Eurographics Conference on Visualization (EuroVis) 2013
B. Preim, P. Rheingans, and H. Theisel
(Guest Editors)

Volume 32 (2013), Number 3

ViviSection: Skeleton-based Volume Editing

A. Karimov[1], G. Mistelbauer[1], J. Schmidt[1], P. Mindek[1], E. Schmidt[2], T. Sharipov[3], S. Bruckner[4] and E. Gröller[1]

[1]Institute of Computer Graphics and Algorithms, Vienna University of Technology, Vienna, Austria
[2]Stockerau Regional Hospital, Stockerau, Austria
[3]Republican Clinical Hospital of G. G. Kuvatov, Ufa, Russia
[4]Department of Informatics, University of Bergen, Bergen, Norway

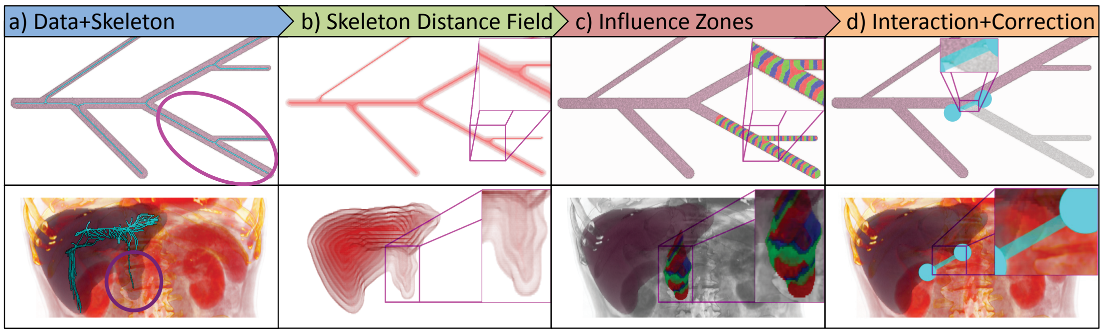

Figure 1: *Example of the correction for a synthetic dataset (top row) and a liver segmentation dataset (bottom row) using our new approach: (a) features for the correction are circled, (b) iso-surfaces of the skeleton distance field are shown, (c) each color band represents a certain influence zone, (d) user interaction is shown together with the correction results.*

Abstract

Volume segmentation is important in many applications, particularly in the medical domain. Most segmentation techniques, however, work fully automatically only in very restricted scenarios and cumbersome manual editing of the results is a common task. In this paper, we introduce a novel approach for the editing of segmentation results. Our method exploits structural features of the segmented object to enable intuitive and robust correction and verification. We demonstrate that our new approach can significantly increase the segmentation quality even in difficult cases such as in the presence of severe pathologies.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.6]: Methodology and Techniques—Interaction techniques

1. Introduction

Volume editing is a highly demanded task during modeling of real-world objects, computer simulation, and volume segmentation. This paper focuses on the correction of volume segmentation in medical image processing. The major problem here is that any segmentation technique may fail due to various pathological conditions and different scanning protocols. The goal of volume editing is to efficiently correct the segmentation. Many segmentation algorithms suffer from two types of potential misclassifications:

Under-segmentation appears in the regions of the object, which are not recognized by a segmentation procedure as being part of the object. Usually it is caused by pathological conditions like cysts or tumors, which change the radiological density of the affected regions. This forces a classifier of the segmentation procedure to believe that the regions contain another type of tissue, than the object itself. A way to treat such a problem is to provide volume editing tools, which allow domain experts to easily add the misclassified regions to the object.

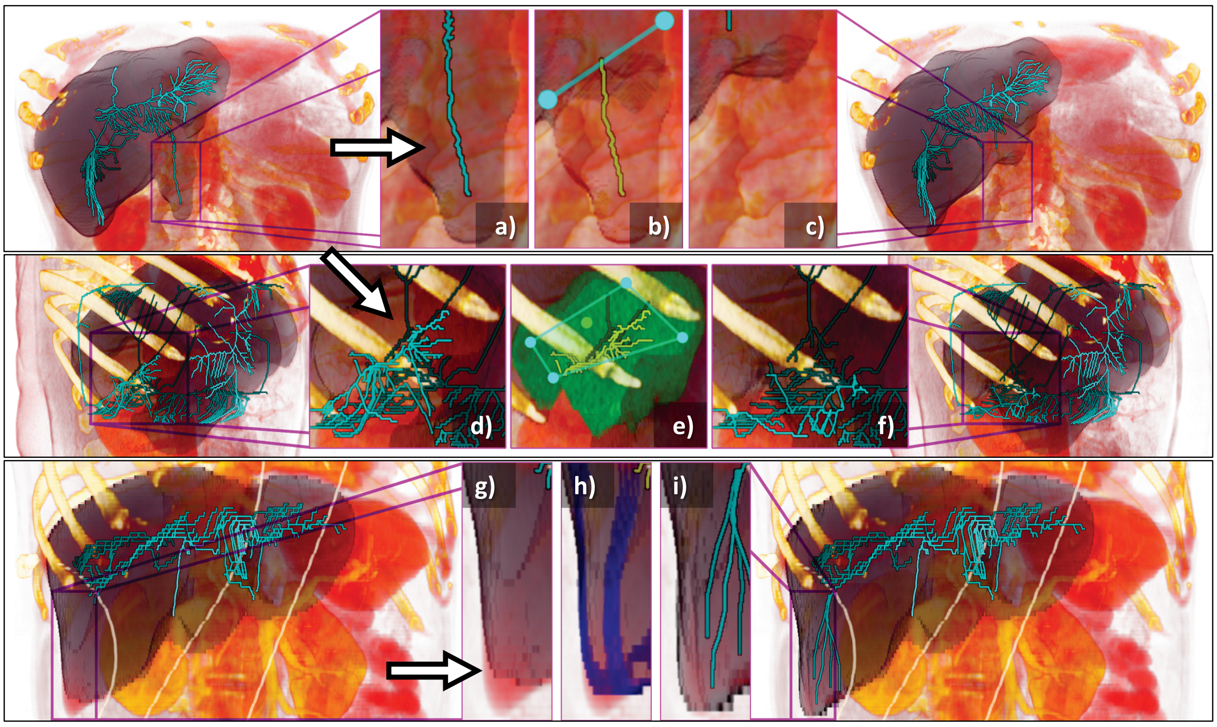

Figure 2: *Manipulation on the skeleton and the structural features for the correction: (a) a region with over-segmentation, (b) manipulation on (a), (c) correction of (a) by result of (b), (d) a region with under-segmentation, (e) manipulation on (d), (f) correction of (d) by result of (e), (g) a region with boundary artifacts, (h) manipulation on (g), (i) correction of (g) by result of (h). (a-c, g-i) depict the skeletons of the objects, (d-f) depict the skeletons of the compliments of the objects in the volumes.*

Over-segmentation happens when some regions of the background are recognized as being part of the object by the segmentation procedure. It can, for instance, be caused by pathological conditions such as fibrosis or steatosis, which change the radiological density of the object itself. This compromises boundary identification between the object and surrounding tissues, because the classifier cannot distinguish them anymore. This problem requires tools which enable the quick removal of background regions from the object.

Additionally, local artifacts in the boundary of the detected object can occur due to noise in the data or pathological conditions like inflammation. The segmentation problems are illustrated in Figure 2 (a, d, g). An effective volume editing approach allows efficient manual corrections of these misclassifications and artifacts.

However, current volume editing approaches suffer from several issues. The main problem is an accurate selection of the features the user wants to correct. In the two-dimensional case the user can work on a pixel level, achieving a high segmentation quality. In 3D, however, the additional dimension makes volume editing on a voxel level laborious, time-consuming, and error prone. The selection should require as little interaction as possible. Thus, it is essential to provide smart tools which ease the selection of features. Furthermore, if the selection or the correction of an isolated feature affects other features of the object as well, then the editing process is not consistent. This happens when the method cannot distinguish different features of the object. These concerns lead to the conclusion that segmentation correction requires an interactive, predictable and robust volume editing method. The selection of the features should be precise and fast. The method has to detect the individual features and to process them separately.

We propose ViviSection, a volume editing method, which uses the skeleton of the object in order to detect structural features. These structural features represent the characteristic parts interesting to the user during correction. The method is based on a novel skeleton distance field technique and an influence zone concept. The components of the technique are illustrated in Figure 1. While our approach is applicable to other scenarios, we specifically focus on liver segmentation in X-ray Computed Tomography (CT) data, which is of high clinical importance. Due to a substantial degree of variation across scanners and patients, particularly in the presence of pathologies, current liver segmentation algorithms frequently require manual corrections [HvGS*09].

The remainder of this paper is organized as follows. The next section discusses related work. Section 3 describes ViviSection in detail. Performance and scalability of the method are analyzed in Section 4. Section 5 presents results and evaluation of our technique. Limitations and potential future extensions of our work are discussed in Section 6. The paper is concluded in Section 7.

2. Related Work

2.1. Intelligent Scissors, Live Wire & Live Surface

Intelligent scissors are a 2D selection technique proposed by Mortensen and Barrett [MB95]. The user selects a "start node", a pixel on the image, where the boundary of the object starts. Then the user places a "free point" where the detected boundary should end. As the user moves the free point, a "live wire" is automatically placed, snapping to the boundary. To achieve such a behavior discrete dynamic programming is used to trace the optimal boundary between the start node and the free point. Finally, the user locks the free point, making it a "goal node". Then the next part of live wire is placed, assuming the last goal node to be a new start node. The boundary, composed of all live-wire segments, runs through all the nodes.

Processing time strongly depends on the image resolution. The interactivity issue is addressed by Wong et al. [WHW00]. They propose a "slimmed graph" in order to process large images interactively. Intelligent scissors use the image gradient, so they are sensitive to noise. Enhanced intelligent scissors proposed by Mishra et al. [MWZ*08] use a phase-based image representation and formulate and solve the optimal boundary extraction problem as an active contour problem. Enhanced intelligent scissors, as stated by the authors, are more tolerant to noise and require fewer nodes than conventional intelligent scissors.

An extension of live wire for volume editing was proposed by Hastreiter and Ertl [HE98]. They focus on brain segmentation from Magnetic Resonance Imaging and Computed Tomography scans. The user defines contours in slices through the live-wire tool. The difference to conventional intelligent scissors is that their live-wire placement procedure takes into account slabs, not individual slices. Then the contours are propagated to adjacent slices automatically.

An analogy of live wire for volume editing was presented by Armstrong et al. [APB07]. They propose a "live surface", which is defined by user interaction and refined by an optimization algorithm. The idea is the following: the user brushes "object" and "background" seeds in a special cross-section plane. Additionally, mouse interaction on the live surface is taken into account to either get more object voxels, or remove background voxels. Although the method allows segmentation correction, it does not extract features for editing. Instead it concentrates on sequential refinements, based on the user input.

2.2. Graph Cuts

For graph cuts a graph is constructed from voxels serving as its vertices, and connections between adjacent voxels serving as its edges. A cost is assigned to an edge according to characteristics of both voxels. A graph cut is basically a graph partitioning: voxels in the first part belong to the object, and voxels in the second part belong to the background. The partitioning depends on user-classified voxels ("object terminals" and "background terminals"). The optimal cut is found with respect to the sum of the costs of the edges in the cut-set. Hard constraints ("seeds") are applied to correct the position of the graph cut locally. Both terminals and seeds are placed by sketching. The method may be utilized for segmentation correction. A drawback is that terminals and seeds have to be placed manually instead of supporting feature-based selection and editing. A survey on graph cuts can be found in the works of Boykov et al. [BFL06, BV06].

2.3. Skeletonization Techniques

For our method we chose a thinning-based skeletonization. The main idea of thinning-based skeletonization techniques [LKC94, PK99, PSB*01] is the iterative removal of surface voxels under geometrical and topological constraints, i.e., preservation of end-points, cavities and holes of the object and desired width, connectedness and centered location of the skeleton [LKC94]. Distance-field-based methods [ST04, vDvdWT06, XT09] treat ridges of a distance field as skeleton voxels. Different types of distance fields result in different skeleton characteristics [JBS06]. Some approaches search for a skeleton by constructing Voronoi diagrams [OK95, AM97, NSK*97]. For a survey on skeletonization we refer the reader to Prohaska's Ph.D. thesis [Pro07].

A main reason for choosing thinning-based techniques is that the given constraints are useful for the detection and editing of segmentation problems. Detection and editing of over-segmentation requires the preservation of end-points in the skeleton. Preservation of holes and cavities in the skeleton is essential for the detection and editing of under-segmentation. Connectedness of the skeleton is necessary for boundary artifacts editing. For distance-field-based methods the given constraints can be violated. For example, connectedness is not guaranteed without additional post-processing. Inaccuracy of the skeleton due to the discrete nature of a volume grid is tolerable, thus time-consuming Voronoi-diagram-based approaches are not necessary.

2.4. Curve Skeletons

The curve skeleton of a three-dimensional object is a set of lines, which are centered around the object's surface [RvWT08]. Cornea et al. [CSYB05, CSM05] propose to decompose an object into segments according to individual lines of the curve skeleton. The segments may represent features like over-segmented or under-segmented regions.

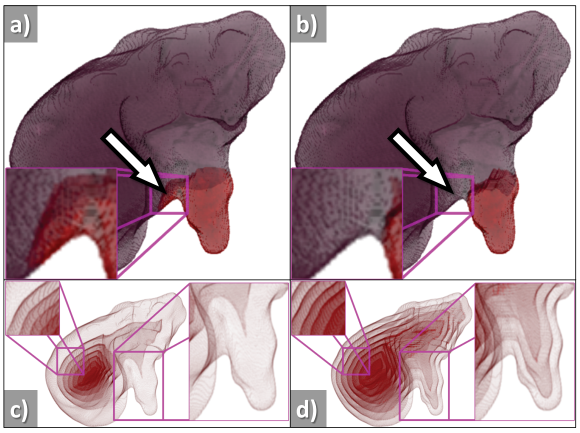

Figure 3: *Comparison of (a, c) the Euclidean and (b, d) the skeleton distance fields for the influence zones calculation. (a, b) The features are selected by the same set of skeleton voxels in both cases. Note the "leak" of the selection if the Euclidean distance is used. (c, d) Iso-surfaces of the same values are shown in both cases. The left close-up views show a region with only one object structure. No strong difference appears. The right close-up views show a region with two object structures. Their delineation based on the skeleton lines in the skeleton distance field causes strong difference.*

Such a segmentation allows volume editing on the level of the detected segments only. It is not sufficient for segmentation correction for two reasons. First, corrections cannot be fine-tuned since segments are not splittable into smaller parts. Second, small-scale corrections are not possible since only large "core" segments are available. Also for this skeletonization one must specify thresholds for divergence and curvature. Cornea et al. [CSYB05] mention that there is no optimum for these thresholds. The presence of segmentation problems may require to tune them during the correction.

Reniers et al. [RvWT08, RT08] propose an importance measure which controls the skeletonization process. Additionally, it matches individual skeleton voxels to the object voxels, which allows volume editing on a more detailed level. The only drawback is that the skeleton is determined by thresholding of importance measure.

3. ViviSection

Our proposed method enables volume editing based on structural features. The pipeline of the method is illustrated in Figure 1 for two datasets: a synthetic dataset with branchings and a liver segmentation dataset. First, skeletonization of the segmentation mask of the object is performed. We use a thinning-based skeletonization [LKC94]. During the skeletonization we also compute a skeleton distance field, which orders the object voxels with respect to the skeleton. Then we build influence zones. Each influence zone is defined by

Algorithm 1: Calculation of the influence zones using the skeleton distance field.

```
forall the voxel $v \in Volume$ do                                  // initialization of
    if $v.SDF = MaxSDF$ then                                         // skeleton voxels
        $v.State \leftarrow Processed$; $v.IZ \leftarrow v$; $F \leftarrow F \cup \{v\}$
    else $v.State \leftarrow None$;                                  // and non-skeleton voxels
$SDF \leftarrow MaxSDF - 1$                                          // begin at the skeleton of the object
while $SDF > 0$ do                                                   // end at the surface of the object
    $Q \leftarrow \emptyset$; $F_{new} \leftarrow \emptyset$
    forall the voxel $v \in F$ do                                    // fill the queue Q
        forall the voxel $w \in N(v)$ do
            if $w.State = None$ then                                 // if unprocessed neighbor w found
                if $w.SDF = SDF$ then                                // then enqueue w for
                    $w.State \leftarrow Ready$; $Q \leftarrow Q \cup \{w\}$   // processing
                else if $w.SDF > 0$ then $F_{new} \leftarrow F_{new} \cup \{v\}$;   // or keep v
    repeat        // process all voxels with skeleton distance equal to SDF
        $Q_{new} \leftarrow \emptyset$
        forall the voxel $v \in Q$ do                                // process the queue Q
            $L \leftarrow \emptyset$
            forall the voxel $w \in N(v)$ do                         // store IZ of processed
                if $w.State = Processed$ then $L.Add(w.IZ)$          // neighbor w
                if $w.State = None$ then                             // if unprocessed neighbor w found
                    if $w.SDF = SDF$ then                            // then enqueue w for
                        $w.State \leftarrow Ready$; $Q_{new} \leftarrow Q_{new} \cup \{w\}$   // processing
                    else if $w.SDF > 0$ then $F_{new} \leftarrow F_{new} \cup \{v\}$;   // or keep v
            $v.IZ \leftarrow L.MostFrequentElement()$                // assign IZ to v
        forall the voxel $v \in Q$ do $v.State \leftarrow Processed$
        $Swap(Q, Q_{new})$                                           // swap the queues
    until $Q = \emptyset$
    $Swap(F, F_{new})$; $SDF \leftarrow SDF - 1$                     // swap the fronts
```

a certain skeleton voxel and contains a set of the object voxels which are linked to this skeleton voxel. The influence zones are determined using the skeleton distance field. They represent low-level structures of the object. A structural feature is defined as follows. First, a set of the skeleton voxels is chosen. The set should contain only one connected component of the skeleton voxels. Then, we add object voxels, which are in the influence zone of any of the chosen skeleton voxels. An incorrect segmentation differs from a correct segmentation by over-segmented, under-segmented regions and boundary artifacts. The regions can be represented as structural features and the artifacts can be selected on the surface of structural features. Operations on the skeleton and the structural features can resolve segmentation problems, as illustrated in Figure 2.

We present several simple selection tools and operations on the skeleton and the structural features. The idea of these tools is to perform a selection operation of the skeleton voxels based on the user input. The structural features, defined by the selected skeleton voxels, are then corrected according to the selected operation. The correction is previewed, so the user may refine it by additional input.

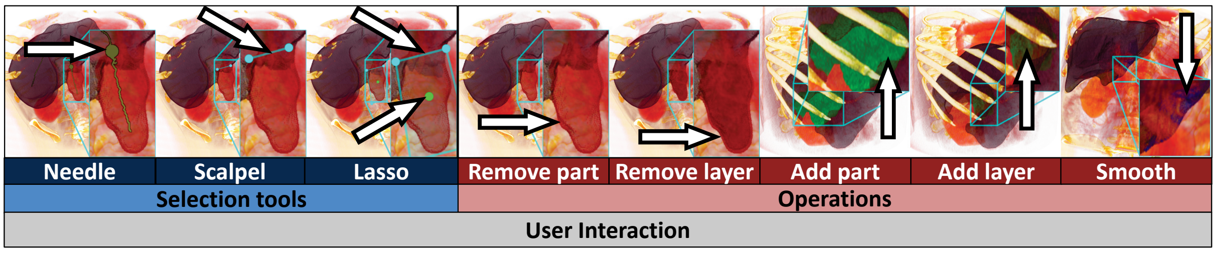

Figure 4: *The selection tools and the operations available during user interaction. Previews of changes are shown.*

3.1. Influence Zones

The influence zones can be determined in various ways. The criterion is that each object voxel should either be a skeleton voxel or belong to one and only one influence zone, i.e., be linked to just one skeleton voxel. To determine influence zones we may use the Euclidean distance. An object voxel o is linked to the skeleton voxel p only if the distance $d(p,o)$ is less than the distance to any other skeleton voxel. This link definition does not take the structures of the object into account, as it simply uses the shortest path between the two voxels. Thus, such a definition may lead to wrong influence zones in regions, where two or more object structures are presented. If some voxels of the first structural feature are incorrectly linked to the second feature due to wrong influence zones, the selection may "leak" leading to undesired result as illustrated in Figure 3 (a). As the selection should not interfere with other features unless explicitly specified to do so by the user input, we use a skeleton distance field to include the delineation, based on the skeleton lines, into the definition of influence zones.

3.2. Skeleton Distance Field

The skeleton distance field can be computed during any thinning-based skeletonization. The main idea is that thinning iterations order the object voxels. By assigning the iteration number to each currently removed voxel, we receive a scalar field defined on the volume grid. We assume that the field is zero for voxels outside of the object. As skeleton voxels are never removed by the skeletonization, we assign them the value $MaxSDF$, which equals to the number of thinning iterations plus one. The influence zone for a voxel o is determined by inspecting its neighborhood. If o has a neighbor s, which is a skeleton voxel, then the voxel o is in the influence zone of s. If o with a skeleton distance a has a neighbor r with a skeleton distance b and $b > a$, then the voxel o is in the same influence zone as the voxel r. With these two rules the influence zones grow from the skeleton voxels into the object voxels. Further the voxels are from the skeleton, earlier they are removed during thinning, so the values of the skeleton distance field decrease. The influence zones grow from voxels with higher skeleton distances to voxels with lower skeleton distances. This is the reason why the $b > a$ condition is sufficient to determine the influence zone in the absence of the skeleton voxels in the neighborhood.

Algorithm 1 provides details about the influence zones calculation using the skeleton distance field. Each voxel v has the following attributes. *v.State* stores a state of the voxel: *None* if the voxel is initialized, *Ready* if the voxel is enqueued for processing, *Processed* if the voxel is processed. *v.SDF* stores the skeleton distance. *v.IZ* stores a link to the skeleton voxel, which influence zone voxel v belongs to. $N(v)$ returns a set of neighboring voxels of voxel v. Sets F and F_{new} ("fronts") store the voxels, which influence zones still propagate. Sets Q and Q_{new} ("queues") store the voxels for processing. List L stores the influence zones in the neighborhood of voxel v and returns the most frequent one by $.MostFrequentElement()$. Voxel v is assigned to this influence zone.

The difference between the skeleton distance field and the Euclidean distance field is caused mainly by the delineation based on the skeleton lines. The Euclidean distance is propagated from voxel to its neighbors. The skeleton distance depends on the skeleton lines, which represent object structures. If only one object structure is located in the region, then the skeleton distance is propagated in the same way as the Euclidean distance. This case is illustrated in the left close-up views of Figure 3 (c, d). There the Euclidean and the skeleton distances change between layers, parallel to the surface of the object.

If two or more object structures are located in the region, then their delineation comes from the skeleton lines. This case is illustrated in the right close-up views of Figure 3 (c, d). The Euclidean distance field does not take the skeleton lines into account, so the Euclidean distance propagates between voxels of different object structures. It forces the delineation of the resulting structural features to fail as illustrated in Figure 3 (a). The skeleton distance field uses information on the skeleton lines, thus the skeleton distance does not propagate between voxels of different object structures. Due to this the skeleton distance field allows the correct delineation of the resulting structural features as illustrated in Figure 3 (b). Here the selection of one feature does not "leak" into other features.

3.3. User Interaction

Selection tools and operations are two main components of our user interface. The selection tools provide an interface for the selection of features to be corrected. Each selection tool defines a set of graphic objects, which are manually placed by the user on the projection of the volume. The operations implement common volume editing tasks. If the user selects a different operation or selection tool or interacts with the selection tool, the selected feature for the correction is computed and displayed interactively. Additionally, the user can inspect the feature and results of the operation in slice views. The selection tools are illustrated in the part "selection tools" of Figure 4.

The **needle** is the most simple tool. The user specifies a point at the surface of the object in 3D space. Then, the skeleton voxel of the influence zone the point belongs to is located. The skeleton voxel splits the skeleton into two or more parts. The smallest part of the skeleton (according to the number of voxels) defines the structural feature selected for correction.

The **scalpel** tool operates on the structural features of the object or the background during the selection instead of defining a simple geometrical delineation as it is done in similar tools of other editing methods. This allows accurate feature selection from the detected structures in two mouse clicks instead of a cumbersome allocation and refinement of a separation plane or surface in slices. The user specifies two points in screen space, which define a line in screen space and a plane in 3D space. The skeleton voxels are split into two parts according to the negative and positive half-spaces of this plane. The smallest part of the skeleton (according to the number of voxels) defines the structural features selected for correction.

The **lasso** is the most powerful tool, but requires more interaction than the needle and the scalpel. The user specifies a selection point and a selection polygon in screen space. The polygon may consist of an arbitrary number of points and can be either convex or concave. It defines a prism in 3D space. The skeleton voxels inside the prism define the candidate structural features. The selection point defines a ray in 3D space to discriminate the features. If the ray crosses one of the candidate features, that feature is exclusively selected. Otherwise all the candidate features are selected.

For the needle and the scalpel selection tools it is assumed that most voxels of the object or the background are correctly classified. Consequently, most skeleton voxels do not require correction.

The operations "add part" and "remove part" are designed for the correction of large-scale structural features. Small-scale structural features can be considered as local artifacts of the boundary of the object. To correct these artifacts "remove layer", "add layer", and "smooth" operations can be used. These three operations modify the boundary of the selected structural feature. To connect or disconnect parts of the object after the operations post-processing is used. The operations are illustrated in the part "operations" of Figure 4.

The **add part** operation resolves under-segmentation problems by adding to the object the selected feature from the compliment of the object in the volume. Morphological closing with a kernel size of 6 voxels is applied as a post-processing step.

The **remove part** operation treats over-segmentation problems by removing the selected feature from the object. Morphological opening with a kernel size of 6 voxels is applied as a post-processing step.

The **add layer** operation grows a new layer, which has a thickness of $\sqrt{3}$ voxels, on top of the surface of the selected feature. Morphological closing with a kernel size of 3 voxels is applied as a post-processing step.

The **remove layer** operation peels away the surface layer of $\sqrt{3}$ voxels thickness from the selected feature. Morphological opening with a kernel size of 3 voxels is applied as a post-processing step.

The **smooth** operation replaces the selected feature with its smooth version. We use Gaussian blurring with a kernel size of 11 voxels and a threshold of 0.5 in order to compute a binary representation of the smooth version of the feature.

4. Performance and Scalability

Influence zones are pre-calculated to provide an interactive selection of the structural features during individual operations. For the performance evaluation we measured the following: pre-calculation time (generation of the skeleton, the skeleton distance field, and the influence zones), preview generation time (transformation of the user input into the selected structural features) and post-processing time (morphological operations).

We investigated twenty datasets in a quantitative evaluation. Pre-calculation times range from 2.8 to 16.7 seconds (average 8.2 seconds) depending on the volume size and the object's complexity. We use the skeletonization algorithm by Homann [Hom07] which is based on the work of Lee et al. [LKC94]. Parallelization is done via OpenMP on the CPU for up to 8 parallel threads. The skeleton and the skeleton distance field are calculated simultaneously. The generation of the influence zones can take up to 25% of the pre-calculation time. Preview generation times are in range from 0.1 to 0.25 seconds depending on the skeleton complexity. Post-processing takes from 2 to 5 seconds depending on the volume size. Post-processing is executed via OpenCL on the GPU. All reported timings were measured on an Intel Core i7-2600K 3.4 GHz CPU equipped with 16 GB of RAM and an NVidia GeForce GTX 680 GPU with 4 GB of video memory.

Table 1: *Measurements for the scalability analysis.*

Dataset	**01**	**02**	**03**	**04**	**05**
# of skeleton lines	13094	6099	2354	371	291
Pre-calculation time, sec.	10.6	6.4	5.5	5.1	4.8
Preview generation time, sec.	0.9	0.8	0.7	0.4	0.3
Post-processing time, sec.	6.8	5.5	5.1	5.1	4.9

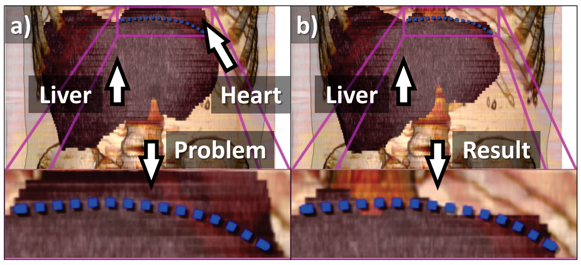

Figure 5: *Comparison of (a) the initial segmentation and (b) the corrected segmentation of the object. The object is a liver, affected by Non-Alcoholic Fatty Liver Disease. The initial segmentation has severe over-segmentation towards the heart. The blue dotted line shows the desired delineation of the organs.*

Scalability is evaluated with respect to the skeleton complexity, measured as the number of individual skeleton lines. Magnetic Resonance Angiography (MRA) datasets with a resolution of $512 \times 512 \times 512$ voxels and containing brain vasculature segmentation are used. Time measurements are given in Table 1. As the skeleton complexity decreases, both pre-calculation and preview generation times decrease. The morphological operations, which are the part of the post-processing time, are independent of skeleton complexity.

Every time the segmentation mask changes during interaction, the skeleton, the skeleton distance field, and the influence zones need to be updated. As this information is global, local partial updates in the region of changes are not feasible.

5. Evaluation & Results

During the evaluation an automatic liver segmentation technique was used for initial segmentation. It is analogous to region growing, but instead of individual voxels the segments from a 3D watershed segmentation are considered. The initial seed point is found automatically from the automatically segmented lung lobes. Then radiological density distributions are used to grow the segmented region and stop at the boundary of the liver. The technique is targeted for Computed Tomography Angiography (CTA), and not standard Computed Tomography.

5.1. Severe Pathological Cases

To demonstrate the value of our approach in difficult clinical scenarios we conducted a study using liver segmentations for patients affected by Non-Alcoholic Fatty Liver Disease (NAFLD) [RS05]. The liver is hardly differentiable from other organs on CT scans. To differentiate it better contrast-enhancing agents are used in CTA. The patient was scanned with a standard CT protocol after the diagnosis confirmation. Due to the NAFLD the liver parenchyma accumulated fat, causing radiological density to drop down to $10-30$ HU from the typical $50-70$ HU [OCWR00]. Usage of the standard CT protocol in this severe pathological case cause a failure of the applied segmentation technique. The problem here is that the boundaries between the liver and other organs (like kidneys, stomach, heart and bowels) are too weak, so that even a trained operator cannot clearly distinguish them. In this case the segmentation technique produced severe over-segmentation, mainly towards the heart. Severe segmentation problems may change the skeleton of the object drastically, as it happens in this case. The correction of such a segmentation provides a robustness test for our method. The domain expert at the hospital used eight operations. He spent approximately seven minutes for the corrections. The calculations required 93 seconds in total. Ultimately, the expert achieved the desired segmentation, as given in Figure 5.

5.2. Quantitative Evaluation

We compare our method with existing approaches in terms of segmentation quality and interaction time. The goal of the SLiver 2007 contest [HvGS*09] was to find a liver segmentation technique which performs best and minimizes user interaction. At the contest twenty datasets were provided together with ground truth segmentations done by domain experts. Each dataset is a CTA scan with a slice resolution of 512×512 pixels.

First the applied segmentation technique provided an initial segmentation for each dataset. Various kinds of pathologies caused over-segmentation, under-segmentation, and boundary artifacts. Then the segmentation was corrected by a trained operator in ViviSection. We evaluated the quality of the segmentation before and after the correction and measured the time spent on the segmentation and the correction. The quality metric is defined as $Q = \frac{|A \cap B|}{|A \cup B|}$, where A is a set of liver voxels in the segmentation, and B is a set of liver voxels in the ground truth. Q equals to one if the segmentation completely matches the ground truth. As the segmentation differs more from the ground truth, Q decreases. Q equals to zero if the segmentation does not contain any voxel from the ground truth. Evaluation results are given in Table 2. Figure 2 partially illustrates the correction of: (a-c) over-segmentation in dataset #13, (d-f) under-segmentation in dataset #16, (g-i) boundary artifacts in dataset #03.

Table 2: *Measurements for the quantitative evaluation. Q_{segm} is the quality of the initial segmentation, T_{segm} is the segmentation time. Q_{corr} is the quality after the correction, T_{corr} is the correction time. t_{calc} is the average calculation time per operation, T_{calc} is the total calculation time. The calculation time is included in the correction time.*

Dataset	Slices	Q_{segm}	T_{segm}, sec.	t_{calc}, sec.	Operations	Q_{corr}	T_{corr}, sec.	**Dataset**	Slices	Q_{segm}	T_{segm}, sec.	t_{calc}, sec.	Operations	Q_{corr}	T_{corr}, sec.	**Dataset**	Slices	Q_{segm}	T_{segm}, sec.	t_{calc}, sec.	Operations	Q_{corr}	T_{corr}, sec.
01	183	90%	106	8.5	3	92%	81	**08**	228	91%	158	15.9	5	92%	148	**15**	394	91%	239	14.4	3	95%	113
02	64	89%	45	4.4	2	90%	37	**09**	210	90%	140	9.3	3	92%	56	**16**	151	79%	118	10.9	4	95%	89
03	79	82%	47	3.8	6	90%	52	**10**	191	87%	126	12.2	1	90%	38	**17**	121	86%	81	5.3	3	91%	47
04	212	79%	118	9.8	4	92%	106	**11**	388	76%	208	19.7	11	94%	576	**18**	245	80%	160	11.8	3	95%	70
05	319	89%	161	13.7	2	93%	56	**12**	220	89%	150	10.7	5	95%	134	**19**	335	88%	215	19.7	2	93%	58
06	111	88%	61	4.7	5	92%	44	**13**	145	88%	96	6.3	6	92%	115	**20**	183	81%	107	5.6	2	90%	22
07	251	89%	154	9.9	4	92%	108	**14**	129	87%	73	3.8	9	92%	103								

Segmentation:	$avg(Q_{segm}) = 86\%$	**Correction:**	$avg(Q_{corr}) = 93\%$
	$sum(T_{segm}) = 2563$ sec.		$sum(T_{corr}) = 2053$ sec.
			$sum(T_{calc}) = 837$ sec.

Several contest entries used interactive segmentation techniques. Beichel et al. [BBB*07] proposed a method built upon a graph-cut segmentation and chunk-based and mesh-based refinements. The initial segmentation takes thirty minutes on average. For the correction the user spends on average six minutes to achieve a segmentation quality of 95%. Beck and Aurich [BA07] proposed a region-growing-based semi-automatic segmentation method with a "virtual knife" for correcting over-segmentation. Under-segmentation correction requires new seed points in the region growing procedure. The segmentation and correction time for this method is seven minutes on average for a segmentation quality of 93%. With our method the average segmentation quality rises from 86% to 93%. The technique only requires a modest amount of time for the correction, i.e., 103 seconds on average, including 42 seconds on average for calculations. For comparison the fully automatic initial segmentation took 129 seconds on average. For all datasets the segmentation quality is equal or above 90%. Before the correction the lower bound was 76%. We conclude that ViviSection allows the user to correct the segmentation for any dataset until a certain quality is reached (about 90%). A further increase of the quality is possible only with voxel-level editing, which takes a significant amount of time. Manual segmentations of the entire volumes on a voxel level takes from twenty five minutes to forty minutes per dataset [SEK*11]. The time of manual editing on a voxel level strongly depends on the quality of the initial segmentation. The voxel-level editing increases the quality to 95%-99%.

5.3. Domain Experts Feedback

Three domain experts provided feedback on our method. They corrected liver segmentations from the applied segmentation technique. Both CT and CTA datasets were used.

The first domain expert, a senior clinical radiologist, was familiar with live-wire-based editing to manually obtain liver contours, which takes about seven minutes per dataset for usual cases. He commented that context information available during the editing in 3D is useful for decision making in cases of severe pathologies. He proposed to add a new selection tool like a "brush" operating on the object's surface to speed up interaction in special scenarios. He preferred live-wire-based editing to our scalpel and lasso tools because of better response times and having full control over the correction. However, in his opinion, the case of the severe pathology was easier corrected with our system than with his current software.

The second domain expert, an intern, used in her routine a slice-based editing technique with a manual drawing of liver contours, which takes about twenty minutes per dataset for usual cases. She stated that the selection based on the structural features is sufficiently precise to correct major segmentation defects. To correct small-scale features she suggested a "zoom-in" mechanism. She found the scalpel selection tool very intuitive and easy to use. She also commented that the needle and the lasso tools require more initial training. She also found that a better understanding of the data can be achieved through interactive 3D rendering. The link between 3D rendering and slice views allowed her to select occluded features, although, in her opinion, this also requires additional training. Finally, she stated that response times during individual operations are appropriate, but response times between operations are too long due to the re-computations.

The third domain expert, a clinical radiologist, used live-surface-based and slice-based techniques for the editing. He commented that a link between the slice views and the 3D rendering is useful for the editing of partially occluded features. The main concern from his point of view is the re-

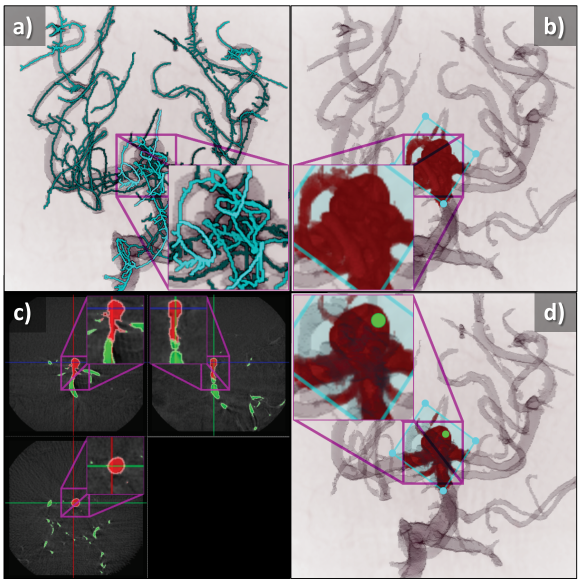

Figure 6: *Selection of a partially occluded structural feature (vessel aneurysm in MRA): (a) skeleton cluttering and self-occlusion, (b) direct interaction with the lasso tool fails to select the feature without the occluder, (c) a link between the 3D rendering and the slice views exploited, (d) the lasso tool with the slice views link allows to select the feature.*

sponse times. He liked the interactive preview of changes in both the 3D rendering and the slice views, as it provided him with the ability to fine-tune operations. He preferred our system over live-surface-based editing, because he can easily and efficiently correct major segmentation defects with our method.

6. Discussion & Future Work

An issue of our method is the dependency on the underlying skeletonization. If the feature of interest is not represented in the skeleton, our approach cannot correct this as no information about the feature is available. For example, small-scale features of the object cannot be properly detected with thinning-based skeletonization techniques. Experimentally we found that features smaller than $5 \times 5 \times 5$ voxels in size are missing in the skeleton. They are not editable using "remove part" and "add part" operations. Treating them as boundary artifacts the user can apply "remove layer", "add layer", and "smooth" operations.

In order to solve the issue of missing small-scale features multi-resolution skeletonization could be utilized. The idea is to merge multiple skeletons with details of different scales across different regions of the volume. As the user concentrates on some region of interest, one can create a high-resolution sub-volume of that region and do a skeletonization on it. With a higher resolution the missing features become larger, so they are detected by the skeletonization. The new skeleton substitutes part of the old one, so the skeletons are merged. Then the user has the possibility to edit the features, which were previously unavailable.

Another issue of our method is that self-obstruction and cluttering in the case of complex skeletons make the selection of structural features more difficult. They might be occluded by other features or located inside the object. However, partial occlusion can be effectively dealt with using our lasso selection tool. This is sufficient, for example, for a vessel segmentation correction as illustrated in Figure 6. For future work we propose to select and temporarily hide occluding structural features. Such a solution could reveal completely occluded structural features, like vasculature, tumors, and cysts in the liver.

Cluttering of the skeleton due to noise or a large amount of details could result in many small skeleton parts. To alleviate this in our method the user does not work with the skeleton directly, but rather selects a region for editing. The scalpel and the lasso tools treat all underlying structural features in the selected region at once.

Skeleton smoothing and pruning methods can simplify complex skeletons generated from the noisy data. Our method can utilize them if the influence zone of each modified skeleton voxel is re-assigned to the appropriate skeleton voxel, which remains after the skeleton simplification. We propose to use the number of voxels in influence zones of each individual skeleton line as a criterion for skeleton pruning.

7. Conclusion

We proposed ViviSection, a novel approach for editing segmentation results based on structural features. For this purpose we introduced the notion of a skeleton distance field, which allows to detect structural features of the object and to operate on them during a correction step. To support the accurate selection of the features our technique employs influence zones. They clearly delineate the structural features. Our approach uses a set of intuitive selection tools and operations, which allow rapid correction of major segmentation defects. We demonstrated the robustness of our method even under severe pathological conditions. Our results show that the presented approach can increase the precision and stability of segmentations with little user interaction.

Acknowledgements

The work presented in this paper is part of the Knowledge Assisted Sparse Interaction for Peripheral CT-Angiography (KASI) project, supported by the Austrian Science Fund (FWF) grant no. TRP 67-N23. The datasets are courtesy of the SLiver 2007 contest and the Republican Clinical Hospital of G. G. Kuvatov in Ufa, Russia.

References

[AM97] ATTALI D., MONTANVERT A.: Computing and simplifying 2D and 3D continuous skeletons. *Computer Vision and Image Understanding 67*, 3 (Sept. 1997), 261–273. 3

[APB07] ARMSTRONG C. J., PRICE B. L., BARRETT W. A.: Interactive segmentation of image volumes with Live Surface. *Computers and Graphics 31*, 2 (Apr. 2007), 212–229. 3

[BA07] BECK A., AURICH V.: HepaTux - A semiautomatic liver segmentation system. In *Proceedings of MICCAI workshop on 3D segmentation in the clinic: A Grand Challenge* (2007), Heimann T., Styner M., van Ginneken B., (Eds.), pp. 225–233. 8

[BBB*07] BEICHEL R., BAUER C., BORNIK A., SORANTIN E., BISCHOF H.: Liver segmentation in CT data: A segmentation refinement approach. In *Proceedings of MICCAI workshop on 3D segmentation in the clinic: A Grand Challenge* (2007), Heimann T., Styner M., van Ginneken B., (Eds.), pp. 235–245. 8

[BFL06] BOYKOV Y., FUNKA-LEA G.: Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision 70*, 2 (Nov. 2006), 109–131. 3

[BV06] BOYKOV Y., VEKSLER O.: Graph cuts in vision and graphics: Theories and applications. In *Handbook of Mathematical Models in Computer Vision*, Paragios N., Chen Y., Faugeras O., (Eds.). Springer US, 2006, ch. 5, pp. 79–96. 3

[CSM05] CORNEA N. D., SILVER D., MIN P.: Curve-skeleton applications. In *Proceedings of 16th IEEE Visualization Conference (VIS 2005)* (2005), pp. 95–102. 3

[CSYB05] CORNEA N. D., SILVER D., YUAN X., BALASUBRAMANIAN R.: Computing hierarchical curve-skeletons of 3D objects. *The Visual Computer 21* (2005), 945–955. 3, 4

[HE98] HASTREITER P., ERTL T.: Fast and interactive 3D-segmentation of medical volume data. In *Computer Graphics International 98* (1998), pp. 78–85. 3

[Hom07] HOMANN H.: Insight Journal - Implementation of a 3D thinning algorithm, 2007. Published: 10.12.2007. Accessed: 25.03.2013. URL: http://hdl.handle.net/1926/1292. 6

[HvGS*09] HEIMANN T., VAN GINNEKEN B., STYNER M., ARZHAEVA Y., AURICH V., BAUER C., BECK A., BECKER C., BEICHEL R., BEKES G., BELLO F., BINNIG G. K., BISCHOF H., BORNIK A., CASHMAN P., CHI Y., CORDOVA A., DAWANT B. M., FIDRICH M., FURST J. D., FURUKAWA D., GRENACHER L., HORNEGGER J., KAINMÜLLER D., KITNEY R., KOBATAKE H., LAMECKER H., LANGE T., LEE J., LENNON B., LI R., LI S., MEINZER H.-P., NÉMETH G., RAICU D. S., RAU A.-M., VAN RIKXOORT E. M., ROUSSON M., RUSKÓ L., SADDI K. A., SCHMIDT G., SEGHERS D., SHIMIZU A., SLAGMOLEN P., SORANTIN E., SOZA G., SUSOMBOON R., WAITE J. M., WIMMER A., WOLF I.: Comparison and evaluation of methods for liver segmentation from CT datasets. *IEEE Transactions on Medical Imaging 28*, 8 (2009), 1251–1265. 2, 7

[JBS06] JONES M. W., BAERENTZEN J. A., SRAMEK M.: 3D distance fields: A survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics 12*, 4 (July 2006), 581–599. 3

[LKC94] LEE T. C., KASHYAP R. L., CHU C. N.: Building skeleton models via 3D medial surface/axis thinning algorithms. *Computer Vision, Graphics, and Image Processing 56* (1994), 462–478. 3, 4, 6

[MB95] MORTENSEN E. N., BARRETT W. A.: Intelligent scissors for image composition. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), SIGGRAPH '95, pp. 191–198. 3

[MWZ*08] MISHRA A., WONG E., ZHANG W., CLAUSI D., FIEGUTH P.: Improved interactive medical image segmentation using enhanced intelligent scissors (EIS). In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (2008), pp. 3083–3086. 3

[NSK*97] NÄF M., SZÉKELY G., KIKINIS R., SHENTON M. E., KÜBLER O.: 3D Voronoi skeletons and their usage for the characterization and recognition of 3D organ shape. *Computer Vision and Image Understanding 66*, 2 (May 1997), 147–161. 3

[OCWR00] O'RIORDAN E., CRAVEN C. M., WILSON D., ROBINSON P. J.: Dual phase hepatic CT: Influence of scanning direction on liver attenuation. *American Journal of Roentgenology 174*, 5 (2000), 1417–1421. 7

[OK95] OGNIEWICZ R. L., KÜBLER O.: Hierarchic Voronoi skeletons. *Pattern Recognition 28*, 3 (1995), 343–359. 3

[PK99] PALÁGYI K., KUBA A.: A parallel 3D 12-subiteration thinning algorithm. *Graphical Models and Image Processing 61*, 4 (July 1999), 199–221. 3

[Pro07] PROHASKA S.: *Skeleton-Based Visualization of Massive Voxel Objects with Network-Like Architecture*. PhD thesis, Potsdam University, 2007. 3

[PSB*01] PALÁGYI K., SORANTIN E., BALOGH E., KUBA A., HALMAI C., ERDOHELYI B., HAUSEGGER K.: A sequential 3D thinning algorithm and its medical applications. In *Proceedings of the 17th International Conference on Information Processing in Medical Imaging* (2001), IPMI '01, pp. 409–415. 3

[RS05] RAMESH S., SANYAL A. J.: Evaluation and management of non-alcoholic steatohepatitis. *Journal of Hepatology 42*, 1, Supplement (2005), S2 – S12. 7

[RT08] RENIERS D., TELEA A.: Segmenting simplified surface skeletons. In *Proceedings of the 14th IAPR international conference on Discrete geometry for computer imagery* (2008), DGCI'08, pp. 262–274. 4

[RvWT08] RENIERS D., VAN WIJK J., TELEA A.: Computing multiscale curve and surface skeletons of genus 0 shapes using a global importance measure. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (Mar. 2008), 355–368. 3, 4

[SEK*11] SUZUKI K., EPSTEIN M. L., KOHLBRENNER R., GARG S., HORI M., OTO A., BARON R. L.: Quantitative radiology: Automated CT liver volumetry compared with interactive volumetry and manual volumetry. *American Journal of Roentgenology 197*, 4 (2011), W706–W712. 8

[ST04] STRZODKA R., TELEA A.: Generalized distance transforms and skeletons in graphics hardware. In *VisSym 2004, Symposium on Visualization* (2004), pp. 221–230. 3

[vDvdWT06] VAN DORTMONT M. A. M. M., VAN DE WETERING H. M. M., TELEA A. C.: Skeletonization and distance transforms of 3D volumes using graphics hardware. In *Proceedings of the 13th international conference on Discrete Geometry for Computer Imagery* (2006), DGCI'06, pp. 617–629. 3

[WHW00] WONG K. C.-H., HENG P.-A., WONG T.-T.: Accelerating "intelligent scissors" using slimmed graphs. *Journal of Graphics Tools 5*, 2 (Feb. 2000), 1–13. 3

[XT09] XIA H., TUCKER P. G.: Distance solutions for medial axis transform. In *Proceedings of 18th International Meshing Roundtable* (2009), IMR '09, pp. 247–265. 3