AN EDITOR FOR 3D MEDICAL VOLUME IMAGES

RUDI DEKLERCK, ALEXANDRU SALOMIE, JAN CORNELIS

Vrije Universiteit Brussel, Dept. ETRO-IRIS
Pleinlaan 2, 1050 Brussel, Belgium

Abstract: An editor for 3D medical images is presented. It allows to visualise anatomical atlases as well as modality images. Objects are delineated slice by slice as a stack of contours edited on the basis of Catmull-Rom splines. The design concepts and the implementation of the editing features are discussed.

1. Introduction

In the context of the European project SAMMIE (HC) 1044 HC, the ETRO department of the VUB (University of Brussels) has been developing a software for the creation of reference sets of anatomical objects (i.e. digital atlases) and for morphometric studies by means of 3D medical volume images. The same principles upon which the tool has been built, are also valid for other imaging domains, where data are acquired as 3D voxel sets, e.g. confocal laser scanning microscopy.

The current tool allows to visualise both atlases and modality images, either separate or with the objects in overlay on the images. In its default appearance the user interface (see Figure 3) consists of a „four window" frame, accompanied by a command area of sliders and buttons. The windows within the „four window" frame are configurable by the user, and can reflect any view on the atlases and the images: a volume rendering, a wiremesh of the object being edited, a slice along the main cuts (transversal, coronal, sagittal) or an arbitrary slice.

With this tool, objects can be delineated slice by slice as a stack of contours along 1 of the main orientations (transversal, coronal, sagittal). An elegant way to edit these contours on the basis of Catmull-Rom splines has been developed and will be explained in detail later on.

Although the contours, based on these splines are very smooth in the direction of construction, it is a major difficulty for the user to design a consistent object in 3D: i.e. to ensure smoothness in the other directions too. However, the 3D editor we developed, distinguishes itself from conventional 2D editors by its features, whichallow to visualise the result of the contour-editing with immediate update inother views along other orientations. The design concepts and the implementation of these editing features will be further discussed in the following sections.

2. General design concepts

In order to meet the requirements the 3D editor has to fulfil, we felt that it was worthwhile to use both volume and contour based approaches. To be more explicit, we need a volume (i.e. voxel) based approach to render the delineated objects and images in 3D. Further more this data representation enables the fast computation of cuts along perpendicular directions or even arbitrary sections along the plane of editing. On the other hand contours provide a superior way of editing compared to pixel- or voxel-painting, they have better scaling properties when they are implemented as splines and are still useful for the display of overlapping objects, which is not the case for filled overlapping volume objects. Moreover, they need much less storage space.

In order to use the advantages of both representations, we had to devise fast conversion algorithms to generate the volume representation from the splines and to convert the voxels back to the contour representation. We were additionally helped by the increased performance of the current computers to ensure the responsiveness of the user interface.

The contours represented by Catmull-Rom splines, are discretised to a polygon of one-pixel- wide line segments (i.e. a chain of adjacent pixels), which is filled by a classical polygon filling algorithm [Foley90]. During this conversion it is unavoidable that some accuracy gets lost.

The algorithm to convert the voxels back to contours is used to generate views on the object in other orientations than its orientation of construction (e.g. sagittal and coronal for an object delineated in the transversal orientation). The contours of the filled objects are retrieved at a sub-voxel level in a way which respects the topology of the objects: the inner and outer contours of objects with holes are discerned by the algorithm, which is described further on. These contours are in fact chains of sub-voxels, through which a Catmull-Rom spline is fitted, in order to make them look smoother when displayed at a larger scale on top of the images.

In our approach we make use of 2 voxel-volumes. One is aimed for the measured modality images and is filled when a volume image is loaded, while the other one can be filled from the contour representation at any time with any combination of 3D objects (e.g. objects from an anatomical atlas can be even combined with objects from a cytological atlas), so there is no need to have several volumes, covering different views, and occupying a lot of disk space on the hard disk.

Of course the volume of the objects can not always reflect the exact geometrical structure of the complete set of objects, as some of them may overlap. In order to solve this problem each object is assigned a certain priority by the user. This priority is used in such a way, that the object with the highest priority will always be displayed. Object attributes such as priorities, colour, opacity and object name can be changed on the fly via a popup-dialog, which also allows the user to make/store and load a hierarchically composed set of objects, like for instance the anatomical hierarchy of the brain and to invoke a Web-browser to display more material (textual descriptions, annotated objects) related to a certain object. In the same way, help descriptions are also made available via the Web-browser.

3. Contours represented by Catmull-Rom Splines

Catmull-Rom splines [Hear94] belong to the family of cardinal splines, which are interpolating piecewise cubics with specified endpoints tangents at the boundary of each curve section. For a cardinal spline, the value for the slope at a control point is calculated from the coordinates of the two adjacent control points.

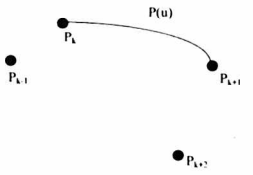

Figure 1. *Curve section between knots*

The class of curves which are obtained when the slopes at the points p_k, and p_{k+1} are set to half the size of their respective chords p_{k-1} ,p_{k+1} and p_k, p_{k+2} as follows:

$$P'(p_k) = 0.5(p_{k+1} - p_{k-1})$$

$$P'(p_{k+1}) = 0.5(p_{k+2} - p_k)$$

are referred to as Catmull-Rom splines or also Overhauser splines. These are in fact the curves one obtains when the tension parameter is set to zero for the cardinal spline. In the first implementation the tension parameter was taken into account, but since it did not offer any extra advantages for our type of application we finally decided to drop it.

The periodic nature of closed contours ensures that the boundary conditions can always be defined. The minimum number of points should be at least 3. Further a cardinal spline will pass through all its points, which is for instance not the case for Bézier-curves and B-splines, where a distinction is made between the points the curve is passing through and the control knots, which allow to directly control the slopes of the curve. These control knots are very useful in applications, where contours are completely imposed by the user, like for instance the design of mechanical parts (CAD) or artistic drawings, but for our type of application, where control knots would have to be defined under the constraint that the spline should follow the trace of a visible edge in the images, they would be more cumbersome than helpful.

In this case Cardinal splines have a clear advantage, and since they can be transformed to 3rd order Bézier-curves, exchange with other applications remains possible.

4. Converting cuts through the volume to contours

The problem we are facing is that it doesn't make sense to display filled objects on top of medical images since the uniformly coloured inner part will

hide the grey value information of the image underneath. In fact the user is mainly interested in the borders or contours of the object. We could compute the borders in a straightforward way via a classical edge detection algorithm and paint these border pixels on top of the images. Yet this approach has the drawback that borders in the edge-image are not more than a collection of pixels, since the method does not generate any additional information about how these pixels are linked to one another. In other words an extra contour scanning algorithm would have to be applied to extract the connectivity information out of the edge-image. Therefore this simple approach works well only when the objects are displayed in their original resolution, but is far from optimal when the objects are displayed at a larger scale.

The algorithm we use, checks for the border of the objects at the same time it is deriving the connectivity of a pixel with it is neighbours.

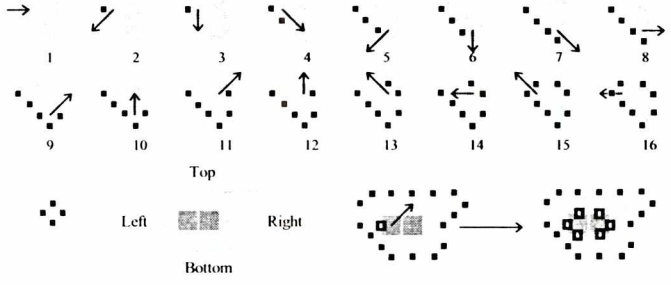

Figure 2. *Scanning the contour at a sub-pixel level.*

In Figure 2 it is shown how the contour is derived for an object consisting of three pixels. The image is traversed from left to right and bottom to top. Whenever a border is encountered (i.e the value to the left is different from the value of the current pixel) and the pixel has not been visited yet, the scanning of the contour is initiated (step 1). The first point of the contour is the point in the middle of the left edge of the current pixel. For an outer contour we then look if there exists a neighbour of the same object in the direction down-left (step 2). Since this is not the case for the object in the example, the direction 45 degrees more anti-clockwise is examined: i.e. direction down (step 3). Again no neighbour of the same object exists. However, since the direction we looked at, is straight (i.e. vertical or horizontal), the point obtained by displacing the center point about half the pixel size along this direction, is added to the contour. In the next step (step 4) the down-right direction is considered. In this case a neighbour is found, which implies that the scanning vector is turned back 90 degrees clockwise in case the neighbour was found along a diagonal and 45 degrees clockwise in the case of a straight vector. This neighbour is now considered as the current pixel and a point obtained by displacing the center point over half the pixel size along a vector, derived by rotating the scanning vector an extra 45 degrees clockwise, is obtained. Again we

start looking for neighbours belonging to the same object (step 5). The same strategies are repeated and the procedure will finally stop when the first point is reached again.

The contour we obtain in this way is indeed defined at a sub-pixel level, since it links the centers of the edges of the pixels. The method even works for a single pixel or thin curves of one pixel thickness. In the case of a single pixel, 4 points are found, so that even the boundary conditions for the fitting of a Catmull-Rom spline are satisfied.

In order to avoid that the same contour would be traced again, when we meet the border on the next line in the image, we mark the visited pixels in an additional image. In addition we discern inner from outer contours by applying an even-odd rule. In the case of inner contours a similar procedure is followed but in opposite directions (i.e. clockwise, while it is anti-clockwise in the procedure for the outer contour and vice-versa).

5. The editing approach

The editing approach is an iterative process. In the first step the user places a number (at least 3) of points along the visible edge in the image. Once this is done, the boundary conditions can be derived and the Catmull-Rom spline between the points is drawn. The user can then see how well the contour is following the edge in the image. For segments of the contour, where the distance from the edge is quite large, the approximation can be improved by adding points on the edge via a left button mouse click. An extra point is inserted between the points of its closest segment, which is continuously updated and highlighted via its different coloured spline points as long as the user moves the cursor with the left mouse button pressed.

For segments already close to the edge, a better match can be further obtained by dragging the spline points with the middle mouse button pressed. In order to allow for fine adjustment, the spline is constantly updated during dragging. Points can be removed too, via a shift-left button mouse click. In this approach parts of the object, which are very smooth, need only a few spline-points, while jagged edges can be approximated at any accuracy by addition or dragging with immediate update of points. These advantages lead to a faster and more accurate delineation compared to outlining methods based on polygons or free hand drawing.

During contour reshaping the cross-section with the other planes (e.g. sagittal, coronal), orthogonal to the plane of the contour (transversal), is continuously computed and displayed on top of the images and contours present in these other planes (see Figure 3). The algorithm we use here is similar to the scan line algorithm used for polygon filling [Foley90] and makes use of the even-odd rule to find out which

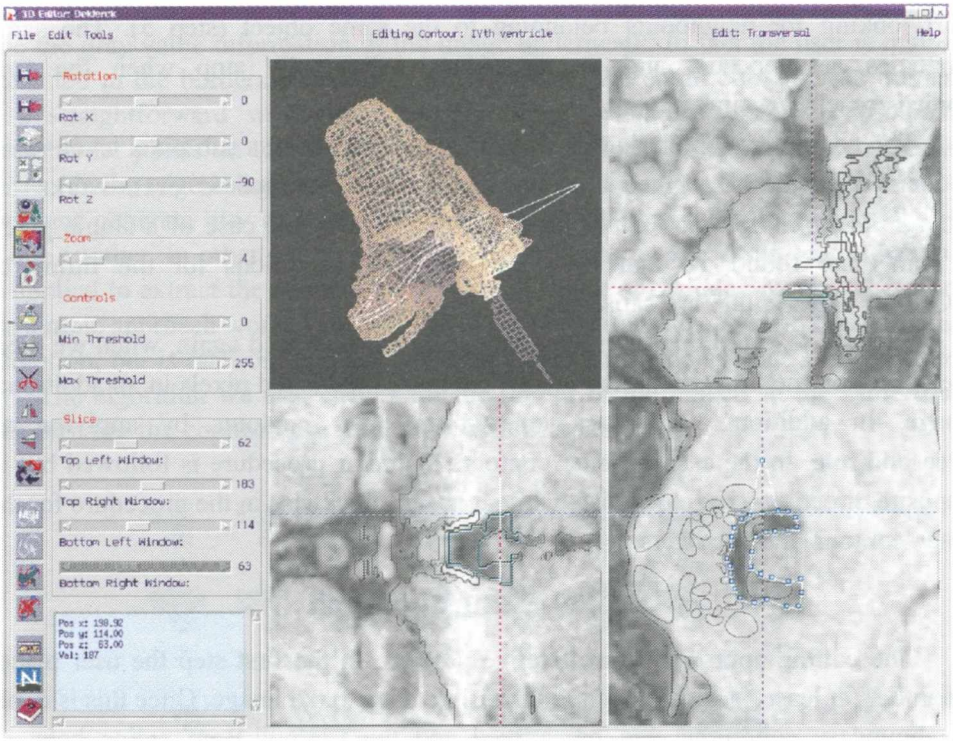

Figure 3. *Display of updates (in white) in different views during contour reshaping*

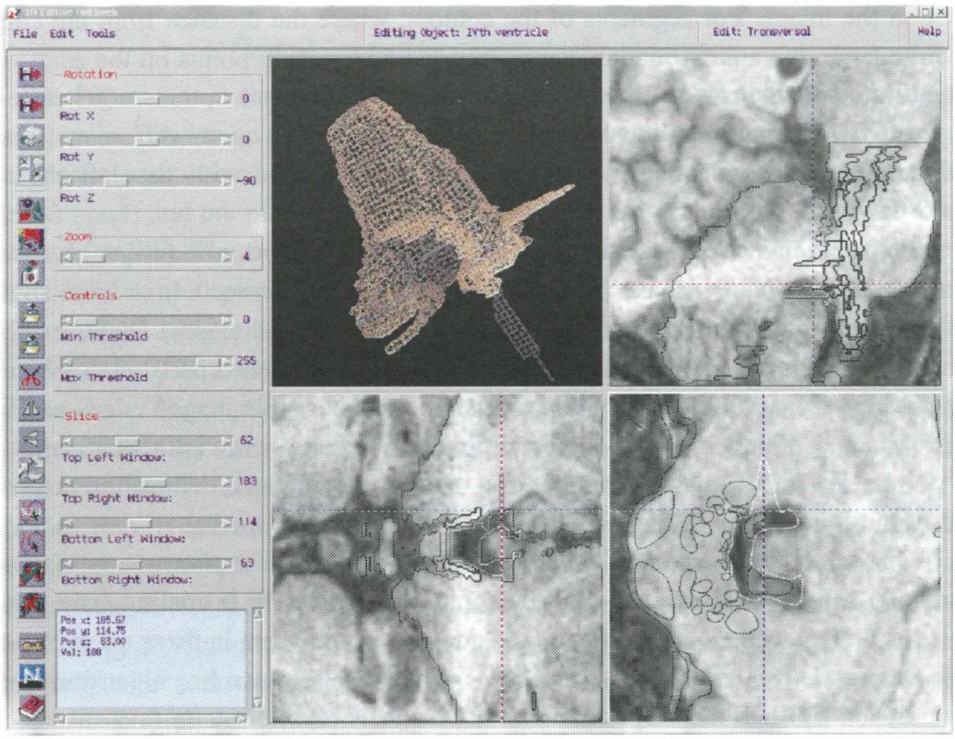

Figure 4. *Conversion of the updated contour (in white) to the volume data*

part of the cross-section is inside or outside the contour. The user may visualise the contour from the previous and the next slice too, in order to get an idea about the variability of the object from one slice to another.

Once the contour has the right shape and the editing is finished the updated contour is filled and put into the 3D voxel volume of the objects. By means of the fast algorithm described in the previous section, we then scan the volume in real-time in order to retrieve and show the updated outlines of the objects in all planes (see Figure 4). The same updates can also be made immediately apparent at the object's surface by means of a 3D wiremesh view, which has in addition some extra 3D editing features.

6. The wiremesh view

The wiremesh view allows editing of the active contour in a similar way to the cut view. The advantage of a wiremesh view is that the user has a global view over the object. Once the modification is accepted, the changes are written into the volume, from which the wiremesh can be generated in real time using again the same fast scanning algorithm. The contours, produced by this scanning algorithm in the three main directions, are rotated according to the angles specified by the user and projected into the viewing window, using a hidden line removal algorithm based on a Z-buffer. The contour points are depth shaded: i.e. points close to the observer are displayed brighter than more distant ones. In this way a good 3D impression of the surface can be obtained. The images (see Figure 3 and Figure 4) show a contour in edit state before and after the update. The time needed to recompute the mesh is less than 1 second on an UltraSparc 140MHz.

The excellent quality of surface rendering can however not be reached. The surface rendering approach is definitely more costly since it has to call upon the marching cubes technique [Lore87], which is a brute force surface construction algorithm generating thousands to millions of polygons for a certain object. The display of such a large number of polygons in less than 1 second demands special graphics acceleration hardware. An alternative is to use algorithms as in [Hopp93] to reduce the number of triangles, without giving up good surface approximation, but they have the drawback to be very slow (several minutes).

7. Conclusion and Future work

The 3D editor has proven to be useful for the elaboration of an anatomical atlas of the brainstem at the University Hospital of Düsseldorf and is currently used at the Erasme Hosipital of the Université, Libre de Bruxelles in the context of a morphometric study of the cerebellar volume of young children.

As the manual delineation of anatomical objects is a tedious task, we are currently combining the useful results produced by image processing techniques

(such as the watershed algorithm, and multivariate techniques for 3D image segmentation based on Fourier descriptors and statistical variables) with the editing approach described above.

We are also implementing a 3D matching algorithm, based on thin plate splines and the interactive definition of 3D landmarks, in order to register the anatomical atlases to the images.

References

[Hear 94] Donald Hearn, M. Pauline Baker, „Computer Graphics, Second Edition", Prentice Hall, 1994, Chap 10, pp 320-327

[Hopp93] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W., „Mesh Optimisation", Computer Graphics (SIGGRAPH '93 Proceedings), July 1993

[Lore87] Lorensen W. E., Cline H. E., Marching Cubes: „A high resolution 3D surface construction algorithm", Computer Graphics, Volume 21, Number 4, July 1987, pp 163-169

[Foley90] Foley J. D., van DAM A., Feiner S. K., Hughes J. F., „Computer Graphics Principles and Practice", Second Edition, Addison-Wesley, 1990, pp 92-99

Acknowledgements

This work is supported by the IWT, under the „IT-TeleVisie" project ITA/950202/KUL and by the EU SAMMIE project HC 1044 (HC).

The paper was presented at TEMPUS JEP 11550 Workshop „Data Acquisition and Visualisation in Medical Diagnostic", 24 September 1997, Ilmenau.