Semi-Automatic Topology Independent Contour-Based 2 ½ D Segmentation Using Live-Wire

Michael Knapp
Vienna University of Technology
Computer Graphics Group
Favoritenstrasse 9-11/E186
1040 Wien, Austria
knapp@cg.tuwien.ac.at

Armin Kanitsar
Vienna University of Technology
Computer Graphics Group
Favoritenstrasse 9-11/E186
1040 Wien, Austria
kanitsar@cg.tuwien.ac.at

Meister Eduard Gröller
Vienna University of Technology
Computer Graphics Group
Favoritenstrasse 9-11/E186
1040 Wien, Austria
groeller@cg.tuwien.ac.at

ABSTRACT

In general three-dimensional segmentation algorithms assume objects to have connected homogeneous regions. However in some cases objects are defined by a fuzzy boundary surface and consist of an inhomogeneous internal structure. In the following a new three-dimensional segmentation technique exploiting the contour detection capabilities of live-wire is proposed. The algorithm consists of two basic steps. First contours are outlined by the user on a small number of planar cross-sections through the object using live-wire. Second the traced contours are used for reconstructing the object surface automatically in each slice using live-wire again. This user-friendly segmentation algorithm is independent from object topology as the topology is implicitly defined during the reconstruction process.

Keywords
Segmentation, Live-wire.

1. INTRODUCTION

Segmentation of objects from volumetric data is an important prerequisite for visualization and still a hot topic in medical investigations. Bony structures in medical datasets have rather steady boundaries but consist of inhomogeneous internal structures. The boundary may have low gradient areas, which make a complete segmentation nearly impossible with region growing. Separating bones from other structures by thresholding is often not applicable. Due to overlapping intensity regions other important features (e.g. contrast enhanced arteries) are affected by this operation too. Especially in the field of *computed tomography angiography* (CTA) the extraction of bones improves the quality of the results dramatically [Kan01].

Journal of WSCG, Vol.12, No.1-3, ISSN 1213-6972
WSCG'2004, February 2-6, 2003, Plzen, Czech Republic.

In Section 2 existing segmentation methods are reviewed. The new method is described in Section 3 and Section 4. The results are presented in Section 5. Section 6 concludes the work.

2. RELATED WORK

Many segmentation methods have been developed for specific structures in medical datasets. Examples for segmenting homogeneous structures are seeded region growing [Ada94] and the water-shed algorithm [Beu92]. The manual boundary tracing of two-dimensional structures is very time consuming. The following acceleration techniques have been developed: Snakes is an active contour model introduced in 1988 by Kass et al. [Kas88]. The model minimizes an energy function based on internal forces (contour curvature) and external forces (image data). Initially the user sketches a contour and the object boundary is approximated automatically by minimizing the energy function. Another approach, known as live-wire [Mor92] or intelligent scissors [Mor98], was introduced in 1992 by Mortensen et al. and Udupa et al. [Udu92]. The live-wire process is interactively steered by the user who can react immediately to automatically suggested contours. The contour represents a minimal-cost path between a

start and an end point. The cost-function is derived from the underlying image.

A straightforward extension of live-wire to three-dimensional objects would be to apply the technique to each volume slice. This requires considerable user interaction and is rather time-consuming especially as the number of slices is ever increasing with improved imaging modalities. One approach for a more efficient three-dimensional extension is the application of the algorithm on a subset of slices. The object boundary is then reconstructed by propagating the contours through the neighboring slices. This can be done by shape-based interpolation [Ray90]. The user traces contours in every *n-th* slice. Distance fields are generated from these contours. These distance fields are interpolated linearly for intermediate slices and contours are constructed from the interpolated distance fields. Another approach for three-dimensional segmentation using live-wire has been proposed by Falcão et al. [Fal00]. The slices of the volume to be segmented are separated into slabs, in which the topology of the object to segment must not change. In each slab the user defines a set of cross-sections, which are perpendicular to the slices. These cross-sections must intersect in a common line. The user traces the object boundary using live-wire in each cross-section of each slab. The resulting outlines of the traced contours are used to reconstruct the object boundary in each slice. This approach requires extensive user interaction, which increases significantly with the complexity of the object to segment. Separating the slices into slabs in which the object topology does not change may even result in dozens of slabs. Another limitation is, that all cross-sections must have a common intersection line.

The proposed new technique targets these two limitations: The arrangement of the orthogonal cross-sections is arbitrary. Furthermore the user is not required to cope with the object topology, as the algorithm itself is not limited by topological constraints.

3. OUR SEGMENTATION METHOD

The basic idea of our technique is a two level application of live-wire: Initially the user chooses an object to segment and defines a set of cross-sections perpendicular to the slices in a way that they intersect the desired object. The object boundary in each cross-section is traced using live-wire. In cases where the cross-section is intersected by another cross-section, the previously calculated contour contributes connectivity points for the currently processed contour (see Figure 1). The set of orthogonal cross-sections and their contours provide vertical connectivity information about the desired object. This information is used to reconstruct the contours in each slice automatically using live-wire again (see Figure 2).

Live-Wire

Live-wire is a method for tracing contours in a 2D gray level image from a fixed starting point to an

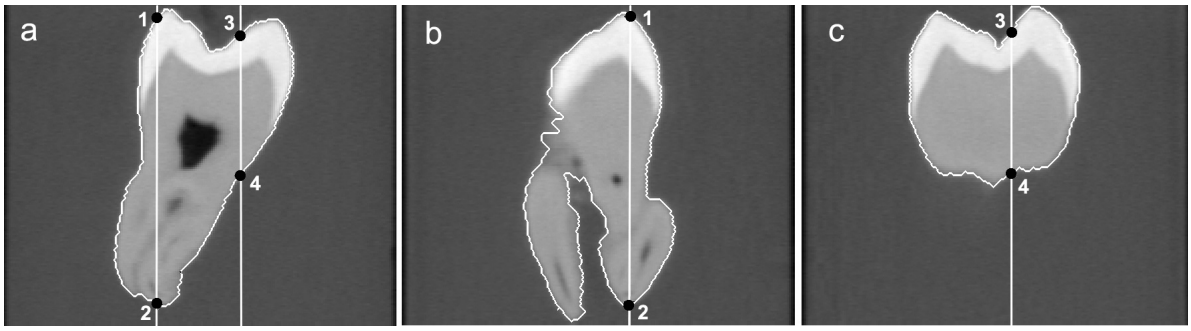

Figure 1. The user traces the object boundary of the tooth dataset in each orthogonal cross-section. The connectivity points are depicted by black dots.

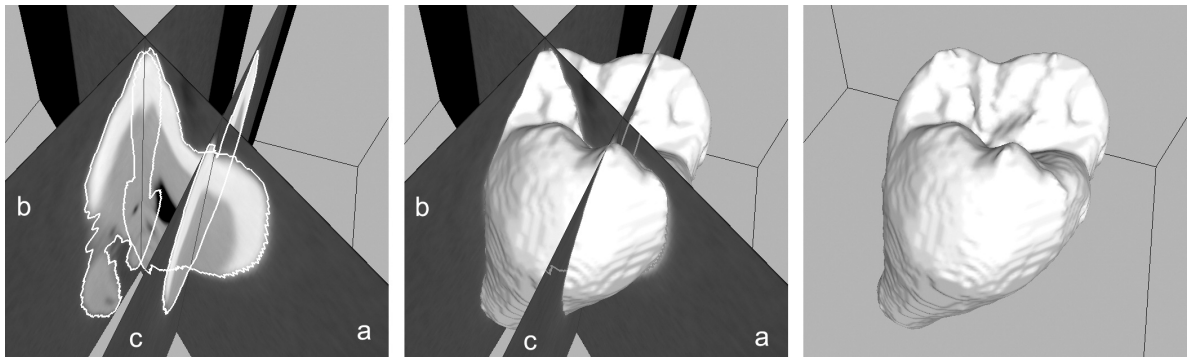

Figure 2. Three-dimensional display of the cross-sections with outlines (left), with the reconstructed surface (middle) and the reconstructed surface (right).

interactively moveable end point. The contour is calculated by applying the single-source shortest-path algorithm on an undirected weighted graph generated from the image. The graph is generated as follows: Each pixel is interpreted as a node which is connected by edges to each of its eight neighboring nodes (horizontal, vertical and diagonal neighboring pixels). The weight of each edge depends on the gradient. Low gradient magnitude means high cost and high gradient magnitude means low costs. Additionally scale space zero-crossing information is included in the weight calculation.

Assumptions

The described method is based on the following assumptions:

The three-dimensional volume is defined by three orthogonal axes denoted by *x*, *y* and *z*. Slices and cross-sections are planar sections through the volume, which are bounded by the volume borders. *Slices* are perpendicular to the z-axis whereas *cross-section*s are parallel to the z-axis.

An object is a structure, which the user wants to segment. An object should not consist of more than one component. Each object has a *boundary surface*. The intersection of the boundary surface with a cross-section results in a *contour* on the cross-section. A traced contour is referred to as an *outline*. A cross-section may contain more than one outline. The outline must not cross itself or other outlines in the same cross-section. Each outline must be closed (i.e. being topologically equivalent to a circle). In general intersecting an outline with a slice produces an even number of intersection points (*outline points*). Tangent points are a special case, which can be handled easily by counting them twice. The user-interface ensures that these requirements are fulfilled. The user is not required to cope with these requirements.

Segmentation Procedure

The segmentation procedure consists of the following four steps (see Figure 8, left side):

Set of orthogonal cross-sections is defined. For segmenting a chosen object, an arbitrary number of orthogonal cross-sections intersecting the object are selected. Typically the user will select at least two up to a dozen representative cross-sections.

The boundary of the desired object is traced. The user traces the boundary of the object in each cross-section. Outlines of two different cross-sections may have points in common. These points are located on the joint intersection line of the cross-sections. They are called connectivity points (see Figure 3). Connectivity points due to previously processed cross-sections guide the user in tracing the outline in the current cross-section.

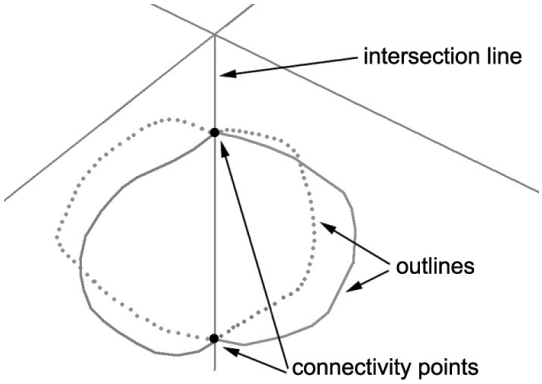

Figure 3: Outlines of two cross-sections may have points in common. These connectivity points are on the intersection line of the two cross-sections

Automatic boundary surface reconstruction. After the object boundary in all cross-sections has been outlined, an automatic boundary surface reconstruction algorithm is done. The algorithm takes the outlines and the spatial positions of the cross-sectional planes as input. Successively for each slice the outlines are computed by connecting the outline-points in each slice using live-wire in an automatic fashion. The output of the algorithm is a set of outlines in each slice. If the result of the algorithm is not satisfying, cross-sections may be deleted or added and the process is repeated from step 2. This may happen in case of an object of high complexity where it is not always easy for the user to find a representative set of cross-sections for reconstructing the object surface. The automatic boundary surface reconstruction is described in detail in the following section.

Visualization. From the stack of outlines a binary mask is generated. For visualization purposes the binary mask is transformed into a binary dataset. After a data-smoothing step an iso-surface is extracted and rendered.

4. AUTOMATIC BOUNDARY SURFACE RECONSTRUCTION

A detailed description of the automatic boundary surface reconstruction within each axial slice is presented in this section. A flow chart of this process is shown in Figure 8 in the gray box on the right.

Connectivity Graph

For each slice a so-called connectivity graph is created. This graph is needed to connect the outline points in a correct order to form a slice outline. To determine the correct order the graph is traversed along the edges starting from a randomly selected

vertex. The topology of the graph is determined in the following way: A slice is intersected by all cross-sections, which split up the slice into a set of convex polygons. Then a binary space partition (BSP) tree is created based on the cross-sections. A cross-section divides the slice along an intersection line into two halves, which are split up recursively by further cross-sections. During the tree creation the neighborhood information of the resulting polygons is tracked within the tree. A leaf node represents a polygon and a branch node an intersection line. For each polygon the adjacent line fragments, which belong to its boundary, are stored in a list.

Since an outline in an orthogonal cross-section created by the live-wire algorithm is closed it separates the cross-section into disconnected regions, which can be classified as inside or outside. An axial slice intersects an orthogonal cross-section in a horizontal line and also the classified regions. Therefore this line contains sections also classified as inside or outside. The first and the last point of each inside section are taken as outline points. Both points belong to the outline in the cross-section.

Based on this information and the binary space partition tree a connectivity graph is build up (see Figure 4): Each polygon in the axial slice is represented by a vertex in the connectivity graph. The outline points described in the previous paragraph are represented by an edge in the connectivity graph. This edge describes a possible transition between two neighboring polygons. In the contour reconstruction step the computed slice outlines are only permitted to cross the polygon boundaries at these outline points.

In cases, where an outline runs along an intersection line several crossings may occur, generating many connectivity points. The user may not be able to trace the contour exactly, which results in a graph containing vertices with an odd number of edges (see Figure 5). As a valid graph must contain only vertices with an even number of adjacent edges, such invalid cases have to be detected: Since an outline in a cross-section is closed, a unique classification in an inside and outside area is possible. An intersection point between two intersection lines

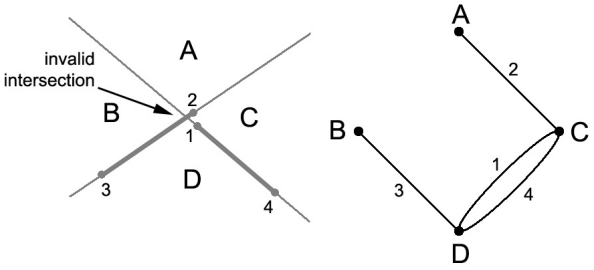

Figure 5. This figure shows a case where an *invalid* intersection occurs (left) and its according connectivity graph (right). Thick lines mean *inside*. Two intersecting cross-sections, where an *inside* line fragment intersects an *outside* fragment.

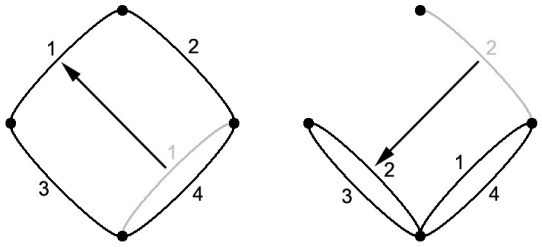

Figure 6. According to Figure 5 two different modifications are possible, depending on the outline point closer to the intersection point. This modification turns an *invalid* case into a *valid* case. Point 1 is closer to the invalid intersection point (left). Point 2 is closer to the intersection point (right).

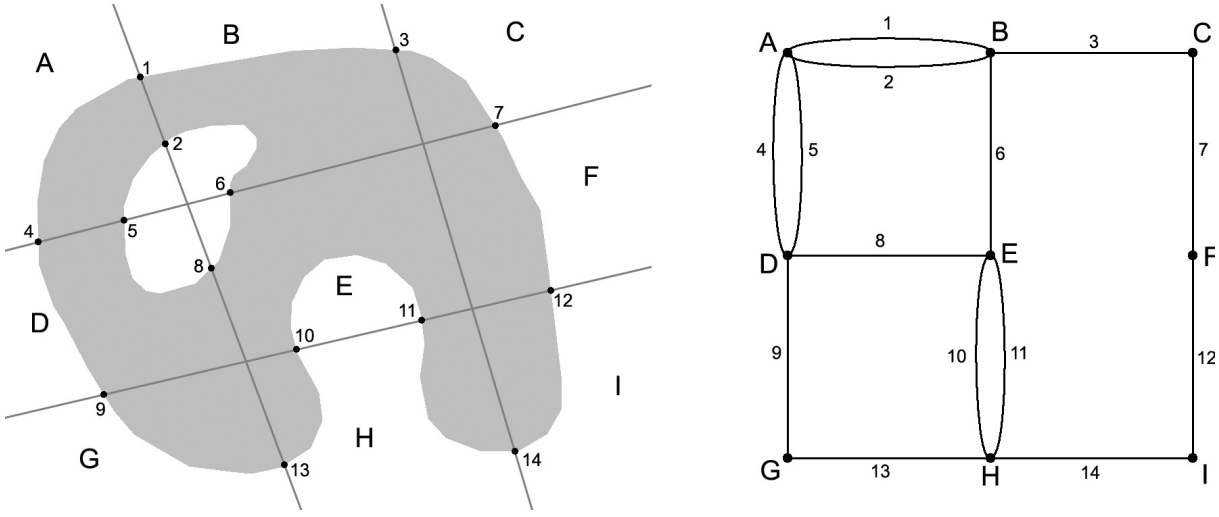

Figure 4. A slice is split into a number of convex polygons by intersecting cross-sections (left) and its connectivity graph (right). Letters indicate polygons and numbers indicate outline points

is valid if inside intersects inside, or outside intersects outside otherwise this intersection point is marked as invalid (see Figure 5).

The process described below is applied until the graph contains only vertices with an even number of adjacent edges. Every invalid intersection point, which was marked in the previous step, is handled in the following way: The outline point, which is closest to this intersection point, is chosen. This outline point is represented by an edge in the connectivity graph. This edge is changed in a way that it connects the vertices, which represent the adjacent polygons across the intersection line (see Figure 6). This computation step results in a topologically consistent connectivity graph. This is an essential prerequisite for the contour reconstruction step.

Contour Reconstruction

The goal of contour reconstruction is to find a set of closed outlines in each slice. Since the connectivity graph contains only vertices with an even number of adjacent edges it is always possible to find these closed outlines (Euler circuits).

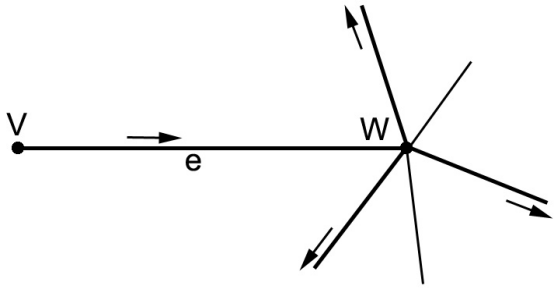

Figure 7. This figure shows the possible succeeding edges indicated by arrows pointing away from *W*.

Before the outlines in a slice are reconstructed using the connectivity graph, the adjacent edges of each vertex have to be sorted. Since the vertices represent convex polygons the center point of the polygon is always inside the polygon. Therefore the

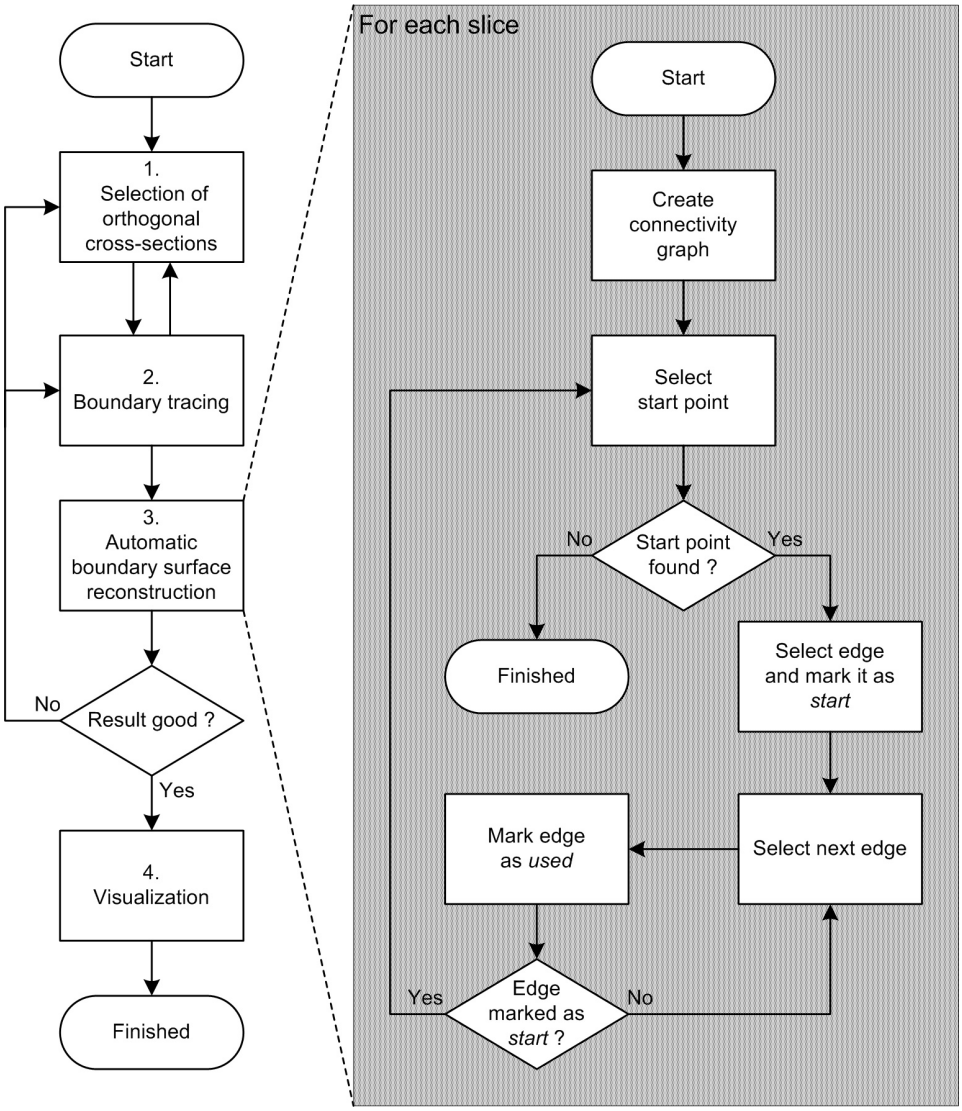

Figure 8: The flow chart of the segmentation process described in section 3 (left side) and 4 (right side).

edges, which represent outline points located on the border of the polygon, can be sorted according to the absolute angle around the center point. Additionally every edge has a flag, which indicates whether an edge is *used* or *unused*, starting edges are marked as *start*. In the following paragraph the connectivity graph traversal for the slice outline reconstruction is described.

For the starting point a vertex containing unused edges is selected randomly. Then an unused edge is taken and marked as start. The variable *V* is set to this vertex (see Figure 7). The succeeding vertex *W* of a vertex *V* is connected by an edge *e*, which represents an outline point. As depicted in Figure 7 every second edge starting from the first edge after edge *e* either marked as unused or start is a possible successor, which is weighted as follows: A live-wire path is generated from the outline point represented by *e* to the outline point represented by each permissible succeeding edge. The average cost per pixel along the previously generated live-wire paths as plausibility metric turned out to generate good results. The path with the lowest cost per pixel is taken as successor because a low cost path usually runs along object boundaries. The cost map used for the live-wire algorithm is based on the gradient-magnitude of the density data filtered with a 3x3x3 Gaussian kernel. Before filtering a windowing function is applied, where values below a given threshold T_0 are mapped to 0, and values above a given threshold T_1, $T_1 > T_0$ are mapped to the maximum possible value. T_0 is the minimum gray value and T_1 the maximum gray value of the whole dataset.

In addition to that, the cost map for the live-wire algorithm is initialized in a way that the area outside of the convex polygon, which is represented by the currently investigated vertex, is weighted with such a high value so that an outline will hardly ever cross the polygon boundary. After the succeeding outline point has been determined, its representing edge is marked as used. If this edge was marked as start before, then the current outline is closed and a new outline is started. Otherwise the variable *V* is set to the current vertex *W* and the algorithm is continued with the determination of the next successor.

5. RESULTS

This section describes the application of the method on two datasets: i.e. a tooth (256x256x161, 16 bit per voxel) and legs (512x512x553, 16 bit per voxel)

The timings are measured on a PC-based workstation: AMD Athlon 1.2GHz (133MHz external bus clock) with 768MB PC133 SDRAM. The sample implementation has been developed using Microsoft Visual C++ 6.0.

In the implementation the calculation time of the automatic surface reconstruction algorithm itself was 20 seconds for the tooth dataset, and 300 seconds for the legs dataset.

As shown in Figure 1 and Figure 2 the tooth can be segmented satisfactory just using three cross-sections. Figure 9 demonstrates the high accuracy of the presented technique in comparison to region growing and thresholding. Bones have an inhomogeneous structure especially at the joints. In this area region growing and thresholding do not generate useful results.

Binary masks generated from the segmented objects where used to fade out the bones from the CTA dataset (see Figure 10). The critical area in close vicinity to the knee-joint was processed by this

Figure 9. Segmented right leg bones seen from the back applying thresholding at 386 Hu (top), region growing (middle), the new method (bottom).

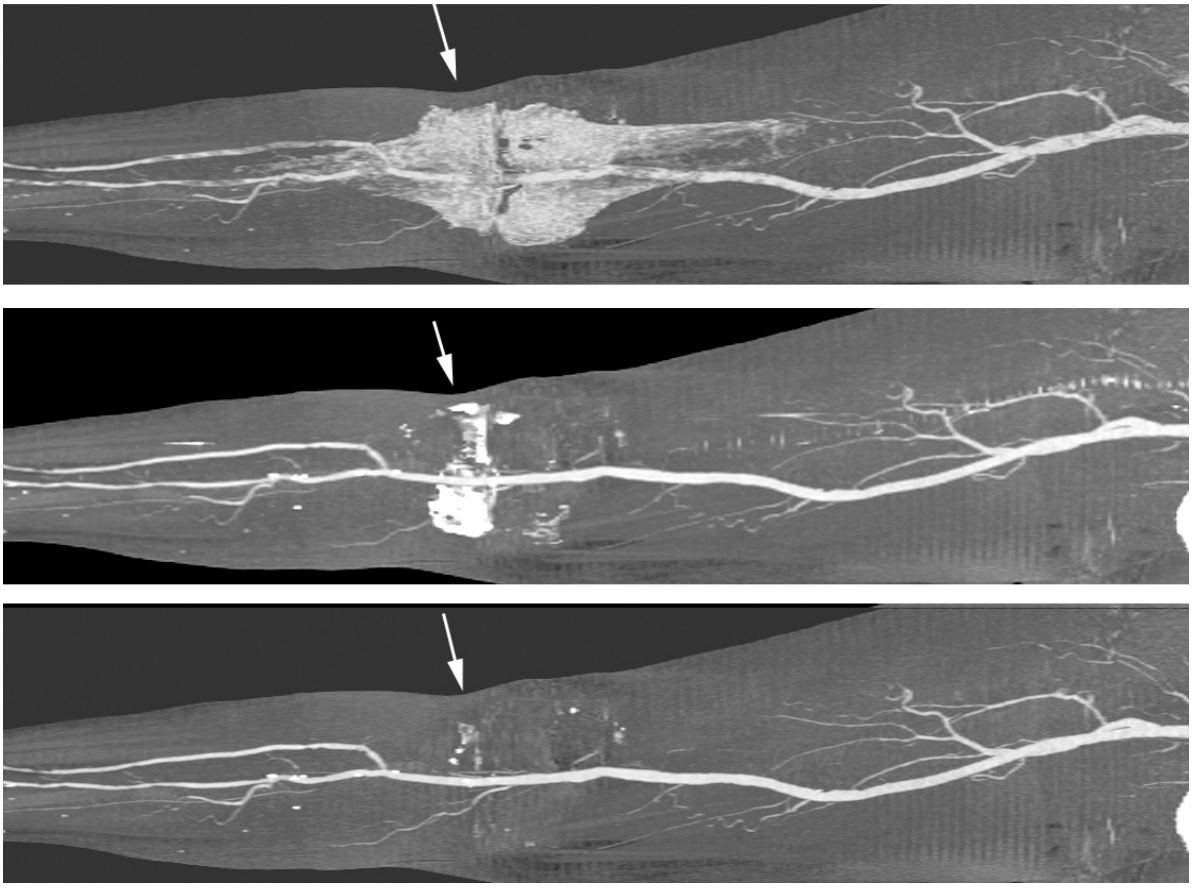

Figure 10. Coronal MIP of a CTA dataset. The mask for removing the bones was generated by thresholding (top), region growing (middle), the new method (bottom).

new technique with only minor errors.

For the legs dataset the total time for segmenting the bones (Figure 10) was 24 minutes: 5 minutes for selecting 14 appropriate cross-sections, 10 minutes for tracing the contours. The automatic reconstruction algorithm took 5 minutes. The surface of the resulting binary mask was smoothed by a 3x3x3 Gaussian kernel and extracted with marching cubes in 4 minutes.

6. CONCLUSION

The presented method provides a robust way to segment objects with a rather steady boundary surface and fuzzy interior. The user interaction effort is much lower than in previously introduced methods. A set of representative cross-sections through the desired object has to be defined. The user does not need to cope with the object topology. Tracing the contours using live-wire is straight-forward. The connectivity points are very helpful to trace the contour in intersecting cross-sections. The accuracy of the reconstruction heavily depends on the live-wire and its cost function but not on the surface reconstruction algorithm itself. This algorithm can also be used with other contour models.

7. ACKNOWLEDGMENTS

The work presented in this publication has been funded by the ADAPT project (FFF-804544).

ADAPT is supported by *Tiani Medgraph* (http://www.tiani.com), and the *Forschungs-förderungsfonds für die gewerbliche Wirtschaft*, Austria.

See http://www.cg.tuwien.ac.at/research/vis/adapt for further information on this project.

8. REFERENCES

[Ada94] Adams. R., Bischof, L.: Seeded Region Growing. IEEE Transactions on Image processing, Vol. 16, No. 6, (1994) 641-647

[Beu92] Beucher, S., Meyer, F.: The morphological approach of segmentation: the watershed transformation. Mathematical Morphology in Image Processing, Chapter 12 (1992).

[Fal00] Falcão, A.X., Udupa, J.K.: A 3D generalization of user-steered Live-Wire segmentation. Medical Image Analysis, Vol. 4 (2000) 389-402

[Kan01] Kanitsar, A., Wegenkittl, R., Felkel, P., Fleischmann, D., Sandner, D., Gröller, E.: Peripheral Vessel Investigation for Routine Clinical Use. Proceedings of IEEE Visualization (2001) 91-98

[Kas88] Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active Contour Models. International Journal of Computer Vision Vol. 4 (1988) 321-331

[Mor92] Mortensen, E.N., Morse, B.S., Barrett, W.A., Udupa, J.K.: Adaptive Boundary Detection Using Live-Wire Two-Dimensional Dynamic Programming. IEEE Computers in Cardiology (1992) 635-638

[Mor98] Mortensen, E.N., Barrett, W.A.: Interactive Segmentation with Intelligent Scissors. Graphical Models and Image Processing, Vol. 60, No. 5 (1998) 349-384

[Ray90] Raya, S.P., Udupa, J.K.: Shape-based interpolation of multidimensional objects. IEEE Transactions on Medical Imaging, Vol. 9 (1990) 32-42

[Udu92] Udupa, J.K., Samarasekera, S., Barrett, W.A.: Boundary Detection via Dynamic Programming. Visualization in Biomedical Computing (1992) 33-39