ELSEVIER

Computers in Biology and Medicine 35 (2005) 329–346

Computers in Biology and Medicine

http://www.intl.elsevierhealth.com/journals/cobm

Interactive 3D segmentation using connected orthogonal contours

P.W. de Bruin[a], V.J. Dercksen[a], F.H. Post[a,*], A.M. Vossepoel[b,c], G.J. Streekstra[e], F.M. Vos[b,d]

[a] *Computer Graphics Group, Delft University of Technology, Room 12250, Mekelweg 4, Delft 2628, Netherlands*
[b] *Quantitative Imaging Group, Delft University of Technology, Netherlands*
[c] *Biomedical Imaging Group, Erasmus Medical Center, Rotterdam, Netherlands*
[d] *Department of Radiology, Academic Medical Center, Amsterdam, Netherlands*
[e] *Medical Physics, Academic Medical Center, Amsterdam, Netherlands*

Received 15 September 2003; accepted 26 February 2004

Abstract

This paper describes a new method for interactive segmentation that is based on cross-sectional design and 3D modelling. The method represents a 3D model by a set of connected contours that are planar and orthogonal. Planar contours overlayed on image data are easily manipulated and linked contours reduce the amount of user interaction. This method solves the contour–to–contour correspondence problem and can capture extrema of objects in a more flexible way than manual segmentation of a stack of 2D images. The resulting 3D model is guaranteed to be free of geometric and topological errors. We show that manual segmentation using connected orthogonal contours has great advantages over conventional manual segmentation. Furthermore, the method provides effective feedback and control for creating an initial model for, and control and steering of, (semi-)automatic segmentation methods.

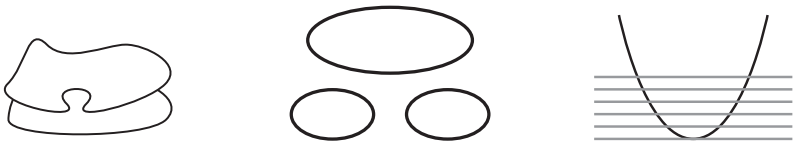
Keywords: Interactive medical image segmentation; 3D geometric modelling; Manual and assisted segmentation; Connected contour model

1. Introduction

Segmentation and visualisation of anatomical structures is important for medical applications such as diagnosis, biomechanical simulation and surgical planning. The data is obtained from medical

* Corresponding author. Tel.: +31-15-278-2528; fax: +31-15-278-7141.
E-mail address: F.H.Post@ewi.tudelft.nl (F.H. Post).

0010-4825/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compbiomed.2004.02.006



(a) (b) (c)

Fig. 1. The classic problems of connecting adjacent contours: (a) contour–to-contour correspondence, (b) branching object, and (c) increasingly smaller contours at object extremes.

imaging equipment, such as Magnetic Resonance Imaging (MRI), Computed Tomography (CT) or ultrasound. A large number of methods is available to segment structures of interest from the data. It is generally not possible, however, to extract a local structure of interest fully automatically. Limiting factors are noise and field inhomogeneities. Additionally, the *as low as reasonably achievable* principle minimises the radiation dose a patient incurs by X-ray imaging. By doing so, the images may not always contain the required detail for automatic segmentation.

Fully manual and semi-automatic segmentation methods usually operate in 2D [1,2]. After a shape is outlined by a set of contours in a stack of 2D images, a 3D shape is constructed by connecting the contours [3,4]. A drawback of such methods is that it is tedious work to construct each contour. In addition, it is not trivial to determine the contour–to–contour correspondence (see Figs. 1(a) and (b)). Finally, it becomes increasingly difficult to draw contours towards the ends of the object, as the object quickly disappears from slice-to-slice (see Fig. 1(c)).

Automatic and semi-automatic segmentation methods often must be initialised, for example, by a geometric model. Manual segmentation methods effectively build such a model, but suffer from the problems indicated. In the fields of cross-sectional design and 3D modelling, methods have been developed to build and deform geometric models [5–8]. It is difficult, however, to guarantee that the model is, and remains, free of geometric and topological errors during interaction.

In this paper we present a new method for interactively segmenting and constructing a 3D model. This method can be used either to segment completely manually, or to create a model to initialise and control more sophisticated (semi-)automatic segmentation methods. The method consists of manipulating a set of connected, planar and orthogonal contours in 3D space. Planar contours are easily manipulated when overlayed on the image data. By using a connected contour data structure the slice-to-slice correspondence problem does not arise. Furthermore, because the contours are connected, the result of manipulating one contour immediately updates the connected counterparts. Thereby, the amount of interaction required to fit the model to an object is reduced. Additionally, it is easy to capture the extrema of objects because contours can be placed in three orthogonal views of the data.

2. Related work

Our interactive segmentation method combines techniques from very different fields: serial section reconstruction [3,4,9–11] and edge detection [12] from medical image processing, cross-sectional

design [6,13,14] and 3D modelling techniques [7,15] as used in geometric modelling for engineering design. Other research on interactive segmentation can be found in [1,2,8]. Several software packages are available for interactive surface extraction [16,17]. All these methods have certain restrictions that make them less useful for our application. For example, all the methods that allow direct manipulation of separate contours require a contour connecting algorithm to (re-)create a surface. The problems related to contour connecting are explained in the previous Section. The 3D modelling approaches allow free-form definition of a surface, where the shape is controlled by a set of parameters. However, in these cases user-interaction is difficult because points need to be moved without constraints in a 3D environment. Furthermore, none of the methods has an explicit mechanism to prevent degenerate shapes from being generated.

What sets our approach apart is that a user has full control over the placement of contour points in such a way that geometric and topological errors are prevented. Contour connecting is not required. Also, the connective structure of the model is constructed incrementally during interaction, and by doing so no singularities or topological errors can occur. Interaction with the 3D model is performed by moving control points on 2D slices. The user gets direct feedback of the effect of her actions.

3. Methods

3.1. Overview

The goal of our method is to extract 3D object shapes by specifying cross-sections in three orthogonal directions. From these cross-sections the object surface model is constructed. The specific application for this method is the segmentation of carpal bones (part of the wrist joint). Therefore, the method assumes that the objects to be modelled have the following properties. The object shape is manifold and topologically equivalent to a sphere, but the geometry may contain protrusions and concavities. The object has no holes and does not self-intersect (genus zero).

An overview of the method is shown in Fig. 2. Starting with an initial template shape (an approximation of a sphere, see Fig. 3(a)) the user can deform the model and add new contours to improve the fit of the model to an object. If contours are planar then manipulation of the entire contour overlayed on the image data is a simple task. By imposing the restrictions that contours are planar, and that at intersections at most two contours are connected, it is possible to add new contours automatically. As the user slices through a dataset the intersections of the model with each plane are shown. At any slice position the user can create a new contour by automatically linking the set of intersections in any of the three orthogonal planes. Next, the new contour can be deformed to improve the fit to the underlying object. The linked contours minimise the amount of user interaction that is required to move control points to desired locations.

3.2. Data structure

An object shape S is defined by a set of oriented contours (see Fig. 3(a)). Given a 3D space spanned by an orthogonal basis $(\vec{x}, \vec{y}, \vec{z})$, each contour is planar and is oriented according to one of

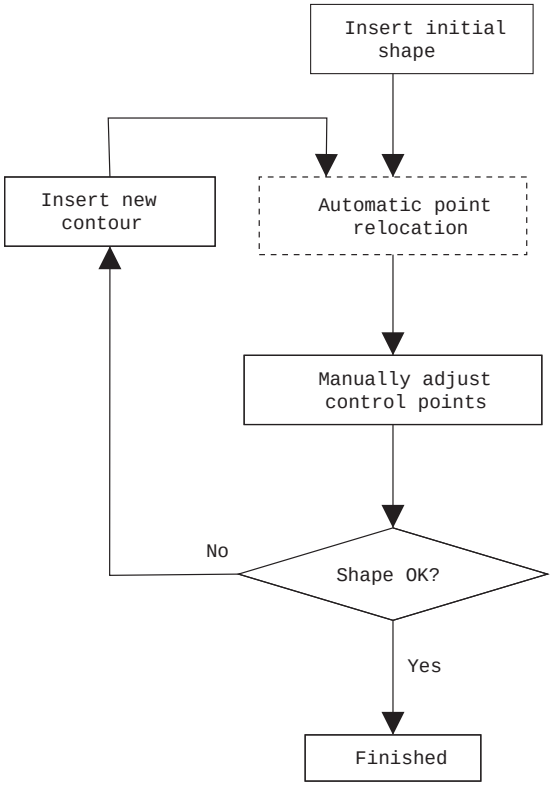


Fig. 2. Overview of the interactive segmentation process.

the base planes. Therefore, all points on a given contour share either the x, y, or z coordinate

$$S = \{C_1^k, C_2^k, \ldots, C_{N_k}^k\}, \quad k \in \{\vec{x}, \vec{y}, \vec{z}\}, \quad \forall N_k \geqslant 1. \tag{1}$$

Each contour C_i^k contains an ordered set P_i^k of control points $\vec{p}_j$ and an interpolation function $I(P_i^k)$

$$C_i^k = I(P_i^k), \tag{2}$$

$$P_i^k = \{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_M\} \quad M \geqslant 4. \tag{3}$$

The interpolation function I determines the location of points between control points. For fast evaluation, a straight line is used. For the generation of smooth curves we use Catmull–Rom splines [18]. This spline has two features that make it suitable for our application: the spline passes through the control points and it has C^1 continuity at the control points. The former makes it easy to exactly place the spline at a user-indicated position. The latter ensures a smooth transition from one spline segment to the next.

Splines are scan converted to the underlying voxel grid using a subdivision algorithm [19]. Each evaluated spline point is given a unique ID that corresponds to the voxel in which it is located. An evaluated spline point is always placed at the centre of the voxel and each voxel contains only one spline point per contour. By doing so, we ensure that each contour is planar and that an intersection

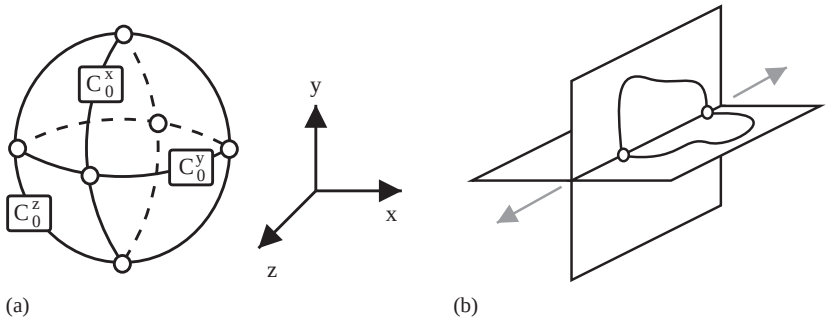


Fig. 3. The initial template model. (a) The initial template shape consists of three planar and orthogonal contours connected at the white dots. Depending on the polarisation it is spherical or octahedral. (b) A control point is allowed to move only along the intersection line of the two planes containing the contours.

is created in each voxel that belongs to two orthogonal contours. The latter is required because two arbitrary splines passing through the same voxel do not always intersect. Furthermore, finding the intersections of a shape with a plane reduces to traversing a list of voxels instead of finding numerical solutions to the spline equation.

The subdivision algorithm proceeds as follows. On each interval of a spline the begin ($t=0$), end ($t=1$), and the midpoint ($t=0.5$) are evaluated. If the voxel containing the midpoint is empty, then the two new intervals are again subdivided. This recursion is continued until all the voxels that the spline passes through are found. The end result is a set of four-connected voxels that form the scan converted planar contour.

By definition, at least one contour of each orientation is needed to build an initial 3D shape (Eq. (1)). The control points are located at the intersection points of contours with different orientations. At each point where contours intersect there must be a control point and at each control point exactly two contours must intersect. Each contour must intersect at least one contour of the other two orientations. Two intersecting contours always have an even number of intersections. Therefore, a contour has at least four intersection points, and consequently, at least four control points (see Fig. 3(a)).

For each control point the four neighbouring control points are stored in a structure. This local connectivity information is used in Section 3.4 to automatically insert contours. For visualisation purposes, a polygon mesh is generated from the linked contours by identifying all polygons enclosed by contour segments. Each polygon is then triangulated using a standard algorithm [20].

3.3. Model shape manipulation

By moving control points the shape model can be adapted to fit a desired object. This manipulation is made considerably easier by the planarity of the contours. Maintaining planarity of all contours requires that the movement of a control point is restricted to the intersection line of the two planes of its contours (see Fig. 3(b)). Therefore, the interaction with a control point is always a 1D task. Note that if three contours were allowed to intersect, no degree of freedom would remain for the resulting control point.

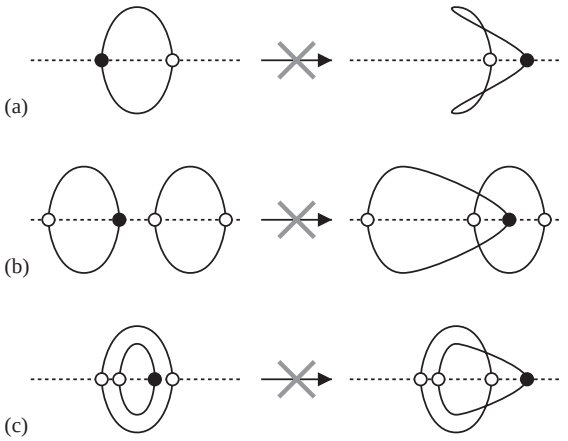


Fig. 4. Constraint on moving a control point. The control points indicated by black dots cannot be moved past any other control points on the line of movement. Self-intersections would result in an invalid model.

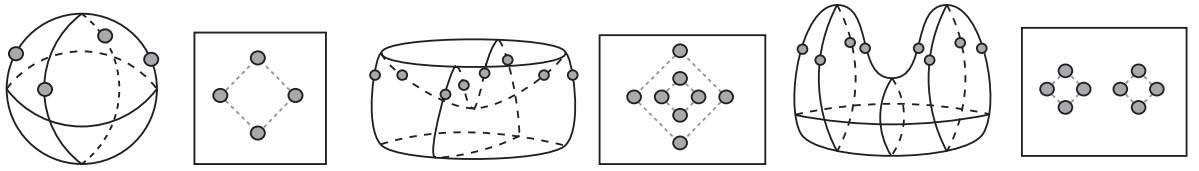


Fig. 5. Three shapes are sliced by a plane, intersections are indicated by the grey dots: (a) a sphere, (b) a bowl shape and (c) a shape with two protrusions. The connections between the intersections are derived from the existing topology.

Apart from the fact that planar contours are easier to manipulate, a manifold shape is easily maintained under planar conditions. Self-intersection is efficiently prevented by requiring that a control point cannot change order with other control points on their common line of movement (see Figs. 4 and 5).

3.4. Adding and removing contours

Adding contours increases the accuracy of the model because more boundary information is provided. Our method can create new contours by automatically connecting the intersections of the shape with a user-indicated plane (see Fig. 6).

A new contour can be added only if two conditions are satisfied. First, the intersection plane for the new contour may not contain any control points. Otherwise, the condition would be violated that at most two contours can intersect in a control point. Second, the number of intersections of the shape with the plane must be at least four, which is the minimum number of control points required for a valid contour (see Section 3.2). Adding a contour to (for example) the initial sphere shape, as shown in Fig. 5(a), is performed by connecting the intersection points in the indicated plane.

The ordering of the control points follows from the existing topology of the contour network (see Fig. 7). In the first step the contour on which the intersection lies (contour a) is traced until a

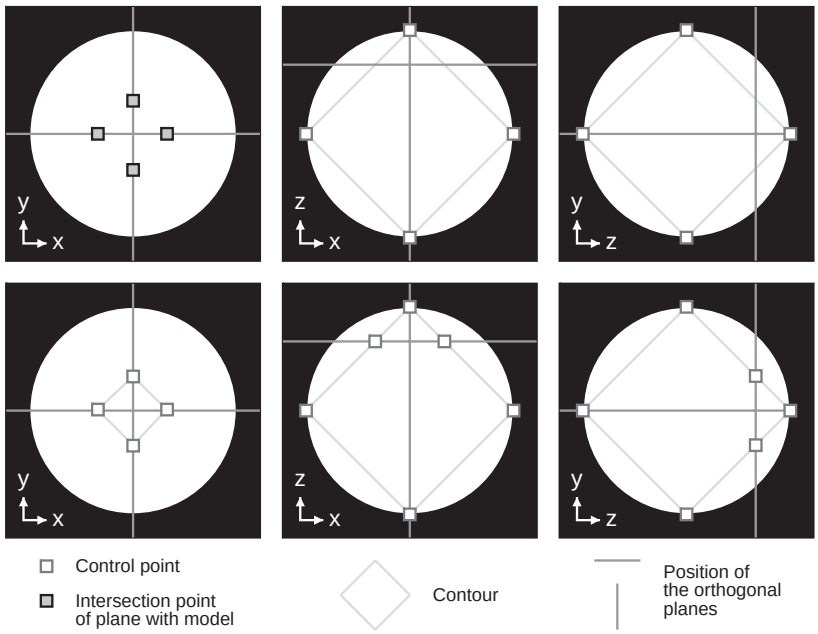


Fig. 6. Automatically creating a contour from intersection points. The three images on the top row show three orthogonal planes before, the three on the bottom row after adding a contour. The intersection points (top left) are automatically connected to form a new contour (bottom left) after the user has indicated a plane.

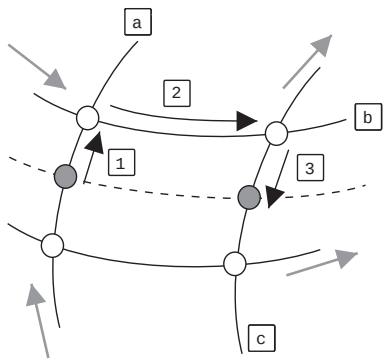


Fig. 7. Determining the ordering of the intersections (grey dots). The dashed line is the new contour to create. The steps are explained in the text.

control point is reached. In the second step, a switch is made to the linked contour (contour b) and this is traced until a control point is reached. Finally, in the third step, again a switch is made to the linked contour (contour c) and it is traced until a new intersection is reached. By doing so, the new intersection points are placed in the correct order and a new contour is formed. This algorithm applies to the case where four control points form a closed-loop. Cases where a different number of control points form a closed-loop are handled in a similar fashion.

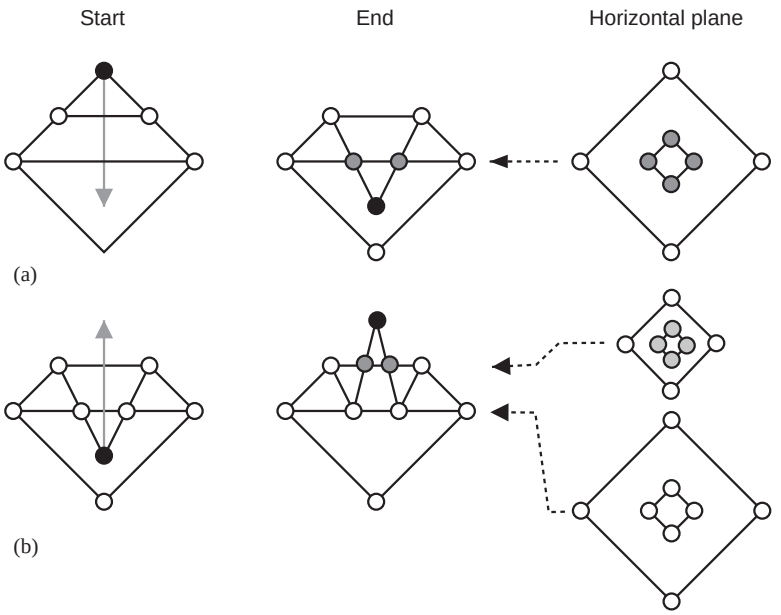


Fig. 8. (a) Creating an indentation followed by (b) creating a protrusion. Appropriate extra contours (grey dots) are generated automatically as the control point (black dot) is moved through intersecting planes.

Fig. 5 shows a number of possible intersection cases. Using the ordering algorithm, a consistent shape is always maintained. Certain shapes require several contours to be added, for example, in the cases shown in Figs. 5(b) and (c) two contours are required (see Section 3.5). Generally, the intersections with a plane form either a single contour (Fig. 5(a)), or (a combination of) non-intersecting sets of concentric (Fig. 5(b)) or disjoint contours (Fig. 5(c)).

A contour can be removed under the conditions that at least one contour of the same orientation remains in the model and that all remaining contours contain at least four control points. Removal of a contour requires the removal of all its control points, and consequently, the removal of these control points from all linked contours.

3.5. Concave shapes

Concave shapes are created either by moving control points deeper into or farther away from the model (see Fig. 5). Consistency requires that contours in an intersecting plane link *all* intersection points. By moving a control point through a plane that already contains a contour, new intersection points are generated (Fig. 8). The new intersections are automatically linked to form the pattern as shown in Fig. 5(b).

Similarly, Fig. 8(b) shows the creation of a protrusion. Again, if a control point is moved through a plane that already contains a contour, extra intersections are formed. Linking the new intersections creates a pattern similiar to the situation shown in Fig. 5(c).

Fig. 9 illustrates a special case of creating a protrusion. If the control point intersects the plane "outside" an existing contour (Fig. 9(a)), then an illegal situation as shown in Fig. 9(b) can occur.

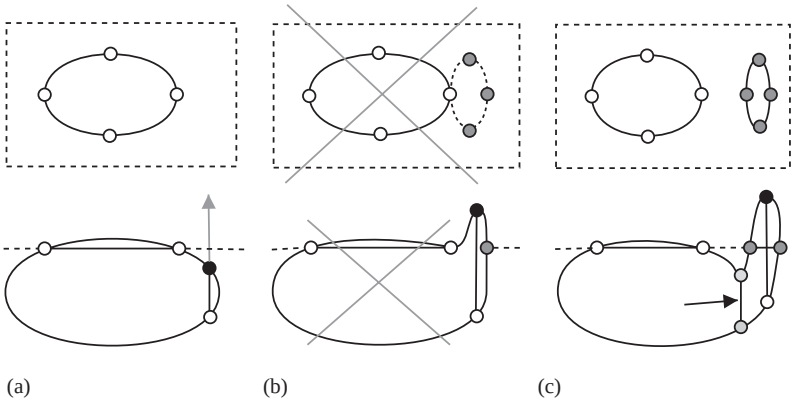


Fig. 9. Moving a control point (black dot) through a plane already containing a control point. The plane is shown in the top row and indicated by a dashed line in the bottom row. (a) When the control point is moved along the grey line passed the dashed line, a situation arises as shown in (b). An illegal situation occurs because a single point is shared between coplanar contours. The solution is to add the contour indicated by the arrow, as shown in (c).

In this case a control point would be part of two coplanar contours, and thereby it would violate the constraints defined in Section 3.2. Fig. 9(c) shows that in order to solve this problem, the user has to add a new contour as indicated by the arrow. By doing so, the control point can be moved through the plane and the automatically generated points form distinct coplanar contours.

3.6. Interaction

The method was implemented using the Visualisation ToolKit [21] and Python [22]. Our implementation will be made available online [1] in the near future. The application runs interactively on a standard PC.

Fig. 10 shows a screenshot of our implementation. Colour is used to guide the user in navigation and orientation in 3D. Each orthogonal plane is (optionally) given a colour. Here, we use blue, red and yellow to distinguish each plane. Each line of a cross-hair is coloured according to its corresponding plane. This makes it easy to determine which plane is orthogonal to which and makes interaction more intuitive. For example, in the blue plane the cross-hair lines are yellow and red. Contours are indicated by green lines and control points by coloured squares. The colour of the control point corresponds to the colour of the line along which it is allowed to move. The intersections with the shape in a plane without contours are indicated by purple circles. A common mode of operation is to scroll through the data and to insert new contours in planes where the intersection points do not line up with the underlying object.

Fig. 11 shows a number of consecutive steps of the segmentation process of a carpal bone. The initial shape in Fig. 11(a) consists of 3 contours and the end result in Fig. 11(e) consists of 20 contours.

[1] At http://visualisation.tudelft.nl/~paul/fastseg, a movie of an example segmentation session is available.

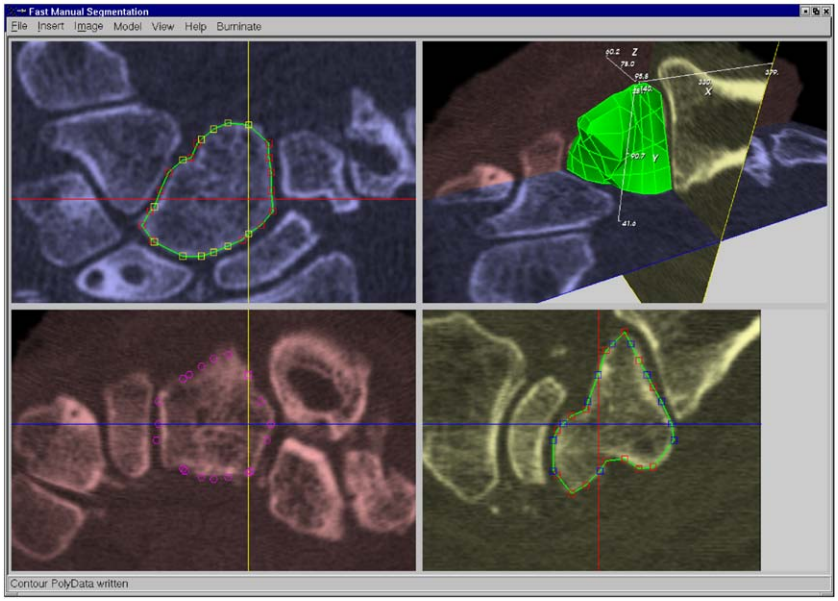


Fig. 10. Screenshot of the segmentation application. A CT dataset of a human wrist and several contours are shown. The colour of the lines of the crosshair and the colour of each control point correspond to the colour of the intersecting orthogonal planes.

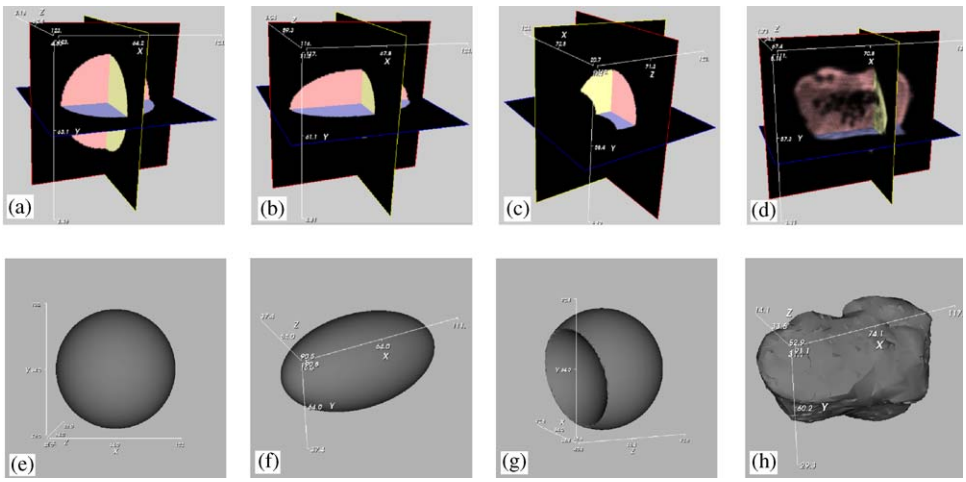


Fig. 13. Objects used for experiments. The top row shows the datasets, the bottom row shows the reference meshes: (a) and (e) Sphere used during training period. (b) and (f) Ellipsoid. (c) and (g) Indented sphere. (d) and (h) Carpal bone.

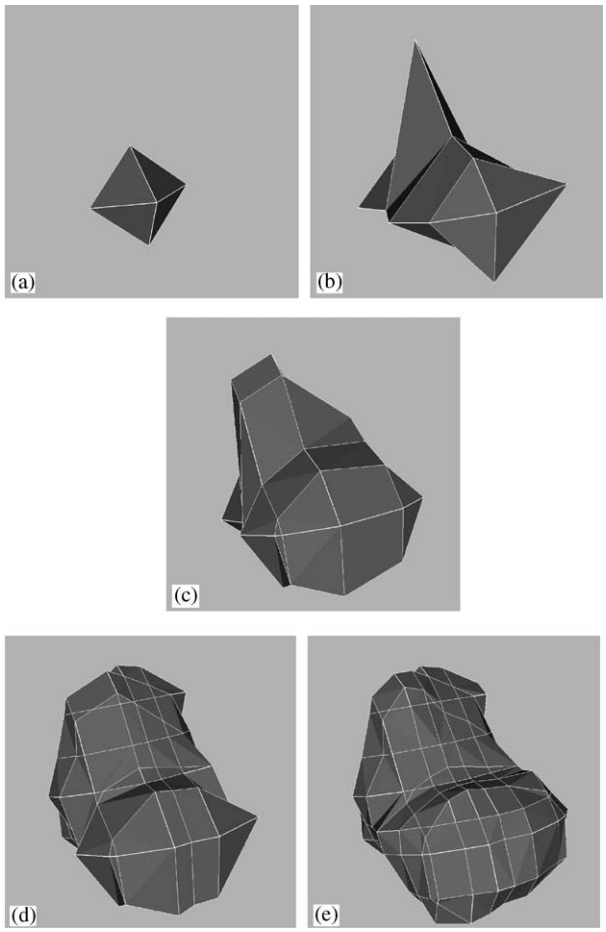


Fig. 11. Illustration of the segmentation process of a carpal bone. (a) Initial sphere. 3 contours, (b) 6 contours. (c) 9 contours. (d) 14 contours. (e) Final segmentation result, 20 contours.

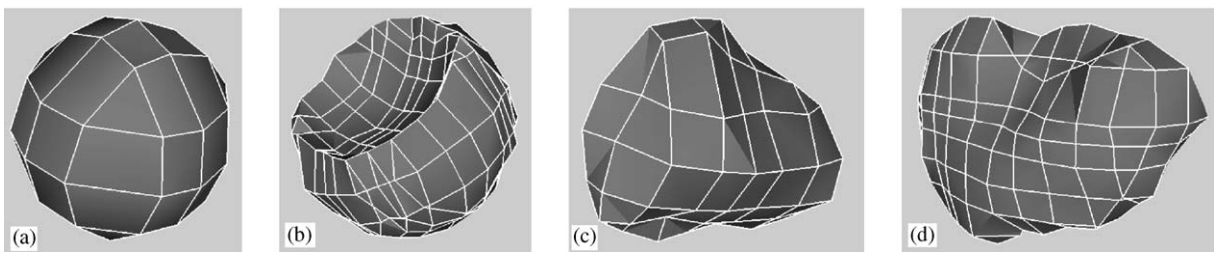


Fig. 12. Segmentations of: (a) a sphere, (b) an indented sphere, (c) a carpal bone in an early stage and (d) the completed carpal bone.

This method is also well-suited to control semi-automatic segmentation methods. Any 2D or 3D method that can calculate a new position for a control point can be applied. For example, we have added to the application an adapted version of an edge-detection algorithm [23] that searches in the direction of movement of a control point for the nearest maximum of the modulus of the gradient. By doing so, the manual segmentation process can be sped up by moving control points using edge-detection enhanced manual segmentation (EDEMS). The user invokes the edge detection in the currently selected orthogonal view by pressing a key. All control points in this view are then moved to the location of the maximum, with a user-selectable search range. The calculation of new locations requires little computer time and the update is instantaneous. By doing so, the user can immediately see the result of the edge detection on *all* the affected points and can manually correct the positions. Using the method in this way speeds up the segmentation process and gives the user effective feedback and control.

4. Results

To test the feasibility of the proposed method we performed an experiment where three subjects are asked to segment a number of objects (see Fig. 12). Synthetic datasets of a sphere, an ellipsoid, and a sphere with an indentation are created together with a reference mesh of each object (see Fig. 13). The accuracy of a segmentation is defined by the average and maximum distance from each point on a contour to the reference mesh. In addition to the synthetic objects, a CT dataset containing a separately scanned carpal bone is used. For this object a reference mesh is created by a multi-scale edge-detection method [23]. Of the three test subjects one had prior experience with our application, the others used it for the first time. After a short introduction to the software using the sphere dataset, the test subjects were asked to segment the three remaining datasets. The results are shown in Table 1. It can be seen that after a short training period test subjects are able to segment a carpal bone in under 10 min time with an average accuracy of 2.5 voxel lengths.

User actions during a segmentation session are logged. For example, we keep track of the distance each control point is moved, when new contours are inserted, and the accuracy of the current model. Fig. 14 illustrates the interaction of a user during a manual segmentation of the indented sphere dataset. It can be seen that the distance that control points are moved decreases as the model

Table 1
Test subject segmentation results

User	Ellipsoid			Indented sphere			Carpal bone		
	time [s]	avg.	max.	time [s]	avg.	max.	time [s]	avg.	max.
		error [voxels]			error [voxels]			error [voxels]	
A	271	0.92	3.1	430	0.96	3.6	533	2.50	6.1
B	284	0.96	2.5	731	0.84	3.7	576	2.65	6.6
C	320	0.88	3.6	591	0.92	4.8	364	2.60	6.8

Subject A has experience using the method, subjects B and C are novices.

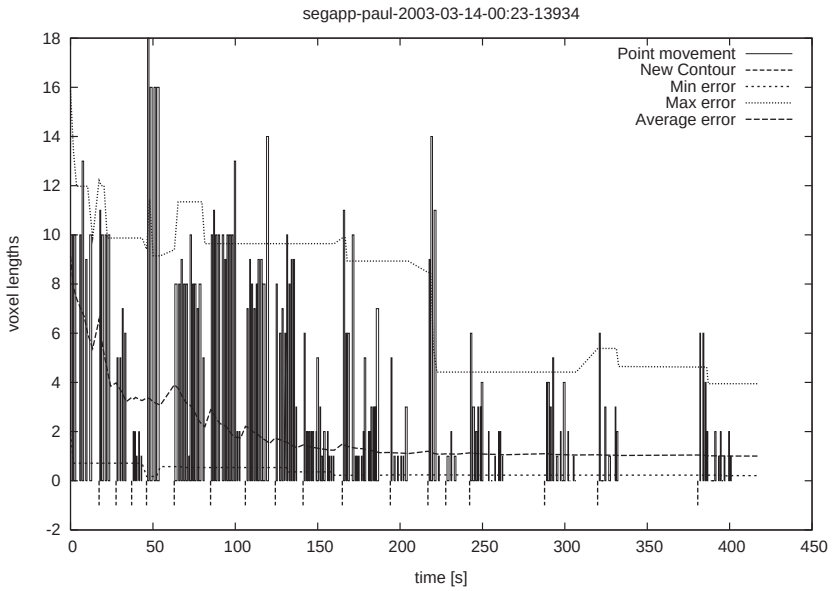


Fig. 14. Graphical representation of a segmentation session. The bars indicate the distance a control point is moved, the dotted vertical lines indicate when a new contour is added, and the lines show the minimum, maximum and average approximation errors. The segmented object is an indented sphere, similar to Fig. 12(b).

is refined. The time interval between contour insertions is initially short, but increases over time. We observed that a user spends this time searching for an appropriate location to insert a new contour. Both the maximum and the average approximation error decrease. The occasional increase of the errors occur because an insertion of a contour adds several control points to the model and consequently, more points are used to calculate the approximation error.

The manual segmentation process can be sped up by using EDEMS. During each segmentation session the distance that each control point travels manually and automatically is logged. A comparison was made by segmenting a carpal bone manually, and by using EDEMS. In both case contours were added in approximately the same locations and the total number of added contours is 15. The results of the manual and EDEMS processes are shown in Figs. 15 and 16, respectively. The manual process takes twice the time of the EDEMS session. In Fig. 15 the distance that each control point travels decreases over time, but after each new contour insertion a set of control points has to be relocated. The EDEMS session in Fig. 16 shows that the amount of manual interaction decreases during the session and is limited to a few corrections near the end. Three carpal bones are segmented using the assisted segmentation. The distance travelled automatically by control points was between 59% and 64% of the total distance. The number of manual control point relocations decreases from 104 in the fully manual session to 26 in the EDEMS session.

The method was also applied to three other datasets. Fig. 17 shows the end result of manually segmenting a MRI dataset of a tumor located between the eyes. Fig. 18 shows a screenshot of our application after manually segmenting a MRI dataset of a brain tumor. During segmentation the surface area and volume of the model can be calculated. In this case, the volume of the model is 12.4 cm^2. Finally, we applied the EDEMS method to a MRI dataset of a large myoma (uterine fibroid). In Fig. 19 the segmented myoma and bladder are shown.

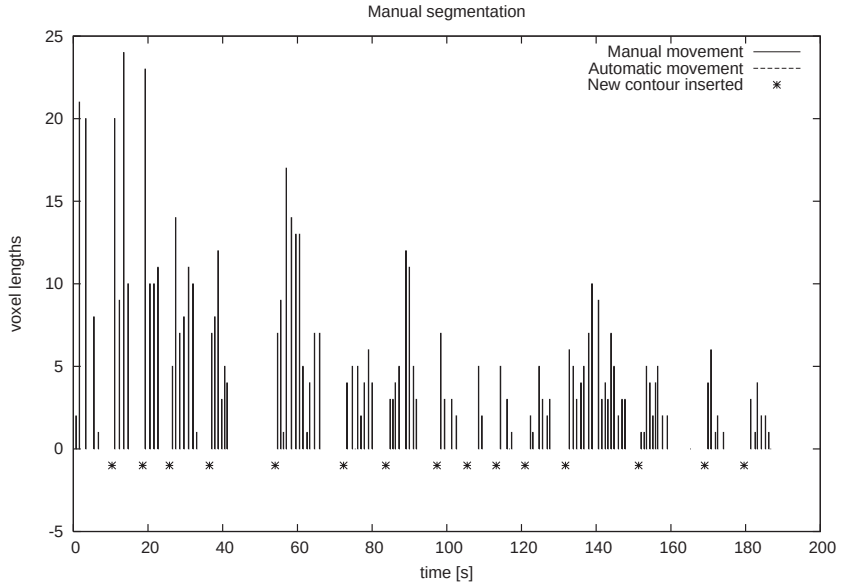


Fig. 15. Manual segmentation of a carpal bone. A total of 104 separate manual control point relocations are performed, the total distance travelled by the control points is 545 voxels, and 15 contours have been inserted.

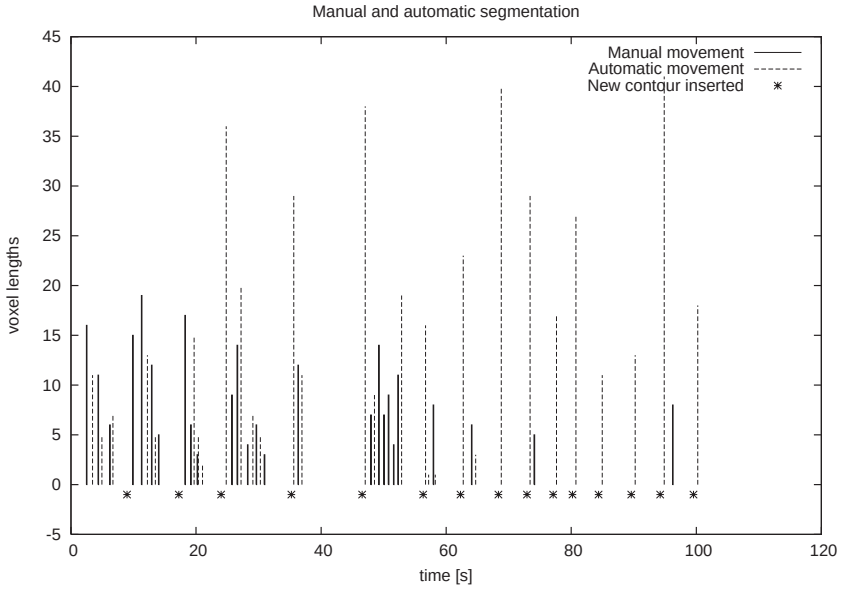


Fig. 16. The effect of adding edge detection on the amount of manual control point movement. The amount of interaction decreases over time and is limited to a few corrections in the final stage. The vertical dashed lines indicate the *total* amount of automatic point movement per invocation. A total of 26 separate manual control point relocations are performed, the total manual distance travelled is 237 voxels, the total automatic distance is 477 voxels, and 15 contours have been inserted.

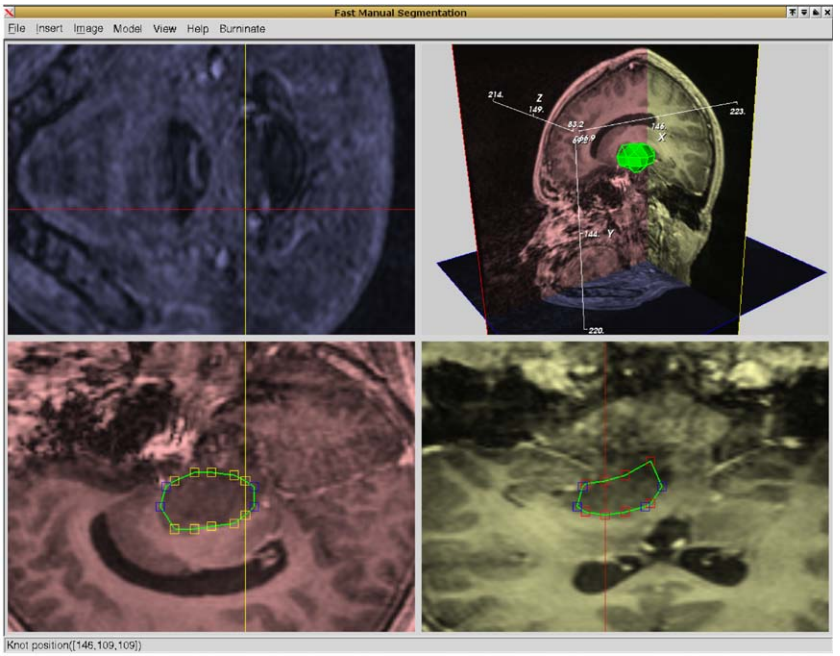


Fig. 18. Segmentation of a tumor located in the brain (MRI dataset). The volume of the model of the tumor is 12.4 cm^3.

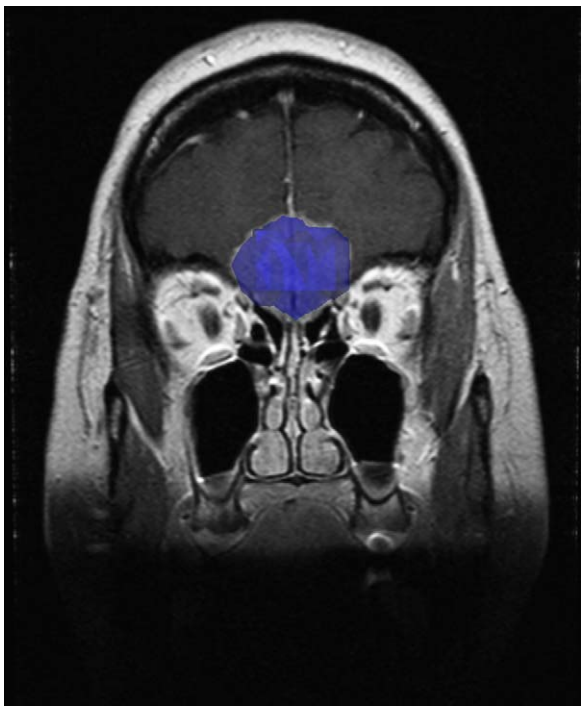


Fig. 17. Segmentation of a tumor located between the eyes (MRI dataset).

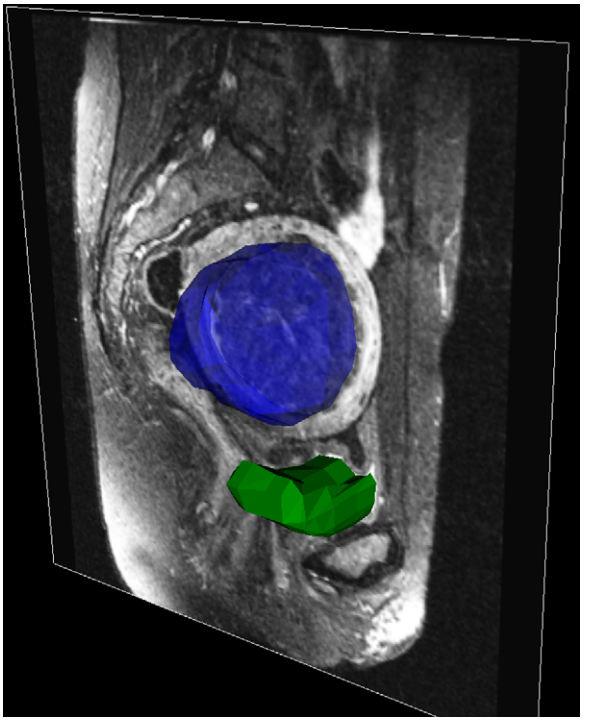


Fig. 19. End result of segmenting a large myoma (blue) and the bladder (green) in a CT dataset.

5. Discussion

5.1. Segmentation strategy

Immediately after adding a new contour, the user should move all the new control points to desired locations. By doing so, the user takes advantage of the better fit of the shape when adding consecutive contours. Thus, the distance new control points have to move decreases steadily.

Also, the user should take advantage of working in 3D. Instead of trying to draw smaller contours at the end of an object in one view, a larger contour could be added in another view to create a better fit.

Applying our method to certain shapes requires planning. For example, the user should identify local minima and maxima and try to place contours accordingly.

Another approach is to reuse the results from previous segmentations. Note that rotation of a shape requires resampling of the data.

Other segmentation techniques could be added to move control points. For example, edge detection and active shapes could be added easily.

6. Conclusions and further research

We have described a new interactive segmentation method. Its data structure consists of a linked set of planar and orthogonal contours. Planar contours overlayed on image data are easily manipulated. User-interaction is kept to a minimum by two features. First, linked contours enable the user to model a 3D shape using 2D slices. Second, contours can be added by indicating a new intersecting plane with the object. The resulting intersection points are automatically linked according to the already present topology. The method can model objects with irregular geometry that are topologically equivalent to a sphere. The end result of the method is always free of geometrical and topological errors.

The use of colour-coding in our application assists a user to navigate and manipulate in a 3D environment. Novice users were able to segment a carpal bone in less than 10 minutes with an average accuracy less than 3 voxel lengths after a short training period of 20 min. The end results of the test subjects are similiar.

Augmenting the method by edge detection speeds up the segmentation process. The manual interaction using edge detection enhanced manual segmentation is limited to initialisation and corrective steering of the results of the edge detection.

Further research will focus on the following topics. First, adding template functionality, where the result of previous segmentation sessions can be reused. This allows building a library of reference objects which can be used for most segmentation tasks. Second, integration of other (semi-)automatic segmentation methods to assist in relocating control points. Third, investigating the possibility of removing the orthogonality restriction. And finally, adding the possibility to model other topologies (objects with holes).

We have shown that manual segmentation using connected orthogonal contours has great advantages over conventional manual segmentation. Furthermore, the method provides effective feedback and control for steering semi-automatic segmentation methods.

Acknowledgements

This research is part of the MISIT (Minimally Invasive Surgery and Intervention Techniques) and the DIPEX (Development of an Improved endoProsthesis for the upper EXtremities) programmes of the Delft Interfaculty Research Center on Medical Engineering (DIOC-9). We would like to thank Mario Maas (AMC), Jaap Stoker (AMC), and the test subjects for their input and time.

References

[1] S.D. Olabarriaga, Human–computer interaction for the segmentation of medical images, Ph.D. Thesis, Universiteit van Amsterdam, 1999.
[2] S. Lobregt, M.A. Viergever, A discrete dynamic contour model, IEEE Trans. Med. Imaging 14 (1) (1995) 12–24.
[3] A.B. Ekoule, F.C. Peyrin, C.L. Odet, A triangulation algorithm from arbitrary shaped multiple planar contours, ACM Trans. Graphics 10 (2) (1991) 182–199.
[4] H. Fuchs, Z.M. Kedem, S.P. Uselton, Optimal surface reconstruction from planar contours, Commun. ACM 20 (10) (1977) 693–702.
[5] J.V. Miller, D.E. Breen, W.E. Lorensen, R.M. O'Bara, M.J. Wozny, Geometrically deformed models: a method for extracting closed geometric models form volume data, in: Proceedings of SIGGRAPH '91, ACM Press, New York, 1991, pp. 217–226.
[6] L.A. Piegl, W. Tiller, Cross-sectional design with boundary constraints, Eng. Comput. 15 (2) (1999) 171–180.
[7] D. Terzopoulos, H. Qin, Dynamic NURBS with geometric constraints for interactive sculpting, ACM Trans. Graphics 13 (2) (1994) 103–136.
[8] M. Harders, S. Wildermuth, G. Székely, New paradigms for interactive 3D volume segmentation, J. Visualization Comput. Animation 13 (2002) 85–95.
[9] J.-D. Boissonnat, Shape reconstruction from planar cross sections, Comput. Vision, Graphics, Image Process. 44 (1) (1988) 1–29.
[10] G. Barequet, D. Shapiro, A. Tal, History consideration in reconstructing polyhedral surfaces from parallel slices, in: Proceedings of the IEEE Visualization '96 Conference, IEEE Computer Society Press, Silver Spring, MD, 1996, pp. 149–156.
[11] D. Meyers, S. Skinner, Surfaces from contours, ACM Trans. Graphics 11 (3) (1992) 228–258.
[12] J. Canny, A computational approach to edge-detection, IEEE Trans. Pattern Anal. Machine Intell. 8 (6) (1986) 679–697.
[13] C. Woodward, Cross-sectional design of *b*-spline surfaces, Comput. Graphics 11 (2) (1987) 193–201.
[14] Y. Zhao, Y. Zhou, J.J. Lowther, C.-K. Shene, Cross-sectional design: A tool for computer graphics and computer-aided design courses, in: 29th ASEE/IEEE Frontiers in Education, November 10–13, San Juan, Puerto Rico, Vol. II, 1999, pp. (12b3–1)–(12b3–6).
[15] T.W. Sederberg, S.R. Parry, Free-form deformation of solid geometric models, in: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, New York, 1986, pp. 151–160.
[16] Surfdriver. URL http://www.surfdriver.com.
[17] 3d-doctor. URL http://www.ablesw.com/3d-doctor/3dhome.html.
[18] E. Catmull, R. Rom, A class of local interpolating splines, in: R. Barnhill, R. Riesenfeld (Eds.), Computer Aided Geometric Design, Salt Lake City, Academic Press, New York, 1974, pp. 317–326.
[19] J.D. Foley, A. van Dam, S. Feiner, J. Hughes, Computer Graphics: Principles and Practice, Second, revised Edition, Addison-Wesley Publishing Company, Reading, MA, 1990.
[20] A. Narkhede, D. Manocha, Fast polygon triangulation based on Seidel's algorithm, in: A.W. Paeth (Ed.), Graphics Gems V, 1995, pp. 394–397.
[21] W. Schroeder, K. Martin, B. Lorensen, The Visualization Toolkit, 2nd Edition, Prentice-Hall PTR, Englewood Cliffs, NJ, 1999. URL http://www.vtk.org.
[22] M. Lutz, D. Ascher, Leaning Python, O'Reilly, 1999. URL http://www.python.org.

[23] P.W. de Bruin, P.M. van Meeteren, F.M. Vos, F.H. Post, A.M. Vossepoel, Accurate and high quality triangle models from 3D grey scale images, in: T. Dohi, R. Kikinis (Eds.), Medical Image Computing and Computer-Assisted Intervention—MICCAI, Part II, LNCS #2489, Springer, Tokyo, Japan, 2002, pp. 348–355.

Paul W. de Bruin (1971) received his M.Sc. in Mechanical Engineering from Delft University of Technology in 1997. In 1998 he started as a Ph.D. student at Delft University of Technology in a project on visualisation for minimally invasive surgery. The title of his thesis is "Accurate and high-quality surface mesh extraction from medical data". Currently, he works as a postdoc at the Orthopaedics Department of the Leiden University Medical Center. His research interests are medical image processing and visualisation.

Vincent Dercksen received his M. Sc. in Technical Informatics from Delft University of Technology, Netherlands, in 2003. Currently he works at the Scientific Visualization department of the Konrad Zuse Institute in Berlin, Germany, where his research includes alignment, segmentation and 3D-reconstruction of large stacks of high-resolution images.

Frits H. Post is an associate professor of computer science (computer graphics) at Delft University of Technology, the Netherlands. He leads a research group in data visualisation since 1990, and conducts research in the visualisation of vector fields and flows, algorithms for feature extraction and event detection, medical imaging and visualisation, virtual environments and 3D interaction techniques. He has been a vice chairman of the Eurographics Association from 1996 to 2000, and was an elected member of the Eurographics Executive Committee from 1992 to 2003. He is currently the chairman of the Eurographics working Group on Data Visualisation. He co-founded the annual joint Eurographics—IEEE TCVG Visualisation Symposium (VisSym). He is an associate editor of ACM Transactions on Graphics (since 1993), and has been an editorial board member of Computer Graphics Forum and IEEE Transactions on Visualization and Computer Graphics.

Geert Streekstra received his M. Sc. in Technical Physics from Twente University in the Netherlands in 1988. From 1989 to 1994 his Ph.D. research at the department of Medical Physics of Utrecht University was devoted to development of methods for measurement of deformation and orientation of red blood cells in flow. In 1994 he worked as a Post-doc at the University of Southern California to investigate the utilization of light scattering theory for measurement of red cell deformability in patients with Sickle Cell Disease. From 1995 to 2000 he worked as a Post-doc at the department of Experimental Cardiology of the Utrecht University, on the development of an automated grafting method for bypass surgery, and at the department of Intelligent Sensory Information Systems the University of Amsterdam where his research activities were in image analysis of 3D confocal microscopic images. Since 2000 Geert Streekstra is an assistant professor at the department of Medical Physics at the University of Amsterdam. His current Research interests are image acquisition and image analysis of 3D and 4D medical images in radiology and orthopedics.

Albert M. Vossepoel (1943) obtained his Masters degree in physics at Leiden University in 1968. He worked as an assistant professor at the Royal Netherlands Naval College in Den Helder and at the Medical School of Leiden University, where, after his Ph.D. in 1987, he became an associate professor. In 1989 he joined Delft University of Technology as an associate professor in the Pattern Recognition (presently: Quantitative Imaging) Group. Recently he was appointed as a part-time professor of Medical Image Processing in Radiology at the Erasmus University Medical Center in Rotterdam. He conducts research in Model-Controlled Image Processing, with as most important applications microscopy, radiology, and minimally invasive surgery.

Frans M. Vos (1969) received his M. Sc. in medical informatics as well as in computer science from the University of Amsterdam in 1993. He was a visiting scientist at Yale University, New Haven, for six months in 1992. In 1999 he received a Ph.D. from the Vrije Universiteit Amsterdam. The title of his thesis was "A system for measuring, modelling and reconstructing corneal shapes based on pseudo-random encoding". Since 1999, he is affiliated as an assistant professor with the Quantitative Imaging Group of Delft University of Technology. Also, he is a staff member at the Department of Radiology of the Academic Medical Center in Amsterdam, The Netherlands.