UNIVERSITY OF CALIFORNIA, SAN DIEGO

Remeshing with Learned Image Boundaries

A thesis submitted in partial satisfaction of the requirements for the degree
Master of Science

in

Computer Science

by

Iman Mostafavi

Committee in charge:

Matthias Zwicker, Chair
Serge Belongie
Mark Ellisman
Ruth West

2008

The thesis of Iman Mostafavi is approved and it is acceptable in quality and form for publication on microfilm:

_____

_____

_____

_____
Chair

University of California, San Diego

2008

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This work was supported by the National Center for Microscopy and Imaging Research, NIH / National Center for Research Resource PHS P41RR04050D ELLISMAN. Special thanks to my advisors Ruth West and Matthias Zwicker for their valuable direction and feedback. Thanks to my other committee members Mark Ellisman, without whom this work would not be possible, and Serge Belongie for sparking my interest in computer vision. I would also like to acknowledge Maryann Martone, Gina Sosinsky, and Masako Terada at NCMIR for their input and help accessing the data sets used in this work.

ABSTRACT OF THE THESIS

Remeshing with Learned Image Boundaries

by

Iman Mostafavi

Master of Science in Computer Science

University of California, San Diego, 2008

Matthias Zwicker, Chair

Meshes generated from a typical manual segmentation process are often unsuitable for simulation purposes due to poor element quality and artifacts. In this thesis we describe a remeshing approach for converting low quality triangular meshes into spatially realistic, simulation quality meshes. We improve upon the Laplacian Mesh Optimization remeshing framework by incorporating topology modification, triangle area equalization, and feature detection using computer vision techniques.

We train a supervised edge learning algorithm with human produced contours and corresponding image data. The resulting classifier is used to generate a probabilistic edge map for the entire image. Salient features in the mesh surface are detected as a weighted combination of surface curvature and the learned image boundary probability. We compare the remeshing performance of our algorithm with the original method on real world data sets, showing that our approach can produce higher quality meshes from extremely irregular input meshes while simultaneously enhancing spatial realism.

Chapter 1.

Introduction

1.1 Motivation

Triangular meshes are important for many scientific applications, from visualization to simulation. Popular simulation techniques utilizing triangular meshes in science and engineering include finite element and Monte Carlo methods. One critical input requirement for both simulation approaches is a mesh which is well behaved with respect to the triangle quality. Elements of a finite element mesh should have neither overly large nor overly small internal angles. Such elements can induce large solution errors and cause slow convergence of the finite element solver [6]. Unfortunately, meshes generated from real world data typically exhibit noisy surfaces and poorly conditioned triangles.

In this work we focus on the case of meshes produced through the triangulation of manually drawn 2D serial contours of 3D images. This is the predominant method of segmentation in the biomedical imaging community. Meshes generated from these hand segmentations often exhibit stair-step artifacts partly due to the deviations in the human produced contours from one slice to the next. Since tracing is a slow and laborious process, it is also common practice to expedite the segmentation process by tracing only a fraction of the imaged slices. This amplifies the artifact problem as it reduces

the number of the sample points from which the mesh is generated along the traced dimension. Figure 1.1 shows an example of a typical mesh produced by this protocol. Currently, considerable effort is required to create high quality meshes which can be used for simulation. Our goal is to develop an approach to allow the conversion of the low quality triangular meshes produced by the current surface reconstruction pipeline into spatially realistic, simulation quality meshes.

1.2 Approach

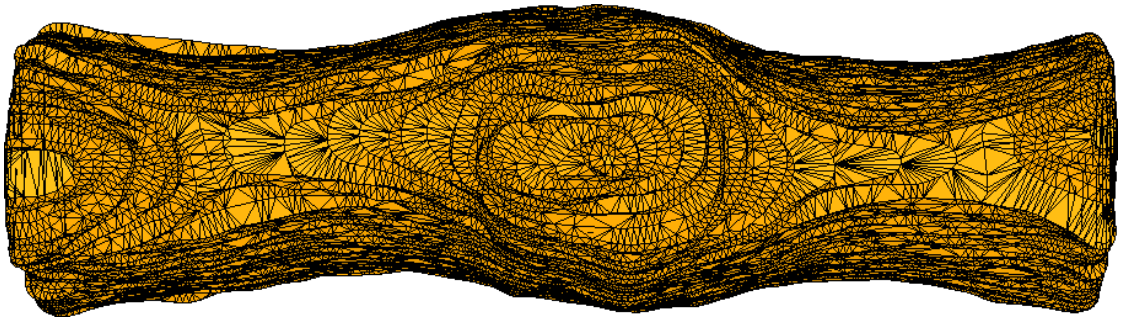
A common solution to the removal of surface noise and artifacts is to smooth the mesh surface. While smoothing may achieve this goal, it may also remove genuine surface features. Our approach differs from most mesh smoothing and optimization methods by revisiting the volume image data in order to improve feature preservation, and potentially enhance spatial realism by restoring surface features lost due to the coarse sampling of the initial segmentation.

Our remeshing approach builds upon Laplacian Mesh Optimization [7], a simple and efficient framework for triangle shape optimization and mesh smoothing. Unlike the original method, we aim to utilize the volume image data from which the input mesh was segmented to further enhance feature preservation. The real world data sets on which we demonstrate our algorithm are acquired via electron tomography. However, our approach is generally applicable to improving mesh quality in scenarios where a 3D volume image is manually segmented to produce a triangular mesh.

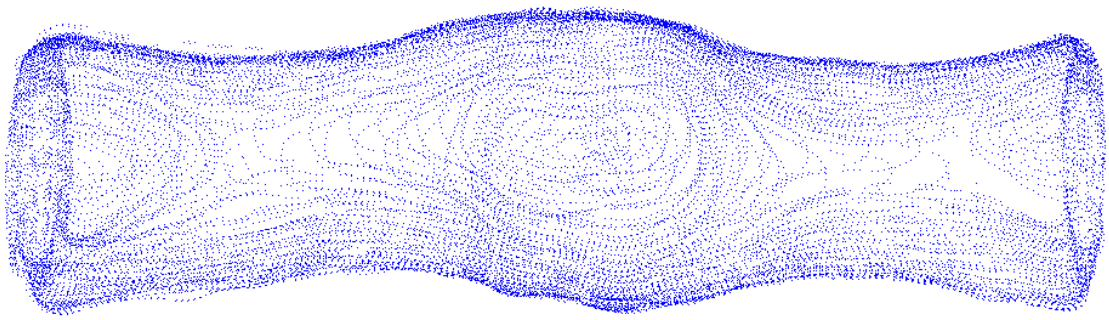
A fundamental way in which our approach differs from the work of Nealen et al. is that we replace the underlying assumption that the input mesh is the ground truth surface with the assumption that the input mesh is only a close approximation of the underlying surface geometry and must be further refined. We supplant their global vertex constraint which penalizes vertices from moving far away from the original surface with one that allows vertices to follow strong edge features detected in the volume image.

Our method takes slices of the volume image data and the corresponding 2D

(a) Surface reconstruction



(b) Surface reconstruction with edges visible



(c) Mesh vertices

Figure 1.1: A typical mesh reconstructed from manual segmentation. Stair step artifacts are shown in (a), while (b) emphasizes the poor mesh quality, and (c) illustrates the uneven distribution of vertices.

contours defined on them during a manual segmentation process, and trains a Boosted Edge Learning classifier [3]. The classifier is tested using all of the slices of the original volume in order to create a series of 2D edge maps, which are combined to form a 3D volume edge map. The volume edge map is converted to a gradient vector flow field [17] which is used to influence the global shape optimization and smoothing framework.

We integrate simplification and subdivision algorithms in order to optimize the mesh connectivity and achieve the desired output mesh resolution. By default our algorithm produces adaptively sampled meshes, placing more vertices at areas of high curvature. However, we implement a triangle area equalization technique to optionally create uniformly sampled meshes for applications which require this type of mesh.

1.3 Contribution

The main contributions of our work are summarized below:

- **Powerful combination of remeshing approaches**

  We combine Laplacian Mesh Optimization [7], a global vertex relocation framework which is effective at triangle shape optimization and feature preserving smoothing, with simplification and subdivision algorithms. As a result we improve the algorithm's performance on highly topologically irregular meshes. With this modification to the approach we can output simulation quality adaptive resolution meshes. We further extend the framework to optionally produce uniformly sampled meshes by applying an iterative local area based remeshing technique which attempts to equalize triangle areas.

- **Learned edge based feature detection for remeshing**

  We introduce the use of a supervised edge learning approach in conjunction with a remeshing algorithm to detect salient surface features and constrain vertex relocation. We show that this constraint can potentially enhance the spatial realism of the resulting mesh by enhancing feature preservation.

1.4 Thesis Overview

The remainder of this thesis is organized as follows. **Chapter 2** will introduce the related meshing, remeshing, computer vision, and learning algorithms. **Chapter 3** covers the entire remeshing process, with specific details about our approach. **Chapter 4** discusses implementation details. **Chapter 5** presents our results and analysis. **Chapter 6** outlines future avenues for improvement.

Chapter 2.

Background and Related Work

2.1 Triangle Quality Measurement

Pebay and Baker [8] demonstrated that there are five major classes of triangle quality measures, all of them based on the dimensionless ratios of various geometric parameters of a triangle. A property common to all triangle quality metrics is that they return a value of approximately one when applied to an equilateral triangle (optimal quality), and are far from one when applied to a "needle" or "sliver" triangle (poor quality). In our work we evaluate the quality of triangles using the radius ratio defined as

$$q_i = 2\frac{r}{R}, \tag{2.1}$$

where R and r are the radii of the circumscribed and inscribed circles for a triangle, respectively. A value of q_i near one indicates a well shaped triangle, while a value far from one indicates a degenerate triangle. For our results in Chapter 5, we normalize the triangle quality histograms to lie between zero and one, where zero represents a degenerate triangle, and one an optimal triangle.

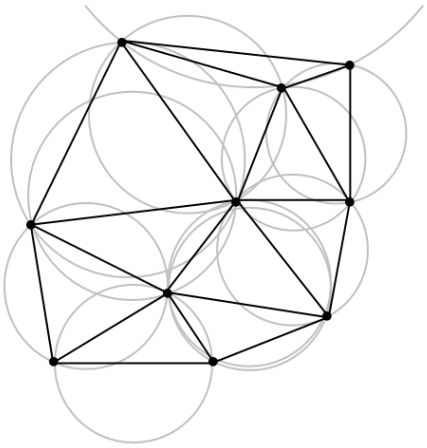


Figure 2.1: Delaunay triangulation of a planar set of points.

2.2 Mesh Generation

Before we investigate remeshing, it is helpful to briefly discuss how meshes are typically generated. Perhaps the most common class of mesh generation algorithms are the ones which generate Delaunay triangulations. An interesting property of Delaunay triangulations is that they maximize the minimum angles of all of the triangles in the mesh, and therefore tend to avoid "sliver" triangles. Delaunay algorithms are an attractive choice for mesh generation due to the angle property, and the fact that efficient $O(n\ log\ n)$ implementations exist for generating triangulations.

In 2D, a Delaunay triangulation for a set of points P is a triangulation $T(P)$ such that no point P is inside the circumcircle of any triangle in $T(P)$. This definition extends to higher dimensions in the Euclidian space, in which a Delaunay triangulation is defined as a triangulation $T(P)$ such that no point in P is inside the circum-hypersphere of any simplex in $T(P)$.

The meshes used in our experiments were created using the software Nuage [2], which uses an algorithm based on Delaunay triangulation for creating surfaces from a series of 2D point contours.

2.3 Remeshing

2.3.1 Vertex Relocation

The idea of repositioning vertices is a subproblem in the context of remeshing [1]. Many algorithms circumvent the problem of relocating the original mesh vertices by resampling the surface [16]. The majority of remeshing algorithms apply a relocation step one vertex at a time, until some criteria are reached. A more recent approach to the vertex relocation problem is to compute a unique solution of a sparse linear system, simultaneously solving for all updated vertex locations. Our vertex relocation framework for both triangle shape optimization and smoothing is based on Laplacian Mesh Optimization [7], one such method.

The Laplacian Mesh Optimization method relies on the least-squares meshes algorithm [12] to perform inner (triangle shape) and/or outer (surface smoothness) fairing. Soft positional constraints are applied to all mesh vertices, where the weights depend on discrete curvature distribution, in order to preserve specific features. The notion of prescribing positional constraints was used in Schneider and Kobbelt's [9] work in geometric fairing for freeform surface design.

2.3.2 Simplification and Subdivision

Simplification algorithms reduce the number of triangles in an input mesh by a desired amount, while striving to remain a geometrically close approximation of the higher resolution initial surface. Also known as decimation algorithms, they generally proceed by classifying all vertices in a mesh and inserting them into a priority queue. The priority is based on the error to delete a vertex and retriangulate the hole. Each vertex in the priority queue is deleted, and the resulting hole triangulated using edge collapse. This process continues until either the priority queue is empty or the desired level of simplification is achieved. In our framework, we utilize the approach described in [11].

Subdivision algorithms refine an initial mesh by subdividing each triangle through the creation new vertices and new faces. This process produces a mesh which is more dense than the original one. The major subdivision schemes fall into two categories– interpolating and approximating. Interpolating schemes are required to match the original position of the vertices in the initial mesh, while approximating schemes are not. We desire to preserve the features in the original mesh as much as possible, therefore we choose to use an interpolating scheme known as Butterfly subdivision. The Butterfly scheme was first proposed by Dyn, Gregory, and Levin in [4]. The 8-point Butterfly scheme we use in our approach was described in [18], and improves on previous similar methods due to special treatment of vertices with valence other than six.

2.3.3 Area-Based Remeshing

The concept of using triangle areas as one criteria for mesh optimization is not new. However, using triangle areas alone cannot be used to obtain high quality meshes. When only triangle areas are optimized, without taking into account the angles, the resulting mesh can have many long and skinny triangles [14]. Surazhsky observed that only when a mesh has almost regular connectivity may uniform triangle areas imply well shaped triangles.

One important property of a mesh containing triangles with closely equal area is that the spatial distribution of the vertices over the total mesh is uniform. This is a desirable property for certain types of simulations, such as the Monte Carlo algorithms used for simulating cellular physiological processes in MCell.

We address the needs of both applications where adaptive and uniform meshes are required. Our general remeshing approach produces high quality adaptive resolution meshes, which can then be used as a preconditioned input to the area equalization method in order to distribute vertices evenly while maintaining good triangle quality.

2.4 Computer Vision and Learning

Our work draws upon two major techniques described in the computer vision literature. The learning component of our framework uses Boosted Edge Learning [3], an algorithm designed for the supervised learning of edges and object boundaries. We train this algorithm with the user produced traces from which the original mesh is generated, and then test the algorithm on all slices in the original image volume. The algorithm outputs an edge map which becomes especially useful in providing boundary knowledge for slices which were not traced in the original image data, but skipped over to expedite the manual segmentation process.

We employ the Gradient Vector Flow technique [17] to produce a smooth vector field from the gradient of the edge map produced by the BEL algorithm. By constraining the input mesh vertex movement based on this vector field, among other smoothness and triangle shape constraints in the Laplacian Mesh Optimization framework, we evolve the mesh in a fashion which enhances spatial realism. Using the GVF as an external force in the case of snakes and active contours has been shown to produce more desirable results compared to simply using the gradient of an edge map. Since the GVF field is derived from a diffusion operation, the vectors tend to be smoother and extend further away from the object, effectively extending the "capture range" (edges that are further away from its initialized position can have some effect). Our goal is to utilize this property to allow the external gradient vector force to inform feature preservation.

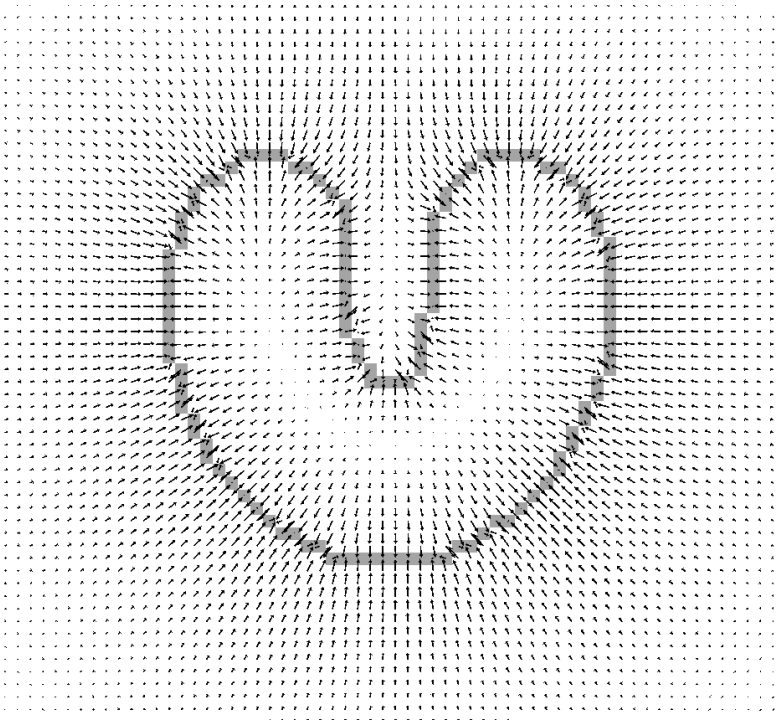


Figure 2.2: Example of a gradient vector flow field computed on the gradient of a U-shaped curve [17].

Chapter 3.

System Overview

Our remeshing system takes as its input a manifold triangular mesh, along with optionally, its corresponding 3D volume image, and the 2D serial contours traced by an expert. As a preprocessing step, the 2D serial contours, along with the original input 3D volume image are used to train the Boosted Edge Learning classifier. The classifier produces an edge map that is converted to a gradient vector flow volume and used by the remeshing algorithm to guide the evolution of the mesh surface. The final output of the algorithm is a new triangular mesh with the desired amount of feature preserving smoothing, improved triangle shape quality, and enhanced spatial realism. Remeshing with no image feature constraints is possible simply by not including the gradient vector flow volume as an input to the remeshing algorithm. In this case we use tangent plane constraints (described in section 3.3.3) for the smoothing and shape optimization.

3.1 Input Data

The pipeline for our remeshing process begins with the acquisition of some form of volumetric imaging data. Common imaging modalities include Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET), Computerized Axial Tomography (CAT), and Electron Tomography (ET). In this work we demonstrate our algorithm

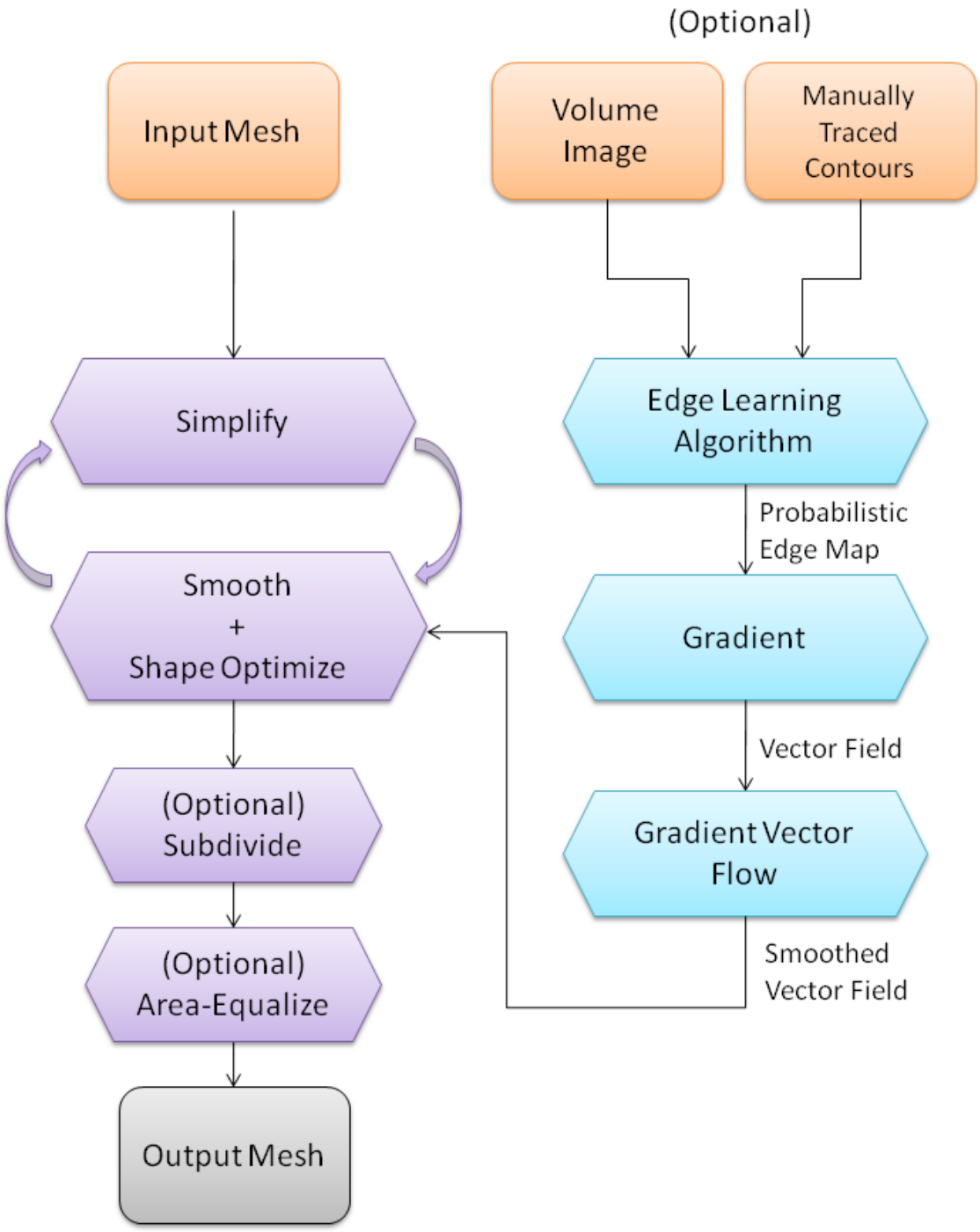


Figure 3.1: System overview.

on data sets acquired through the Electron Tomography technique, although our method should be generally applicable to other imaging modalities as well. Electron Tomography is a technique for obtaining detailed 3D images of objects in the nanometer scale. A beam of electrons is passed through a sample at incremental degrees of rotation around the center of the target sample. The collected information is then used to assemble a 3D image of the target.

The data sets used in this study were provided by the National Center for Microscopy and Imaging Research. In particular we perform experiments with two data sets–a Node of Ranvier, and a spiny dendrite. Nodes of Ranvier are known as the gaps (about 1 micrometer in diameter) formed between myelin sheath cells along axons or nerve fibers. Dendrites are the branched projections of a neuron that act to conduct the electrical stimulation received from other neural cells to the cell body. A dendritic spine is a small membranous protrusion from the central shaft of a dendrite that is typically electrophysiologically active and synapses with a single axon. The study of these structures is an active area of research and critically important to the understanding of their function in living organisms.

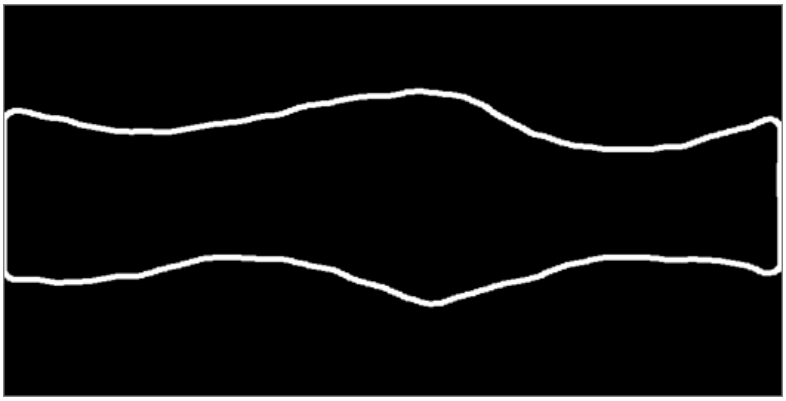
Within the Node of Ranvier data set, the region of interest which has been segmented to create one of our input meshes is a section of an axon. We also evaluate our remeshing algorithm on a portion of the shaft, and two different spines from the spiny dendrite data set.

3.2 Preprocessing

The Boosted Edge Learning algorithm is a standalone Windows executable provided by Dollar et al., which requires a series of 2D images along with corresponding binary labeled images as training input. Some processing is required to convert the contour files generated by the tracing tool (Xvoxtrace in our case) into binary labeled edge map images which the BEL algorithm can use.

Upon successful training of the BEL classifier on the traced slices of the original

(a) Training image



(b) Human labeled contour

Figure 3.2: Example pair of training images for the BEL algorithm.

Figure 3.3: Example output of the BEL algorithm on a different test image.

image data, the classifier is executed in test mode on all available image slices in the original volume. The resulting output is a series of 2D edge probabilities, which are then combined to form a 3D edge map.

Finally, we use the Image Processing Toolkit [5] to generate a gradient vector flow volume from the edge map volume. This gradient vector flow volume is provided as an optional additional input to the remeshing algorithm, where it can be utilized as a constraint in the vertex relocation framework.

3.3 Remeshing Algorithm

The three primary goals of the remeshing algorithm are to improve the triangle quality, remove noise and artifacts (but not features), and to enhance the spatial realism of the mesh.

3.3.1 Notation

A mesh is represented as a graph $G = (V, E)$, with vertices V and edges E. $V = [v_{1x}, v_{1y}, v_{1z}, v_{2x}, v_{2y}, v_{2z}, ..., v_{nx}, v_{ny}, v_{nz}]^T$, a vector of length $3n$ containing the original geometry, and V' denotes the displaced geometry. The Laplacian, or in other words the result of applying the discrete Laplace operator to an individual vertex $v_i = [v_{ix}, v_{iy}, v_{iz}]^T$ is:

$$\delta_i = \sum_{i,j \in E} w_{ij}(v_j - v_i), \tag{3.1}$$

where the sum of the weights equals one and the choice of the weights

$$w_{ij} = \frac{\omega_{ij}}{\sum_{i,k \in E} \omega_{ik}} \tag{3.2}$$

defines the nature of δ_i. Two popular choices for the weights are

$$\omega_{ij} = 1, \tag{3.3}$$

$$\omega_{ij} = \cot\alpha + \cot\beta, \tag{3.4}$$

where (3.3) are the uniform and (3.4) are the cotangent weights. We will refer to the uniform and cotangent Laplacians as δ_u and δ_c respectively.

The Laplacians for an entire mesh can be computed using an $n \times n$ Laplacian matrix L with elements

$$L_{ij} = \begin{cases} -1 & i = j \\ w_{ij} & (i,j) \in E \\ 0 & \text{otherwise} \end{cases} . \tag{3.5}$$

We denote L_u and L_c the Laplacian matrices with uniform and cotangent weights. $V_d = [v_{1d}, v_{2d}, ..., v_{nd}]^T$ where $d \in x, y, z$, is an $n \times 1$ vector containing the x, y, or z coordinates of the n vertices. The x, y, and z Laplacians $\Delta_d = [\delta_{1d}, \delta_{2d}, ..., \delta_{nd}]^T$ where $d \in x, y, z$ are computed separately as

$$\Delta_d = LV_d. \tag{3.6}$$

The uniform Laplacian of a vertex v_i points to the centroid of its neighboring vertices. The cotangent Laplacian is known to be a good approximation of the surface normal. When scaled by the Voronoi region as described by Meyer et al., we obtain the discrete mean curvature normal, which is the unit length surface normal n_i scaled by the discrete mean curvature κ_i.

$$\kappa_i n_i = \frac{1}{4A(v_i)} \sum_{i,j \in E} (\cot\alpha + \cot\beta)(v_j - v_i). \tag{3.7}$$

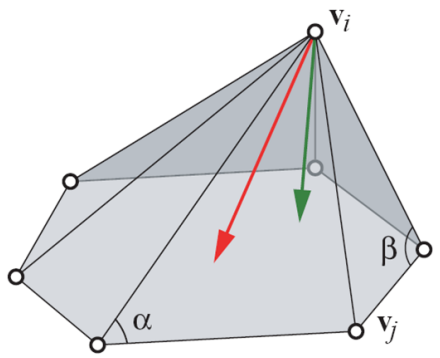


Figure 3.4: Uniform (red) and cotangent (green) Laplacian vectors for a vertex v_i and its 1-ring neighborhood, as well as the angles used in Equation 3.4 for one v_j.

3.3.2 Background

Sorkine and Cohen-Or demonstrate that solving the $(n+m) \times n$ overdetermined linear system

$$\frac{L_u}{I_{m \times m}|0} \cdot V'_d = \frac{0}{V_{(1...m)d}} \tag{3.8}$$

in the least square sense using the method of normal equations $V'_d = (A^T A)^{-1} A^T b$ can reconstruct a mesh from connectivity information alone using a small subset m of geometrically constrained anchor vertices. The reconstructed shape is smooth, as the minimization procedure moves each vertex to the centroid of its 1-ring since the uniform Laplacian is used.

Nealen et al. then show how least squares optimization can improve triangle quality in a small mesh region with negligible vertex drift. They modify the linear constraints in (3.8) by

$$\frac{L_u}{I_{m \times m}|0} \cdot V'_d = \frac{\Delta_{d,c}}{V_{(1...m)d}}, \tag{3.9}$$

asking the uniform Laplacian of each deformed vertex position to resemble its undeformed cotangent Laplacian as closely as possible. Since the uniform Laplacian has a

tangential component and the cotangent Laplacian does not, the optimization attempts to remove the tangential components while preserving the surface details in the normal direction.

Nealen et al. generalize Equation 3.8 and 3.9 in their Laplacian Mesh Optimization approach, where they solve the $2n \times n$ system

$$\frac{W_L L}{W_p} \cdot V_d' = \frac{W_L f}{W_p V_d}. \tag{3.10}$$

The main modification is that instead of having a subset of vertices as positional constraints, all vertices appear both as Laplacian and positional constraints. By introducing the diagonal positional constraint weight matrix W_p and weighting the Laplacian matrix L and right hand side with the diagonal matrix W_L, precise control can be achieved regarding the enforcement of regular triangle shapes and/or surface smoothness on a per vertex level. In general larger weights in W_p enforce positional constraints which preserve the original geometry, ideal for high curvature regions and sharp features. Larger W_L weights enforce regular triangle shapes and/or surface smoothness. Setting $f = \Delta_{d,c}$ maximizes detail preserving triangle shape optimization, whereas setting $f = 0$ performs mesh smoothing.

The choice of weight assignment is critical to the behavior of the mesh optimization algorithm. Nealen et al. demonstrated good feature preservation in general when using the cumulative density function (cdf) of the mean curvature κ of the mesh vertices to map from κ_i to $w_i \in [0, s]$, resulting in $W_p = W_{cdf}$. The logic behind using the cdf based positional weights is that if a mesh with a large amount of low curvature vertices is input to the algorithm, those vertices should be assigned a larger weight than if a simple linear ramp is used for weight assignment (which does not take relative frequency of mean curvature into account). The linear weighting scheme would in this case allow too much freedom for the low curvature vertices and may result in excessive loss of detail.

An additional trick Nealen et al. employ aims to improve the quality of the worst triangle in the mesh, which is an important criterion for a simulation quality mesh. To (heuristically) maximize the quality of the worst triangles, the positional weights W_p

are modulated by another diagonal matrix W_t where the entry for $w_{t,i}$ for vertex v_i is set to the minimal triangle quality t of its adjacent triangles. In essence this allows a vertex attached to triangles with a small t (poor quality) to have more freedom than without the modulation. The intuition for this modification is that such a vertex may more easily be relocated to a position which results in overall improved triangle quality for its neighborhood.

3.3.3 Additional Constraints

Nealen et al. use the notion of adding additional constraints to the system in order to reduce the geometric error caused by vertex displacement during remeshing. They couple the additional n constraints (one for each vertex) of the form

$$n_i \cdot v_i' = n_i \cdot v_i, \tag{3.11}$$

resulting in a $7n \times 3n$ system. The constraints penalize displacement perpendicular to the tangent plane defined by the original vertex position v_i and the local surface normal n_i, assuming a first order approximation of the surface around v_i. Although adding the constraints involves significant extra computational overhead, it was shown to reduce geometric error as they defined it (excessive displacement of vertex positions in the final mesh).

In our work we replace the tangent plane constraint with the constraint in Equation 3.12 when a gradient vector flow volume is available. By using the gradient flow vector g_i at each vertex v_i instead of the normal, we constrain the vertices to move perpendicular to the direction from the vertex to the closest edge feature in the mesh. In the case that the edges detected by the boundary learning algorithm are completely in agreement with the initial input mesh, this constraint becomes equivalent to Equation 3.11. However, in the cases where the detected image feature edges are not in agreement with the initial segmentation, the deformed vertex normals will be optimized to match the image gradient. This should produce a surface with normals which are in agreement with the image where reliable features are detected.

$$g_i \cdot v_i' = g_i \cdot v_i \tag{3.12}$$

3.3.4 Feature Based Positional Weight

We introduce an additional per vertex weight matrix W_g which scales the overall positional constraint weights. We assign a per vertex weight value based on the magnitude of the gradient flow vector. A greater magnitude implies a strong edge is already very close to the vertex, so its positional weight is adjusted accordingly to prevent it from moving too much. A small gradient magnitude at a vertex means that no strong feature edges have been detected in the vicinity of the vertex and it is permitted to move more freely for the benefit of the smoothing and shape optimization. This image feature based constraint is included to enhance the feature preserving smoothing capabilities of the remeshing algorithm.

3.3.5 Triangle Area Equalization

One of the goals of our remeshing framework is being able to produce meshes with uniform sampling. Equivalently, we can solve an optimization where we attempt to equalize the areas of the triangles in the mesh. Our approach is based on a local technique applied for 2D triangle area equalization on parameterized mesh patches [14]. Surazhsky et al. use an iterative method where each vertex's one ring neighborhood is analyzed, and a new vertex position $p = (x, y)$ is found which equalizes the areas of the triangles in the one ring as much as possible. Denote the area of triangle p, p_i, p_{i+1}:

$$A_i(x, y) = \frac{1}{2} \begin{vmatrix} x_i & y_i & 1 \\ x_{i+1} & y_{i+1} & 1 \end{vmatrix} \tag{3.13}$$

It then follows that:

$$(x, y) = \underset{x,y}{\operatorname{argmin}} \sum_{1}^{k} \left(A_i(x, y) - \frac{A}{k} \right)^2 \tag{3.14}$$

where A is the total area of the one ring neighborhood around the vertex. We extend this formulation to 3D and apply it directly to the mesh vertices, rather than to a 2D parameterization of the mesh.

3.3.6 Algorithm Overview

Input: original mesh, (optional) gradient vector flow volume

Output: new mesh

for $n = 1, 2, ...simplify - smooth - optimize - iterations$ **do**
- simplify
- compute Laplacian matrices L_u and L_c
- compute mean curvature normals
- generate cdf and apply mapping of curvature to weights
- compute and apply triangle modulation weights
- **if** *GVF file provided* **then**
  - apply GVF based weights
  - set GVF constraints
- **else**
  - use tangent plane constraints
- **end**
- setup and solve for final vertex positions

end

//Optional subdivision to increase mesh resolution if desired

for $s = 1, 2, ...subdivision - iterations$ **do**
- subdivide

end

//Optional triangle area equalization to produce uniform sampling

for $s = 1, 2, ...area - equalize - iterations$ **do**
- area-equalize

end

Chapter 4.

System Implementation

Our algorithm has been implemented in a prototype tool as a class which extends the functionality of the Visualization ToolKit (VTK) [10]. Helper scripts for contour and image data file conversion to the format usable by the Boosted Edge Learning classifier are written in MATLAB, and the utility we implement to generate a gradient vector flow volume from the BEL classifier's output is based primarily on the functionality in the Insight Segmentation and Registration ToolKit (ITK) [5].

4.1 Boosted Edge Learning

The Boosted Edge Learning classifier is provided as an executable for Windows by Dollar et al. Initially demonstrated in the context of natural images, we apply this algorithm to the Electron Microscopy data because it performs significantly better than any alternative edge detection methods, such as the classic Canny edge detector, or simple gradient magnitude thresholding of our images. The BEL executable takes as its input n pairs of uncompressed .TIF files, where each pair is an input image (grayscale or color), and a companion binary image in which the pixels are labeled positive or negative examples by their pixel intensities. Values greater than one represent positive examples, and zero values represent non-boundary pixels.

Our contour data comes from the NCMIR tool Xvoxtrace and exists as proprietary format .trace files. Within the trace files, the closed contours are represented as a varying number of equally spaced points in the shape of the object's cross section. A MATLAB script was developed to parse the .trace files and output an uncompressed .TIF binary image for each contour slice such that a smooth curve passes through the points.

Training is performed on all of the slices for which human produced contours are available. After training, the entire data set is provided as input to the classifier in test mode, producing a series of 2D edge map images where probable edges are represented as pixel values greater than one. The stack of edge map images are combined using the tool MRIcro to produce a 3D Analyze formatted volume.

4.2 VTK

The Visualization ToolKit is an open source software system for 3D computer graphics, image processing, and visualization. It consists of a C++ class library which supports a wide variety of visualization algorithms and modeling techniques. VTK contains file loaders for a variety of file formats, and its own OpenGL based viewer which makes rapidly prototyping an interactive application fairly easy and efficient. The object oriented, pipelined architecture makes it straightforward to perform a series of complex operations on data with any of the built in classes. Developing a new class, which is what we have done in order to create our remeshing algorithm, is the primary way to expand or customize the library's functionality.

In our case, we derive a new class from the vtkPolyDataAlgorithm class, and implement our algorithm inside it according to VTK specifications. In VTK, a vtkPolyData object is a concrete implementation of vtkDataSet, and represents a geometric structure consisting of vertices, lines, polygons, and/or triangle strips. VTK provides convenient and efficient methods for traversal and manipulation of geometric data such as triangular meshes, which is why we choose to implement our algorithm within its framework.

Most operations in VTK are performed by instantiating a "filter" class and connecting the appropriate input(s) and output(s) to the filter. Most filters have one input and one output, as is true in our remeshing filter implementation.

4.3 Core Application Pipeline

The core application pipeline consists of several interconnected filters. The first stage includes the vtkVRMLImporter filter, which loads our VRML formatted input mesh from disk. The output of the file loader filter is connected to a vtkGeometryFilter in order to ensure that our input mesh contains only triangles. The GeometryFilter's output is then connected directly to our remeshing class, vtkLaplacianMeshOptimize. Optionally, the output of our smoothing and shape optimization filter is passed as input to one ore more of the following filters: the vtkAreaEqualize filter, which performs triangle area equalization, vtkDecimatePro for simplification, or vtkButterflySubdivisionFilter for subdivision. The output of the last filter in this chain is connected to the vtkMeshQuality filter, which performs several triangle quality measures and prints the results. The output of our filters can also be mapped to a vtkActor, which represents an object (geometry and properties) in a rendered scene. If this option is selected the output mesh is visualized using a simple graphical user interface. We export the mesh to a VRML or PLY formatted file by passing the PolyData object which represents the mesh to the relevant file exporter filter.

4.4 Gradient Vector Flow

We call upon the itkImageFileReader filter to load our gradient vector flow volume, which is the second optional input to the algorithm besides the initial input triangular mesh. The gradient vector flow image is a volume image with the same dimensions as the initial scalar volume, however at each voxel coordinate there exists a 3D vector instead of a scalar value.

The generation of the GVF volume consists of several steps. First, an itk image filter which reads normal grayscale Analyze formatted volume images is used to load the 3D edge map produced as the output of the BEL algorithm. The output of the Analyze volume loader is attached to the itkGradientRecursiveGaussianImageFilter, which computes the gradient of the edge map using a simple finite difference method. The output of this filter is a 3D vector field, which is then connected as input to the itkGradientVectorFlowImageFilter. This filter generates the resulting desired smooth flow field by applying generalized diffusion equations to the gradient of the edge map.

4.5 vtkLaplacianMeshOptimize

This class encapsulates the smoothing and shape optimization portion of our remeshing algorithm. The workhorse of the smoothing and triangle shape optimization functionality is the TAUCS [15] library for solving sparse linear systems of equations. TAUCS allows us to store and carry out the matrix operations in a compressed format, minimizing compute time and system memory requirements. By setting up the appropriate matrices as described in Chapter 3, we can solve for the new vertex positions in this filter.

4.6 vtkDecimatePro

Decimation occurs through a series of edge collapse operations such that the edges which minimally increase the geometric distance of the simplified mesh to the input mesh are removed first. The decimation criterion is based on the vertex distance to plane for simple vertices inside the mesh, or distance to edge for boundary vertices. In the simple vertex case, if the vertex is within the specified distance to the average plane it may be deleted, otherwise it is retained. In the boundary case, the algorithm determines the distance to the line defined by the two vertices creating the boundary edge. If the distance to the line is less than a specified amount, the vertex can be deleted.

4.7 vtkButterflySubdivisionFilter

This filter implements the 8-point butterfly subdivision scheme described in [18]. Each input triangle is divided into four triangles. Determining where and how to split the triangles depends on the local topology.

4.8 vtkAreaEqualize

This class implements Equation 3.14 and iteratively performs the area equalization on each vertex of the input mesh.

Chapter 5.

Results and Discussion

We evaluate our remeshing algorithm on several real world data sets–the axon from the Node of Ranvier, as well as the shaft and two different spines from a spiny dendrite. Our experiments are designed to explore the parameter space of our remeshing algorithm and to discover the values which yield the best results. We remesh the input meshes with varying degrees of smoothing, gradient weights, and blur kernel sizes for the gradient vector flow volume. We choose a metric for determining the correlation of a mesh with the edge map produced by the BEL algorithm. The purpose of this metric is to verify that our learned image boundaries constraint increases the remeshed result's correlation with the learned edges in the volume data as compared to traditional Laplacian Mesh Optimization, our control method. We also analyze triangle quality statistics for selected meshes to quantitatively measure mesh improvement.

A higher correlation with the edge map implies that the remeshed result is being faithful to the salient edge features learned from the training data. We compute this correlation energy by simply taking 2D cross sections of the remeshed results, convolving them with a small Gaussian kernel, and taking the product of the blurred mesh cross section with the corresponding slice of the learned edge map. The sum over the entire product image is a single number which reveals the correlation between a cross section of the mesh and the edge map. We calculate this measure over all slices for all of

the experimental meshes, and compare the mean correlation energy over the mesh for different remeshing parameters.

Table 5.1: Parameters varied during remeshing experiments.

Smoothing	Topology Optimize	Image Boundaries	Gradient Weights
Low	Off	None	N/A
Low	On	None	N/A
Medium	On	None	N/A
High	On	None	N/A
Low	On	Small Kernel Smoothed	Low
Low	On	Small Kernel Smoothed	Medium
Low	On	Small Kernel Smoothed	High
Medium	On	Small Kernel Smoothed	Low
Medium	On	Small Kernel Smoothed	Medium
Medium	On	Small Kernel Smoothed	High
High	On	Small Kernel Smoothed	Low
High	On	Small Kernel Smoothed	Medium
High	**On**	**Small Kernel Smoothed**	**High**
Low	On	Large Kernel Smoothed	Low
Low	On	Large Kernel Smoothed	Medium
Low	On	Large Kernel Smoothed	High
Medium	On	Large Kernel Smoothed	Low
Medium	On	Large Kernel Smoothed	Medium
Medium	On	Large Kernel Smoothed	High
High	On	Large Kernel Smoothed	Low
High	On	Large Kernel Smoothed	Medium
High	On	Large Kernel Smoothed	High

Our tests show that using high smoothing and gradient weights, while minimally blurring the learned edge map, emphasize the benefits of remeshing with learned image boundaries. As a result, our axon and dendritic shaft results shown here are remeshed with high weights on the gradient vector flow field, and high smoothing. This behavior can be described by the notion that the greater the amount of smoothing, the more the mesh will diverge from the edge map as small features are removed. Therefore, by increasing the weight of the gradient vector flow field, the detected features in the edge map will be more strongly preserved. Increasing the smoothing weights while enabling

large gradient weights further magnifies the beneficial effect of the learning based feature preservation.

The original axon mesh is shown in Figure 5.1 for comparison to the remeshed results. Our reference remeshing result for the axon is shown in Figure 5.2. This mesh represents the output of our implementation of Laplacian Mesh Optimization, upon which we add topology optimization, the learned image boundary constraints, and area based remeshing. We discover that even with high levels of smoothing, ultimately the poor topology of the input mesh does not permit the traditional LMO algorithm to produce good results.

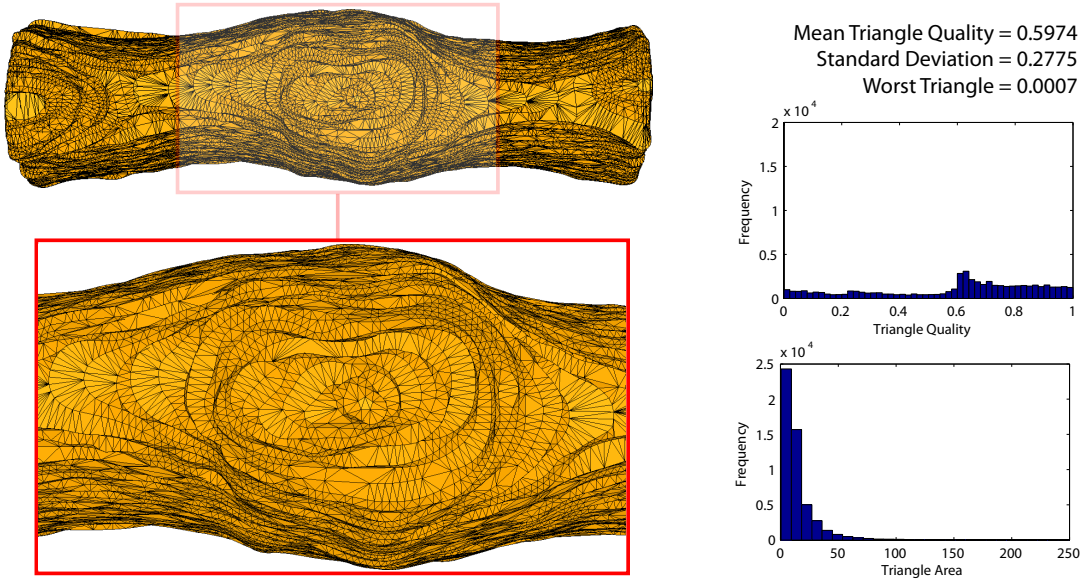


Figure 5.1: Original axon mesh.

Figure 5.3 illustrates the significantly improved remeshing performance we achieve when combining LMO with topology optimization. The histogram of triangle qualities reveals that most of the triangles in the mesh are now nearly equilateral, and the worst triangle in the mesh (often the bottleneck for finite element simulations) is significantly closer to optimal shape compared to the result achieved by the previous method.

Figure 5.4 introduces the results of using learned image boundaries as an ad-

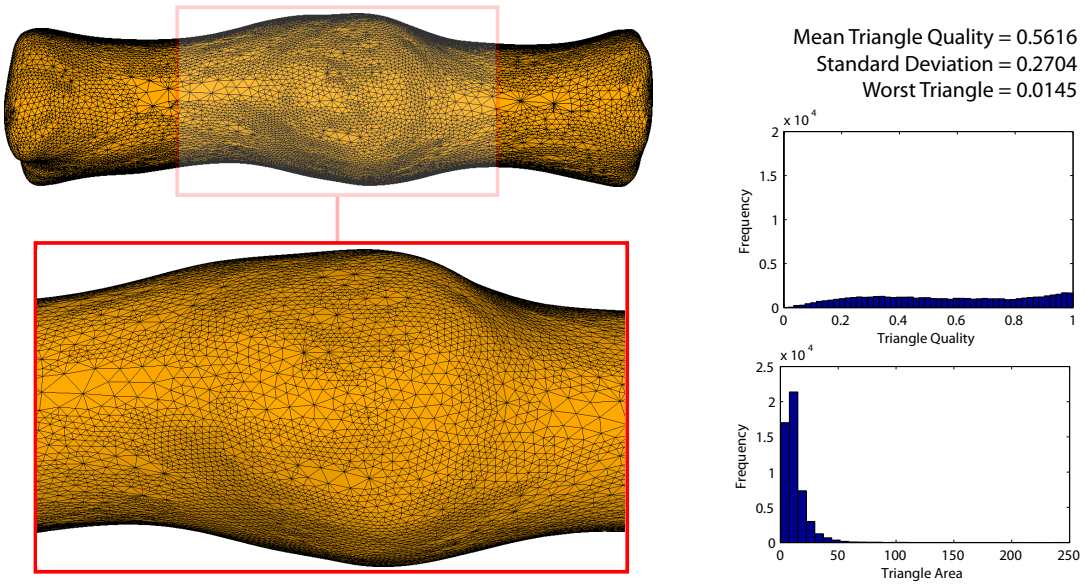


Figure 5.2: Smoothed axon mesh.

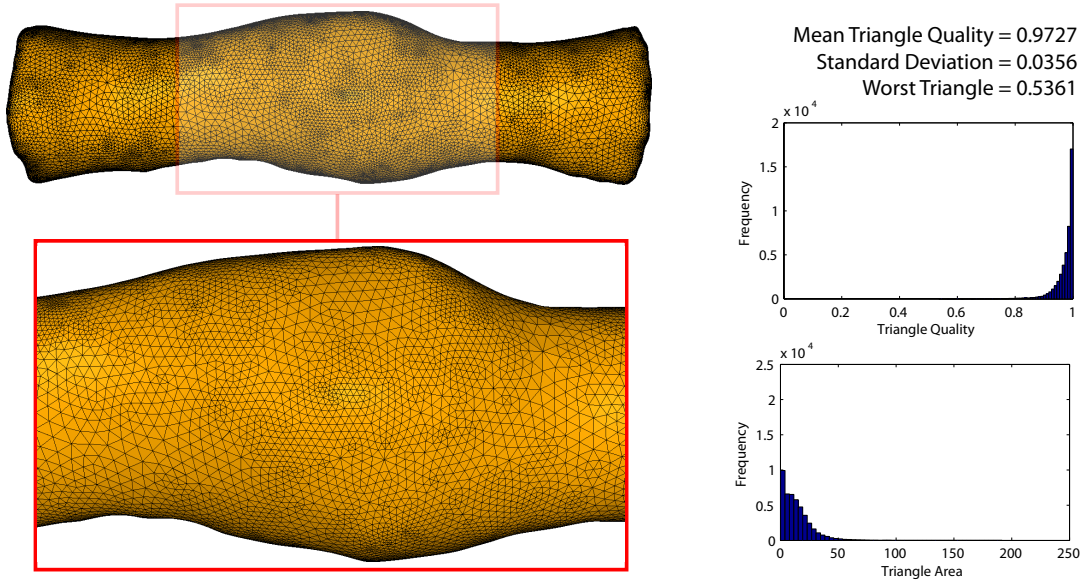


Figure 5.3: Smoothed axon with topology optimization.

ditional constraint on the remeshing process. Overall mesh quality is slightly reduced due to the extra constraints placed on the optimization. However, we observe increased surface feature details. We highlight one particular visible feature in Figure 5.5, a sizeable bump visible on the remeshed surface only when using the learned image boundary constraints. We investigate the this feature on a 2D slice in Figure 5.6 and verify that it is indeed a desired feature and not noise introduced by our algorithm.

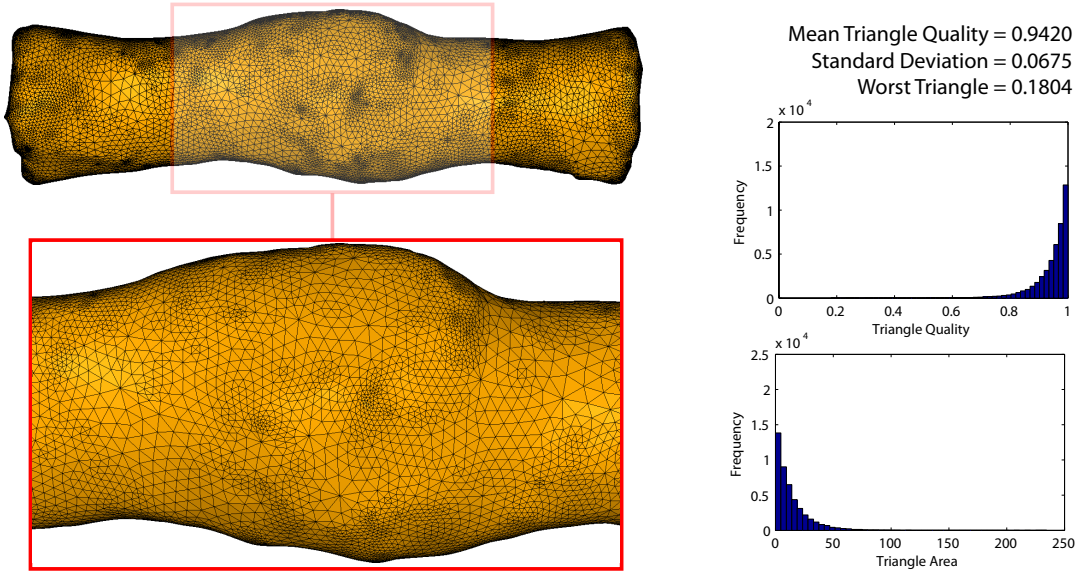
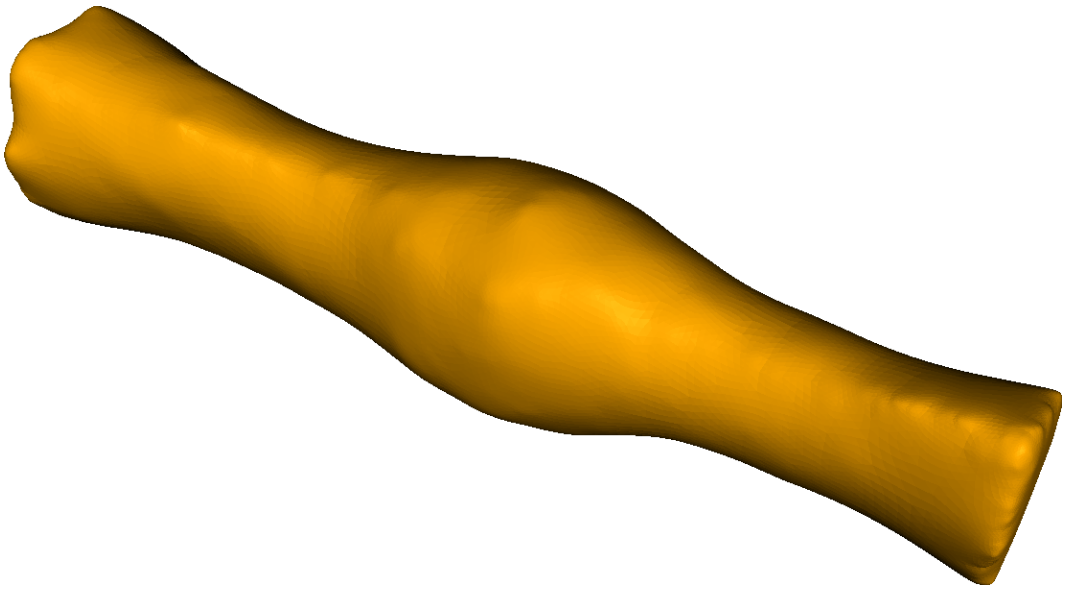
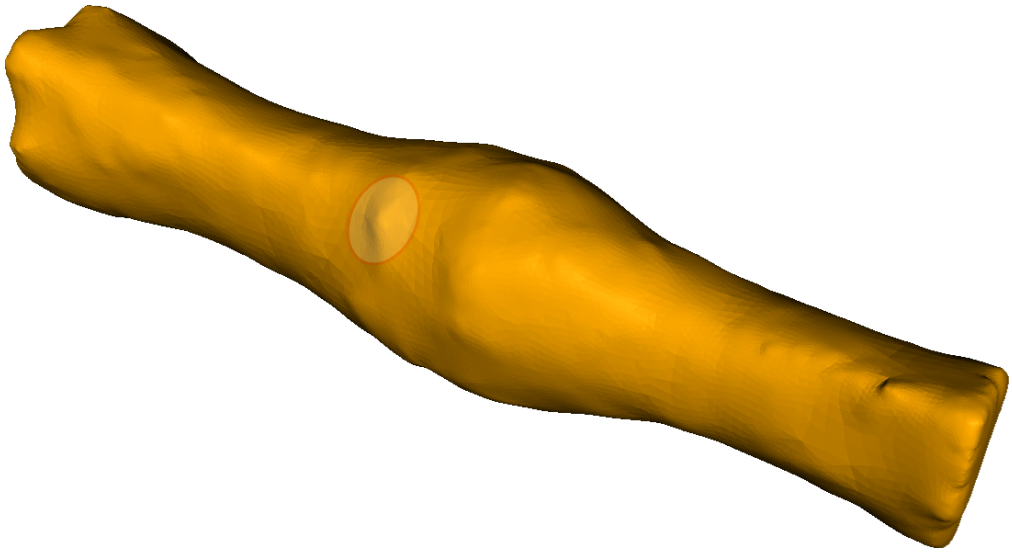


Figure 5.4: Smoothed axon with topology optimization and learned image boundaries. The learned image boundaries constraint reduces overall mesh triangle quality slightly in order to enforce increased feature preservation.

We perform a slice by slice comparison of our correlation metric between the original axon mesh, the remeshed axon using Laplacian Mesh Optimization with only topology optimization, and the remeshed axon using LMO with topology optimization and our learned image boundaries constraint. Figure 5.7 shows that the remeshed axon using learned image boundaries has higher overall correlation to the edge features than without. While the original mesh has the highest correlation with edge features, it achieves this by having a greater surface area which is largely noisy. We show that

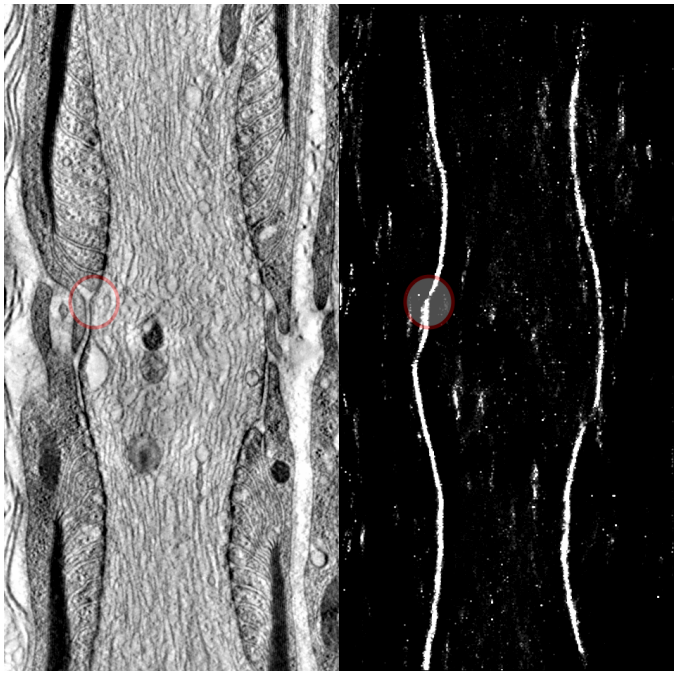


(a) Remeshed without learned image boundaries

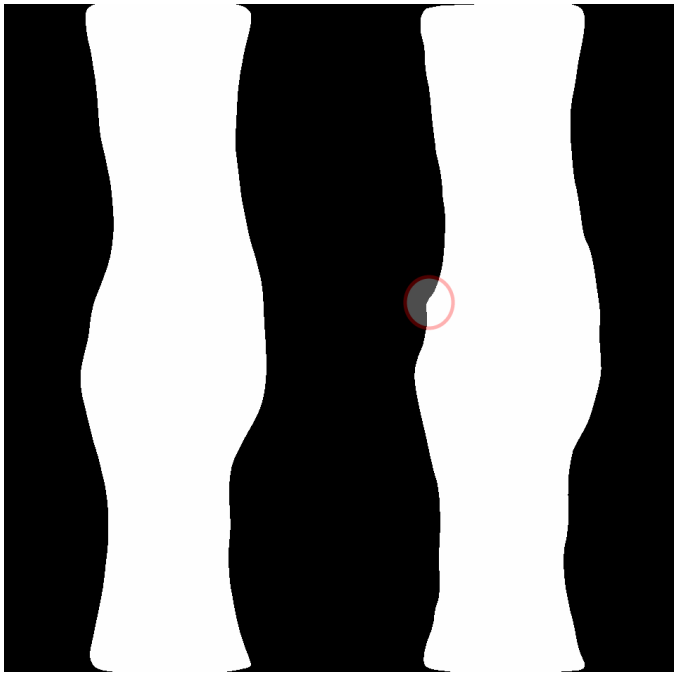


(b) Remeshed with learned image boundaries

Figure 5.5: Remeshed axon showing increased surface detail preservation when using learned image boundaries. Highlighted feature shown in Figure 5.6.



(a) Original image (b) Learned image boundaries



(c) Remeshed without LIB (d) Remeshed with LIB

Figure 5.6: The highlighted feature preserved by our remeshing method is verified to be a genuine feature by inspecting the original volume at slice 105.

we can reduce the noise and artifacts, improve the triangle quality, as well as preserve the most salient features better than the control method. Note that our Node of Ranvier image was acquired by imaging one half of the entire structure, then mirroring the volume to complete an entire image. As a result, the increased noise and distortion along the central slices in the volume (slices 110-130) produce an interesting effect in our plot. Even along the central slice areas, however, we see some benefit to using our method.

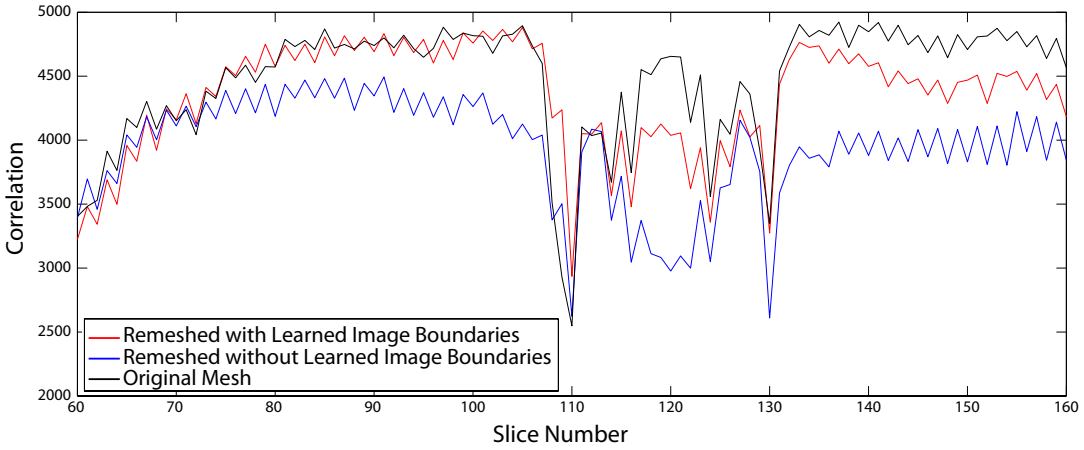


Figure 5.7: Correlation of axon meshes with learned image boundaries across all 2D slices. Note that slices 110-130 in the image volume are more blurry and distorted than other slices due to the imaging of this specific sample.

Figure 5.8 is our reference simulation quality mesh, a uniformly sampled remeshed version of the axon created by Sosinsky et al. for their studies of the Node of Ranvier in 2005 [13]. Using approximately the same number of vertices (22,120), we apply our area based remeshing algorithm to produce a comparable mesh, Figure 5.9. From a mesh quality standpoint, our result has removed the stair stepping artifacts, and exhibits preferable triangle quality according to the radius ratio metric. Note that the reference mesh is clipped at the left and right ends (not visible), while our result is a completely

closed surface. Figure 5.8 therefore appears more densely sampled than our remeshed result due to the fact that the same number of vertices are distributed over a smaller surface area.

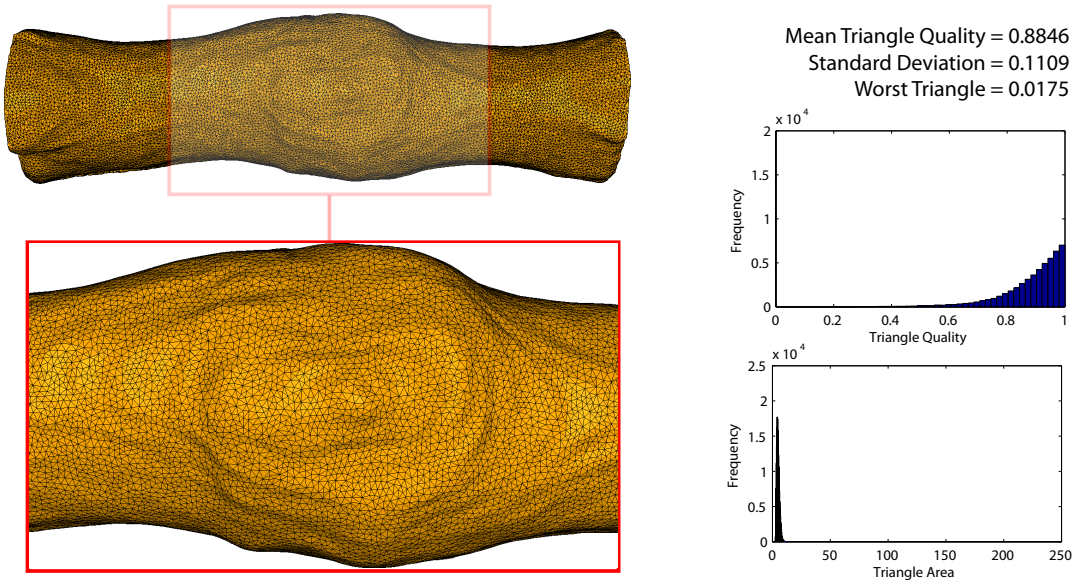


Figure 5.8: Reference axon simulation quality mesh [13].

Our triangle area equalization result in Figure 5.9 is shown here after only two iterations. Continued application of the equalization algorithm would increase the level of triangle area uniformity until the desired results are achieved.

Figures 5.10, 5.11, and 5.12 show the replication of our learned image boundary comparison results on a shaft from the spiny dendrite data set. Figures 5.13 and 5.14 visually demonstrate our remeshing results on two spines from the spiny dendrite data set.

5.1 Computational Cost

The sparse matrix routines used for the mesh smoothing and triangle shape optimization make up the majority of the remeshing algorithm's run time. On our P4

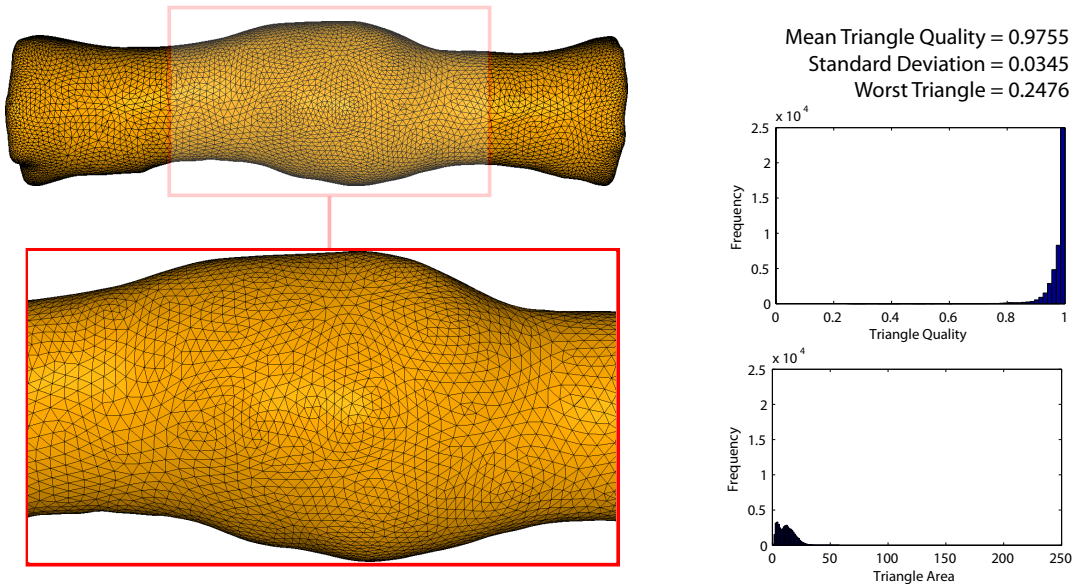
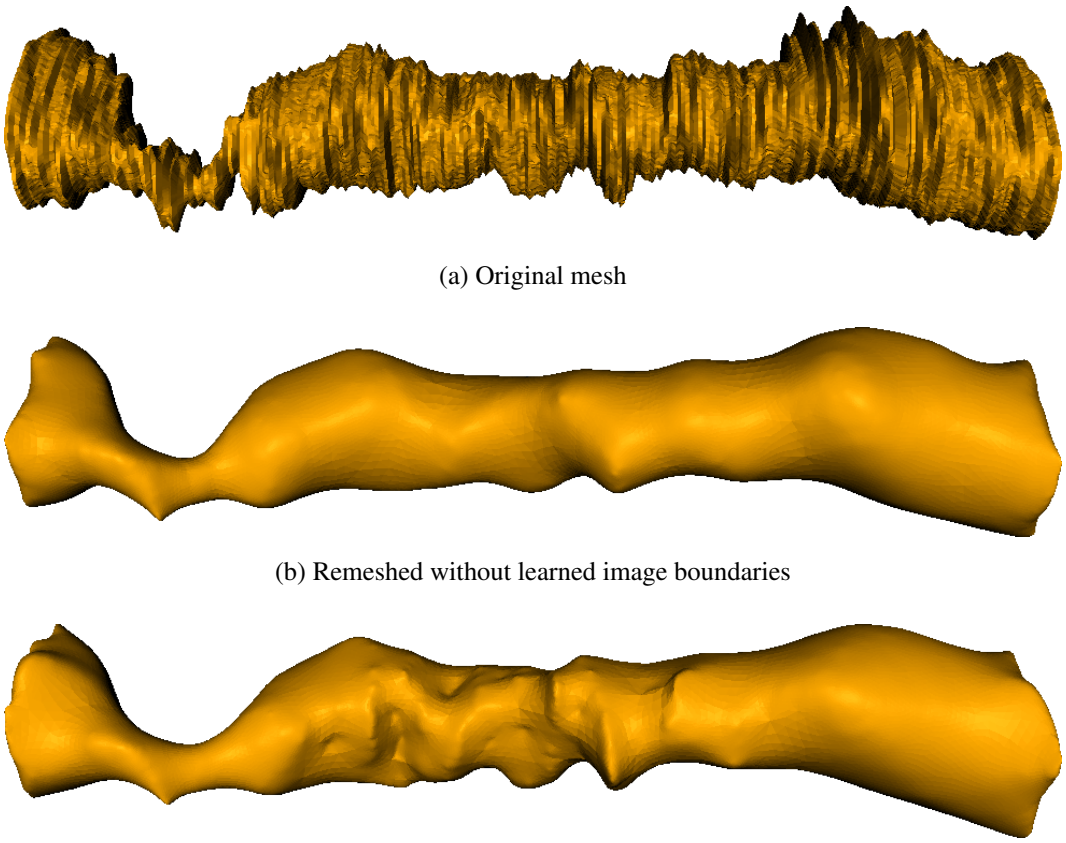


Figure 5.9: Area equalized remeshed axon.

3.4GHz test machine we can factorize the system matrix in approximately 20 seconds for a mesh with approximately 50,000 triangles.

The batch offline edge learning algorithm used for generation of the probabilistic edge map takes several hours to train on one data set, and several minutes to convert into a gradient vector flow volume. This entire process however, can be completely automated.

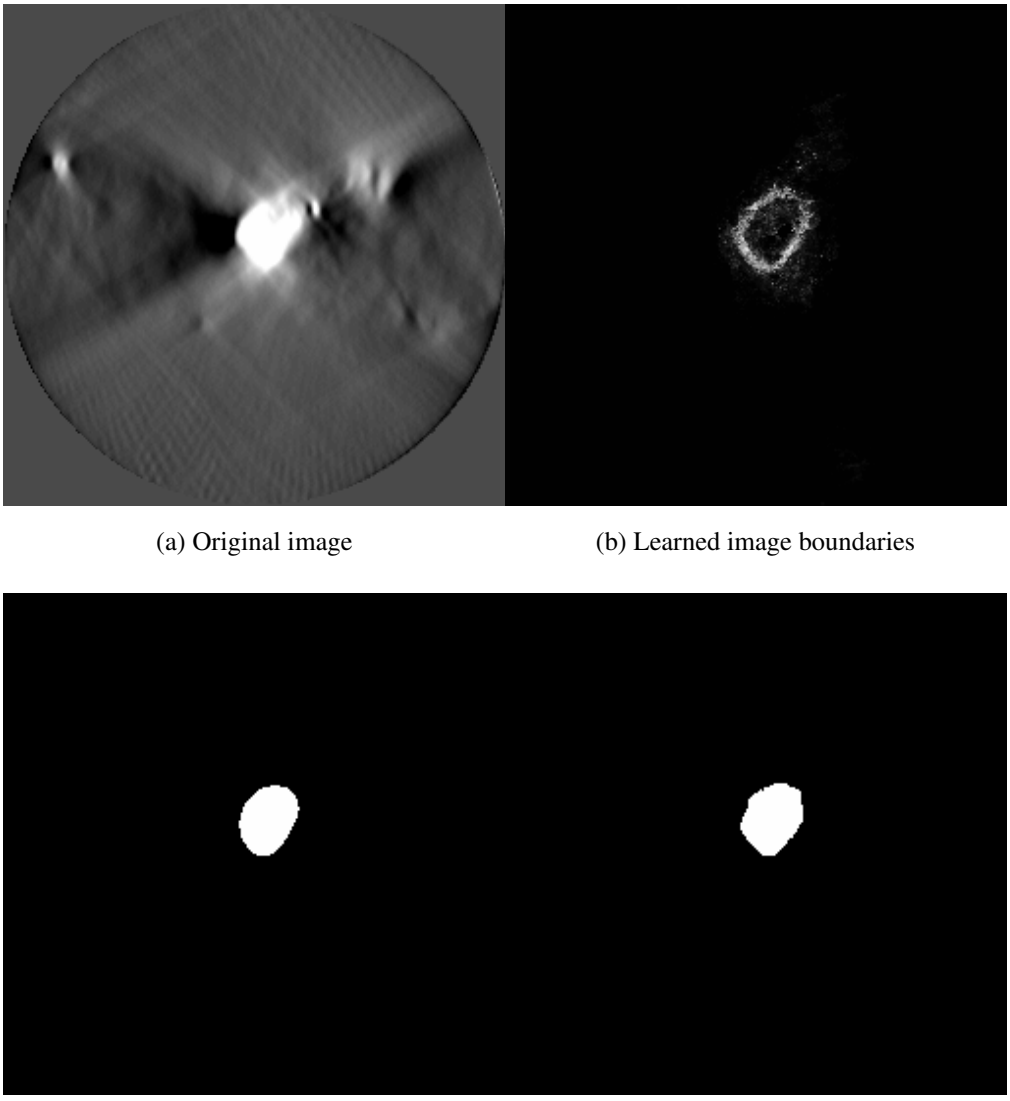


(a) Original mesh

(b) Remeshed without learned image boundaries

(c) Remeshed with learned image boundaries

Figure 5.10: Remeshed dendritic shaft showing increased surface detail preservation when using learned image boundaries.



(a) Original image

(b) Learned image boundaries

(c) Remeshed without LIB

(d) Remeshed with LIB

Figure 5.11: Comparison of dendritic shaft mesh cross sections (c) and (d) with image data (a) and learned edge map (b) on slice 109 of the spiny dendrite data set.

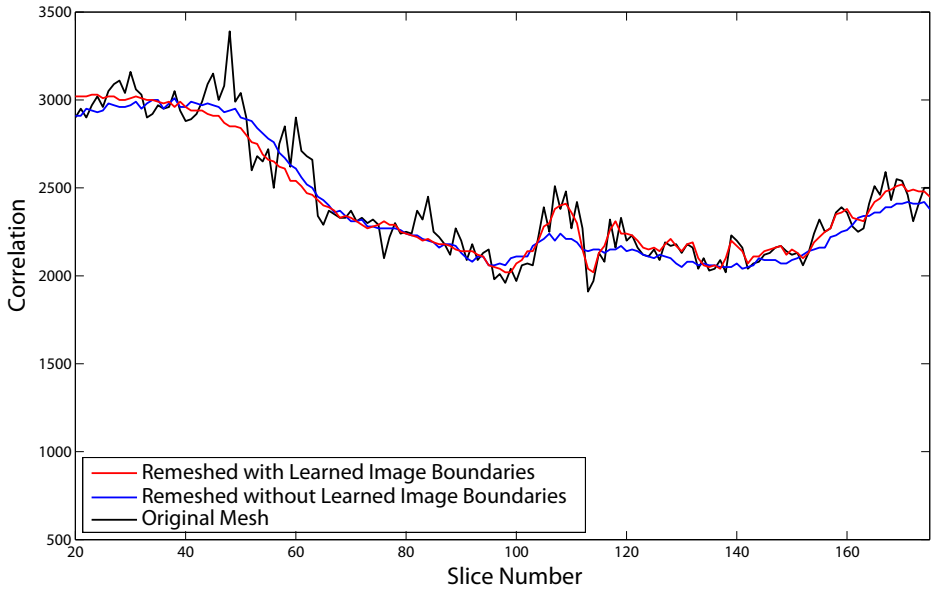


Figure 5.12: Correlation of dendritic spine meshes with learned image boundaries across all 2D slices. The correlation of the remeshed dendritic shaft mesh with the learned image boundaries (red) is shown to be equal to or higher than without learned image boundaries (blue) across most slices.

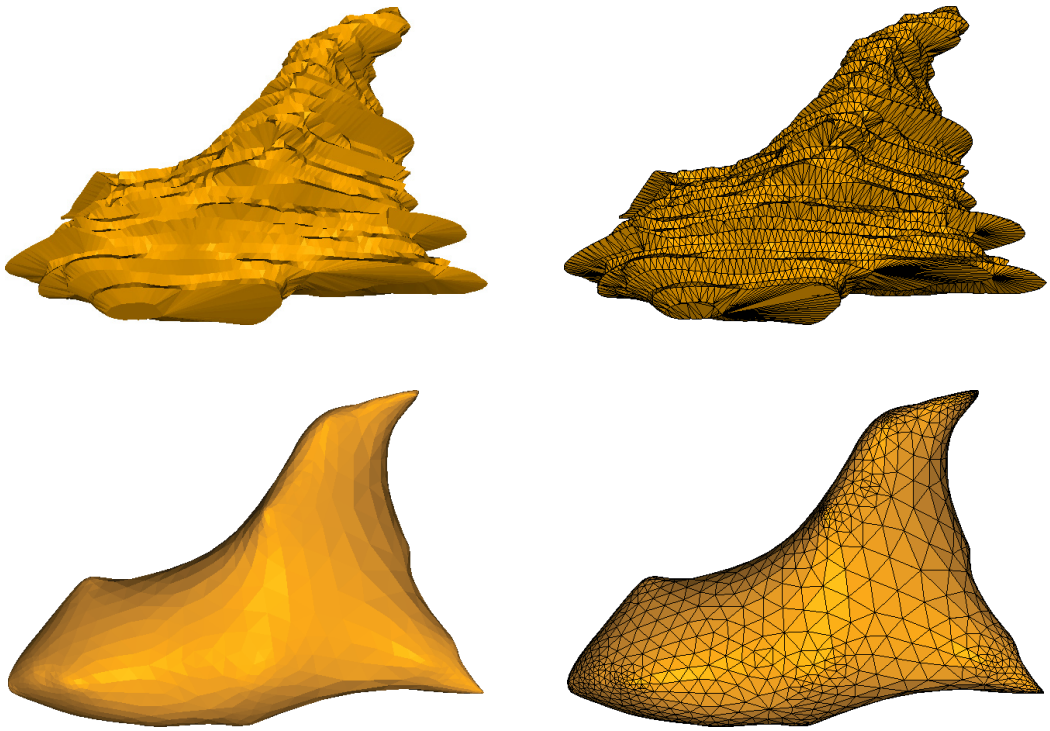


Figure 5.13: Set one of original (top) and remeshed dendritic spine meshes (bottom).

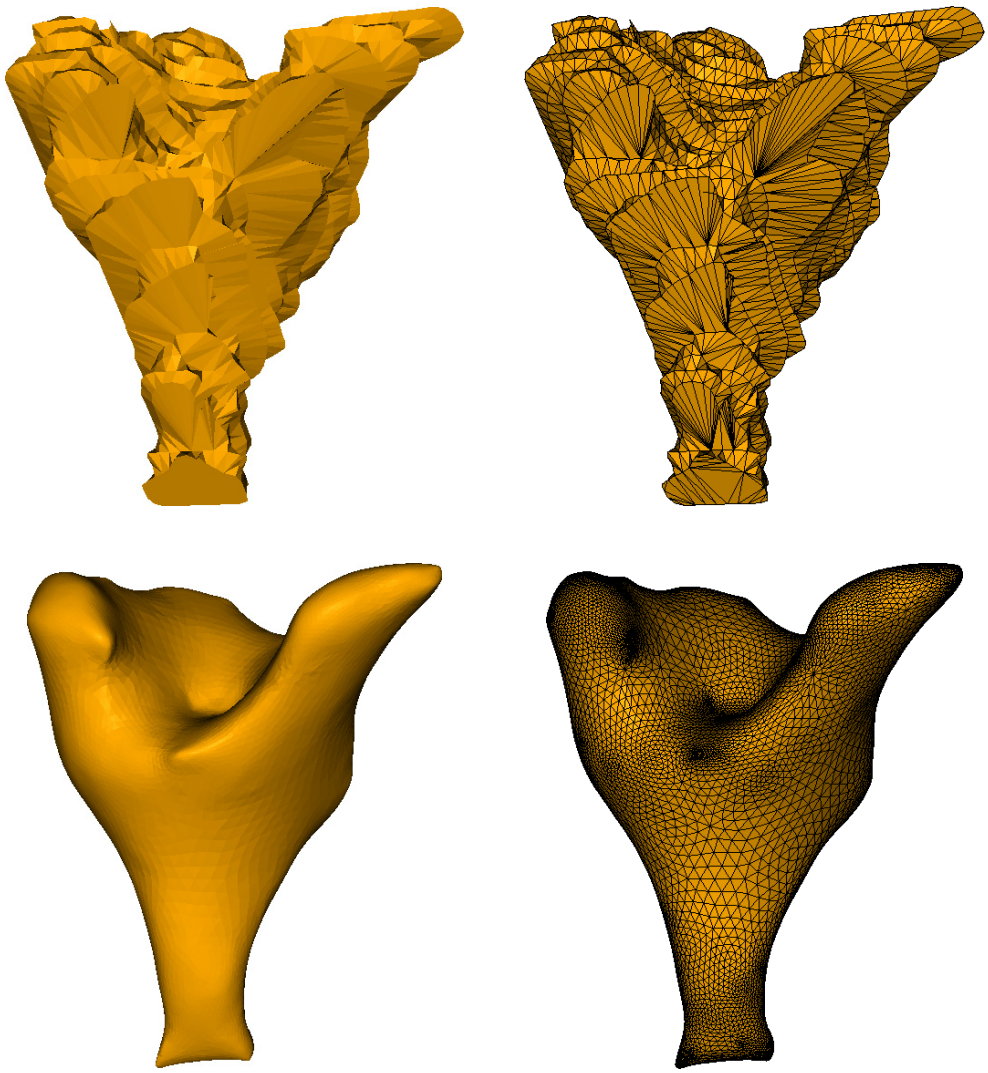


Figure 5.14: Set two of original (top) and remeshed dendritic spine meshes (bottom).

Chapter 6.

Conclusion and Future Work

We have demonstrated a flexible remeshing algorithm which combines global vertex relocation with local topology optimization and area based remeshing to produce either adaptive or uniformly sampled simulation quality meshes from highly irregular input. In addition, we integrated an edge learning framework to perform feature detection and enhance the feature preservation capability of the remeshing algorithm. We conclude that remeshing with learned image boundaries becomes increasingly useful when high amounts of smoothing must be performed on a mesh, assuming that the edge detection algorithm can learn some salient edge features from the training data.

Improving feature detection remains an open avenue for future work. Good feature detection allows for the smoothing of noise without removal of desired image features. As the state of the art in learning based feature detection improves, better algorithms can be substituted into our remeshing approach to increase the quality of our results. While in our approach we use an offline batch learning process, we foresee a movement towards online learning approaches which can train on-the-fly, learning the salient image features at the time of manual segmentation.

Appendix A

LIBRemesh Manual Page

A1 NAME

LIBRemesh – Learned Image Boundaries Remeshing tool

A2 SYNOPSIS

LIBRemesh [option | filename]...

A3 DESCRIPTION

Optimizes the triangle quality of an input mesh by performing smoothing, triangle shape optimization, area equalization, and/or topology optimization. Optionally constrains the remeshing procedure using a gradient vector flow field derived from a 3D probabilistic edge map. Performs mesh visualization and mesh statistics reporting.

A4 OPTIONS

```
-a Area equalization enabled
-b Disable curvature based weighting
-d [0.0-1.0] Simplify mesh by percentage (default: 0.0)
-g [filename] Specify gradient vector flow volume
-m [filename] Specify input mesh
-n Disable smoothing and triangle shape optimization
-p Print triangle areas
-q Print triangle qualities using radius ratio metric
-s [1.0-10.0] Smoothing weight (default: 2.0)
-t [1.0-10.0] Triangle modulation weight (default: 1.0)
-v Visualize output mesh
-w [0.0-10.0] Gradient vector flow weight (default: 1.0)
-x Subdivide mesh
```

A5 EXAMPLES

A5.1 One iteration of smoothing and shape optimization using default parameters

```
LIBRemesh -m mesh.wrl
```

A5.2 One iteration of smoothing and shape optimization using default parameters and using learned image boundaries

```
LIBRemesh -m mesh.wrl -g gvf.img
```

A5.3 One iteration of smoothing and shape optimization with 20 percent decimation, using learned image boundaries, moderate smoothing and increased weight for gradient vector flow

```
LIBRemesh -m mesh.wrl -d 0.2 -g gvf.img -s 4.0 -w 1.25
```

A5.4 One iteration of smoothing and shape optimization with 20 percent decimation, followed by area equalization, a second iteration of smoothing (with reduced weight), and a second iteration of area equalization

```
LIBRemesh -m mesh.wrl -d 0.2 -s 2.0
LIBRemesh -a -m output.wrl
LIBRemesh -m output.wrl -s 1.0
LIBRemesh -a -m output.wrl
```

References

[1] Pierre Alliez, Giuliana Ucelli, Craig Gotsman, and Marco Attene. Recent advances in remeshing of surfaces. Research report, AIM@SHAPE Network of Excellence, 2005.

[2] Jean-Daniel Boissonnat and Bernhard Geiger. Three-dimensional reconstruction of complex shapes based on the delaunay triangulation, 1992.

[3] Piotr Dollar, Zhuowen Tu, and Serge Belongie. Supervised learning of edges and object boundaries. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1964–1971, Washington, DC, USA, 2006. IEEE Computer Society.

[4] Nira Dyn, David Levine, and John A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Trans. Graph.*, 9(2):160–169, 1990.

[5] Luis Ibanez, Will Schroeder, Lydia Ng, and Josh Cates. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-15-7, http://www.itk.org/ItkSoftwareGuide.pdf, second edition, 2005.

[6] Ashraf Mohamed and Christos Davatzikos. Finite element mesh generation and remeshing from segmented medical images. *Biomedical Imaging: Nano to Macro, 2004. IEEE International Symposium on*, pages 420–423 Vol. 1, 15-18 April 2004.

[7] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh

optimization. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 381–389, New York, NY, USA, 2006. ACM.

[8] Philippe Pébay and Timothy Baker. Analysis of triangle quality measures. *Math. Comput.*, 72(244):1817–1839, 2003.

[9] Robert Schneider and Leif Kobbelt. Geometric fairing of irregular meshes for free-form surface design. *Computer Aided Geometric Design*, 18(4):359–379, 2001.

[10] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit, Third Edition*. Kitware Inc.

[11] Will Schroeder, Jonathan Zarge, and Bill Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, 1992.

[12] Olga Sorkine and Daniel Cohen-Or. Least-squares meshes. In *Proceedings of Shape Modeling International*, pages 191–199. IEEE Computer Society Press, 2004.

[13] Gina Sosinsky, Thomas Deerinck, Rocco Greco, Casey Buitenhuys, Thomas Bartol, and Mark Ellisman. Development of a model for microphysiological simulations. *Neuroinformatics*, 3(2):133–162, 2005.

[14] Vitaly Surazhsky and Craig Gotsman. High quality compatible triangulations. *Engineering with Computers*, 20(2):147–156, April 2004.

[15] Sivan Toledo. Taucs: A library of sparse linear solvers, version 2.2. 2003.

[16] Greg Turk. Re-tiling polygonal surfaces. *Computer Graphics*, 26(2):55–64, 1992.

[17] Chenyang Xu and Jerry L. Prince. Gradient vector flow: A new external force for snakes. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 66, Washington, DC, USA, 1997. IEEE Computer Society.

[18] Denis Zorin, Peter Schröder, and Wim Sweldens. Interpolating subdivision for meshes with arbitrary topology. *Computer Graphics*, 30(Annual Conference Series):189–192, 1996.