See new software: http://www.turtleseg.org

# Segmentation of Complex Objects with Non-Spherical Topologies from Volumetric Medical Images using 3D Livewire

Miranda Poon[a], Ghassan Hamarneh[b], Rafeef Abugharbieh[a]

[a]Biomedical Signal and Image Computing Lab, University of British Columbia, Canada
[b]Medical Image Analysis Lab, Simon Fraser University, Canada

## ABSTRACT

Segmentation of 3D data is one of the most challenging tasks in medical image analysis. While reliable automatic methods are typically preferred, their success is often hindered by poor image quality and significant variations in anatomy. Recent years have thus seen an increasing interest in the development of semi-automated segmentation methods that combine computational tools with intuitive, minimal user interaction. In an earlier work, we introduced a highly-automated technique for medical image segmentation, where a 3D extension of the traditional 2D Livewire was proposed. In this paper, we present an enhanced and more powerful 3D Livewire-based segmentation approach with new features designed to primarily enable the handling of complex object topologies that are common in biological structures. The point ordering algorithm we proposed earlier, which automatically pairs up seedpoints in 3D, is improved in this work such that multiple sets of points are allowed to simultaneously exist. Point sets can now be automatically merged and split to accommodate for the presence of concavities, protrusions, and non-spherical topologies. The robustness of the method is further improved by extending the 'turtle algorithm', presented earlier, by using a turtle-path pruning step. Tests on both synthetic and real medical images demonstrate the efficiency, reproducibility, accuracy, and robustness of the proposed approach. Among the examples illustrated is the segmentation of the left and right ventricles from a T1-weighted MRI scan, where an average task time reduction of 84.7% was achieved when compared to a user performing 2D Livewire segmentation on every slice.

**Keywords:** segmentation, other (3D Livewire, semi-automatic segmentation, user-interaction)

## 1. INTRODUCTION

In medical image processing, segmentation is vitally important for quantifying and visualizing 3-dimensional (3D) biological structures. The two most common approaches are fully-automated segmentation and manual tracing. Fully-automated segmentation tools work best when calibrated for very specific image properties. However, biological structures are typically affected by significant variations due to subject diversity and pathology. Automated methods are also sensitive to parameterization and may not produce correct results due to lack of human intervention. Manual tracing, on the other hand, is time consuming because delineation is required for each 2D slice in a volume. Also, this method suffers from inter-operator variability.[1] Due to these difficulties, semi-automated methods have recently drawn interest as a way to facilitate computer-based segmentation of a 3D object of interest using minimal human interaction. However, these techniques are too restrictive on the object's shape.[2–5] Basic groundwork has been proposed to extend 2D Livewire[6] to 3D by utilizing 2D Livewire contours to automatically generate additional seedpoints in orthogonal orientations.[5] However, medical images often contain objects with complex 3D shapes (e.g. deep concavities and protrusions and non-spherical topologies), which none of the earlier Livewire extensions can handle.[2–5] In this paper, we present an enhanced robust 3D Livewire-based segmentation algorithm that is capable of handling arbitrarily complex 3D object geometry and topology. An interactive and intuitive segmentation tool based on the proposed framework has been developed. The tool allows for easy transition between 2D and 3D Livewire modes, concurrent viewing of contours in all three orthogonal orientations, efficient on the fly correction of user mistakes, and 3D visualization of the segmentation results.

The rest of this paper is organized as follows: Sections 2.1 and 2.2 provide an overview of the 2D and 3D Livewire extension technique on which our method is based. Our algorithm modifications and improvements are discussed in sections 2.3-2.5. Next, accuracy, reproducibility, and efficiency results are given in section 3.

## 2. METHODS

The proposed improved Livewire algorithm is based on previous work by Falcao *et al*[7] and Hamarneh *et al*.[5] The user begins the segmentation process by performing 2D Livewire segmentations on slices in any two orthogonal orientations. These 2D contours are then used to determine the Livewire seedpoints to be used in the third orthogonal orientation. This is achieved by determining all intersection points between the available contours and an arbitrary unseen orthogonal slice. These points are connected and arranged in two dimensions. This resulting 'seedpoint map' is then used to determine the ordering of all seedpoints, thus mimicking a user-guided (manual) Livewire segmentation but on a new slice orientation not visited by the user. Since these new seed points are a subset of the contours seen and approved by the user, they are therefore a suitable choice of seedpoints for guiding the Livewire segmentation in unseen slices. In the proposed enhanced scheme, user-generated contours that are circumscribed inside another contour are automatically flagged with this information and these flags are used to split and merge sections of the map. By doing so, multiple contours can be processed for each unseen slice. The result is that objects with non-spherical topologies such as a vertebra (toroidal topology due to the spinal canal) are processed correctly. In real medical image data, intersection points found as described above may occupy consecutive voxels on a line and form clusters rather than isolated points. A new pre-processing pruning step and an improved turtle algorithm for ordering the seedpoints were designed to overcome this issue, which substantially increased the method's robustness.

We first present an overview of 2D Livewire in section 2.1. Next, the basic idea of using orthogonal contours to generate Livewire seedpoints is discussed in section 2.2. Modifications of the above method to increase robustness and to enable the handling of arbitrary object topology is then provided in sections 2.3 and 2.4. Our technique was developed into an intuitive user interface; we will present some of the features of this software tool in section 2.5.

### 2.1. 2D Livewire

While our proposed approach segments 3D objects, it does use 2D Livewire (in orthogonal orientations) to compute each segmentation contour. The original Livewire technique[6] was not significantly altered. In any 2D slice $S(q)$ of a volume, where $q = (x, y)$, edge and smoothness terms were used to create a local cost map $C(p, q)$ of the original image (equation 1). Here, *p=(x',y')* represents a neighboring pixel to pixel *q*. In our implementation, the gradient magnitude cost $C_G(q)$, gradient direction cost $C_{GD}(p, q)$, Canny edge detection cost $C_C(q)$, Laplacian of Gaussian (LoG) cost $C_{LoG}(q)$, and Euclidean distance (smoothness) cost $C_d(p, q)$ were used. Each term is then weighted by $w_G$, $w_{GD}$, $w_C$, $w_{LoG}$, and $w_d$ respectively, as follows:

$$C(p,q) = w_C C_C(q) + w_{LoG} C_{LoG}(q) + w_G C_G(q) + w_{GD} C_{GD}(p,q) + w_d C_d(p,q) \quad , \tag{1}$$

where $C_G(q)$ is defined as

$$C_G(q) = 1 - \frac{1}{max(G)} \sqrt{\left(\frac{dS(x,y)}{dx}\right)^2 + \left(\frac{dS(x,y)}{dy}\right)^2} \quad \Bigg|_{(x,y)=q} \quad . \tag{2}$$

$max(G)$ denotes the largest gradient magnitude found in the image. The gradient direction cost is then

$$C_{GD}(p) = \frac{1}{\pi} acos\left(\frac{S'_x(p)}{G(p)}\frac{S'_x(q)}{G(q)} + \frac{S'_y(p)}{G(p)}\frac{S'_y(q)}{G(q)}\right) \quad , \tag{3}$$

where $S'_x(x, y)$ and $S'_y(x, y)$ are the partial derivatives of $S(x, y)$ with respect to $x$ and $y$, and $G(p)$ and $G(q)$ denote the gradient magnitude (not gradient cost) of the image at pixel $p$ and $q$ respectively. The LoG cost of the image is defined as

$$C_{LoG}(q) = 1 - (LoG_{kernal}(x,y) * S) \quad |_{(x,y)=q} \quad , \tag{4}$$

where $S$ is the original image and the LoG kernal is defined as

$$LoG_{kernal}(x,y) = -\frac{1}{\pi\sigma^4}\left(1-\frac{x^2+y^2}{2\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}} \quad . \tag{5}$$

With a seedpoint located at pixel $q$ on the slice, a cost map $M(q)$ is created by determining the minimal accumulated path cost from pixel $q$ to all other pixels on the slice. This uses the graph search algorithm proposed by Barrett and Mortenson.[6] By repeating this operation, the minimal-cost path connecting several sparsely separated seedpoints on the object boundary that delineates the object of interest is found. The number of seedpoints needed to accurately segment the object depends on the image quality and object size and shape. A simple extension of this scheme to 3D is to iterate this process for each slice of the volume.

## 2.2. Basic 3D Livewire Extension

Building on the ideas of Falcao,[7] the general framework for extending Livewire to 3D was proposed by Hamarneh.[5] Initially, the user performs 2D Livewire on select slices in any two orthogonal orientations ($yz$ and $xz$, $yz$ and $xy$, $xz$ and $xy$). An automatic Livewire segmentation can then be performed on a slice in the third orientation: $xy$, $xz$, or $yz$ respectively. To achieve this, the intersection points between 2D Livewire contours and the unvisited orthogonal slice are determined and used as seedpoints for a fully automatic Livewire operation. For example, if 2D Livewire is used to create two contours $C_{yz}$ and $C_{xy}$ on arbitrary $yz$ and $xy$ slices, then given a slice $S_{x,y_0,z}$, in the $xz$ orientation, the intersection points $I_{x,y_0,z}$ between $C_{yz}$ and $S_{x,y_0,z}$, and $J_{x,y_0,z}$ between $C_{xy}$ and $S_{x,y_0,z}$ can be calculated using

$$I_{x,y_0,z} \Leftarrow C_{yz} \bigcap S_{x,y_0,z} \qquad J_{x,y_0,z} \Leftarrow C_{xy} \bigcap S_{x,y_0,z}. \tag{6}$$

Similarly, equations 7 and 8 define the intersection points $I$ and $J$ on slice $S$ if different orientation combinations are chosen.

$$I_{x,y,z_0} \Leftarrow C_{yz} \bigcap S_{x,y,z_0} \qquad J_{x,y,z_0} \Leftarrow C_{xy} \bigcap S_{x,y,z_0} \tag{7}$$

$$I_{x_0,y,z} \Leftarrow C_{yz} \bigcap S_{x_0,y,z} \qquad J_{x_0,y,z} \Leftarrow C_{xy} \bigcap S_{x_0,y,z} \tag{8}$$

In order to mimic a user-guided 2D Livewire segmentation task in an automatic fashion, the seedpoints, $I$ and $J$, need to be ordered either clockwise or counterclockwise in the 2D space of slice $S$. This is similar to the

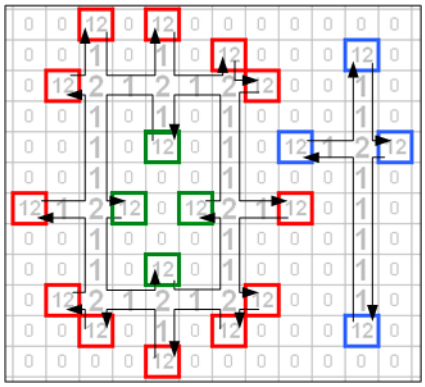


**Figure 1.** Example seedpoint map showing three separate contours (red, green and blue) to be computed on the same unvisited 2D slice. Arrows indicate ordering of seedpoints (boxed) along the turtle path (1's and 2's). The red and green contours make up the outer and inner contours of a toroidal object. The blue contour is considered a separate object on this slice.

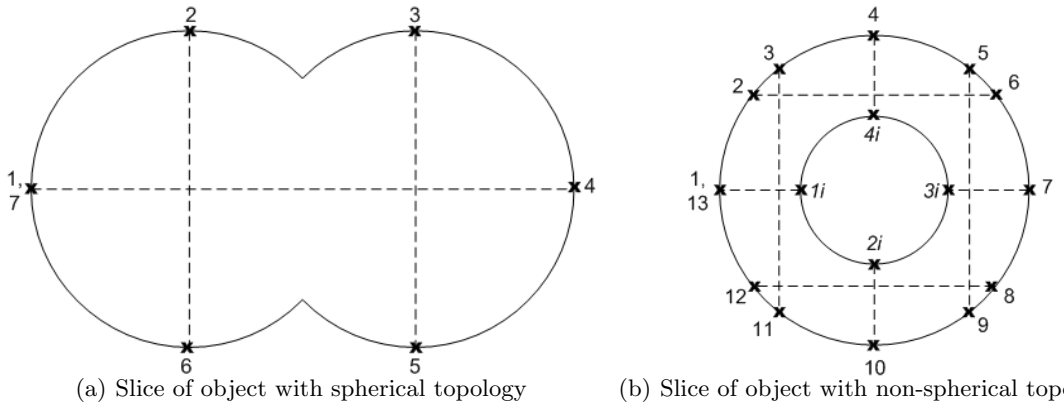
seedpoint order in which a user would manually segment the object with manual 2D Livewire. To accomplish this, seedpoints on the 2D space which belong to the same user-guided Livewire contour subset are paired and connected by lines (tracks). Since there are seedpoint contributions from two orthogonal orientations, these tracks will themselves be orthogonal on slice $S$. Figure 1 shows how these seedpoints are connected into a map.

To process this map, an L-systems 'turtle' algorithm is used. Starting at an arbitrary seedpoint or anchor, the turtle traverses the orthogonal tracks on the turtle map such that it can only move forward and turn left at path intersections. If it encounters another seedpoint, it will reverse direction. As the map is traversed, the sequence in which the seedpoints are visited determines their order (Figure 2). Lastly, this ordered seedpoint list is used by the 2D Livewire algorithm to determine the segmentation contour on this unvisited slice.

### 2.3. Intersection Point Pruning

Ideally, a 2D contour would intersect with an unvisited orthogonal slice at an even number of locations, as described in earlier work.[2,5] However, objects with cusps can cause singular intersection points to exist as well. Also, while a user-guided contour will always be orthogonal to the slice in question, contiguous colinear contour pixels may intersect with this slice. In the extreme example of a cube, a user-guided contour (a square) may be orthogonal to an unseen slice, but their intersection may comprise the entire square side of the contour. This results in a set of intersection points that span multiple contiguous voxel locations.

The problem of contiguous intersection points was treated in a different implementation of 3D Livewire, where two parallel reference frames projected onto the problematic slice was used to correct this problem.[2] Our method automatically discards any extraneous points by analyzing only the intersection point groupings and locations on a given slice. To discard these extraneous points, a pre-processing pruning operation is performed. Rather than treating intersection points individually, our technique assumes intersection points appear in cluster(s) or occupy consecutive pixel locations such as in Figure 3. Since the intersection of two slices will always be a line, the set of intersection points found between a user-guided Livewire contour and the slice being processed will also be along a line. These colinear points are first sorted in ascending pixel location order. Traversing this list in increasing order, cluster boundaries are easily determined by their position values found on the original image: non-consecutive pixel location values signify the start or end of clusters. Knowing where clusters start and end allows us to prune the unnecessary points in between. The result is an even number of intersection points with no contiguousness. An exception to the rule is when only one cluster is found, which corresponds to a cusp (singular point). In this case, the start and end of the cluster are kept and the middle points are discarded. However, if an odd number of clusters (greater than 1) is found, then it does not seem possible to determine which cluster(s) are cusps without analyzing information from adjacent slices or texture information.




(a) Slice of object with spherical topology (b) Slice of object with non-spherical topology

**Figure 2.** Two examples of turtle maps used to order seedpoints on an unvisited slice. Seedpoints are numbered, and 'turtle tracks' are denoted using dashed lines. Slices of objects with spherical topology, (a), will not result in circumscribed contours. Circumscribed contours of toroidal objects, (b), must be processed differently since these contours do not encompass the object of interest.

With the extraneous contiguous points removed, the ideal case of having even intersection occurrences is reached. This allows for each automatically processed slice to be independent of all other parallel slices. Therefore, shape and topology changes (e.g. branches, cusps, saddle points) not observed in adjacent slices are preserved.

### 2.4. Handling Arbitrary Topology

Biological diversity and pathology often give rise to extremely complex objects with deep concavities, protrusions, non-spherical topologies, and discontinuity (e.g. due to tissue atrophy). While the determination of intersection points between user-guided contours and an orthogonal unvisited slice is straightforward, complex topologies directly affect the creation of the turtle map, and consequently, the ordering of these points. For purely convex objects (e.g. sphere), it is guaranteed that there will only be one or two clusters of seedpoints; thus, a turtle map can be easily generated using the technique described in our previously proposed framework.[5] For objects with concavities or protrusions (e.g. U-shaped tube), there may be situations where a slice will appear to capture two objects. Here, our approach will process each structure separately in a given point map, thus all the parts will be segmented correctly. Similarly, if the object of interest is comprised of two disjoint parts, then each part will be segmented separately as our approach does not assume connectivity between seedpoints on the turtle map unless explicitly joined by turtle tracks. What this implies is that even though a volume may contain multiple objects that are not related to each other, they can be segmented at the same time with no interference. An example of this is shown in Figure 5, where both ventricles, although structurally disjoint, are segmented in the same binary output mask.

Objects with non-spherical topology (e.g. torus) pose the biggest problem for previous Livewire implementations. In order to correctly segment these types of objects, during the user-guided Livewire process, our method first identifies all contours that are circumscribed within another. To do this, pairwise comparisons are performed for all user-guided contours on a given slice. Let $C_1$ and $C_2$ represent two contours on the same image slice that are subjected to a pairwise comparison. They are first converted to binary masks $M_{C_1}(x, y)$ and $M_{C_2}(x, y)$ respectively, where

$$M_A(x, y) = \begin{cases} 1 & \text{if } (x, y) \text{ is inside contour } A \\ 0 & \text{otherwise} \end{cases} . \tag{9}$$

If $M_{C_1}(x, y) \bigcap M_{C_2}(x, y) = M_{C_1}(x, y)$, then $C_1$ is wholly situated inside $C_2$, and if $M_{C_1}(x, y) \bigcap M_{C_2}(x, y) = M_{C_2}(x, y)$, then $C_2$ is wholly situated inside $C_1$. This step is critical because these 'inner' contours delineate pixels that do not encompass the object of interest, but rather a hole in the object. Due to this, these inner contours and their derived seedpoints are flagged as 'negative', whereas the contours and seedpoints that actually

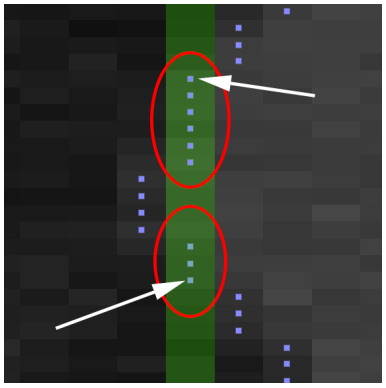


**Figure 3.** A user-steered 2D Livewire contour (blue dots) shown on an image slice during segmentation. An orthogonal unvisited slice intersects with this slice (green band). The red circles indicate clustering of the intersection points which is problematic but handled by our proposed scheme. Arrows indicate the two remaining intersection points after pruning.

delineate the object are flagged as 'positive'. Although all intersection points, regardless of their classification, are anchor points on the turtle map, negative seedpoints do not create turtle tracks. Instead, they in effect negate a section of the otherwise longer track line, correctly splitting the turtle map into two distinct parts. An example is shown in Figure 2(b), where seedpoints $2i$ and $4i$ negate the otherwise longer turtle track between seedpoints 4 and 10. This process results in a central cavity, depicted in Figures 1 and 2, which now correctly represents the object. Although all the seedpoints in Figure 1 are connected by tracks, the turtle algorithm will always process the outer contour first due to the turtle's pre-defined movement restrictions (section 2.2). After these points are processed and subsequently disregarded, the inner anchor points will then be processed. This particular algorithm's use is limited to the handling of non-spherical topologies such as a toroid, but it can be used in conjunction with the different object shape cases shown above. The combinations of these different topologies are representative of most if not all anatomical shapes.

### 2.5. Implementation and Visualization

This proposed framework was developed in MATLAB and offers the standard concurrent orthogonal views of a volume as shown in Figure 4. As an overlay on top of the image data, user-guided contours are clearly demarcated in all views, regardless of their orientation. One criticism of the original framework[5] was that the slices used for user-guided contours had to be carefully selected otherwise the segmentation will fail.[2] By displaying these

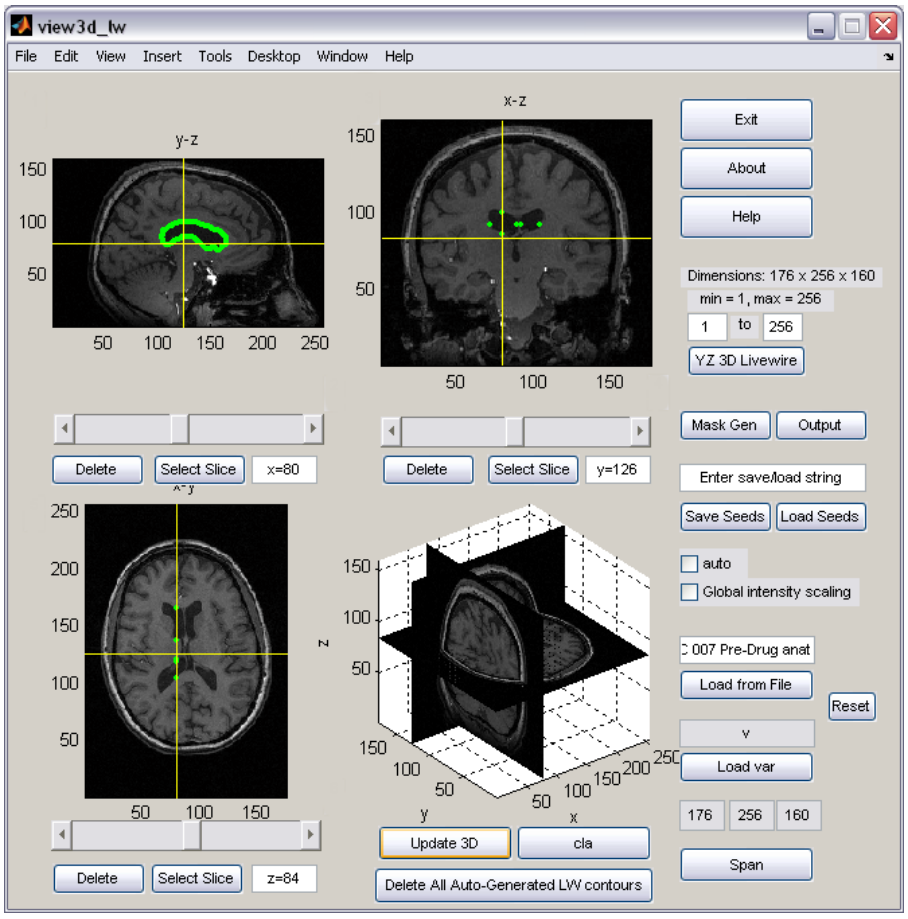



**Figure 4.** Screen-capture of the proposed segmentation tool's graphical user interface during the segmentation of the left and right ventricles. Completed 2D contours, regardless of their orientation, are displayed in green for the three orthogonal views. This provides feedback on the accuracy and effect of these contours throughout the segmentation task. Yellow lines indicate the current slice indices of the other two orientations.

contours in this manner, our application gives users a clear idea of which areas have been segmented and which areas exhibit more topological features. In our findings, these feature-rich areas, if segmented correctly by the user, usually allow for higher accuracy. Also, this software feature is very useful for visually judging the accuracy of the delineation result. Additionally, our user interface is able to display a 'point cloud' representation as well as a surface rendering of the object of interest after the 3D Livewire procedure is completed.

In our developed tool, the 2D Livewire component supports additional features such as point deletion and contour closing. During this user-guided 2D Livewire stage, if the user selects a seedpoint erroneously, he/she can revert the segmentation process to an earlier state, similar to the 'undo' command found in many common applications. Also, in earlier versions of Livewire, the contour had to be finished or closed by having the user select the initial seedpoint again; however, in high resolution images, this is often difficult and problems such as unclosed contours or contour overlapping can easily occur. Here, the tool automatically and accurately closes the contour as the last step.

While our technique is extremely flexible and robust, errors are bound to occur due to human error and poor image quality. Our tool offers an undo operation described above, as well as the ability for users to remove entire contours for re-computing. From the rendered result, users can quickly identify problematic areas and increase the segmentation accuracy by providing additional user-guided contours in these areas and re-running the 3D Livewire algorithm. For minor errors, users can overwrite the automatically generated contour(s) using the 2D Livewire procedure.

## 3. RESULTS

The proposed method was applied to a variety of synthetic images and real medical image data to demonstrate its capabilities. The application's performance during these tasks was quantified based on the three main recommended criteria for semi-automatic segmentation.[8] These criteria are reproducibility, efficiency, and accuracy, and they are discussed in sections 3.1-3.2. A basic example is the segmentation of a mask of a left caudate nucleus (Figure 5(a)). While not difficult to segment, it does highlight the need and importance of placing contours at 'critical' locations within slices that demonstrate the highest and most complex shape characteristics. For instance, while user-guided contours are needed to capture the body shape of the caudate nucleus, extra contour(s) at the nucleus tail ensures the 3D Livewire's ability to segment these areas properly.

Another segmentation example is a synthetic torus under high levels of noise (SNR = 0dB), as shown in Figure 5(d). This object is an example of toroidal topology. Here, the planes perpendicular to the axis of the torus were chosen for segmentation by our 3D algorithm because this highlights our technique's ability to segment image slices with negative spaces (holes) inside. For this scenario, eight user-guided 2D Livewires are needed to provide the seedpoints required for the 3D algorithm. Other orientations may require only six user-guided contours. Figure 2 depicts how the turtle map looks like for a slice near the middle of a torus.

Lastly, the left and right ventricles of a T1-weighted MRI scan of a human brain are segmented using our method (Figure 5(g)). Here, additional contours are required due to the ventricles' shape complexity and less than ideal image quality. User-guided Livewire contours were done for the sagittal and axial orientation, and the 3D Livewire algorithm was applied to the coronal orientation.

### 3.1. Reproducibility

Since our 3D Livewire method is deterministic and produces identical results given the same input contours, we measured reproducibility with different user-guided contours and seedpoints as input. This simulates the interaction of real users to our application because not all operators will choose the same slices nor will they choose the same locations for seedpoints. The orientation of the 3D segmentation and the actual human operator were kept constant. Since reproducibility does not compare segmentations to the absolute truth, the resulting binary masks were subjected to pairwise Dice similarity tests and a similarity coefficient was found using:

$$C_{Dice} = \frac{2vol_{sim}}{2vol_{sim} + vol_A + vol_B} \quad . \tag{10}$$

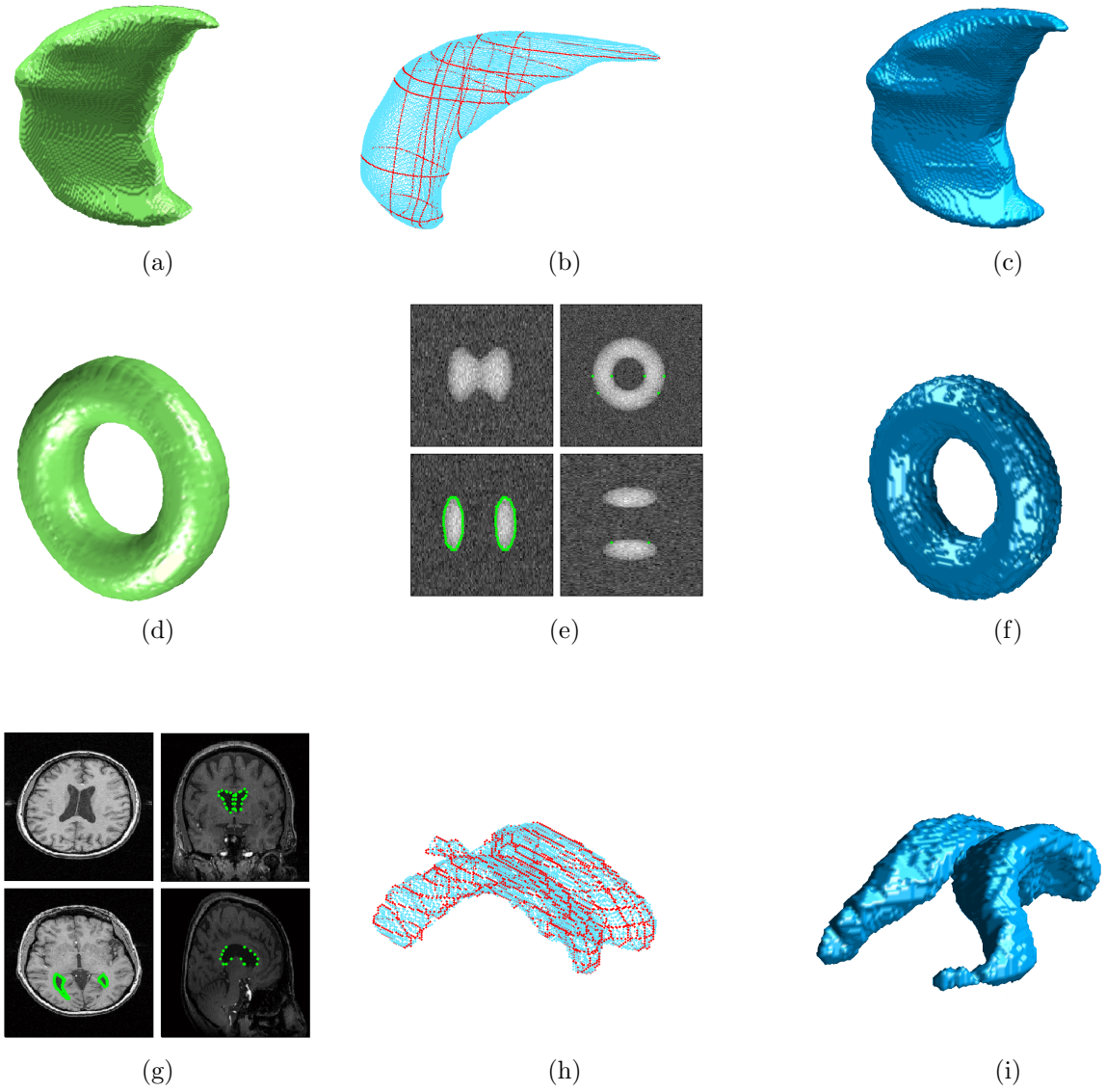



**Figure 5.** (a) Rendering of the left-caudate mask. (b) 3D plot of user-guided Livewire input (red) and automatically generated contours (light blue) of the left-caudate mask. (c) Rendering of the segmented left-caudate mask. (d) Rendering of a synthetic noise-free torus. (e) Views of the torus subjected to noise with SNR of 0dB. Green markers denote the presence of user-guided Livewire contours. (f) Rendering of the torus segmentation result. The roughness is due to the high level of simulated noise. (g) Views of a T1-weighted MRI scan of a brain. Green markers denote the presence of user-guided Livewire contours. (h) 3D plot of the left and right ventricles: user-guided Livewire input is in red and automatically generated contours are in light blue. (i) Rendering of both segmented ventricles as separate objects. Both ventricles were segmented during the same task.

$C_{Dice}$ is the Dice similarity coefficient, which is a measure of voxel agreement. $vol_{sim}$ is the sum of the voxels at the intersection between trial A and trial B, and $vol_A$ and $vol_B$ represent the sum of the voxels in trials A and B respectively. The Dice similarity coefficients were then averaged over all trials (table 1). In binary images, the reproducibility rate is high because the exact locations of the user-guided contours are not as important due to the high gradient properties of the data.

**Table 1.** Summary of the average reproducibility and accuracy results of our proposed method. Reproducibility is represented by the Dice similarity coefficient, and expressed as a percentage

| | **# of Trials** | **Reproducibility %** | **Avg. Accuracy %** |
|---|---|---|---|
| Left-caudate Mask | 5 | 98.7 | 98.5 |
| Torus (SNR=0dB) | 5 | 93.2 | 73.4 |
| Left + right ventricles | 3 | 93.4 | n/a |

### 3.2. Accuracy

Accuracy is simply defined here as the voxel agreement between our technique's segmentation result with either the original synthetic data or a mask that is generated using manual tracing by an expert. Here, the caudate segmentation result is compared to the original noise-free data, and the torus segmentation result is compared to the noiseless image. Since the left-caudate mask was pre-segmented, the accuracy of its segmentation was naturally high. In the torus segmentation example, the heavy noise applied to the image corrupted much of the object's outer edge; however, we were still able to recover a large portion of the torus. Table 1 summarizes our averaged results over multiple trials.

### 3.3. Efficiency

Efficiency was calculated by comparing the time required for our technique to segment all slices of an orientation to the time needed for performing 2D Livewire on each slice. Due to poor image quality or user mistakes, contour errors may occur with 3D Livewire; thus, the time it takes to correct them is included in the time measurements as well. Trials for each task were performed and their task times were averaged prior to comparison (table 2). Total processing time naturally increased for volumes with high shape complexity. This is because more user-guided Livewire contours are needed to fully characterize the object and connect the various turtle tracks together into valid maps. As the number of user-guided contours increases, so does the total amount of intersection points found for each unseen slice. Since each seedpoint must ultimately be subjected to the computationally-expensive graph search algorithm during the Livewire processing step, computation time grows. However, this higher processing time is counterbalanced by the fact that manual tracing of complex objects requires more user attention and segmentation time for an accurate delineation. Our results show that our method is able to achieve these complex segmentation tasks in roughly 20% of the time it takes to delineate all slices using 2D Livewire.

## 4. CONCLUSIONS

We presented a new powerful highly-automated 3D segmentation technique that extends and enhances our 3D Livewire-based semi-automated segmentation technique that was previously proposed.[5] This enhanced algorithm is capable of handling arbitrary object shapes, including object concavities, protrusions, and non-spherical topology, and is robust to handling real medical images. Moreover, we have shown that this method results in a high degree of segmentation reproducibility, task time reduction, and accuracy.

Development of this framework is ongoing. Enhancing accuracy and robustness are the primary focus, and techniques to automatically refine the generated contours are being explored. Also, the algorithm is being further modified to be more tolerant to errors attributed to user-mistakes. Lastly, efficiency is being improved so that real-time, on-the-fly segmentation capabilities can be added to our application.

**Table 2.** Summary of segmentation times required to delineate the examples of the left-caudate mask, the noisy torus, and the left and right ventricles. Also, the task time reduction of our proposed method compared to performing 2D Livewire on each slice is provided.

| | **# of Trials** | **Avg. user interaction time with our tool (s)** | **Avg. automatic processing time of our tool (s)** | **Avg. time required for manual corrections (s)** |
|---|---|---|---|---|
| Caudate mask | 5 | 128 | 22 | 80 |
| Torus(SNR=0dB) | 5 | 82 | 13 | 19 |
| L+R ventricles | 3 | 537 | 75 | 42 |
| | **# of Trials** | **Avg. total time using our tool (s)** | **Avg. task time using 2D Livewire for all slices (s)** | **Fraction of time needed compared to 2D Livewire** |
| Caudate mask | 5 | 230 | 1265 | 18.2 % |
| Torus(SNR=0dB) | 5 | 114 | 523 | 21.8 % |
| L+R ventricles | 3 | 654 | 4267 | 15.3 % |

## 5. ACKNOWLEDGEMENTS

The medical data presented in this paper was provided by Dr. Martin McKeown of the Department of Medicine at the University of British Columbia, Vancouver.

## REFERENCES

1. J. Rajapakse and F. Kruggel, "Segmentation of MR images with intensity inhomogeneities," *Image and Vision Computing* **16**, pp. 165–180, 1998.
2. K. Lu and W. Higgins, "Improved 3D live-wire method with application to 3D CT chest image analysis," *Proceedings of SPIE Medical Imaging: Image Processing* **6144**, pp. 189–203, 2006.
3. R. Saboo, J. Rosenman, and S. Pizer, "Geointerp: Contour interpolation with geodesic snakes," *The Insight Jounal* , July 2006.
4. A. Souza, J. Udupa, G. Grevera, D. O. Y. Sun, N. Suri, and S. M, "Iterative live wire and live snake: New user-steered 3D image segmentation paradigms," *Proceedings of SPIE Medical Imaging: Physiology, Function, and Structure from Medical Images* **6144**, pp. 1159–1165, March 2006.
5. G. Hamarneh, J. Yang, C. McIntosh, and M. Langille, "3D live-wire-based semi-automatic segmentation of medical images," *Proceedings of SPIE Medical Imaging: Image Processing* **5747**, pp. 1597–1603, 2005.
6. W. Barrett and E. Mortensen, "Interactive live-wire boundary extraction," *Medical Image Analysis* **1**, pp. 331–341, 1997.
7. A. Falcao and J. Udupa, "Segmentation of 3D objects using live wire," *Proceedings of SPIE Medical Imaging: Image Processing* **3034**, pp. 228–235, 1997.
8. S. Olabarriga and A. Smeulders, "Interaction in the segmentation of medical images: A survey," *Medical Image Analysis* **5**, pp. 127–142, 2001.