as discrete values. From this discretized volume, a surface-based model can be constructed by identifying collections of voxels corresponding to features of interest. Such features can be as small as blood vessels in an angiogram, or as large as a kidney in an MRI scan. This process of labeling features of interest is termed *segmentation* or *classification*.

Volumetric datasets are typically segmented one slice at a time via manual algorithms, rather than as an entire volume all at once. That said, there have been some note-worthy semi-automatic volumetric segmentation algorithms. These algorithms have worked by growing segmented regions from seeds via level set methods [14], by evolving snakes through planar slices to bound segmented regions [8], and by various statistical methods [12, 24]. While these methods are of substantial utility on well-behaved datasets, they tend to be of limited use on noisy or artifact-laden datasets, or on datasets through which the feature sizes can shrink to sub-voxel sizes. As automatic segmentation is a fundamentally difficult research problem, there will always be a place for processing the output of manual segmentation tools.

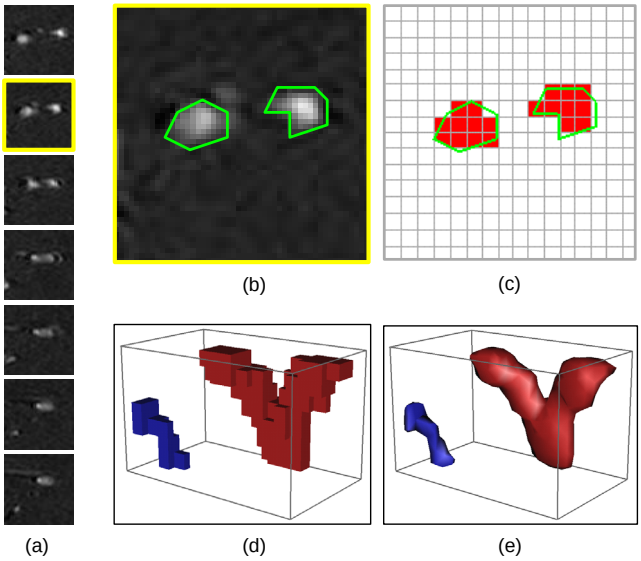


Figure 1: Stages of our scanline surfacing algorithm. From slices of raw data (a), an expert draws contours around features (b), which are scan converted into images (c). After stacking these images into a classified volume (d), separating surfaces are extracted and smoothed (e). (See also colorplate.)

The most common manual segmentation tools are contour drawing programs. An expert views planar slices of the volume (such as the angiography slices shown in Figure 1(a)) and draws polylines (*contours*) to outline regions of interest (such as the vessel contours shown in Figure 1(b)). The unique power of manual segmentation is that the expert can draw these curves despite noise, low-resolution data, signal fallout, and other pathologies which would confound automatic techniques.

As the expert outlines slice features, multiple features may appear in each slice, and individual features may appear more than once. To identify attached features between as well as within slices, the user typically selects a particular color or classification tag for each feature of interest. Figure 2 shows a single planar slice of a pelvic dataset. Note that the dark blue contours (also shown in colorplate Figure 10), which correspond to a single structure, the right pelvis, appear three times in this single slice. The user proceeds through all of the slices, consistently labeling regions of interest in each. At the end of this process, the user has created a set of planar contours uniquely identifying every feature of interest in the volume. The slice contours can then be aligned in 3D, as shown in

Email: dmw@cs.utah.edu
WWW: http://www.cs.utah.edu/∼dmw

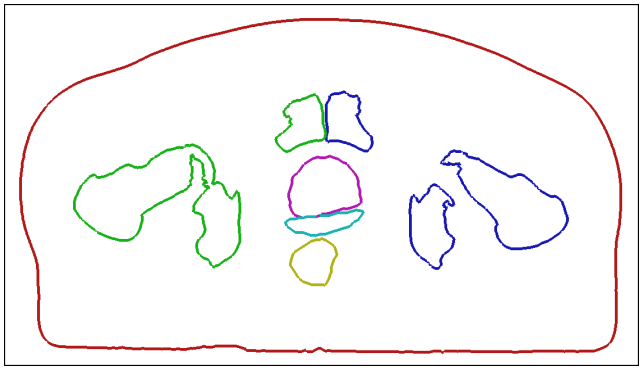


Figure 2: Hand drawn planar contours of pelvic structures. Skin (red), left pelvis (green), right pelvis (dark blue), rectum (yellow), prostrate (cyan), and sacrum (magenta) have been segmented.

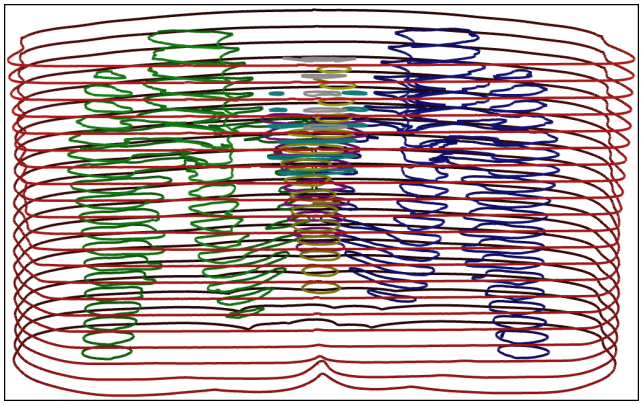


Figure 3: Planar contours outlining structures of interest are aligned in 3D. (See also colorplate.)

Figure 3.

While manual segmentation can take hours of user time for a dataset like that shown in Figure 3, constructing surfaces from those contours (such as those in Figure 4) rarely requires more than a minute of CPU time on modern architectures. As such, it is of limited benefit to produce faster algorithms for constructing surfaces from contours. Rather, what matters more is that the surfacing algorithm produces a *correct* result.

The traditional method for building triangulated surfaces from planar contours consists of solving three subproblems for each structure in the volume: correspondence, tiling, and branching [4, 6, 15, 21]. However, because these methods process the structures *serially*, rather than processing them *simultaneously*, they do not necessarily produce correct results for volumes in which surfaces abut.

Full volumetric models, such as torso models, cranial models, or pelvic models are often comprised largely of such abutting surfaces. The human body is *not* comprised of well-separated, free-floating surfaces, but rather of tightly packed organs which often conform to one another's shapes. The surfaces are merely the boundaries between structures within the volume. When these surfaces are constructed from a segmented volumetric dataset (such as a tetrahedral mesh or a voxelized volume), they are termed *separating surfaces* [18]. Separating surfaces can be smoothed and simplified without producing spurious gaps or surface interpenetrations [23].

2 Background

Constructing surfaces from planar contours has traditionally been decomposed into three subproblems: correspondence, tiling, and branching. Excellent reviews of these problems were presented by Sloan and Painter [21], as well as by Meyers [15]. We describe each subproblem briefly below.

The correspondence problem aims to determine which contours belong to the same structure. Simply stated: *Which contours in one slice are to be connected to which contours in neighboring slices?* The question of correspondence applies within a slice, as well as between slices. From the magnetic resonance angiography dataset shown with a maximum intensity projection in Figure 5, we extracted the indicated subvolume which intersects a pair of blood vessels. For each slice of the subvolume, we hand drew corresponding contours around the vasculature, as shown in Figure 6. All of the red contours correspond to one vessel, while all of the blue contours correspond to a different vessel. While the correspondence for this simple example is trivial, in general, algorithms for determining correspondence can be under-constrained. For example, imagine trying to determine correspondences for tomographic slices of a bowl of spaghetti. Algorithms for resolving correspondence when there is ambiguity resort to heuristics based on shape fitting [2] or graph analysis [20]. In contrast, if an unambiguous correspondence does exist, it can often be found by simply evaluating overlaps of contours in consecutive slices [22, 25].

The second subproblem in constructing surfaces from contours is the tiling problem. *What is the optimal triangle strip for joining corresponding contours from consecutive slices?* There has been much research targeted at solving this problem efficiently. The problem was reformulated by Keppel [11] as finding a path through a toroidal graph. Formalization of this method and efficiency improvements were subsequently introduced [3, 6, 21]. These methods vary in the heuristics used for optimizing the tiling, as well as in the complexity of their search. The simplest method is a greedy algorithm that marches through pairs of contours. For each step, it chooses whether to advance along the top contour or the bottom contour by picking the one which will contribute the shortest edge, and then building the resulting triangle. More complex global algorithms seek to minimize the surface area of the triangle strip, or to maximize the volume it encloses.

The simplest greedy methods are not robust against certain complexities (such as pathological concavities) and even the complex methods fail to handle abutting contours. Figures 7(a) and (b) depict a simple case for which the greedy heuristic has produced an

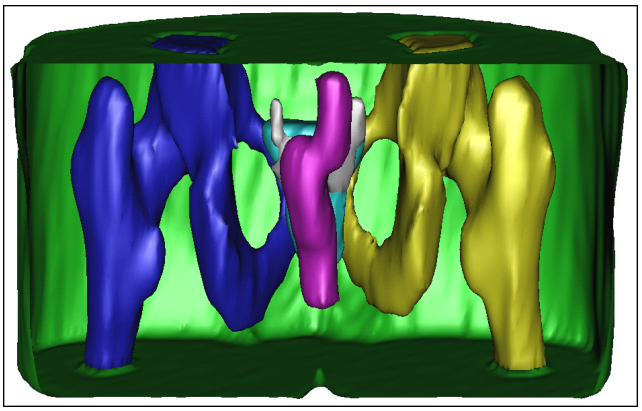


Figure 4: Reconstructed surfaces generated with our scanline surfacing algorithm. (See also colorplate.)

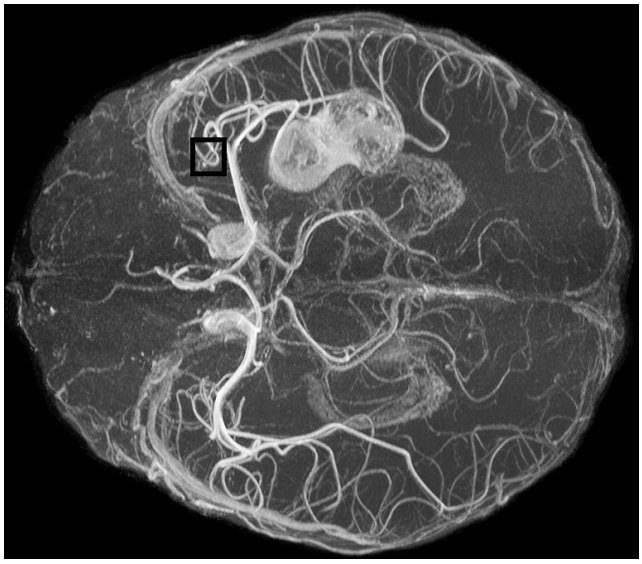


Figure 5: Maximum intensity projection of a masked, contrast enhanced magnetic resonance angiography dataset. The dark square indicates the subvolume which will be processed below.

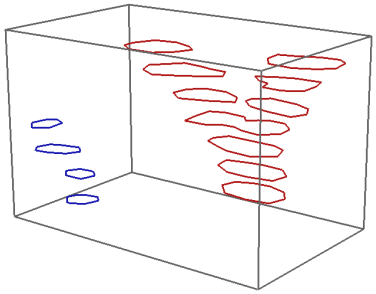


Figure 6: Planar contours of blood vessels from the angiography dataset shown above.

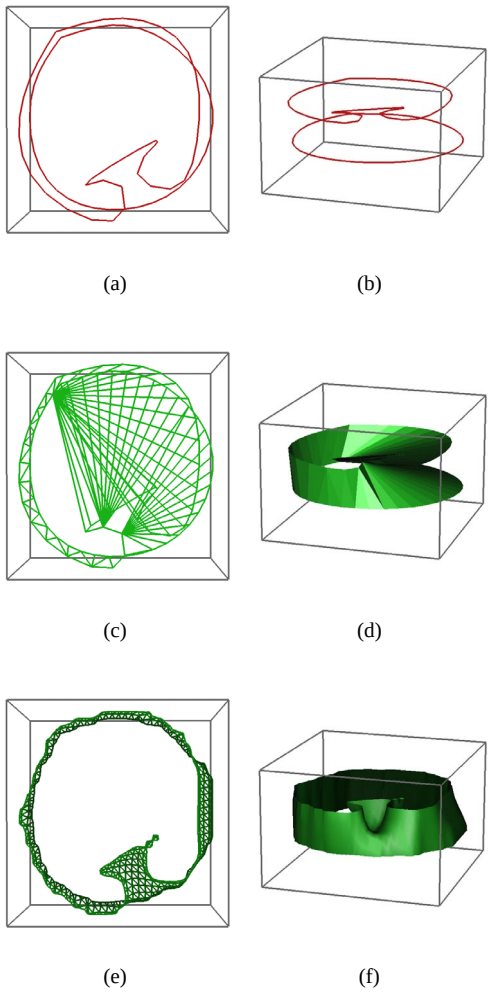


Figure 7: For the two simple contours shown in (a) and (b), a greedy tiling algorithm gets stuck in local concavities, producing the unacceptable tiling shown in (c) and (d). In contrast, our new scanline surfacing algorithm produces the reasonable tiling shown in (e) and (f).

unacceptable tiling: a self-intersecting surface composed largely of triangle fans, as seen in Figures 7(c) and (d).

A specific case of tiling that is typically handled separately from the methods described above is branching. *How should m contours in one slice be attached to the n corresponding contours in a consecutive slice?* Branching is typically handled through either a contour compositing scheme, or through Delaunay triangulation. In contour compositing, multiple contours are merged into a single contour [19]. This method requires user intervention to resolve complex cases. The Delaunay method uses Delaunay tetrahderalization to mesh the volume between the contours, after which only those triangles which span the two contours are preserved [1].

3 Methods

The traditional methods described above are useful for reconstructing unconstrained triangulated surfaces, with an independent surface for each structure. However, none of the above algorithms correctly handle contour sets with abutting structures. Since the traditional methods reconstruct each surface independently, the prostate (cyan) and sacrum (magenta) contours in Figure 2 will likely result in different tilings at their interface. Such discrepancies are problematic, as they can result in interpenetrations or gaps between the surfaces. An example interpenetration is shown for two abutting synthetic structures in Figure 8. Furthermore, if the contour points for the two abutting surfaces are not precisely coincident (*e.g.*,if contours were generated by uniformly resampling polylines with cubic b-splines), there would be no hope of producing consistent triangulations across boundaries. Ultimately, what we need is an algorithm that will recognize and correctly maintain abutting surfaces, and that will be capable of producing shared surface patches between them. Separating surface algorithms [17, 18, 23] offer just such a solution.

3.1 Algorithm Overview

In order to produce separating surfaces, we must first build a classified volume. That is, instead of building surfaces *directly* from contours, we take the novel approach of first *voxelizing* the con-

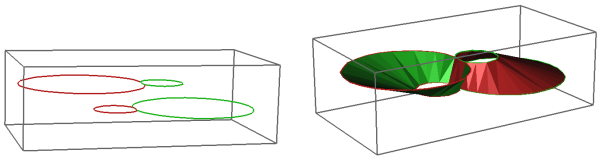


Figure 8: Interpenetration at an abutting interface. If adjacent structures do not have the exact same contour points or tilings along abutting regions, interpenetrations (as seen here) or gaps between the structures can result.

tours and then *extracting* separating surfaces from that voxelization. Our approach is illustrated with the angiography subvolume in Figure 1. For each slice of the volume (a), contours are hand-drawn around blood vessels of interest (b). These contours are then scan converted into low-resolution images (c), which are stacked into a classified volume (d). At this point, the original contours are said to be *voxelized*. Extracting separating surfaces from the classified volume, we recover approximations to the true surfaces, which are then smoothed, resulting in the final surfaces shown in (e).

In contrast to the unacceptable tiling produced by the traditional algorithm in Figures 7(c) and (d), our new scanline surfacing algorithm produces the reasonable output surface seen in Figures 7(e) and (f).

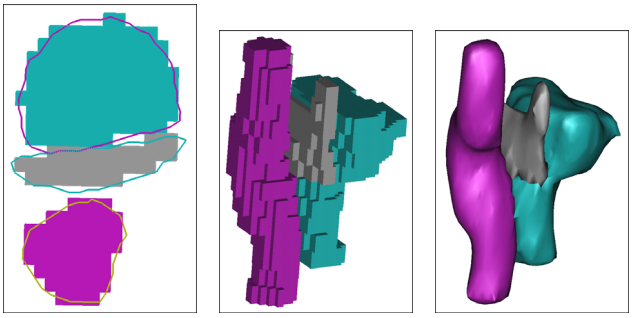
There has been much previous research done on designing fast and accurate voxelization algorithms for converting polygonal models into volumes [5, 7, 9, 10]. Those reports have focused on scan converting polyhedral or parametric models, and on developing methods with proofs of accuracy for those voxelizations. The work presented here is somewhat tangential. Rather than voxelizing polyhedral models to produce volumes, we are interested in voxelizing planar contours, and then from the voxelization extracting polyhedral models. Fortunately, voxelizing planar contours is very straight-forward and can be implemented using a standard graphics scan conversion algorithm. Because of this simplicity, the above voxelization research results are not required for our approach.

The premise for our scanline surfacing algorithm is that when building complex surfaces, it is much easier to scan convert the set of contours (as polygons) than it is to determine correspondences, tilings, and branchings.

3.2 Input

Instead of constructing the surface two slices at a time for each structure independently, as is done with tiling, our algorithm processes all of the contours on one slice at a time. For each slice, all of the contours on that slice are rendered into the frame buffer.

Our scanline surfacing algorithm takes as input a list of features that have been manually contoured. For each feature, there is a list of slices that that feature appears in, and for each of its slices, there is a list of all of the contours for that feature in that slice. Each contour is composed of a list of (x, y) coordinates in a consistent polyline ordering (*e.g.*,clockwise). The only user-chosen parameters for the algorithm are the number of pixels per scanline, nx, and the number of scanlines, ny. To store the classified volume, our algorithm allocates an $(nx \times ny \times nz)$ array of voxels, where nz is two greater than the number of input slices. (The outermost voxels of the volume function as the layer of padding needed for producing capped surface models.)



(a) (b) (c)

Figure 9: Sacrum (blue), prostrate (gray), and rectum (magenta). The contours of each feature are scan converted into image slices (a). These slices are then stacked into a volume (b), and the features are extracted and smoothed as separating surfaces (c).

3.3 Initialization

In order to retain classification information in the volume, the algorithm assigns classification numbers to each of the features before rendering them. The features are numbered from 1 to n, where features with higher indices correspond to features which are *outside*. One contour is considered outside of another if all of the inner contour's points are to the right as one traverses the outer contour in clockwise order. The sorting order of the features can either be provided by the user, or can be determined using quicksort with the inner/outer comparison operator described above. This sorting produces a "nesting" that is guaranteed to be acyclic. Features contained entirely inside of other features (such as ribs inside of the skin, or blood inside of the veins) have lower indices. We note that sibling relationships (features which share external boundaries, such as the prostate and sacrum) require no particular ordering.

The final steps of our setup stage are to allocate an image buffer with $(nx \times ny)$ pixels, and to define an orthographic projection oriented with the image plane parallel to the slice and such that the minimum and maximum corners of the contour set's bounding box will project to the centers of pixels $(1, 1)$ and $(nx - 1, ny - 1)$, respectively.

3.4 Slice Scan Conversion

Our scanline surfacing algorithm was implemented using the OpenGL graphics pipeline for scan conversion. Each slice of contours is transformed into a slice of our classified volume by rendering the contours as closed polygons and then reading the image buffer into a volume slice. Contours and their respective scan conversions are shown for a subset of the pelvic volume in Figure 9(a), and for the angiography subvolume in Figure 1(c).

To render each slice, our algorithm clears the image buffer and then scan converts the contours of that slice. Rather than rendering the contours in out-to-in order, our method renders each contour with its sorted index as its "color" value as well as its z-value, and relies on the z-buffer for sorting. After all of the contours for a slice have been rendered, the image buffer is copied into the classified volume.

The contours from the slice shown in Figure 2 have been scan converted, resulting in the fully segmented slice shown in Figure 10. (The contours have been super-imposed in distinct colors for reference.)

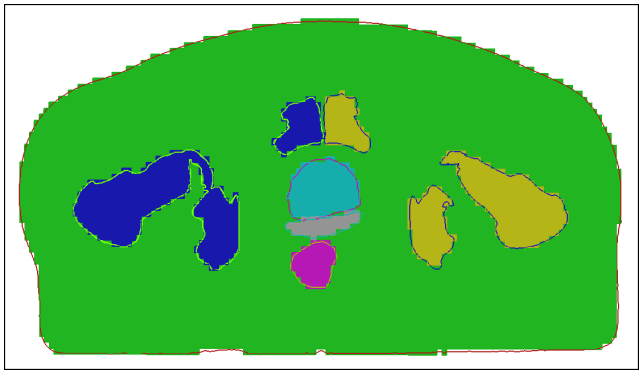


Figure 10: Scanline fill of all of the contours for a slice of the full pelvic models. Contours and filled regions are shown in different colors for visual contrast. (See also colorplate.)

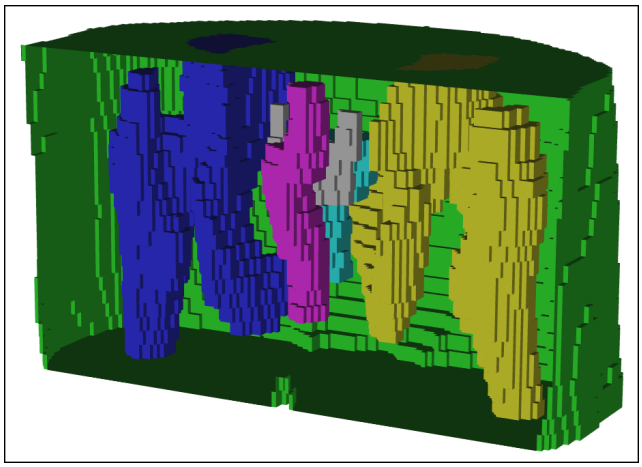


Figure 11: The scan converted images are stacked together, producing a classified volume of the pelvic dataset. (See also colorplate.)

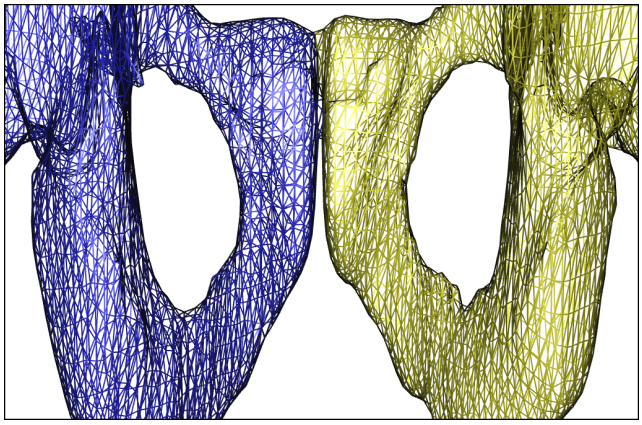


Figure 12: Close-up wireframe view of the smoothed pelvic surface. (See also colorplate.)

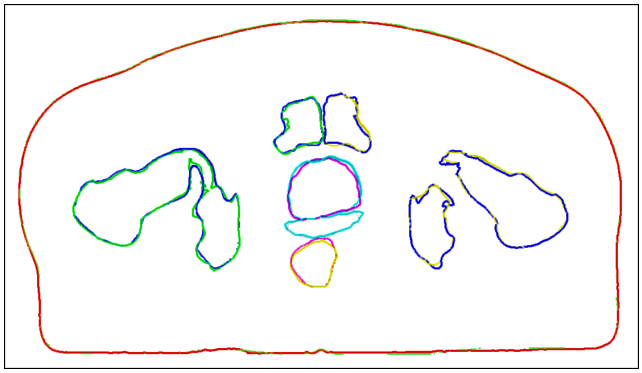


Figure 13: A cross-section of the final smoothed separating surface superimposed on the original planar contours.

This process is repeated for all of the slices, resulting in the segmented volume shown in Figure 11. The 21 slice pelvic volume was reconstructed at 64×128 resolution in 1.78 seconds using an SGI O2 with a 300 MHz R5000 processor and CRM graphics. This timing included contour sorting, scan conversion, and reading back the frame buffer. As mentioned above, the surface generation time is essentially negligible when compared to the user-intensive time required to generate the initial planar contours.

3.5 Separating Surface Extraction and Smoothing

Having voxelized the planar contours, our algorithm must now extract the separating surfaces from the segmented volume. For volumes consisting of only two levels of classification (background material and non-abutting foreground features), an isosurface extraction algorithm such as Marching Cubes [13] can be used. However, for more complex volumes in which features can be nested or can abut, a surfacing algorithm designed for classified volumes is required. An algorithm specifically designed for separating surface extraction was developed by Müller [17] and independently by Nielson and Franke [18]. This algorithm builds triangulated faces between homogeneous regions. As a preprocess, their algorithm transforms hexahedral voxels into tetrahedra. It then marches through each tetrahedron, building triangles for the separating surfaces by evaluating the classifications of the tetrahedron's nodes against a mask and a case table. Using this method, our algorithm extracts separating surfaces from the classified angiography subvolume in Figure 1(a); the sacrum, prostate and rectum in Figure 9(b); and the full pelvis in Figure 11.

As a final step, our algorithm employs a constrained separating surface smoothing algorithm [23] that relaxes the vertices with respect to their neighbors. Conceptually, an ellipsoid is centered at the original position of each vertex, such that the ellipsoids are packed together but do not overlap. The motion of each vertex as it is smoothed is constrained to remain within its ellipsoid. This constraint prevents nodes from crossing and thus prevents surfaces from interpenetrating. Smoothing the surfaces to best conform to the original contours remains a topic for future research. The final smoothed separating surfaces for the pelvis and blood vessels are shown in Figures 4, 9(c), and 1(e). Figure 12 shows a close-up of the triangulation of the pelvic bones in order to reveal the smooth, regular structure of the triangulation. The deviation between the original contours and cross-sections of the separating surfaces are due to two factors. First, the new vertices are located at the centers and edges of the scan converted pixels, and not at the locations of the original contour vertices. Second, the vertex positions move in order to produce a smoother surface. Thus, as the scanline resolution increases, the disparity between the original contours and

cross-sections of the separating surfaces diminishes. A typical result is shown in Figure 13.

4 Discussion and Conclusion

By voxelizing the planar contours, we have gained straight-forward solutions to the traditional surfaces-from-contours subproblems of correspondence, tiling, and branching that were discussed in Section 2. Correspondence and branching are implicit in the connected components of the voxelization, and the tiling is produced by the separating surface extraction. We note that while these solutions are straight-forward, they do not always produce the desired result. Specifically, if corresponding features do not overlap in their scan conversions between consecutive slices, they will not be tiled together, and they may not even be identified as corresponding. In short, this algorithm is not robust against datasets which are exceedingly under-sampled between slices. Extensions to this algorithm which will generate interpolated z-slices is a topic for future research.

We also note that whereas traditional algorithms produce interpenetrating surfaces when presented with overlapping contours, our algorithm does not. This property is a direct result of using separating surfaces. While this is certainly a desirable characteristic for many applications, it is possible that it could be undesirable for others. In general, we believe that our method has utility beyond medical imaging, and it is a topic for future work to identify and evaluate other appropriate applications.

Unlike traditional methods, our algorithm does not preserve the original contour vertices or edges in the final surface. The original vertices are replaced with new approximating vertices located at the centers and edges of the classified voxels. By choosing a small enough pixel size (large enough nx and ny parameters), we can produce vertices that come asymptotically close to the original contour vertices. More important than duplicating the original vertices, though, is the ability to replicate features of the original contours. This can also be controlled by varying nx and ny, and in the case of feature reproduction there is a bound on how large nx and ny must be. Specifically, the upper-bound on a pixel's dimension is the diameter of the largest alpha-shape that can trace the minimal feature of interest [16]. As we reduce the scan conversion pixel resolution in order to capture smaller features, our algorithm produces ever finer triangulations throughout the entire model. We note that regions which are tessellated too finely can be decimated with the constrained simplification method described in [23].

In conclusion, we have presented a fresh approach to solving a classic problem. By combining a standard scanline algorithm from computer graphics with separating surface methods, we have built a novel technique for efficiently generating robust surfaces from arbitrarily complex planar contour sets. Using this method, we have been able to automatically construct smooth, non-interpenetrating surfaces from sets of complex planar contours.

5 Acknowledgments

This work was supported in part by awards from the National Institutes of Health, and the National Science Foundation. The author would like to thank Gordon Kindlmann, Chris Johnson, Laura Traynor, and Helen Hu for their valuable comments and suggestions. The author would also like to extend special thanks to the reviewers for their constructive criticism and meticulous edits. The pelvic and angiography datasets were generously provided by Ismail Khalil Bustany from the Radiology Department at UCSF, and Dennis Parker from the CAMT at the University of Utah, respectively.

References

[1] Jean-Daniel Boissonnat. Shape reconstruction from planar cross sections. *Computer Vision, Graphics, and Image Processing*, 44(1):1–29, October 1988.

[2] C.A. Brebbia and J. Dominguez. *Boundary Elements: An Introductory Course*. McGraw-Hill, Boston, 1989.

[3] Larry T. Cook, Samuel J. Dwyer, III, Solomon Batnitzky, and Kyo Rak Lee. A three-dimensional display system for diagnostic imaging applications. *IEEE Computer Graphics and Applications*, 3(5):13–19, August 1983.

[4] A. B. Ekoule, F. C. Peyrin, and C. L. Odet. A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Transactions on Graphics*, 10(2):182–199, 1991.

[5] V. Filippov and R. Yagel. Accurate methods for the voxelization of planar objects. Technical report, Ohio State University, 1997. OSU-CISRC-2/97-TR08.

[6] H. Fuchs, Z. M. Kedem, and S. P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, October 1977.

[7] Jian Huang, Roni Yagel, Vassily Filippov, and Yair Kurzion. An accurate method for voxelizing polygon meshes. In *IEEE Symposium on Volume Visualization*, pages 119–126. IEEE, ACM SIGGRAPH, 1998.

[8] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

[9] Arie Kaufman. An algorithm for 3d scan-conversion of polygons. In G. Marechal, editor, *Eurographics '87*, pages 197–208. North-Holland, August 1987.

[10] Arie Kaufman. Efficient algorithms for 3D scan-conversion of parametric curves, surfaces, and volumes. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, pages 171–179, July 1987.

[11] E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM Journal of Research and Development*, 19(1):2–11, January 1975.

[12] D.H. Laidlaw. *Geometric Model Extraction form Magnetic Resonance Volume Data*. PhD thesis, California Institute of Technology, 1995.

[13] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[14] R. Malladi and J. A. Sethian. Level set methods for curvature flow, image enhancement, and shape recovery in medical images. In H.-C. Hege and K. Polthier, editors, *Proc. of Conf. on Visualization and Mathematics*. Springer-Verlag, Heidelberg, June 1995.

[15] David Meyers, Shelley Skinner, and Kenneth Sloan. Surfaces from contours. *ACM Transactions on Graphics*, 11(3):228–258, July 1992.

[16] P. J. Moran and M. Wagner. Introducing alpha shapes for the analysis of path integral monte carlo results. In R. Daniel Bergeron and Arie E. Kaufman, editors, *Proceedings of the Conference on Visualization*, pages 52–60, Los Alamitos, CA, USA, October 1994. IEEE Computer Society Press.

[17] Heinrich Müller. Boundary extraction for rasterized motion planning. Technical Report 566, University of Dortmund, January 1995. published in: H. Bunke, T. Kanade, H. Noltemeier, Modelling and Planning for Sensor Based Intelligent Robot Systems, World Scientific Publ. Co., 1995.

[18] G. M. Nielson and R. Franke. Computing the separating surface of segmented data. In *Visualization '97*, pages 229–233. IEEE Press, 1997.

[19] M. Shantz. Surface definition for branching, contour-defined objects. *Computer Graphics*, 15(3):242–259, August 1981.

[20] Shelley Marie Skinner. The correspondence problem: Reconstruction of objects from contours in parallel sections. Master's thesis, University of Washington, 1991.

[21] Kenneth R. Sloan and James Painter. Pessimal guesses may be optimal: A counterintuitive search result. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-10(6):949–955, November 1988.

[22] Y. F. Wang and J. K. Aggarwal. Surface reconstruction and representation of 3-D scenes. *Pattern Recognition*, 19:197–207, 1986.

[23] D.M. Weinstein. Separating surfaces: Extraction, smoothing and simplification. *University of Utah Technical Report*, UUCS-00-009, 2000.

[24] W.M. Wells, W.E.L. Grimson, R. Kikinis, and F.A. Jolesz. Statistical intensity correction and segmentation of MRI data. In *Visualization in Biomedical Computing*, pages 13–24, 1994.

[25] Michael J. Zyda, Allan R. Jones, and Patrick G. Hogan. Surface construction from planar contours. *Computers and Graphics*, 11(4):393–408, 1987.