Surface Construction from 3D Ultrasound Images

Yasmin Halwani
Electrical and Computer Engineering
University of British Columbia
yhalwani@ece.ubc.ca

Sara Sheikholeslami
Mechanical Engineering
University of British Columbia
ssheikholeslami@mech.ubc.ca

Abstract

Surface extraction from ultrasound volumetric data is one of the active research areas due to the variety of ways through which it can be achieved. The objective of this paper is to present a simple method for surface extraction from 3D ultrasound data through the following steps: image segmentation, contour extraction and de-noising, surface construction and re-meshing. The results show that re-meshing improves the generated mesh only to some extent, but requires further mesh de-noising to generate a smooth output.

Keywords

3D Ultrasound · Volumetric Data · Marching Cubes · Surface Construction · Remeshing

Introduction

Ultrasound imaging is a medical imaging modality that views internal soft tissues through the reflection of emitted high-frequency sound waves. It is one of the most widely used medical imaging modality due to its non-invasive and non-ionizing nature. Given ultrasound imaging's ability to distinguish soft tissues effectively, areas that include the usage of ultrasound imaging include, but not limited to, cardiography, transrectal ultrasound, intravascular ultrasound, obstetrics and breast ultrasound. Ultrasound imaging started with 2D imaging and evolved to 3D imaging to yield volumetric data with different acquisition methods that could be based on free-hand motion or mechanically-based.

Surface extraction from 3D ultrasound images is one of the challenging problems in the field of ultrasound imaging. This is due to the need to segment images prior to surface construction and the noisy nature of ultrasound images, which makes it difficult to accurately segment an object or an area of interest in a set of ultrasound slices for volumetric representation [1, 2]. Many solutions for segmentation were proposed in this area, such as those reviewed in [3]. Most of the reviewed methods for segmentation of 3D ultrasound images depend on 2D segmentation of the slices. An example includes techniques based on edge-based semi-automatic methods and automated clustering methods. Another approach performs segmentation directly on volumetric data, such as using a 3D deformable model initialized with a user-defined starting model. Previous work, such as the ones presented in [4] and [5] are examples of approaches taken to construct a mesh out of volumetric ultrasound data. The latter used a combination of image segmentation and a level set method for contour extraction to highlight the area of interest before mesh construction, then used a variation of the marching cubes algorithm to construct the desired mesh, which is a main source of influence for the work that will be presented in this paper in terms of the steps taken to generate a mesh from 3D ultrasound data.

Marching cubes algorithm is one of the widely-used algorithms for extracting iso-surfaces from volumetric data. It was first introduced in 1987 [6] and many variations of the algorithm resulted in the following years. One of the popular variations that is used in the work introduced in [5] is the regularized marching tetrahedra [7], which combines marching cubes with vertex clustering to produce enhanced meshes in terms of topological consistency.

In most computer graphics applications, triangular surface meshes are used to represent shapes. Many of these meshes are generated by scanning devices or by isosurfacing implicit representations. Unfortunately, such processes - especially if automated - are error-prone, and the resulting “raw” meshes are rarely satisfactory [8]. For instance, these raw meshes are often over-complex and contain many redundant vertices. As a result, most existing meshes can be considerably improved in terms of their complexity [9], size, vertex sampling, regularity and triangle quality. The process that corrects the given mesh geometry and connectivity, while providing decent fidelity (keeping the mesh geometrically close to the original surface), is commonly known as re-meshing (Figure 1). Commonly, the focus in efficient re-meshing frameworks is on the trade-off between the visual quality of the result, the speed of the re-meshing operation, and the optimization of the number of polygons in order to achieve interactive rendering speeds [2].

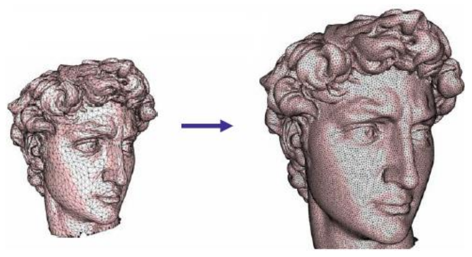


Figure 1: Uniform remeshing of the Digital Michelangelo David model. Figure reproduced from [2]

Depending on the specific application for a given mesh, a variety of re-meshing algorithms have been proposed. Among those are semi–regular re-meshing schemes. The main techniques for semi-regular re-meshing can be classified into two categories according to the way they find correspondences between the input and output meshes. One group of algorithms, e.g. [10-12], is based on partitioning the original mesh into regions (using Voronoi diagram), and treating each region separately, usually with subdivision techniques. While these techniques yield reasonable results, they are very sensitive to the patch structure. The vertex sampling is also sensitive to control.

The other, more recent group of semi–regular re-meshing algorithms, e.g. [13-15] is based on a parameterization of the original mesh on a global planar domain. The parameter domain is then resampled, and the new triangulation is projected back into 3D space, resulting in an improved version of the original model. The main drawback of the global parameterization methods is the sensitivity of the result to the specific parameterization used, and the metric distortion that arises due to the fact that the 3D structure is forced onto a foreign parameter plane [2]. Even if the parameterization minimizes the metric distortion of the 3D original in some reasonable sense, it is impossible to eliminate it completely. Additionally, most of these techniques involve solving a large set of equations, resulting in expensive computation. Sander et al. [16] proposed a hierarchical approach based on multigrid methods, which could accelerate the even for large input meshes. However, this approach still suffers from numerical precision issues when dealing with extremely large meshes, or meshes with severe isoperimetric distortion. In such cases, a global parameterization is almost impossible to perform without using multi-scale or precise arithmetic representation of the parametric domain [8].

Another re-meshing framework is to operate directly on the mesh surface and execute a sequence of local modifications on the mesh [2, 17-19]. This approach is also known as the mesh adaptation process and is the one we implemented in this work. Our scheme is similar to those in [2, 19] performing a series of local modifications on the mesh. The most well-known and commonly used local modifications are (1) edge-flip, (2) edge-collapse/vertex removal, (3) edge-split and (4) vertex relocation. Modifications are applied sequentially in order to achieve desirable mesh characteristics. We apply local modifications on the new mesh, while the original provides a reference to the geometry of the original mesh. To keep the new mesh geometrically close to the original surface, we apply a modification only if all faces created or affected by the modification satisfy the error conditions defined in the following section.

This paper aims to demonstrate a simple approach to construct and refine a 3D mesh from volumetric ultrasound data of a phantom using automatic segmentation, contour extraction and manual contour de-noising, surface construction and re-meshing. An edge-based segmentation method and an active contour are used to generate the iso-surface in the US slices. A mesh is then constructed from the resulting images through extracting the iso-surface using a basic marching cubes algorithm. We finally apply re-meshing to enhance the resulting mesh as previously described in sequential steps.

Method

In this section, we present in details the methods used in each step of the visualization process of the 3D US images. We start with a brief description of the data obtained and the desired output for our work, and then explain the approaches adopted for image segmentation, contour extraction and de-noising, mesh construction and re-meshing.

1. Data Source

The source of the data used for this paper is a phantom of a prostate, as shown in figure 2. The data was acquired at the Robotics and Control Lab at UBC. Given the homogenous density of the material of this phantom, the speckle and noise is lower and the SNR is higher in comparison to images taken of prostates in-vivo. Images were obtained in B-mode with a mechanical probe. A total of 113 slices were obtained.

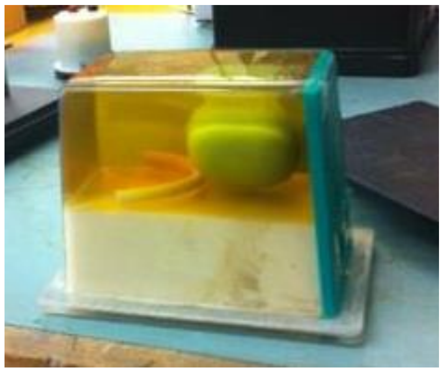


Figure 2: Prostate Phantom at RCL

2. Desired output

Randomly-selected 13 slices were manually segmented and a surface later was constructed using a specialized software named Stradwin [20]. This output is later to be compared to the output of the described algorithm to assess the performance and the output quality of the produced work. Figure 3 and figure 4 show a set of manually-sliced images and the output after surface construction with Stradwin.

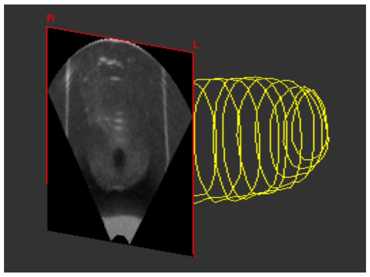


Figure 3: Manual segmentation using Stradwin

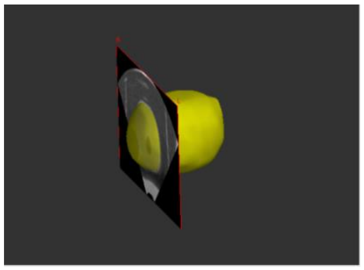


Figure 4: Mesh generation using Stradwin

3. Selected approach

The following steps describe, in order, the methods used for mesh generation and refinement of volumetric 3D ultrasound images:

3.1. Watershed Segmentation

Given that we are interested in the edges of the object in an image, an edge-based segmentation approach is selected. Namely, we select watershed segmentation with foreground markers to generate images that highlight the foreground shapes in an image. Figure 5 shows some sample inputs and outputs of the watershed segmentation method used.

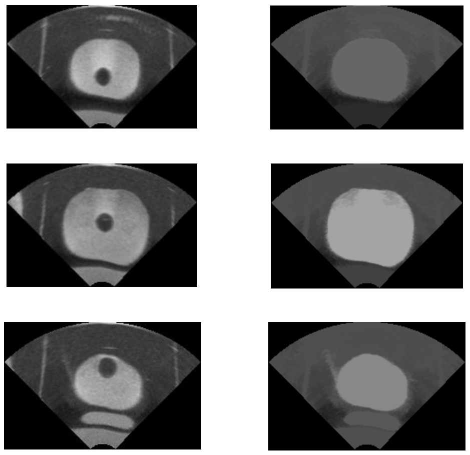


Figure 5: Input slices and output results with watershed segmentation with foreground markers for slices 17, 60 and 97 (from top to bottom)

This method is effective for finding the edges of the foreground shapes in the image. However, in some of the segmented images, such as those slices shown in figure 6, we can distinguish two objects in the image with almost the same intensity. One is the object of interest (the phantom) and another is an elliptical shape below it that is not of an interest in this context. Therefore, it is necessary to perform another step to eliminate any other objects in the image, other than the object of interest, and to uniformly represent that object of interest with an iso-level to be later extracted with the marching cubes algorithm.

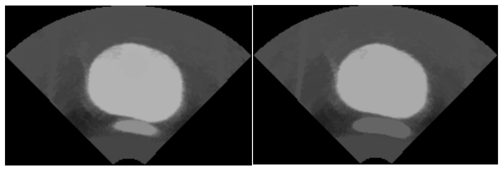


Figure 6: In some slices, we find multiple segmented objects within an image

3.2. *Contours Extraction*

The method used for contour extraction is active contours, also known as "snakes". The concept of active contours is to generate a curve that moves within an object until it reaches its boundaries, based on the minimization of internal and external energies. The internal energy is at its minimal when the curve is at the boundary position of the sought object. The external energy is based on the shape of the contour, and it should be minimal when it closely resembles the shape of the sought object.

An available implementation based on "Active contours without edges for vector-valued images"[21] has been used in this paper to find the contours of a single object in the segmented prostate phantom images. Figure 7 shows how the contour evolves through the image to settle after a few iterations at the edges of the prostate phantom. Figure 8 shows the final result of a binary image where all points outside the contour are black and all the points inside the contour are white. It is necessary to have a uniform iso-value which forms the basis for the next step of mesh construction with marching cubes.

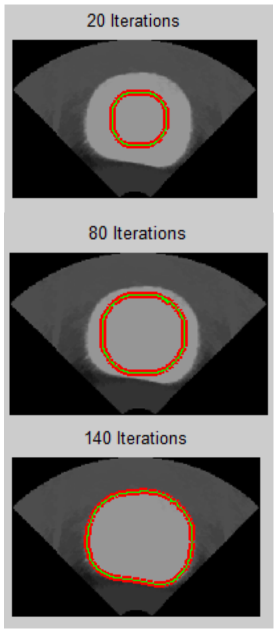


Figure 7: Using active contours

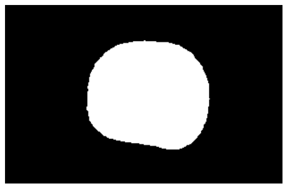


Figure 8: Contour extraction output

Given that the size of the prostate changes temporally across the slices, and the shape is not clearly visible at the two ends of the prostate; the number of iterations for the active contours implementation has been manually set according to the displayed graph in figure 9.

Change of iterations per slice

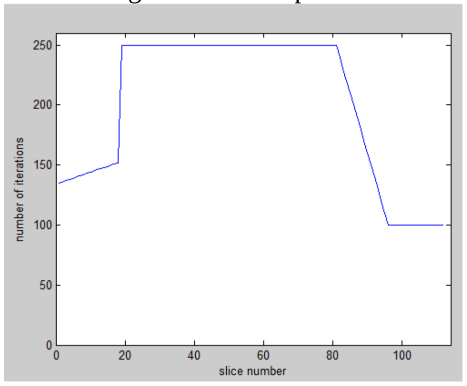


Figure 9: Number of iterations at each slice

The number of iterations is limited at the beginning and at the end of the contour fitting to minimize the amount of false segmentation and noise at the ends of the prostate. Therefore, the contour maintains its original elliptic shape without attempting to segment ambiguous areas and resulting in a noisy output. An example is shown in figure 10 of a mesh generated in MATLAB with a constant number of iterations for all slices during contour fitting (250 iterations in this case).

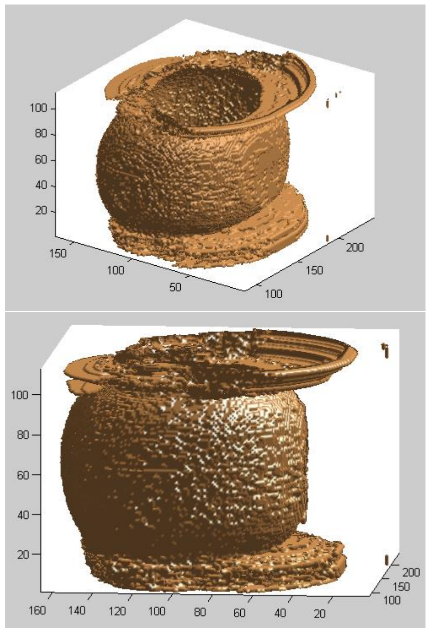


Figure 10: Artifacts at the ends of the constructed prostate in MATLAB

3.3. *Surface Extraction*

Surface extraction is performed with a basic marching cubes algorithm. The algorithm “marches” 8-pixels at a time along a grid of cubes, to decide the number and orientation of the triangles to be built at each cubic grid element. Although the algorithm is considered one of the simplest and most efficient methods for mesh construction, one main drawback is the noisy or “voxelated” output in the resulting mesh, which requires more refinement and mesh de-noising.

One approach taken to reduce some noise in the mesh is limiting the number of slices used for constructing the mesh, which will in turn reduce the resolution of the mesh. As shown in figures 11, 12 and 13, there is a constant trade-off between the mesh resolution and the amount of noise produced. Higher resolution always generates more noisy meshes.

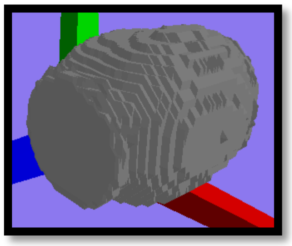


Figure 11: Generated mesh with 113 slices

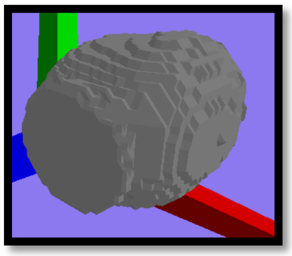


Figure 12: Generated mesh with 56 slices

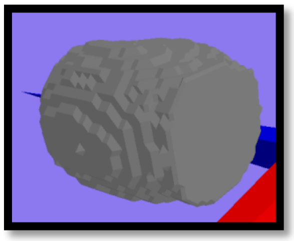


Figure 13: Generated mesh with 30 slices

3.4. Manual Contour De-noising

Another approach for reducing noise in the generated mesh is by reducing the noise in the binary images that form an input to the marching cubes algorithm. By manually modifying the edges of the noisy contours, as shown in figure 14, less noisy resulting meshes are obtained. Figure 15 shows the effect of using noisy binary images and filtered binary images on the resulting mesh. We notice that the noise is significantly reduced at the two ends of the prostate shape, since the noisy contours are present most in those areas. However, the noise in the overall mesh is still present due to the marching cubes algorithm.

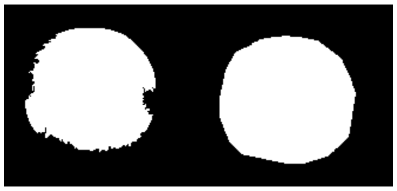


Figure 14: Manually-modified contours in the binary images

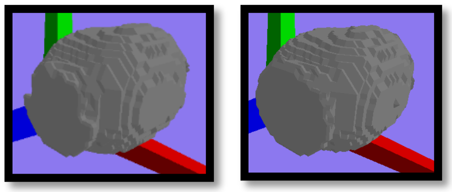


Figure 15: Reduced noise at the two ends of the shape due to contour de-noising

3.5. Re-meshing

The focus of our re-meshing scheme is to generate a better mesh with better sizing and better shape of the triangles of the mesh while preserving the geometry of the original surface. The main stages of our re-meshing scheme are as follows:

1. Refine/Coarsen to adjust the number of vertices of the mesh

2. Smooth Mesh to improve sizing and quality

3. Perform flips after every other operation

4. Repeat entire sequence several times.

Refinement (edge-split) and coarsening (Edge-collapse) are used to change the number of mesh vertices to reach desired sizing and element count of the mesh. However, it is difficult to achieve good spacing of mesh vertices via mesh refinement/coarsening. Vertex relocation via mesh smoothing can be implemented to improve the spacing between the vertices. Edge-flip then improves the quality of the mesh triangles further. Repeating this sequence several times polishes the mesh to obtain the optimal mesh geometry without changing its connectivity. Later in this section, we will further discuss each of these local operations in more details

The newly generated mesh via these local operations should approximate the original shape geometry as closely as possible, while keeping the mesh complexity below a given threshold. Ideally, "just enough" resolution for the problem being solved is sought. This involves choosing some error metrics.

3.5.1. Error Conditions

To ensure fidelity of the new mesh to the geometry of the original mesh, two error metrics are used to evaluate the distance between the two meshes (1) Normal Error, and (2) Smoothness error. Our measures are conceptually similar to those of [2].

(1) Smoothness Error:

Let f = ($v1$, $v2$, $v3$) be a face whose error is to be estimated. The first measure E_{smth} captures the degree of smoothness and should not exceed some threshold angle θ_{smth}:

$$E_{smth}(f) = \max_{i \in \{1,2,3\}} \langle Nf\ , N_{vi} \rangle < \cos \theta smth \quad (1)$$

Nf *is* the unit normal of the current face, and N_{vi} are unit normals of the vertices of the face taken from the original surface. *Esmth* is a measure of how well *f* coincides with tangent planes of the surface at the vertices of *f*.

(1) Normal Error:

The second measure Egap captures the gap between a face f and the surface:

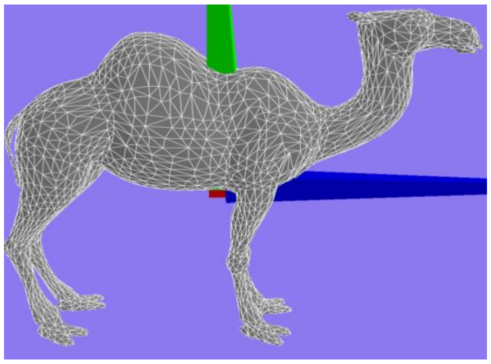
$$E_{gap}(f) = i \in \overset{max}{i \in \{1,2,3\}} \langle N_{vi} , N_{vi+1} \rangle < \cos \theta gap \quad (2)$$

A greater value of the gap angle between the normals of two face vertices corresponds to a more curved surface above face *f*, and thus, to a bigger distance.

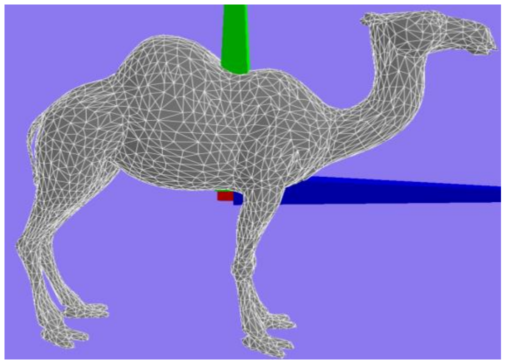
The two error metrics, when implemented together, give accurate results. In addition, since these error metrics involve only normal directions, they are computationally very efficiency.

3.5.2. Mesh Refinement/coarsening

To obtain a mesh with the number of vertices specified by the user, we apply local refinement or simplification operations to the mesh. If an edge is too long, we split it by adding a vertex. Then, we compute the new location of the vertex in terms of barycentric coordinates of new face vertices. Next, we locally parameterize the old mesh containing all 3 new face vertices and compute the corresponding location of the added vertex on the 3D mesh. Edges whose faces have minimal error metrics are simplified first. Every time we perform a series of local refinement, we should follow it by a series simplification operations – either edge-flip or vertex removal- to satisfy the required mesh sizing. Figure 16 shows a camel model after performing one round of mesh refinement.



Original (2862 vertices)



Refinement (3010 vertices)

Figure 16: A re-meshing example for a camel model after one round of refinement.

3.5.3. Mesh Smoothing

The goal in mesh smoothing is to move vertices on the mesh surface to improve mesh sizing and quality. To perform mesh smoothing, we do the following steps [22]:

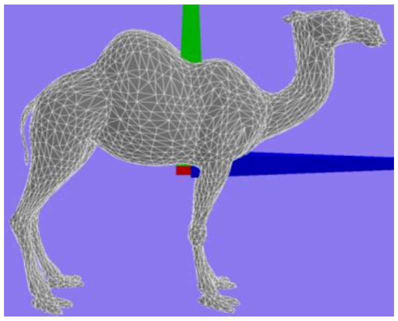
1- Project the vertex to be relocated in addition to its neighbors to the normal plane of this vertex.

2- Relocate the vertex on this plane to be at the center of mass of its neighbors.

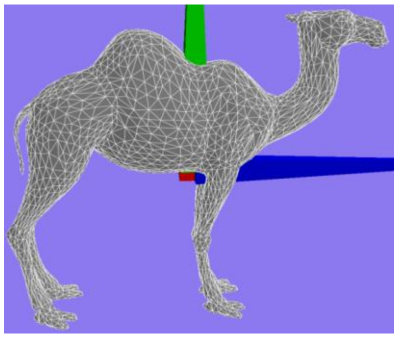
3- Search for the triangle in which the vertex is now located.

4- Project back to the 3D mesh.

5- Check the normal error. If the error is too large, don't move the vertex



Original (2862 vertices)



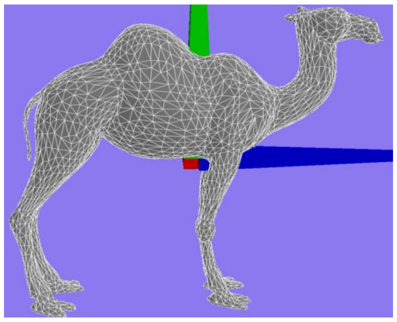
Smoothing (1020 vertices were relocated)

Figure 17: A re-meshing example for a camel model after one round of smoothing.

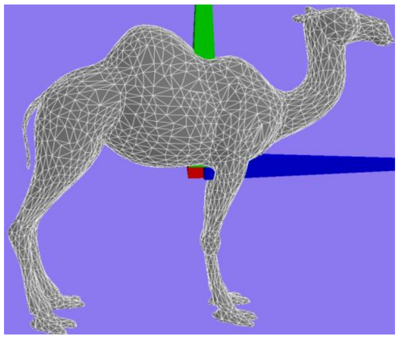
3.5.4. Edge-Flip

In this operation, for any given 2 neighboring triangles, as shown in figure 16, we flip one diagonal if longer than the other. We apply edge-flips only if the Smoothness Error for all the vertices of the triangles adjacent to the edge is smaller than some predefined threshold.

When combined, the sequence of these local operations provides an accurate and robust re-meshing algorithm that can be applied to meshes of arbitrary genus and geometries. This re-meshing scheme is very efficient, allowing re-meshing at interactive rates for meshes of up to tens of thousands of vertices.



Original (8,580 edges)



Edge-flip (1,431 edges were flipped)

Figure 19: A re-meshing example for a camel model after one round of edge-flip

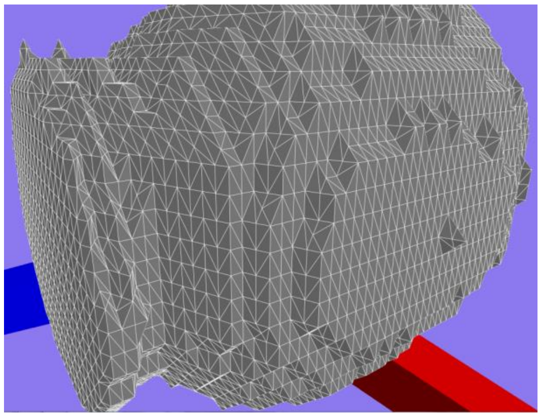
4. Implementation Tools

We use MATLAB to perform image segmentation on each slice and contour extraction. The Image Processing Toolbox is used for image segmentation, Volume Visualization tools are used for some initial tests of mesh construction and an implementation of contour extraction using active contours written in MATLAB and published in MATLAB Central [23].

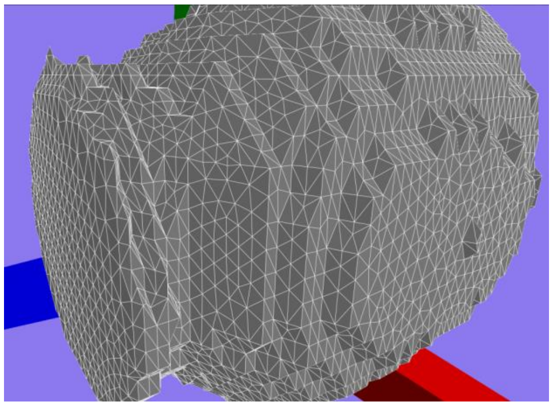
The final results for mesh construction with marching cubes and re-meshing are all written in C++ using Cartel library developed at UBC [24].

Results

After performing surface extraction with a marching cubes algorithm on de-noised contours, we performed our adaptive re-meshing algorithm. Figure 20 shows the final results. Before implementing our re-meshing scheme on the marching cubes, we tried it on other less noisy meshes to ensure the algorithm works, as shown in figure 21.

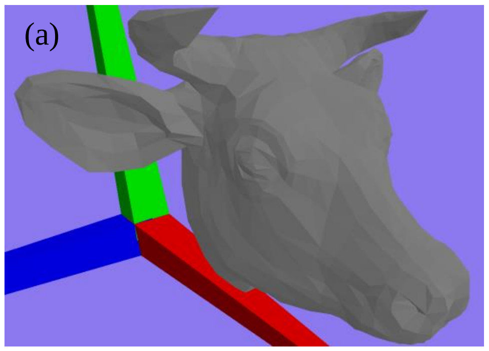


Original (7,394 vertices)

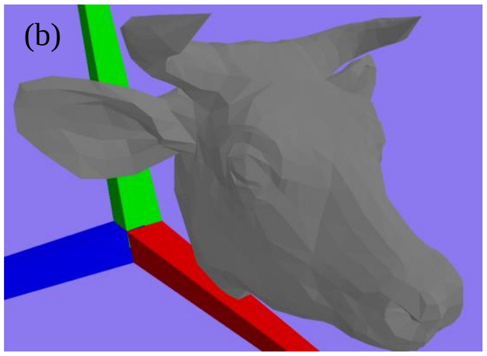


Re-mesh (7,996 vertices)

Figure 20: The final re-meshed result for the prostate model generate by marching cubes. The overall mesh looks better and some sharp spikes are smoothened, however the overall noise is still clearly present in the mesh due to the marching cubes (even after countour denoising).



Original (995 vertices)



Re-mesh (1005 vertices)

Figure 21: A re-meshing example for the cow model after 7 iterations. The head and the ears are much smoother in the re-meshed model

Discussion

Although the resulting mesh is similar to the original shape of the prostate phantom, there are some limitations to the selected approach for mesh construction. Two approaches were taken to reduce the noise in the resulting mesh: manual contour de-noising and resolution reduction. However, noise is still clearly present in the mesh due to the marching cubes algorithm, which couldn't be easily eliminated with the described re-meshing algorithm as the re-mesher preserves the general topology of the shape. In such cases, a mesh de-noising algorithm would be ideal to create a smooth surface that resembles the surface of the modelled prostate phantom.

Another apparent limitation for the implemented marching cubes algorithm is the manual setting of the spacing between the slices. Using the same proportions between the slices as between the pixels in each image might result in some undesired output, such as shown in figure 17, with an overall stretched mesh in one direction.

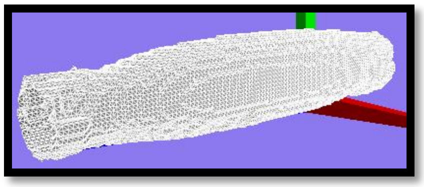


Figure 22: Undesired output due to uniform spacing between slices and pixels

There were some limitations to our adaptive re-meshing scheme as well. We found it difficult to control the resulting number of vertices as well as the spacing between these vertices. Also, the selected threshold angle greatly affects the output result, and should change from one mesh to another for optimal results. In other words, it was very difficult to come up with a good global threshold on our error metrics. Lastly, there is no single rule that determines the sequence of the local re-meshing operations, and the number of iterations of each operation. We found that the sequence and the number of iterations of these operations worked differently on different meshes, and had to be manually set with trial and error. Therefore, there was no guarantee to a global optimal solution.

Conclusion and Future Work

This paper presented a simple method for the extraction and enhancement of a 3D mesh of an area of interest from ultrasound volumetric data. The steps used to achieve this goal included image segmentation, contour extraction and de-noising, surface construction and re-meshing. The output highly resembled the shape of the prostate imaged with a little bit of artifacts that can be further improved with mesh de-noising and refinement.

Future work would include mesh refinement techniques, such as mesh de-noising and further smoothing. Also, it is essential to experiment later with images of prostates in-vivo to evaluate the effectiveness of this method with real data obtained from patients.

Reference

1. L.B. Cruz, “Basics on 3D Ultrasound,” *Jaypee Journals*, vol. 2, no. 4, 2008, pp. 83.

2. V. Surazhsky and C. Gotsman, “Explicit surface remeshing,” *Proc. Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, Eurographics Association, 2003, pp. 20-30.

3. A. Fenster, et al., “Visualization and Segmentation Techniques in 3D Ultrasound Images,” *Computer Vision Beyond the Visible Spectrum*, Springer, 2005, pp. 241-269.

4. R. Chaves, et al., “Structured mesh generation from Doppler ultrasound images,” *Proc. submitted to 15 th International Conference on Experimental Mechanics, ICEM15*, 2012.

5. Y. Zhang, et al., *Direct surface extraction from 3D freehand ultrasound images*, IEEE, 2002.

6. W.E. Lorensen and H.E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” *Proc. ACM Siggraph Computer Graphics*, ACM, 1987, pp. 163-169.

7. G.M. Treece, et al., “Regularised marching tetrahedra: improved iso-surface extraction,” *Computers & Graphics*, vol. 23, no. 4, 1999, pp. 583-598.

8. P. Alliez, et al., “Recent advances in remeshing of surfaces,” *Shape analysis and structuring*, Springer, 2008, pp. 53-82.

9. D.P. Luebke, *Level of detail for 3D graphics*, Morgan Kaufmann, 2003.

10. M. Eck, et al., “Multiresolution analysis of arbitrary meshes,” *Proc. Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, 1995, pp. 173-182.

11. I. Guskov, et al., “Normal meshes,” *Proc. Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 2000, pp. 95-102.

12. A.W. Lee, et al., “MAPS: Multiresolution adaptive parameterization of surfaces,” *Proc. Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, 1998, pp. 95-104.

13. K. Hormann, et al., “Remeshing triangulated surfaces with optimal parameterizations,” *Computer-Aided Design*, vol. 33, no. 11, 2001, pp. 779-788.

14. P. Alliez, et al., "Interactive geometry remeshing," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3, 2002, pp. 347-354.

15. X. Gu, et al., "Geometry images," *Proc. ACM Transactions on Graphics (TOG)*, ACM, 2002, pp. 355-361.

16. P.V. Sander, et al., "Signal-specialized parameterization," 2002.

17. H. Hoppe, "Progressive meshes," *Proc. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, 1996, pp. 99-108.

18. A. Rassineux, et al., "Surface remeshing by local Hermite diffuse interpolation," *International Journal for numerical methods in Engineering*, vol. 49, no. 1-2, 2000, pp. 31-49.

19. P.J. Frey, "About surface remeshing," 2000.

20. "Stradwin," http://mi.eng.cam.ac.uk/~rwp/stradwin/.

21. T.F. Chan, et al., "Active contours without edges for vector-valued images," *Journal of Visual Communication and Image Representation*, vol. 11, no. 2, 2000, pp. 130-141.

22. A. Sheffer, "Remeshing," 2014; http://www.cs.ubc.ca/~sheffa/dgp/.

23. Y. Wu, "Chan Vese Active Contours without edges," 2009; http://www.mathworks.com/matlabcentral/fileexchange/23445-chan-vese-active-contours-without-edges.

24. R. Gillette and D. Harisson, "Cartel," 2014; http://www.cs.ubc.ca/labs/imager/tr/2014/CartelModeling/Cartel_website/.