Sketch-Line Interactions for 3D Image Visualization and Analysis

T. McInerney and Y.S. Shih

Dept. of Computer Science, Ryerson University, Toronto, ON, Canada, M5B 2K3

Abstract. This paper explores the effectiveness of an interaction model based on user-sketched line segments and curve segments, together known as sketch-lines. Directly sketching line segments on image slices, or curve segments on the surfaces of objects in volume rendered or surface rendered 3D data, is an effective means by which to quickly and simply transfer accurate information, such as position, object surface orientation, and surface region width, from the user to a visualization algorithm. This information can be used for fast, intuitive and precise object-relative image slice positioning and for precise, editable region of interest (ROI) delineation in a volume image. The results from two user studies are presented that analyze the efficiency, intuitiveness and precision of the sketch-line interaction model as well as quantitatively and qualitatively compare aspects of the model to other interaction techniques.

1 Introduction

Efficient and intuitive user interactions that aid in the visualization and analysis of 3D data can often be effectively realized using data surface relative input actions that mimic familiar constrained physical actions such as drawing, sliding, and painting [1]. Furthermore, several 3D view generation and region delineation tasks (for example, for surgical planning or for segmenting objects in noisy images) not only require efficiency and simplicity, but also precise control and accuracy. These additional requirements can complicate the design of a 3D interaction model as they often conflict with efficiency and simplicity.

This paper explores the effectiveness of an interaction model based on user-sketched line segments and curve segments, together known as sketch-lines. Directly sketching line segments on image slices, or curve segments on volume rendered or surface rendered object surfaces, is an effective means by which to quickly and simply transfer accurate information, such as position, object surface orientation, and surface region width, from the user to a visualization algorithm. In the context of volume image visualization and analysis, this information can then be used for fast, intuitive and precise object-relative image slice positioning or for precise, editable region of interest (ROI) delineation. These two primary user operations underlie many common volume image visualization tasks, including 3D image slice-based exploration and inspection, visualization via cutaway, surgical planning, and deformable model-based segmentation [2]. We combine the sketch-line interaction model with the use of an interpolating spline curve, an interpolating subdivision surface and an extrusion process to support the implementation of the two primary user interactions. We demonstrate the use of sketch lines with several examples. We also present results from two user studies in order to gain insight into the efficiency, intuitiveness and precision of the sketch-line interaction model as well as compare it to other interaction techniques. A conclusion and future work section summarize the advantages and current drawbacks of the model and its implementation.

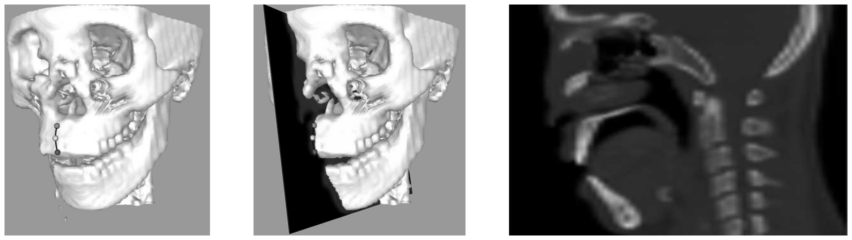

Fig. 1. Left: a sketch-line is drawn on the volume rendered skull surface and an image slice (middle) is instantly positioned and oriented such that it is approximately orthogonal to the surface. Right: the image slice is also displayed in a 2D window.

2 Related Work

Image slice views remain an integral part of the detailed inspection and analysis of volume images. Many volume image visualization packages provide widgets that support the translation and arbitrary rotation of a 3D image slice. The 3D image slice is typically rendered together with a volume rendering of anatomical structures and optionally as a 2D image slice rendered in a separate window. These 2D/3D view combinations are effective for volume image navigation and exploration [3]. It is often useful to orient the image slices orthogonally to a curved surface region of an anatomical structure in order to examine and measure these "natural" object cross sections. Image slice widgets typically provide controls to "push" the slice plane in a direction along its normal vector as well as controls to rotate it. However, direct links between the slice plane manipulation and the 3D object typically do not exist. Orienting and positioning are performed in "absolute" 3D world coordinates. Repeatedly positioning the slice back and forth along a structure in order to explore it therefore relies on a series of independent world space manipulations. Direct object surface relative positioning, on the other hand, is potentially more efficient and intuitive [1].

Selecting or enveloping an arbitrarily shaped volume of interest (VOI) in volumetric data sets in a simple, intuitive and precise way is a complex 3D interaction task. This task is further complicated if the user wishes to represent the VOI with a precisely and easily editable model. Common representations of a VOI include a triangle mesh, a set of voxels, or blended implicitly functions. These VOI representations can be too low level, often have limited editing support, and often cannot be precisely defined without considerable user effort. A selected VOI can be cutaway to remove unwanted data in order to visualize hidden objects. In addition, a geometric representation of the VOI can be used as input to a segmentation algorithm to constrain the algorithm when it is applied to noisy images, in an effort to prevent "leaks" into neighboring objects.

Most visualization systems support standard tools such as cropping boxes, controlled using widgets, that can be used to select a box-shaped VOI in real time. Fuchs [4] extended this idea to allow the generation of a non-convex polyhedral VOI. Many 3D region selection techniques use some sort of interaction metaphor. Sketching [5], tracing [6], painting [7], sculpting [8], are among the most common. In the sketching metaphor, the user draws lines or contours on the screen. These contours are then connected and "inflated" to form a 3D envelope. The sculpting metaphor simulates cutting tools with convex-shaped tool "tips" that are positioned and/or moved within the volume. Any voxels inside the tool tip are selected and sculpted away. Similar to sculpting, painting defines a surface region, such as a circle, as the "brush tip" and as the brush is moved along the data object surface, voxels inside the brush region are rendered transparent. In tracing the user draws a contour on the object surface to outline a region. Although many of these interaction metaphors support fast, approximate VOI specification, most do not support precise and efficient editing of the VOI, often due to the low level VOI representation.

3 Sketch Line Methodology and Examples

The sketch-line interaction model presented in this paper is an extension of the sketch-line model in [2] used for image segmentation. The basic idea is to maximize the amount of accurate information (position, orientation, width, surface normal etc.) from the user to the algorithm with the least amount of effort by using only simple strokes. Sketching lines is a simple and familiar action for users and the lines can be quickly and precisely drawn.

The sketch-line system was implemented using VTK 5.6.0. The sketch-line system interface is designed with two window views: the left window displays a volume rendering of the data and a 3D image slice plane that can be positioned, scaled and rotated using a sketch line or using standard widget controls, and the right window shows a 2D image slice view (Figure 1). The 2D window camera position and orientation are linked to the 3D slice plane orientation so that a consistent 2D window view "up" direction is maintained.

3.1 Sketching Line Segments on Image Slices

Sketch-lines are formed by clicking and holding the left mouse button (or finger/stylus), dragging the mouse to another location to stretch out the line segment, and finally releasing the button. Sketch-lines can be connected together

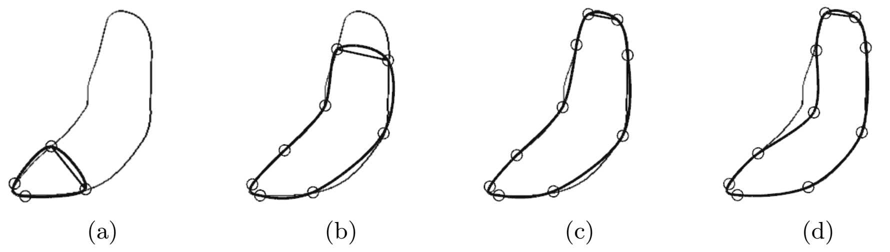

Fig. 2. (a)-(c) Sketch lines are drawn across the width of an object and connected to form a closed spline. Only a few lines are need to quickly and accurately delineate complex contours and the spline can be input to a segmentation algorithm. (d) The spline control points provide precise editing capability.

with a spline to form a closed contour (Figure 2). To maximize the amount of user-provided information needed to create a contour envelope that accurately delineates an object cross-section, sketch-lines should ideally be drawn across the object cross-section so that they are approximately orthogonal to its primary medial axis (Figure 2). As is clear in the figure, the resulting spline curve accurately delineates the object cross-section using very few control points. This sketching process takes less than a second per sketch line and has a short learning curve - after some practice it is often obvious where to place the lines and how many to use. As the user adds a sketch-line, the spline curve is updated and rendered. Most simple shaped contours require only 2 or 3 sketch lines (in Figure 2 6 are used). The straight line segment joining the front two control points in Figure 2 is called the *active edge*. When a new line is sketched, its endpoints are connected to the active edge endpoints and a new spline contour is created. The user can click on the spline contour in another location to change the active edge, allowing for the creation of more complex shapes. Finally, the spline contour can be intuitively edited, at any time, by selecting and dragging the control points (Figure 2d).

3.2 Sketch-lines on Object Surfaces

A sketch-"line" is drawn on the surface of the volume rendered object using a similar mouse click and drag action as the image slice case. Because object surfaces are curved, a sketch-line is visually represented as an interpolating quadratic spline curve segment so that it conforms to the surface (Figure 1). The spline curve is constructed from 3 sampled surface points. Along with surface point samples, object surface normals are also sampled along the sketch-line path and are used to construct a 3D coordinate system for an image slice (Section 3.3).

Surface Sketch-Line Construction To construct the spline curve segment, the first point picked on the object surface (e.g. Figure 3, leftmost sphere) is used as the start control point. As the user drags the mouse and moves the cursor

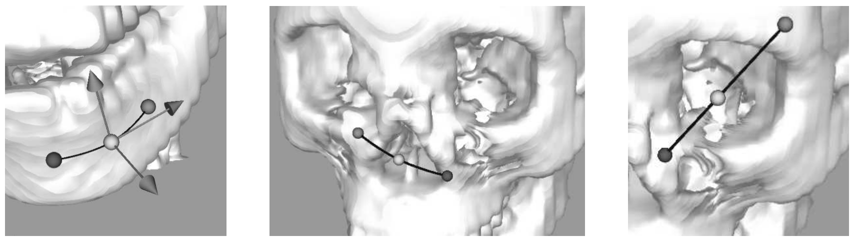

Fig. 3. Examples of sketch lines with the spline control points highlighted. The left image shows the basis vectors of a image slice coordinate system. The middle and right images show sketch lines automatically spanning concave regions.

along the surface, surface points and normals are sampled and an average surface normal is calculated. Similar to an image-slice sketch line, as the user drags the cursor over the surface the end control point (e.g. Figure 3, rightmost sphere) is continuously updated (and the spline continuously redisplayed). A 3rd middle control point (e.g. Figure 3, middle sphere) is continuously calculated as well using the sampled surface point currently furthest away (in a positive average surface normal direction) from the 3D line segment formed by subtracting the start and end control points (that is, furthest above the surface).

The average surface normal is calculated "on the fly" from surface normals sampled roughly along the surface path from the start control point to the end control point. The surface sampling process is as follows. The 2D screen window points corresponding to the 3D start and end control points are connected to form a line in screen space. Points are evenly sampled along this screen space line and are projected back onto the object surface and the normal samples are gathered at these locations. Similar to the image slice sketch lines, this sampling process allows users to stretch and drag the spline segment along the surface until they are satisfied with its position, length and orientation.

The sketch-line construction process was designed to create a spline curve that is always visible on the surface while also conforming to the surface shape. If the sketch-line is drawn over a concave region (i.e. an indentation such as the eye socket (Figure 3 middle) or a hole in the surface), the sketch-line will be constructed across the region rather than bend inwards. From experiments, 5-10 sampled surface normals are sufficient to form an accurate average surface normal. To increase accuracy and reduce susceptibility to noise, the number of sample points along the line can be increased, surface points within a small neighborhood of these sample points can be included, and/or a pre-smoothed image volume can be used.

3.3 Image Slice Positioning using Surface Sketch Lines

To position and orient an image slice using a sketch line (Figure 1) such that it is roughly orthogonal to the object surface, an image slice coordinate system is constructed (Figure 3 left) using the average surface normal as the u axis (Figure

3 left, vector coming away from the surface), the normalized line segment formed from subtracting the spline start control point from the end control point as the initial "up" vector (v axis - Figure 3) left, vector pointing to the right) and the cross product of these two vectors as the slice plane normal (n axis - Figure 3 left, vector pointing up). The standard orthogonal image slice plane (sagittal, axial, coronal) closest to this coordinate system is determined and this slice is then translated and rotated to match the constructed coordinate system. The initial up vector of the image slice is then adjusted by rotating the slice around its normal vector until the u and v axes are parallel to the sides of the volume image bounding box. This image slice "spin angle" constraint makes the slice appear "upright" in the 3D window from the user's perspective rather than tilted and establishes an up direction in the 2D window view. In a final step, the image slice is translated with respect to the u and v axes so that it is centered within the volume image bounding box. The image slice may then be edited by the user with the standard widget controls.

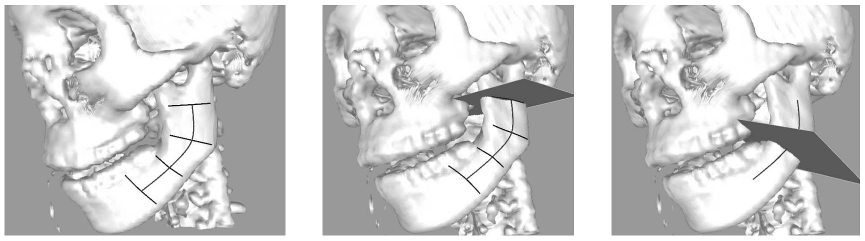

Fig. 4. Left: a series of sketch-lines (4, in this example) are drawn along the curving surface of the lower jaw and their midpoints are connected to form a smooth spline. Middle, Right: the user can slide the slice along the curve and it is constrained to remain approximately orthogonal to the jaw surface.

Image Slice Plane Sliding If the user wishes to examine object cross-sections along a curving part of a target object surface, they would typically use widget controls to push and rotate the image slice in 3D world coordinates until the slice is visually orthogonal to the surface, and then repeat this process back and forth at different points along the target region. Using sketch-lines, this inspection of various cross-sections orthogonal to the target surface is simple and efficient. The user quickly draws a series of sketch-lines along the target region (4 sketch lines in Figure 4 left). The midpoint of each sketch-line is automatically connected and used to form an interpolating spline. The average surface normal calculated from each sketch-line is also smoothly interpolated using this interpolating spline. The user can now push the image slice as usual and the slice center point automatically follows the spline path while the slice normal is automatically smoothly interpolated between the sketch-lines. The image slice can now slide back and forth along the object surface and is automatically constrained to remain approximately orthogonal. This is an example of how sketched input typically contains more ”structured” information that can be interpreted by the visualization algorithm.

3.4 VOI Selection using Sketch Lines

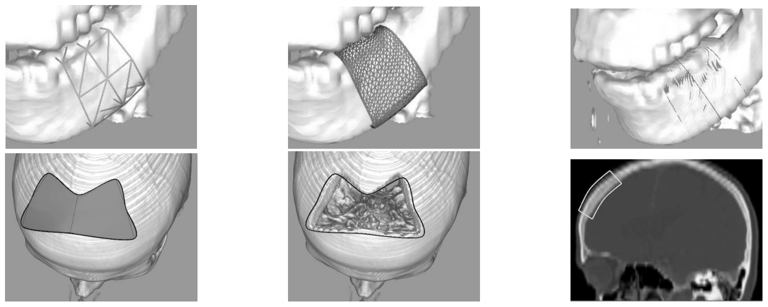

Fig. 5. Top row: 3 sketch lines are converted to profile curves and connected to form a sleeve control mesh. Middle: subdivided mesh and (right) the result of fitting the mesh to the jaw. Bottom row: a patch is created with 3 sketch lines, extruded inward and used to cut away the skull. The right figure shows a cross sectional view.

Similar to the contour constructed from lines sketched on an image slice, multiple sketch-lines can be quickly and precisely drawn on the object surface along the length of a region of interest (Figure 5). The corresponding control points of the 3D spline curves can be connected to form a mesh of triangles. This control mesh is fed into an interpolating subdivision surface algorithm [2] which finely subdivides the triangles to form a smooth surface (Figure 5 top row). This subdivision surface delineates and/or envelopes the region and can be used to either cut the region away or it can be converted to a deformable surface model [2] and fitted to the region (segmenting it).

The sketch-line system currently supports 3 types of meshes: a closed mesh, an open cylindrical mesh known as a “sleeve”, and an open surface mesh known as a “patch” (Figure 5). A patch can be used to delineate a curving region of the target object surface such as the top of the skull. The patch can be copied, the copy moved into the object surface, and the two patches connected together to form a closed “thin shell” mesh envelope. This type of envelope is useful for cutting away thin shell structures such as the skin or the skull. A sleeve is an open cylinder mesh that can be used to define a section of a curving cylindrical structure such as an artery. The sleeve can be fitted to the object section using the deformable surface model segmentation algorithm [2], segmenting it. The open cylinder sleeve mesh can also be capped at its ends to form a closed mesh which can then be used to cutaway sections such as a portion of the jaw. Finally, a closed profile curve can be constructed from the open sketched spline segment.

The 3 control points of the sketched spline curve are copied and moved inward (i.e. in the direction of the negative average surface normal vector) a user-defined distance (using a slider) into the object. The copied spline curve is also reflected across the axis formed by the line joining the spline end control points. The copied spline curve segment is connected to the sketched spline to form the closed curve. A series of 3D sketch-lines can be converted to profile curves, connected and subdivided to form the smooth closed surface mesh.

The subdivision surface envelope or patch mesh can be flexibly edited in several ways, either in the 3D window or 2D window via the profile curves. The envelope control points can be repositioned and the envelope is updated in real time. Entire profile curves can be translated/rotated via the slice plane in which they are embedded. If the initially sketched envelope does not quite cover the ROI or if the depth of the envelope needs adjusting, the user can quickly adjust control points to correct it. Finally, sketch lines can be quickly undone to return the envelope to its previous state and then redrawn if desired. Further details on editing can be found in [9].

4 Evaluation

Two user studies were conducted to quantitatively measure the efficiency of the sketch-line interaction technique compared to several other common interaction models, as well as to qualitatively assess its effectiveness. The first user study was conducted to compare the sketch-line slice plane positioning and orienting technique with a standard widget push/rotate technique. A volume rendered skull was used as the target object for slice positioning as it contains easily visible surface features. In the push/rotate technique, the user selects any point on the slice plane and moves the mouse to "push" the slice in its normal vector direction. To rotate the slice for both push/rotate and for fine-tuning the result of the sketch line slice positioning, a rotation model using precisely controllable circular rotation handles was used [9].

A target slice plane is presented to the users, positioned and oriented approximately orthogonally to some part of the skull surface. For the push/rotate technique, a reference slice plane in a standard position is provided and the users must manipulate it to match the target plane. For the sketch-line technique the user sketches a line on the skull surface near the target slice in order to match it. The users are permitted to fine-tune the sketch-line result using the standard push/rotate technique. A total of 3 trials were performed, each consisting of 10 target slice test cases. Users were allowed to practice the two techniques before the trials. The presentation order of the target slices, as well as the technique used first, was randomized.

The basic hypothesis in this study was that the sketch-line technique would need fewer mouse clicks and less overall time to position and orient the slice plane with respect to the skull surface. The hypothesis was verified ($p < 0.03$) (see [9] for details of the study). As expected, the Push/Rotate technique was dominated by slice rotation input actions whereas the sketch-line technique was almost entirely dominated by the quick curve sketching action and seldom required fine tuning with the standard widget controls. A questionnaire asked the participants whether the sketch-line and push/rotate techniques were easy to control and easy to learn. The results (Table 1) were roughly equal with Push/Rotate scoring better on both questions. However, the participants were also asked to state their favorite technique. The results were that 2/11 listed push/rotate as their favorite while 9/11 favored sketch-line. The two who preferred push/rotate commented that they found it difficult to know what to expect for the slice position and orientation after sketching.

Table 1. Sketch-lines vs. Push/Rotate questionnaire results

5-Point Likert Scale	Push/Rotate	Sketch-Line
Easy to Learn	Avg. 4.64 (Std. 0.67)	Avg. 4.18 (Std. 0.75)
Easy to Control	Avg. 4.18 (Std. 0.87)	Avg. 3.91 (Std. 0.54)

4.1 Sketch-Lines versus Tracing and Painting

A second user study compared sketch-lines against two other common interaction metaphors: tracing and painting. Users were asked to delineate 2D contours using the 3 techniques. We were interested not only in efficiency and simplicity but also in issues of control and precision. The time taken to delineate, the editing time required, as well as other interaction model specific quantities were measured. A 2D delineation task was chosen for several reasons. The most basic reason was that equivalent systems of 3D tracing and painting that were readily comparable to our VTK-based implementation were not available at the time of the user study. We were also concerned that different systems would have different interaction controls and/or differences in interaction model and volume rendering efficiency and quality, especially compared to our somewhat restrictive and slow VTK-based prototype system. We conjectured that a 2D contour delineation task, where all methods were implemented within our VTK framework, would use highly similar input actions as their 3D counterparts and therefore useful information could still be gleaned from such an experiment. Furthermore, there is no depth control requirement and highly similar interface controls using a mouse could be used for all 3 models. In addition, the 2D contour delineation task is very simple to explain and to grasp by naive users. Finally, designing target contours, measuring delineation time and precision is very simple in 2D.

In our tracing implementation the user used a mouse to move the cursor along the target contour, tracing out a smooth spline curve with a user-definable control point spacing. The user can, at any time, stop and select a spline control point and edit the shape of the spline by repositioning the control point. The user may also use a slider positioned in a window region to the immediate right of the target contour to change the distance between control points as the tracing proceeds. In the painting implementation, the uses moves a circle (a "paintbrush tip") around the interior of the target contour and the circles are blended to form a smooth closed contour. To edit the painted contour, the user holds down the right mouse button and the circle becomes an "eraser". The user may also use a slider to change the size of the brush tip. Tracing and Painting were chosen for comparison as they are both common, are familiar to non-expert users, and are simple to learn.

Eleven people (undergrad. and grad. computer science students, with no overlap from user study 1) participated in the within-subjects study, 8 males and 3 females with an average age of 22 years, 35 hours of mouse usage per week and 8 hours of video games per week. The study took approximately one hour to complete. Each trial consisted of 10 contour matching tasks, with contours of different shapes and lengths and shape complexity (see [9] for details). The users were asked to delineate each target contour as quickly and as precisely as possible, initially without editing it. In the last two trials, the users could edit their result. Each user performed 3 trials and the delineation technique (sketch, trace, paint) as well as the presentation order of the contours was randomized. Each user was allowed to practice each of the 3 techniques. The first hypoth-

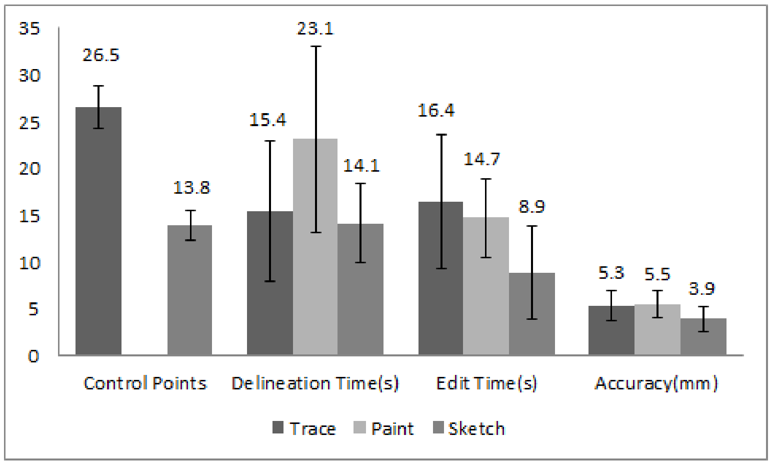

Fig. 6. Sketch-lines vs. Tracing and Painting

esis we formed was that the sketch-line model would be more precise (without contour editing) than paint and trace due to the simple and precisely controllable input actions. Similarly, a second hypothesis was that sketching would be faster (with editing) than paint and trace and that the editing time would be less. A final secondary hypothesis was that sketching would require fewer control points to form the spline curve than tracing, assuming no post processing of the traced spline curve. The results are summarized in Figure 6. The first hypothesis was validated ($p < 0.005$ for sketch vs. trace and $p < 0.003$ for sketch vs. paint). The sketched contour was more accurate than the traced or painted contour. Sketch-lines were also faster (without editing) and the variation in contour delineation times smaller than either trace or paint but this result was only statistically significant for sketch vs. paint ($p < 0.02$). The second hypothesis was also validated. When editing time is included, both the average time required to delineate the contours using sketch-lines as well as the average editing time required were significantly less for sketch-lines than trace ($p < 0.003$) and paint ($p < 0.001$). The final hypothesis was also validated (see "Control Points" in Figure 6). Sketch-lines required significantly fewer control points to accurately delineate the contours than tracing ($p < 3.9E-07$). According to

Table 2. Averaged responses to a questionnaire. Standard deviation is in brackets

5-Point Likert Scale	Trace	Paint	Sketch
Easy to Learn	4.91 (0.30)	4.36 (0.92)	4.82 (0.42)
Easy to Control	4.09 (0.94)	3.18 (1.33)	4.27 (0.79)

a questionnaire (Table 2), all participants scored Tracing and Sketching easy to control and learn. Painting scored slightly lower on easy to control although this result must be considered in the context of the precise delineation task. The participants were also asked to state their favorite technique. Since tracing and painting techniques are similar to simple pen tracing and painting on paper, we expected sketch-lines might not be chosen as a favorite. Nevertheless 4/11 users preferred sketching while 2/11 liked painting and 5/11 preferred tracing. Some comments were that painting is easy to use for a quick rough outline of a target contour. People who liked sketch commented that it was fast, precise, easy to edit, easy to learn, easy to visualize and convenient. Those who liked tracing commented that it was efficient and easy to focus resulting in fewer mistakes. Participants were also asked to name their least favorite technique. The results were that 7/11 disliked painting for the delineation task commenting that paint was hard to predict and control precisely without making mistakes. This result reinforces the idea that a painting style interaction is perhaps best used for fast exploration where precision is not as important. Only one user disliked sketching because it required the user to sketch precisely. For tracing, 3/11 disliked it and commented that it was hard to trace using the mouse and get an accurate result. Our experience also found that long tracing-like input actions where precision is important are tedious and require a higher level of user concentration.

5 Conclusion

The sketch-line interaction model supports simple, direct and precise input actions that nonetheless transfer a significant amount of user information to volume visualization algorithms. It complements and augments existing image slice widget interaction by allowing slices to be quickly positioned orthogonally to object surfaces. The object-relative sketch input actions provides a more seamless bridge between a volume rendered data views and (oblique) image slice views. By combining the interaction model with a spline curve, a subdivision surface and an extrusion process, the user is able to sketch editable geometric meshes that accurately delineate or surround regions of interest. Two user studies support the effectiveness and efficiency of the sketch-line model for tasks that require precise control and accuracy. The algorithm is not limited to volume images - sketch lines can be drawn on triangulated surface meshes.

There are many areas for improvement and extension. Firstly, it is somewhat difficult to sketch on narrow anatomical structures, such as small arteries. An accurate volume image interpolation and zoom feature is needed. Also, in noisy medical images, the volume rendering of objects results in "fuzzy" surfaces that are also difficult to sketch on. More surface samples within a larger region around the cursor and a smoothed volume image may help with this issue. The VTK-based implementation is rather limiting and needs to be replaced with a more powerful volume rendering system. While envelopes can be created and manipulated in real time, both the VTK cutaway operation and the deformable model segmentation algorithm require GPU acceleration. Finally, local or global self-intersection prevention of the various envelopes is currently not implemented and relies on user editing.

References

1. Hinckley, K., Pausch, R., Goble, J., Kassell, N.: A survey of design issues in spatial input. In: 7th annual ACM symposium on User interface software and technology (UIST'94). (1994) 213–222
2. Aliroteh, M., McInerney, T.: Sketchsurfaces: Sketch-line initialized deformable surfaces for efficient and controllable interactive 3d medical image segmentation. In: International Symposium on Visual Computing (ISVC). (2007)
3. Tory, M., Moller, T., Atkins, M., Kirkpatrick, A.: Combining 2d and 3d views for orientation and relative position tasks. In: CHI '04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. (2004) 73–80
4. Fuchs, R., Welker, V., Hornegger, J.: Non-convex polyhedral volume of interest selection. Computerized Medical Imaging and Graphics **34** (2010) 105–113
5. Pühringer, N.: Sketch-based modelling for volume visualization. Master's thesis, Vienna University of Technology (2009)
6. Bruyns, C., Senger, S.: Interactive cutting of 3d surface meshes. Computer & Graphics **25** (2001) 635–642
7. Chen, H., Samavati, F., Sousa, M.: Gpu-based point radiation for interactive volume sculpting and segmentation. Visual Computer **24** (2008) 689–698
8. Weiskopf, D., Engel, K., Ertl, T.: Interactive clipping techniques for texture-based volume visualization and volume shading. IEEE Transactions on Visualization and Computer Graphics **9** (2003) 298–312
9. Shih, Y.: A sketch-line interaction model for image slice-based examination and region of interest delineation of 3d image data. Master's thesis, Dept. of Computer Science, Ryerson University, Toronto, ON, Canada (2012)