Learning Global Features for Neuron Reconstruction in EM Images

Brian Patrick Matejek

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Master of Science in Engineering

Recommended for Acceptance
by the Department of
Computer Science
Adviser: Professor Thomas Funkhouser

May 2016

Abstract

Connectomics, the study of the neural connections, has gained significant attention in the neuroscience community as a method to understand the complex workings of the brain. Recent advancements in neural image acquisition has created a surplus of data too expensive to manually annotate, leading to an influx of automatic reconstruction methods. Most current methods employ hierarchical agglomeration techniques that focus on extracted local features. These techniques fail to utilize the overall global structure of neurons in the brain and other valuable shape descriptors, leading to early errors which compound over time. This paper explores novel shape descriptors and global features to predict merge candidates in electron microscopy images. A new global hierarchical merging algorithm considers the global ramifications of each merge. Testing on the SNEMI3D challenge dataset confirms the methods proposed in this paper outperform the hierarchical clustering baseline.

Acknowledgements

I would like to thank Professor Thomas Funkhouser for advising me the last few years and teaching me many valuable research skills. I have greatly enjoyed working on the Google Street View and Neuron Reconstruction projects and I am very thankful for the opportunity he gave me years ago to join his team.

I would like to thank Professor Sebastian Seung for meeting with me frequently over the past year to help me better understand the current problems facing the Connectome Project.

I would like to thank Jonathan Zung for providing me with the complete SNEMI3D datasets along with the affinity and watershed images. I appreciate how quickly he responded to questions I had on the data.

Finally, I would like to thank my family for helping me through the years to realize my potential. I cannot imagine being where I am without their consistent support and love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Connectomics, the study of the neural connections, has gained significant attention in the neuroscience community as a method to understand the complex workings of the brain. Understanding the structure of the brain may help illuminate the causes and potential treatments of serious mental illnesses [21]. Several diseases, such as schizophrenia, arise from synaptic perturbations. Neuroscientists hope that understanding the structure of the brain will advance drug research and development for these diseases [24]. The earliest study in connectomics focused on the brain of the *Caenorhabditis elegans* roundworm. Over the course of 10 years, White et al. completed the connectome with 302 neurons and approximately $7,000$ chemical synapses [30]. In 2013, Takemura et al. reconstructed the optical lobe of the *Drosophilia* fly. Machine learning algorithms partitioned the stacks of electron microscopy (EM) images into superpixels, generating an oversegmentation of the images. After $\sim$14, 400 manual hours of agglomerating superpixels, researchers had a densely reconstructed volume with 379 neurons and $8,637$ chemical synaptic contacts [28].

Recent advancements in large-volume staining and electron microscopy image acquisition has created a surplus of neuroimage datasets [13] [8] [11]. Using 61 parallel electron beams, Harvard researchers are approaching a collection speed of nearly one gigabyte per second. These recent advancements have opened the possibilities for neuroscientists to image more complex mammalian brains. The manual reconstruction process of the fly and worm brains provided invaluable information on neuron connections. However, mammalian brains are much larger and require a more automated approach. Mice typically have around $\sim$70 million neurons while an average adult human has around $\sim$86 billion neurons [12] [1]. With brains of this magnitude, manual reconstruction efforts are infeasible.

Dendrite neurons can span several millimeters in mouse brains and yet have diameters on the nanometer scale. This produces massive datasets that require several thousands of hours of manual labor to segment even modestly-sized brain sections. Helmstaedter, Briggman et al. acquired EM images of a mouse retina covering an area of 132 x 114 x 80 μm at a resolution of 16 nm x 16 nm x 25 nm. Students spent $20,000$ hours manually generating skeletons for the neurons. The data helped the researchers further understand the variation of neuron cells beyond the simple morphology. However, a full mouse brain is $\sim$400,000 times larger than this already substantial reconstructed volume. Automated approaches are therefore needed to speed up the reconstruction process.

Neuroscientists are optimistic about the advancements possible with more labeled neuron image data. However, the accuracy needed far exceeds the current state-of-the-art algorithms and manual segmentation is far too costly. Current hierarchical clustering algorithms make early local decisions without context from the full image volume. Early errors can therefore compound over the course of the algorithm producing suboptimal results that are not nearly accurate enough. This thesis presents a system that uses both local and global features to learn a merge classifier between oversegmented supervoxels. Various global clustering algorithms then use the learned classifier to produce a segmentation of the electron microscopy images that is more robust and therefore less susceptible to local errors.

Chapter 2

Related Work

A substantial amount of research has focused on predicting similarity between neighboring voxels in electron microscopy images. Some early membrane detectors used graph-cuts to produce a probability that a voxel belonged to a neuron membrane [5]. Further advancements added additional cost terms to leverage the expected geometry of neurons [17]. More recent approaches have focused on training deep neural networks to classify pixels [6]. Most EM datasets are anisotropic, with an xy resolution approximately 5 to 10 times greater than the z resolution. This disparity has caused many researchers to predict pixel similarity in a given image slice. The image slices are linked afterwards to create a topologically consistent boundary across the image stack [18]. However, Lee et al. constructed a recursive 2D-3D convolutional neural network for boundary detection in three dimensions [22].

The affinities generated by the above algorithms produce probabilities that neighboring voxels belong to the same neuron. Many current methods rely on a watershed algorithm [29] to produce an oversegmentation of the images [23]. Frequently, a hierarchical clustering algorithm agglomerates these supervoxels to create a labeled segmentation. GALA, an open-source Python library created by Juan Nunez-Iglesias et al., trains a multi-level merge priority function [23]. Another system, named LASH, learns a similarity function by reformulating the problem as a Markov decision process where a state's reward depends on the decrease in Rand error [14].

Many automatic reconstruction methods focus on segmenting 2D EM images to account for the anisitropic nature of the data [15] [17]. These 2D predictions are stitched together afterwards in a topologically consistent manner. Knowles et al. created a dynamic program to find the shortest path through the image stack [19]. Funke et al. generated an integer linear program to minimize the cost of labeling an image stack [10]. Kaynig et al. used a random forest classifier to produce multiple segmentations of each EM image and used segmentation fusion to group the segmentations into geometrically consistent 3D objects [18].

The current state-of-the-art methods do not produce segmentations at a high enough accuracy. Recently, crowd-source platforms such as EyeWire and Mojo have introduced semi-automatic pipelines for volumetric reconstruction [25] [18]. A trained affinity function creates probabilities that neighboring voxels belong to the same neuron. A watershed algorithm with a conservative high threshold agglomerates voxels to produce an oversegmentation of the image. Users further connect the agglomerated supervoxels to produce a complete neural reconstruction. Although these interfaces provide an easy and interactive way for researchers to acquire fully segmented images, they are still too slow for large-scale reconstruction efforts.

2.1 SNEMI3D Challenge Dataset

There are many different datasets containing electron microscopy neural images with accompanying human labels. This paper focuses on the SNEMI3D challenge dataset. The Lichtman Lab at Harvard University produced the raw image stacks. [16]. The dataset comes with both the original EM images and hand labeled ground truth. An actively maintained online leader board displays the results of several different

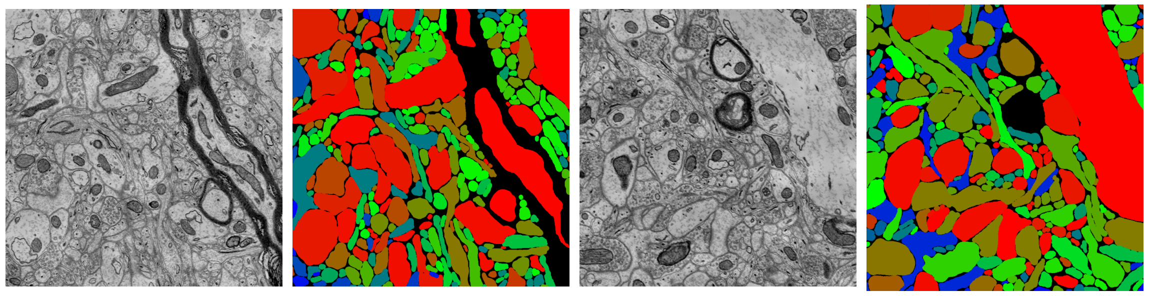


Figure 2.1: A sample image slice of the training dataset with the colorized ground truth (left) and the testing dataset with the colorized ground truth (right).

segmentation methods. Both datasets provide a densely reconstructed volume of size 1024 x 1024 x 100 at a resolution of 6 nm x 6 nm x 30 nm. To reduce running times, both volumes were split into four equal sizes of 512 x 512 x 100. Figure 2.1 shows an example image slice from both the training and testing datasets, as well as the colorized human labels.

2.2 Evaluation Metrics

There are several different evaluation metrics used to measure the success of a segmentation against ground truth. The SNEMI3D challenge uses the Rand F-score to evaluate submissions. Consider all $\binom{n}{2}$ pairs of voxels in the entire image stack. Call the ground truth labeling T and the segmented prediction P. A voxel pair that maps to the same segment in both P and T is a true positive (TP). A pair that maps to the same segment in P but not in T is a false positive (FP). A pair that maps to the same segment in T but not in P is a false negative (FN). A pair that maps to different segments in both P and T is a true negative (TN). Precision and recall are respectively defined as:

$$precision = \frac{TP}{TP + FP}$$
$$recall = \frac{TP}{TP + FN}$$

and the Rand F-score is:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Note, for the SNEMI3D challenge dataset, extracellular voxels that are not labeled as neurons in the ground truth do not count towards the number of voxels. Labeling a voxel as a given neuron when it has no label does not disadvantage the submission.

Chapter 3

Algorithmic Overview

This paper presents a multi-step system for segmenting EM images. Current state-of-the-art segmentation algorithms focus on local hierarchical agglomeration techniques that do not leverage the overall repetitive global structure present in neural images. This system augments the local features of these algorithms with additional shape descriptors and global features.

Figure 3.1 presents a graphical overview of the system. The EM images are first preprocessed to generate affinities between neighboring voxels. A watershed algorithm creates a conservative supervoxel oversegmentation. Learning merge predictions is one of the main challenges in hierarchical clustering. To this end, this paper explores various boundary features and shape descriptors. A random forest classifier learns on these attributes to produce a mapping from supervoxels to predictions. A global merging algorithm presented in section 4.3 transforms the results from the classifier into a 3D segmentation of the EM image. This algorithm improves on hierarchical clustering by considering the global implications of individual merges.

The output of the global merging algorithm produces a reasonable segmentation of the images. However, the algorithm does not exploit certain global invariants of the neurons in the image, namely every neuron must leave the volume. Therefore, a graph

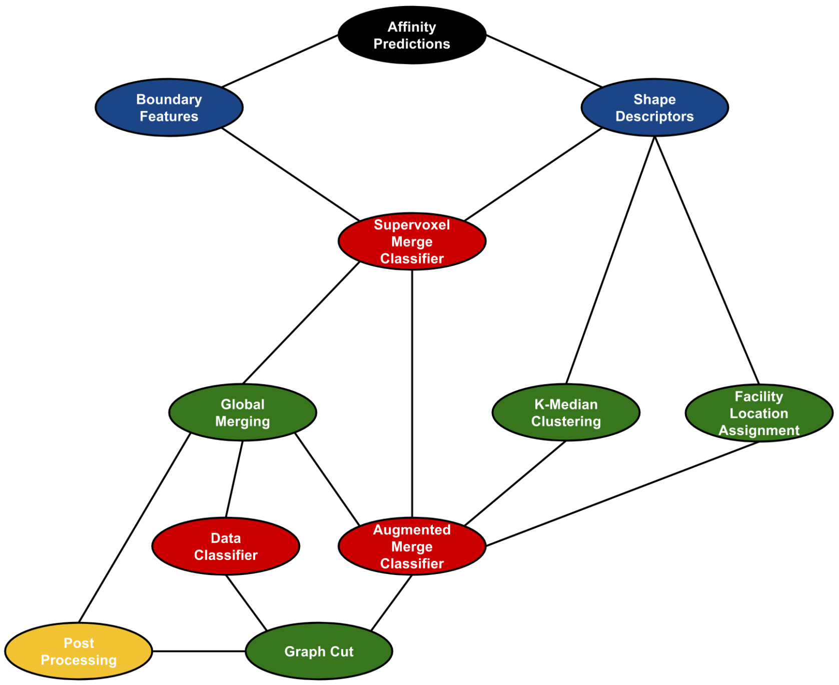


Figure 3.1: An overview of the neuron segmentation algorithm. Blue circles represent local features, red circles represent machine learning classifiers, and the green circles represent global approximation algorithms.

cut algorithm assigns each interior supervoxel a label corresponding to an exterior supervoxel. Graph cuts require data terms and smoothing terms. In this formulation the data terms correspond to the penalties for assigning an interior supervoxel the label of a given exterior supervoxel. The graph cut energy function enforces smoothing penalties when neighboring supervoxels have different labels. The global merging algorithm is input into the graph cut algorithm to provide data terms. The smoothing terms follow the output of an augmented merge classifier which includes additional global features. A post processing step merges the output of the graph cut and global merging algorithms to produce a final predicted segmentation.

Chapter 4

Implementation

This chapter provides a detailed description of the algorithm described in Chapter 3. The chapter first presents the preprocessing steps which generate more useful information from raw electron microscopy images. Preprocessing returns a set of “supervoxels” that represent voxel regions with high likelihood of belonging to the same neuron. The rest of the chapter focuses on clustering these supervoxels to produce a correct segmentation of the whole EM image stack. Section 4.2 considers the task of predicting which neighboring supervoxels belong to the same neuron and the training of a machine learning classifier based on boundary features and shape descriptors. Section 4.3 describes a function that maps the output of the machine learning algorithm to a fully segmented image stack. Although this function produces promising results, sections 4.4 and 4.5 analyze additional methods to further improve the results.

4.1 Image Preprocessing

Image segmentation algorithms need similarity measures between neighboring voxels to produce coherent segmentations. Many researchers have focused on optimizing these affinity predictions using an array of machine learning algorithms ranging from random forest classifiers [18] to convolutional neural networks [6] [31] [22]. Developing additional algorithms for boundary predictions is an open research area. However, this paper will focus on segmentation after receiving voxel affinities and does not attempt to produce a novel affinity function.

4.1.1 Affinity Generation

Kisuk et al. created MattNet, a convolutional neural network, to predict affinities between voxels in EM images. The network has an architecture of 5 hidden layers with convolution filters of size 5 x 5 x 3. Thus, the field of view is 25 x 25 x 13 voxels for each prediction. MattNet is a recursive 2D-3D neural network that takes as input the original EM image and the IDSIA boundary map [6]. The neural network was trained and validated on the SNEMI3D training dataset to produce the affinities used in this paper. There are three affinities generated for each voxel: one for each non-diagonal negative direction. These affinities are combined to generate three affinity "images" or "graphs" where the value at (x, y, z) represents the affinity between voxel (x, y, z) and $(x-1, y, z)$, $(x, y-1, z)$, or $(x, y, z-1)$ respectively. Figure 4.1 shows a sample image slice and the corresponding x-affinity image output by the neural network. This paper will refer to these affinity images as images or graphs interchangeably. A graph is constructed by making every voxel a node with edges between non-diagonal voxel neighbors. The edge weight equals the affinity output by the convolutional neural network.

4.1.2 Watershed Segmentation

Researchers frequently use naive clustering algorithms to produce a conservative oversegmentation of the affinity graph. The segments output from the algorithm are called supervoxels because they represent a set of individual voxels. The clusters create an oversegmentation of the image meaning that voxels in the same supervoxel belong

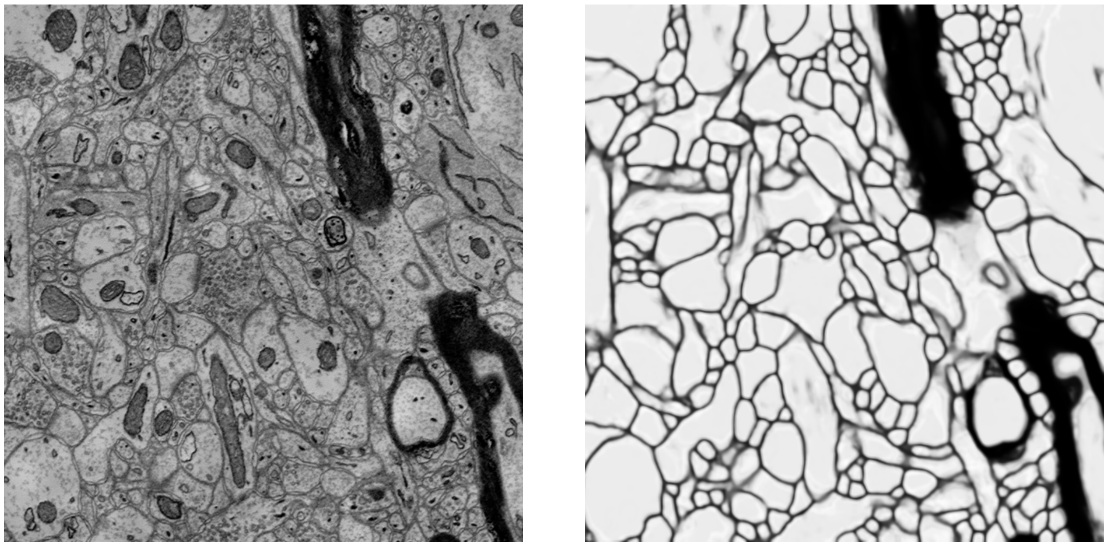


Figure 4.1: Comparison of an electron microscopy image (left) and the corresponding x-affinity image (right) produced by MattNet, a convolutional neural network.

to the same neuron but voxels in different supervoxels do not necessarily belong to different neurons. This relaxes some of the computational challenges of working with such large image datasets. It is computationally cheaper to cluster $\sim 10^4$ supervoxels than to cluster $\sim 10^7$ voxels so using supervoxels provides more opportunity for sophisticated agglomeration algorithms.

The watershed algorithm is one such clustering algorithm which can produce a conservative oversegmentation of the affinity graphs [29]. The watershed algorithm takes three parameters: a high threshold, a low threshold, and a minimum supervoxel size. Consider the graph formulation of the affinity image presented in section 4.1.1. All edges are sorted in descending weight order so voxel pairs with high affinity are considered first. Any edge with a weight greater than the high threshold is collapsed and the voxels merge into the same supervoxel. This edge collapsing continues to create larger supervoxels until there are no more edges greater than the high threshold. Voxels with edges less than the low threshold are labeled extracellular (i.e. not neurons). Supervoxels smaller than the minimum supervoxel size are merged to their best neighbor. All affinities between the high and low threshold now contribute to a boundary between two supervoxels. This system uses a high threshold of 0.915, a low threshold of 0.25, and a size threshold of 250.

Table 4.1 displays the number of correctly predicted neuron and extracellular voxels by the watershed algorithm. As discussed in section 2.2, the commonly used evaluation metrics do not penalize the assignment of extracellular voxels to neurons. Thus, the large number of cellular predictions that are in fact extracellular will not affect the success of further clustering techniques presented in this thesis. Voxels clustered with other voxels from a different neuron produce another source of error for the watershed algorithm. Since the system presented here agglomerates supervoxels, these voxels will not be labeled correctly. 3.0% of voxels in the training dataset and 2.5% of voxels in the testing dataset meet this error criteria.

The watershed algorithm produces supervoxels with boundaries between neighboring supervoxels. Every pair of neighboring supervoxels either does or does not belong to the same neuron. For the purposes of this paper, a boundary between supervoxels in the same neuron is called an oversegmented boundary, and a boundary between supervoxels in different neurons is labeled a true boundary. Any such boundary is defined by the set of affinities between voxels in the different supervoxels. Taking the mean of the boundary affinities produces a simple value representing the overall affinity between two supervoxels. Sorting these boundaries in descending order by the mean produces a ranking of boundaries by decreasing likelihood of being an oversegmented boundary. The simplest hierarchical agglomeration algorithm considers boundaries in order and continually merges clusters across the boundary. Rather than define criteria for the cessation of merging, one can consider the complete sequence of all merges to find the optimal result. Figure 4.2 presents the best Rand F-score over the course of the entire hierarchical clustering algorithm for the training and testing sets. This will serve as a benchmark against the algorithms described later in this section.

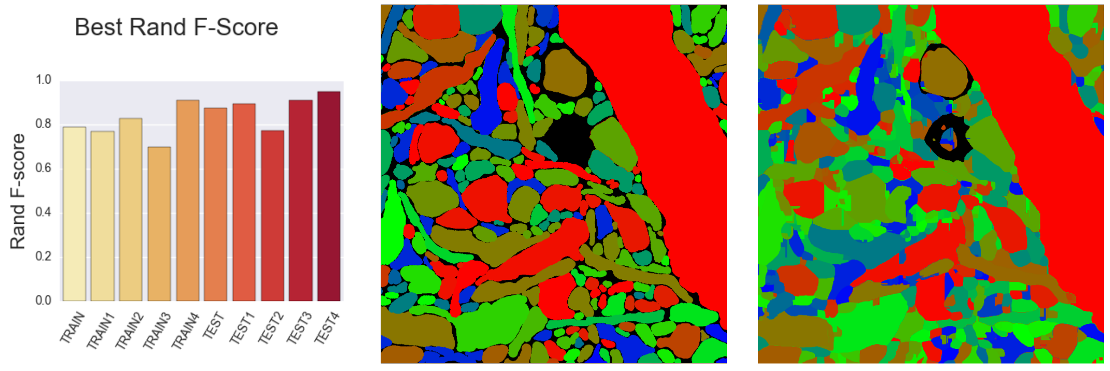


Figure 4.2: The best Rand F-score for the training and testing datasets for the baseline hierarchical clustering algorithm (left). Also, the human labeling for one slice of the testing dataset (center) adjacent to the optimal segmentation prediction (right) of the described hierarchical clustering algorithm.

	Predicted Class	
Actual Class	Cellular	Extracellular
Cellular	89,009,158	33,398
Extracellular	12,316,481	3,498,563

	Predicted Class	
Actual Class	Cellular	Extracellular
Cellular	93,160,164	24,216
Extracellular	9,556,150	2,117,070

Table 4.1: Results of the watershed algorithm run on the above convolutional neural network for the training dataset (left) and the testing dataset (right).

4.2 Predicting Merges

Section 4.1.2 showed the results of a very simple hierarchical clustering algorithm on the training and testing datasets. That algorithm only used the mean of the affinities along the boundary between two neighboring supervoxels. Most sophisticated hierarchical clustering algorithms do not rely on a single attribute but accumulate many features to define a given boundary. Machine learning algorithms take these features along with the label for each boundary (oversegmented or true) to learn a function that maps features to labels. Nunez-Iglesias et al. trained a random forest classifier to predict which neighboring supervoxels to merge [23]. For the duration of the paper their system will be referred to as GALA. Random forest classifiers are frequently used for machine learning problems since predicting labels for new data points is linear in the number of trees (if the tree depth is bounded by a constant). For this reason, this system trains a random forest classifier to predict which boundaries are oversegmented.

When analyzing features for input into a machine learning algorithm, it is important to evaluate the usefulness of individual features. There are several available evaluation metrics for this task, but most tree-based learning algorithms use information gain [26]. Define the entropy (S) of a set of data points with classes $\in C$ by:

$$S = \sum_{c \in C} -p_c \log_2 p_c \tag{4.1}$$

A feature will split the "parent" data into two "children" sets. A good feature will create children datasets that have highly homogeneous labels. The information gain is defined for a feature as:

$$G = S(parent) - \text{avg}_w \left[S(children) \right] \tag{4.2}$$

A dataset with only one label will have zero entropy.

Most current clustering algorithms extract local features for training the machine learning classifier. These algorithms, including GALA, only rely on geometric features and affinity statistics along the boundary and do not consider overall global structure. The next two subsections consider various features on which to learn, and the final subsection analyzes the results of this implementation's machine learning classifier.

4.2.1 Boundary Features

As described in section 4.1.2, the watershed algorithm produces a set of supervoxels and boundaries between those supervoxels. These boundaries contain a set of affinities between voxels in the different supervoxels. These affinities provide an array of features that can help classify boundaries as oversegmented or true. Define $\mathcal{A}$ as the set of affinities between two neighboring supervoxels. There are several features about the distribution $\mathcal{A}$. The naive hierarchical clustering algorithm serving as the baseline considered only the mean of $\mathcal{A}$. However, GALA extracted additional features from this distribution including the maximum, median, and minimum values, along with the standard deviation (second moment), skew (third moment) and kurtosis (fourth moment). That system also computed the fraction of affinities below the thresholds of 0.1, 0.5, and 0.9.

Consider the following unweighted graph formulation: every supervoxel is a node and every pair of neighboring supervoxels are connected by an edge. Extracting features about this graph can produce additional information about the local region around two neighboring supervoxels. To this end, GALA extracted two more features corresponding to the degree difference of the nodes in the graph and the number of mutual neighbors between two nodes in the graph.

The system presented here extracted all of these same features as GALA. However, the thresholds of 0.1, 0.5, and 0.9 were altered to 0.4, 0.6, and 0.8 to better reflect the affinity graph produced by MattNet. Table 4.2 shows the result of training a random forest classifier with 1000 trees, a max depth of 7 splits, and $\sqrt{n}$ attributes per split (where n is the total number of features). The table also shows the information gain for each of these features. A boundary is a true positive if the classifier correctly predicted merge, a false positive if the classifier incorrectly predicted merge, a false negative if the classifier incorrectly predicted segmented, and a true negative if the classifier correctly predicted segmented. The most useful boundary features based on information gain are boundary maximum, boundary median, boundary mean, and proportion of affinities less than 0.60. Note, GALA also extracts a 25 bin histogram

Dataset	True Positives	False Positives	False Negatives	True Negatives	Proportion Correct
Training 1	26629	849	951	15784	95.929
Training 2	22370	1053	1118	17355	94.818
Training 3	30799	1197	2437	15709	92.753
Training 4	38858	1406	1802	14659	94.345
Testing 1	43425	1361	1544	14747	95.244
Testing 2	42002	1732	1816	18972	94.501
Testing 3	40645	1201	875	15110	96.410
Testing 4	46015	1460	2205	12040	94.062

Feature Name	Information Gain	Proportion Correct	Split Value
Boundary Maximum	0.573233	0.924861	0.652071
Boundary Median	0.551384	0.914207	0.489334
Boundary Minimum	0.244607	0.690682	0.484131
Boundary Mean	0.557886	0.918902	0.465879
Boundary Standard Deviation	0.073787	0.529330	0.116110
Boundary Skew	0.151836	0.659481	-0.779639
Boundary Kurtosis	0.104266	0.564671	-0.000175
Affinities Less Than 0.40	0.024620	0.666788	0.004137
Affinities Less Than 0.60	0.575200	0.928457	0.726534
Affinities Less Than 0.80	0.181572	0.604801	0.999873
Degree Difference	0.068339	0.654040	9
Mutual Neighbors	0.006368	0.650988	9
Number of Voxels	0.100011	0.691361	5390
Proportion of Voxels	0.054988	0.640137	0.191060
Boundary Rank	0.145567	0.712104	14
Boundary Scaled Ranking	0.117105	0.708617	0.498934

Table 4.2: The results of the random forest when trained using only boundary features (top). The set of extracted features with the corresponding information gain, proportion correct, and split value (bottom).

of the affinities for its random forest classifier. However, these bin features proved detrimental to the classifier for the affinity graph output by MattNet.

4.2.2 Shape Descriptors

While these boundary features provide a good baseline for predicting merges among supervoxels, additional features can further improve the results by considering the broader context between two supervoxels. Bogovic et al. saw an increase in classification accuracy for 3D agglomeration after adding derivative features [2]. Boundary features only consider the narrow ridge between two neighboring supervoxels. How-

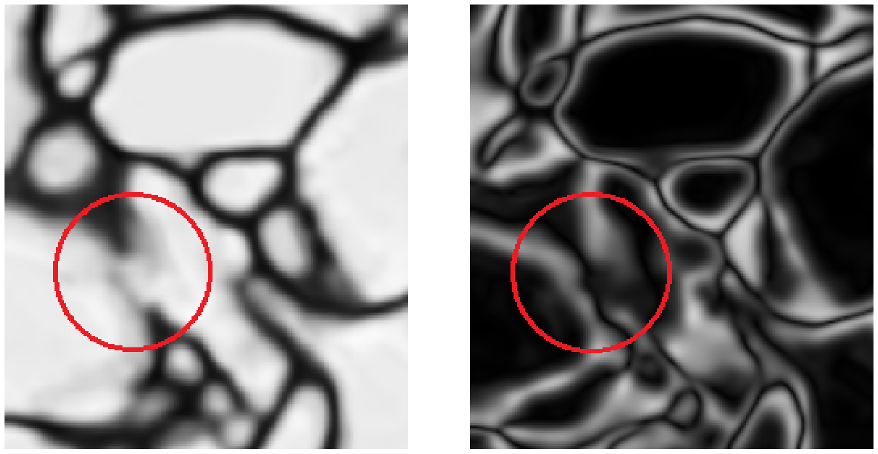
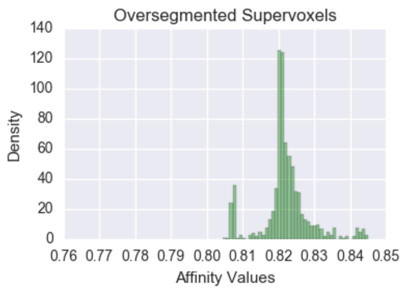


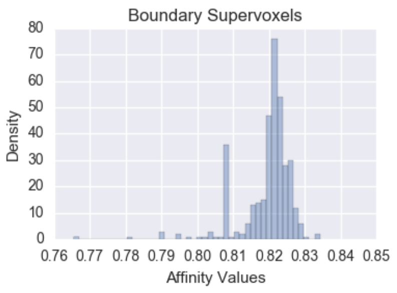
Figure 4.3: The first derivative (right) of the affinity image (left) accentuates the true boundary circled in red.

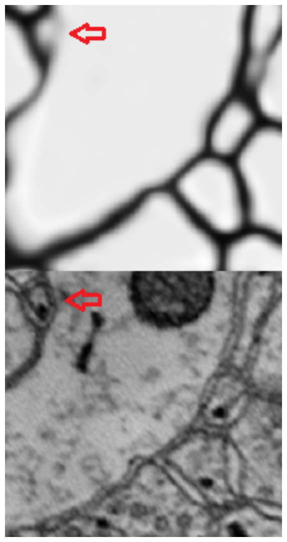
ever, consider the situation presented in figure 4.3. The circled region highlights a true neuron boundary with relatively high affinities based solely on the boundary features. The boundary appears more prominent compared to its neighbors after taking the derivative of the affinity image using a Sobel filter [27]. The oversegmented boundaries do not produce large contrast in the derivative image. After taking the first and second derivative of the affinity image, the following features were extracted from the boundaries: maximum, minimum, median, mean, standard deviation, skew, and kurtosis.

These derivative features do improve on the traditional boundary features but even they only rely on a small 3 x 3 x 3 window around each voxel. These features depend on a good watershed threshold to produce reasonable boundaries between supervoxels. However, there is additional valuable information in the affinity graphs which can be exploited by our machine learning algorithms. For simplicity, supervoxels are assigned a central voxel that minimizes the $L1$ distance to it from all other voxels in the supervoxel. Consider again the formulation of the affinity image as a graph with voxel nodes and edges between neighboring voxels. However, the edges in this construction have weight $1.0 -$ `affinity`. Intuitively, neighboring voxels with a low edge weight are “closer” to each other in some similarity space. Every pair of supervoxels has a shortest path between them (starting at the central locations of each). The shortest path traverses from the center of one supervoxel to the other and therefore considers a much wider region in the affinity image than any of the features previously discussed. This shortest path produces a distance between supervoxels, where a shortest distance indicates a higher probability that two supervoxels belong to the same neuron. In addition to the distance, consider the distribution of affinities along the shortest path and extract the following features: maximum, median, minimum, mean, standard deviation, skew, kurtosis, first quintile, second quintile, third quintile, and fourth quintile. True boundary supervoxels should have a low minimum affinity representing the traversed boundary. However, as discussed earlier, the affinity map does not perfectly assign boundary voxels low affinities. Figure 4.4 shows two shortest path distributions between pairs of supervoxels. Two oversegmented supervoxels produced the distribution on the left while two boundary supervoxels created the distribution on the right. In both pairs of supervoxels, the boundary maximum and mean affinity suggest that the supervoxels are oversegmented (and the random forest classifier from the previous section incorrectly merged both). However, the shortest path statistics provide additional insight into the supervoxels. The shortest distance and the shortest path skew indicated the right boundary is in fact true. Skew measures the asymmetry of a distribution compared to a normal distribution. A true boundary will have slightly lower affinities at least. A shortest path that crosses such a boundary will have affinity values that trend low (even if not low enough to register solely based on boundary features).

Despite the fact that the shortest path algorithm improves on boundary features by considering more than just the affinities on the boundary, it is susceptible to small holes in the boundary predictions. If a boundary contains a wrongfully high affinity between two voxels, the shortest path will exploit this error creating a smaller distance







Metric	Oversegmented	Boundary
Boundary Max	0.807364	0.807397
Boundary Mean	0.771332	0.755981
Distance	1.170613	**1.596400**
Mean	0.822043	0.819276
Standard Deviation	0.006919	0.007619
Skew	0.400652	**-2.353312**

Figure 4.4: Two distributions of affinities along the shortest path (top left). The left distribution is a oversegmented boundary and the right distribution is a true boundary. The boundary location incorrectly predicted as oversegmented by the boundary only classifier (right). A table with some of the attribute values for the true boundary shown on the right (bottom left). Bold values indicated shortest path attributes which predict true boundary.

than otherwise desired. Random walks can minimize the influence of membrane holes by exploring a variety of paths between two supervoxels. Consider the following three different random walk algorithms. The first is a true random walk between the center of one supervoxel to the center of the other supervoxel. At every step, the walker moves towards the other supervoxel in a random direction with equal probability. The random walk returns the minimum affinity along the walk. The algorithm considers 250 such iterations and averages the minimum affinity. The next random walk proceeds in the same constrained region of the affinity graph but does not progress in an arbitrary direction as above. This random walk favors directions of high affinity. These walks explore a much larger region of the affinity graph but are still constrained to the bounding box framed by both supervoxels. The final randomized walk conducts a Monte Carlo simulation over the entire voxel region. For each iteration, a voxel is randomly chosen to start a 1000 step random walk.

The walk favors directions of high affinity to encourage all steps to remain within a neuron. The process iterates over a billion times so that in expectation ∼20 iterations start at each voxel. The machine learning algorithm trains on two features extracted from the Monte Carlo simulation. The first is the total number of steps across a given boundary. The second is the total number of steps across a given boundary normalized by boundary size to approximate a unit flow.

The results of training a random forest classifier on only the shape descriptors are presented in table 4.3. The random forest had the same parameters as described in section 4.2.1 (1000 trees, max depth of 7, and $\sqrt{n}$ variables per split). There is an improvement by 1.266% and 0.818% on the training and testing datasets respectively. Also in that table is the complete set of features and their corresponding information gain. The most useful features are the shortest path distance, the minimum affinity along the shortest path, the standard deviation of the affinities along the path, and all three random walks.

4.2.3 Random Forest Results

Table 4.4 displays the results of the random forest classifier trained on both boundary features and shape descriptors. The random forest classifier had the same parameters as the above two trained random forests. Combining shape descriptors with boundary features improves performance on average by 1.439% and 1.174% on the training and testing datasets respectively. Figure 4.5 provides a visual of the random forest results overlay on both the original EM image and affinity graph.

4.3 Global Merging

A naive hierarchical clustering algorithm sorts all boundaries in descending order of merge probability. Supervoxels are blindly merged creating larger and larger cluster

Dataset	True Positives	False Positives	False Negatives	True Negatives	Proportion Correct (±)
Training 1	26908	687	672	15946	96.926 (+0.997)
Training 2	22588	995	900	17413	95.477 (+0.648)
Training 3	31737	1178	1499	15728	94.661 (+1.908)
Training 4	39350	1041	1310	15024	95.855 (+1.510)
Testing 1	43387	940	1582	15168	95.871 (+0.627)
Testing 2	42143	1232	1675	19472	95.495 (+0.994)
Testing 3	40600	816	920	15495	96.998 (+0.588)
Testing 4	46264	1053	1956	12447	95.125 (+1.063)

Feature Name	Information Gain	Proportion Correct	Split Value
First Derivative Maximum	0.109945	0.539316	0.363275
First Derivative Median	0.281139	0.725126	0.252299
First Derivative Minimum	0.313644	0.756332	0.030775
First Derivative Mean	0.186230	0.649863	0.387998
First Derivative Standard Deviation	0.085195	0.570755	0.159791
First Derivative Skew	0.114192	0.567107	1.433026
First Derivative Kurtosis	0.092367	0.546534	4.611641
Second Derivative Maximum	0.150815	0.625285	20.586449
Second Derivative Median	0.314093	0.766173	9.597646
Second Derivative Minimum	0.335419	0.771946	0.694593
Second Derivative Mean	0.285962	0.763484	14.620527
Second Derivative Standard Deviation	0.115152	0.629700	5.210494
Second Derivative Skew	0.122325	0.605350	1.201753
Second Derivative Kurtosis	0.100682	0.572968	4.248445
Shortest Path Distance	0.566285	0.924286	1.551222
Shortest Path Maximum	0.215426	0.704487	0.816454
Shortest Path Median	0.152763	0.713685	0.811474
Shortest Path Minimum	0.487327	0.873829	0.784011
Shortest Path First Quintile	0.075158	0.700191	0.810058
Shortest Path Second Quintile	0.132685	0.717182	0.811594
Shortest Path Third Quintile	0.168628	0.714006	0.812130
Shortest Path Fourth Quintile	0.191101	0.707539	0.813733
Shortest Path Mean	0.079704	0.718939	0.800415
Shortest Path Standard Deviation	0.425453	0.871829	0.011535
Shortest Path Skew	0.137815	0.677095	-1.361089
Shortest Path Kurtosis	0.125459	0.664590	4.747231
Shortest Path Number of Steps	0.163528	0.721732	66
Weighted Random Walk	0.545440	0.909160	0.451746
Unweighted Random Walk	0.543179	0.905714	0.431717
Monte Carlo Ratio	0.558173	0.921058	3947.285645
Monte Carlo Raw	0.150644	0.736252	197737

Table 4.3: The results of the random forest when trained using only shape descriptors (top). The last column shows the percent increase on using shape descriptors instead of the traditional boundary features. The set of extracted features with the corresponding information gain, proportion correct, and split value (bottom).

Dataset	True Positives	False Positives	False Negatives	True Negatives	Percent Correct (±%)
Training 1	26929	691	651	15942	96.965 (+1.036)
Training 2	22635	945	853	17463	95.708 (+0.890)
Training 3	31781	1082	1455	15824	94.940 (+2.187)
Training 4	39380	997	1280	15068	95.986 (+1.641)
Testing 1	43627	976	1342	15132	96.205 (+0.961)
Testing 2	42409	1245	1409	19459	95.887 (+1.386)
Testing 3	40797	828	723	15483	97.318 (+0.908)
Testing 4	46487	1044	1733	12456	95.501 (+1.439)

Table 4.4: Results for the random forest classifier when trained on both boundary features and shape descriptors. The final column also presents the percent increase over just using boundary features.

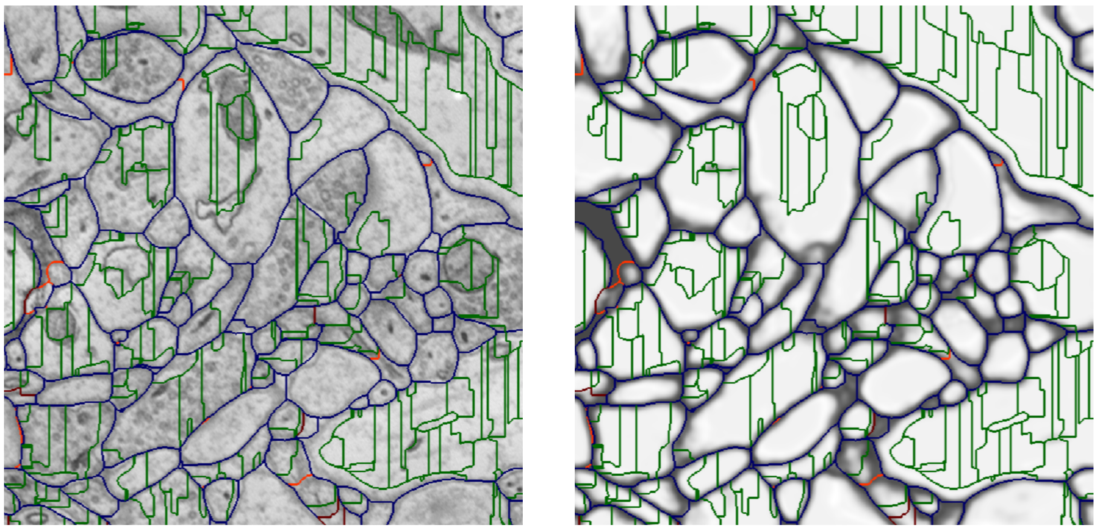


Figure 4.5: Results of the random forest classifier on the Testing 4 dataset overlay on the raw image (left) and affinity graph (right). Dark green lines represent correctly predicted merges, dark blue line correctly predicted boundaries, dark red lines incorrectly predicted merges, and dark orange lines incorrectly predicted boundaries.

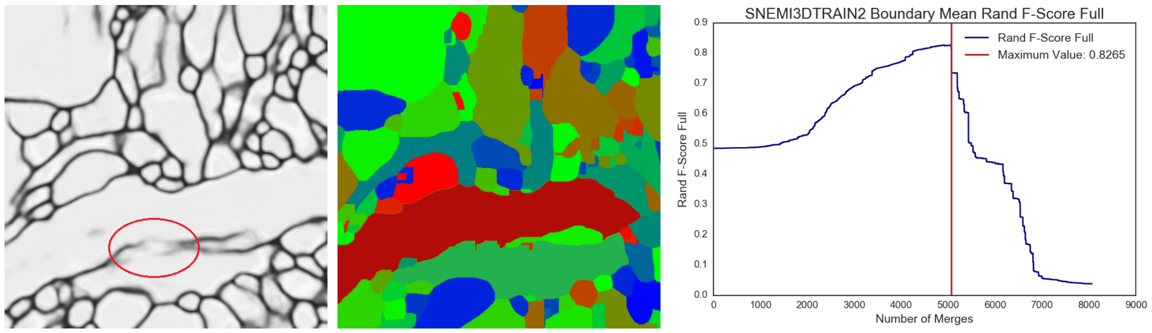


Figure 4.6: A large hole in the affinity map (left) that produces errors in traditional hierarchical clustering algorithms (center). A plot of the Rand F-score versus the number of merges (right) that shows a large drop off.

predictions. Merging ceases when the merge probability reaches a sufficiently low threshold. This algorithm is particularly susceptible to wrongly classified boundaries. If the classifier says a boundary is oversegmented, that boundary will collapse without regard to the surrounding information available. Consider figure 4.6 which shows the results of the naive hierarchical clustering algorithm. The circled region in the leftmost image shows a large "hole" in the affinity map—a location where two distinct neurons have unusually high affinities across their boundary. In this example, the supervoxels on opposite sides of the hole are sufficiently small that the boundary between them is dominated by that mistake. Call the boundary containing the hole H. For the purposes of the merging process, a "boundary collapse" defines a merge between the two clusters across a boundary. The center image in figure 4.6 shows the predicted segmentation of the hierarchical clustering algorithm before H collapses. Both neurons on opposite sides of the hole are correctly segmented. The plot on the right shows the Rand F-score for the hierarchical clustering algorithm after each merge. The algorithm peaks just after 5000 merges with a score of 0.8265 and quickly plummets. The collapse of boundary H produces this drastic fall in Rand F-score.

The hierarchical clustering algorithm does not adequately exploit all of the infomration available. An algorithm should consider the entire surrounding before

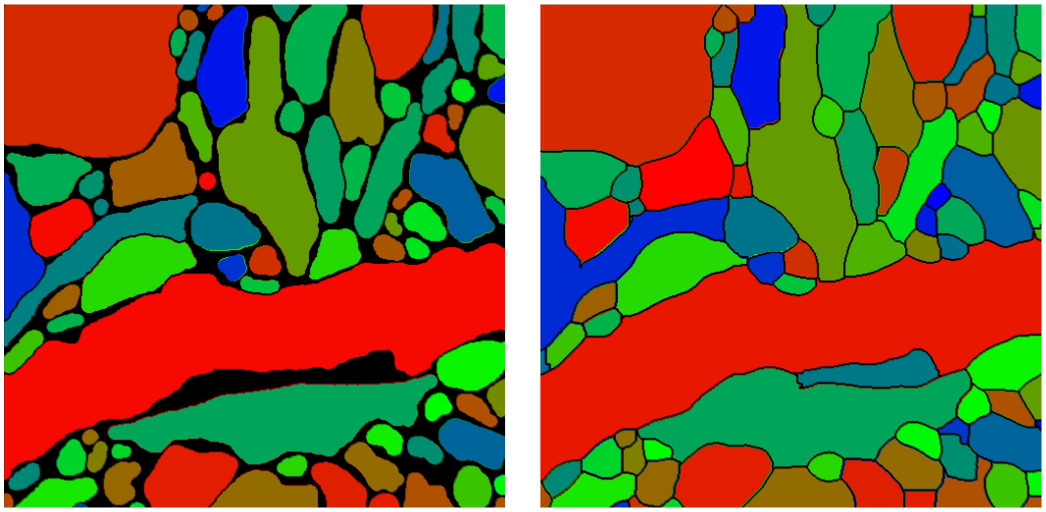


Figure 4.7: The results of the global merging algorithm (right) on the dataset shown in figure 4.6 compared to the ground truth (left). This algorithm correctly segments the two neurons across the hole.

collapsing the boundary H in figure 4.6. In this example, both of these neurons have undergone numerous correct internal merging, causing the well-defined structure visible in the figure. Many of the superovels in these neurons maintain boundaries with supervoxels in the adjacent neuron. These boundaries are correctly labeled as true segmentations, otherwise they would have already merged before reaching H. When a given boundary collapses, all of the boundaries between the two clusters also collapse. Any boundary that would collapse because of a merge should vote on the merge. Boundaries predicted as true boundaries vote no for merging and those predicted as oversegmented vote yes.

Figure 4.7 shows the results of the global merging algorithm on the image with the boundary hole. The boundary H does not collapse since the votes from the other supervoxels prevent the merge. The Rand F-score for this dataset using the global merging algorithm is 0.8684, an improvement of 0.0419. The Rand error decreases by 24.15%. The global merging algorithm does not require any stopping conditions like the hierarchical clustering algorithm. Rather, every boundary is iterated over in merge probability order as predicted by the classifier in section 4.2. Any boundary

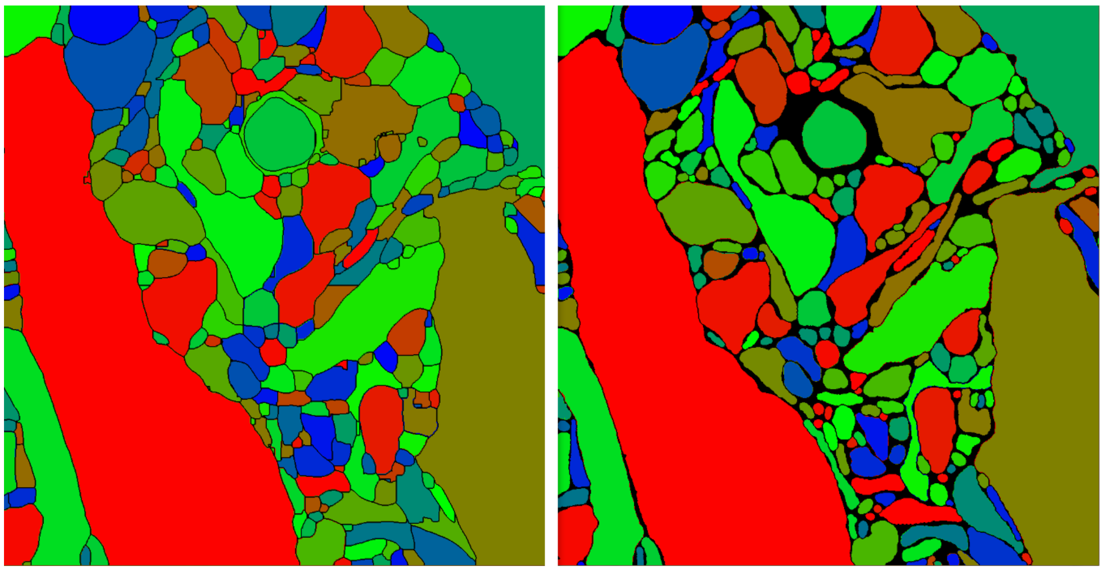
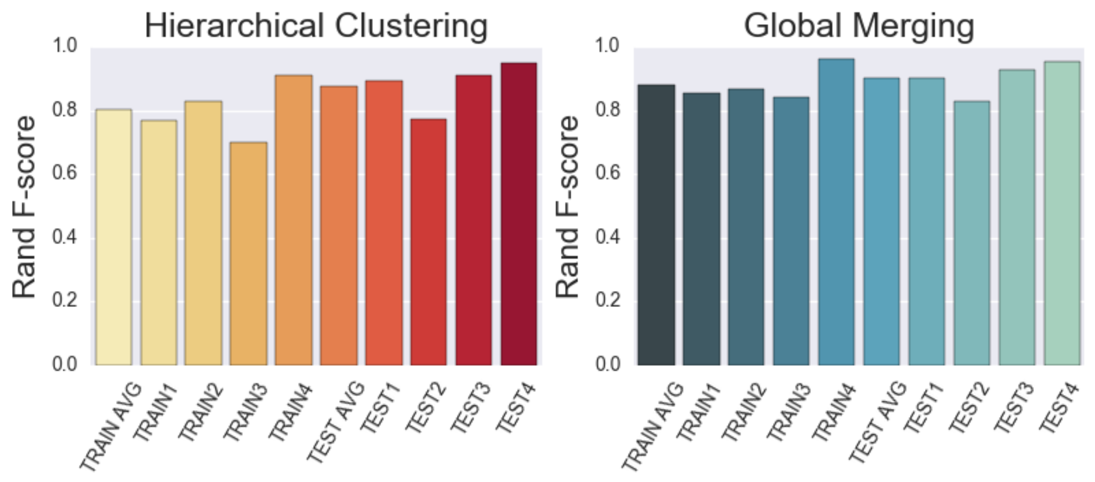


Figure 4.8: Output of the global merge algorithm (left) compared to the human labels (right).

that accumulates enough votes among its neighbors collapses. However, there is one parameter that needs to be addressed: the proportion of positive votes required for merging. After validation on the training dataset, a threshold of 0.80 was chosen. If over 80% of the other affected boundaries vote to merge, a boundary collapses. Otherwise, that boundary remains and the algorithm proceeds to the next candidate.

4.3.1 Results

Figures 4.8 and 4.9 display a visual sample and the numerical results for the global merging algorithm respectively. For the training datasets, the above algorithm has a Rand F-score that on average is 9.87% higher than the baseline. Likewise, the Rand F-score increases on average of 2.35% on the testing dataset. However, the naive hierarchical clustering algorithm performs better on the testing dataset so the smaller increase is expected. Looking at the Rand error (1 - Rand F-score) shows how much the global merging algorithm decreases the mistakes of the hierarchical clustering



	Hierarchical Clustering		Global Merging		
Dataset	Rand F-Score	Rand Error	Rand F-Score	Rand Error	Error Decrease
Training 1	0.769	0.231	0.852	0.148	35.941%
Training 2	0.827	0.174	0.869	0.131	24.286%
Training 3	0.699	0.301	0.841	0.159	47.215%
Training 4	0.911	0.089	0.960	0.040	55.033%
Training Average	0.801	0.199	0.880	0.120	39.808%
Testing 1	0.893	0.107	0.902	0.098	8.395%
Testing 2	0.774	0.226	0.826	0.174	23.091%
Testing 3	0.909	0.091	0.928	0.072	20.730%
Testing 4	0.948	0.052	0.951	0.049	5.529%
Testing Average	0.881	0.119	0.902	0.098	17.417%

Figure 4.9: A bar plot of the Rand F-score for the hierarchical clustering algorithm (left) and the global merging algorithm (right). (bottom) The numerical results of both algorithms.

algorithm. The training and testing datasets decrease Rand error by 39.808% and 17.417% respectively.

4.4 Graph Cut

Neuron cells are exceptionally long compared to the relatively small diameter of the reconstructed volume. Therefore, no neuron should both begin and end within the same volume. Rephrased, every supervoxel inside the volume should belong to a neuron which at one point leaves the volume. Any prediction from section 4.3 that does not intersect the volume boundary should merge with a neighboring segment that does. The above algorithm does not guarantee this neuron invariant. However, reformulating the problem can allow us to use a graph cut algorithm which guarantees the desired properties. Graph cuts are frequently used in image segmentation and graph partitioning problems [3] [5]. Graph cuts minimize the following objective function:

$$Energy = \delta * \sum_{p \in \mathcal{P}} d_p(a) + \beta * \sum_{p,q \in \mathcal{N}} v_{p,q}(a,b) \tag{4.3}$$

Above, δ, β are constants, a, b are labels, $\mathcal{P}$ is the set of pixels, and $\mathcal{N}$ is the set of neighboring pixels. The term $d_p(a)$ is the cost of assigning pixel p label a and the term $v_{p,q}(a,b)$ is the cost of assigning pixel p label a and pixel q label b.

We can choose the labels strategically to force every interior supervoxel to belong to a neuron that intersects the boundary. Every exterior supervoxel receives a unique labeling. Interior supervoxels must receive one of these labels. Neighboring supervoxels that have different labels incur a smoothing penalty $v_{p,q}(a,b)$. Assigning a label a to an interior supervoxel p contributes a cost of $d_p(a)$ to the energy function. Since every interior supervoxel receives a label, all supervoxels belong to predictions that intersect the boundary. This implementation uses the GCoptimization library with alpha-beta swaps [5] [20] [4]. Although this produces a convenient formulation of the problem, it creates two additional problems of assigning data and smoothing costs for all supervoxels.

4.4.1 Graph Cut Data Terms

The graph cut procedure described in this section produces a random forest classifier to predict if an interior supervoxel belongs to the same neuron as an exterior supervoxel. There are several possible features on which to train, although most only

Dataset	True Positives	False Positives	False Negatives	True Negatives	Percent Correct
Training 1	611,011	10,330	100,396	24,172,938	99.555
Training 2	592,809	13,935	82,614	24,925,082	99.623
Training 3	1,081,484	23,717	165,501	25,357,580	99.289
Training 4	4,566,029	49,178	319,865	3,078,7091	98.967
Testing 1	11,323,771	147,023	903,941	49,637,938	98.305
Testing 2	5,922,484	40,631	667,485	58,026,375	98.905
Testing 3	9,545,290	90,499	314,437	37,336,860	99.144
Testing 4	10,157,766	103,915	571,205	43,696,962	98.762

Table 4.5: Results for the data term random forest classifier.

introduce noise. For example, determining the shortest path statistics between all supervoxels creates data highly dependent on the location of the interior supervoxels. That is, distances are arbitrarily small to nearby supervoxels that might belong to different neurons. Therefore, training the random forest on two simple features provides the best results. The first feature is merely the output of the global merging algorithm. If an exterior supervoxel belongs to the same prediction as an interior supervoxel, the data cost is 0. Otherwise, the cost is 1. This graph cut algorithm is used to merge predicted segments that do not overlap the boundary. With this data term, these interior segments will receive an equal cost no matter the label. Thus, the interior segment will receive the labeling that minimizes the smoothing terms between all the neighbors. The second feature comes from a slight modification of the global merging algorithm. The slight variant guarantees that every interior supervoxel has a path to the volume edge by growing segments from the boundary. Once every supervoxel is mapped to an exterior supervoxel, the global merging takes over and starts merging these segments. The prediction results of the data term are presented in 4.5. This simple random forest performs exceptionally well for the data term labeling problem.

4.4.2 Augmented Merge Classifier With Global Features

The random forest classifier from section 4.2 only considers local information about the supervoxel boundaries. However, neurons have a highly repetitive shape and overall global structure that can be exploited by choosing additional features. The following two subsections focus on approximation algorithms to the NP-Complete clustering problem that provide additional information for the merge classifiers.

k-Median Clustering

The k-median segmentation problem focuses on generating k clusters from n possible locations. A distance metric maps two locations to a real valued similarity. For this paper, the shortest path distance served as the distance metric. The problem focuses on choosing k centers from the n locations to serve as cluster centers. The goal is to choose centers that minimize the average distance of all other points to the nearest cluster center. Although this problem is NP-Complete, there are several heuristics that approximate the optimal solution. One such heuristic, a Lloyd-style expectation and maximization iterative cycle, converges to a local minimum by choosing an optimal center for a cluster, reassigning the other n points to the nearest cluster, and repeating until convergence. This implementation uses 100 random restarts since the E & M algorithm only promises convergence to a local minimum. The k-median problem requires a parameter k to denote the number of clusters in the data. k is unknown for most image segmentation problems. In particular, the number of neurons is not known beforehand. Rather than choose an arbitrary value of k, all values of k from 2 to 1000 were considered.

Facility Location Assignment Problem

The facility location problem is a slight variant of the k-median segmentation problem. The cluster centers from the k-median problem are restricted to a certain supervoxel

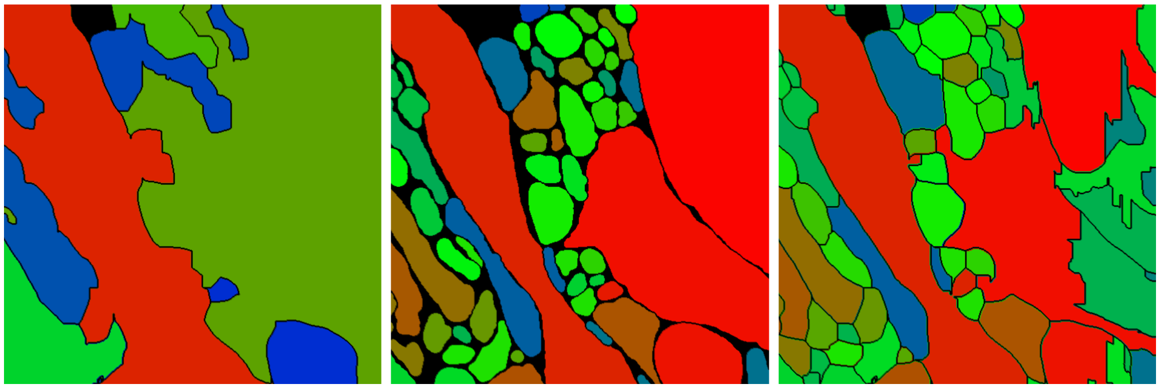


Figure 4.10: The undersegmented results for the facility location assignment heuristic with $k = 25$ (left), next to the ground truth (center), and the oversegmentation when $k = 600$ (right).

subset. Again, since neurons should extend past the volume edges, each interior supervoxel should belong to a neuron that intersects the boundary. Therefore, the exterior supervoxels are the only available cluster centers forcing every interior supervoxel to belong to a segment that exits the volume. Farthest-first is a simple heuristic to the facility location problem. For each new value of k, the facility farthest from all others that is not currently assigned becomes the new cluster center. As in the above problem, k is not known so this paper considers all values of k from 2 to 1000.

Extracting Features From Global Algorithms

The previous two sections outlined global approximation algorithms that produce nearly a thousand predictions varied on the parameter k. A random forest uses random splits in the decision trees so that many features would overwhelm any other local features extracted earlier. Figure 4.10 shows the output of the facility location assignment algorithm for $k = 25$ and $k = 600$ on the Training 4 dataset. There are actually 145 neurons in this dataset, so $k = 25$ provides an undersegmentation while $k = 600$ provides an oversegmentation. A boundary between supervoxels that splits for a small value of k is more likely to represent a true boundary than one that splits first for a large value of k. Therefore, the first value of k for which a boundary splits two segments provides some useful information. In addition to this, the random forest trains on a feature representing the proportion of times a boundary splits two clusters. In addition to the NP-Complete approximations above, the global merging algorithm produce three features for the new classifier. Varying the voting threshold provides a similar parameter to the k in the above approximations. In addition, the entire output of the global merging algorithm for threshold 0.80 creates an additional feature. Table 4.6 shows the information gain for these feature for the k-median, facility location, and global merging algorithms. The boundary merge algorithm was briefly described in section 4.4.1

Augmented Merge Random Forest Classifier

Table 4.6 displays the results of training a random forest classifier on only the global features outlined in this section. The classification rates are greater than when using boundary and shape descriptors alone. The percent increase over using only boundary features is shown in the last column. The information gain for the global features shows just how powerful they are. The global merging and boundary merging attributes far outperform any of the other features considered in this thesis. Even the facility location boundary proportion feature performs on par with the best boundary and shape features. Creating a better smoothing term function for the graph cut algorithm motivated this exploration into the global approximation algorithms. The following section analyzes the graph cut results using the new classifiers.

4.4.3 Graph Cut Conclusions

The parameters δ and β can be adjusted depending on how much the smoothing or data terms should be trusted relative to each other. Validating on the training dataset shows that the Rand F-score values level off around a $\frac{\beta}{\delta}$ ratio of $\sim$3. The appendix

Dataset	True Positives	False Positives	False Negatives	True Negatives	Percent Correct (±%)
Training 1	27041	417	539	16216	97.838 (+1.909)
Training 2	22772	666	716	17742	96.701 (+1.883)
Training 3	32278	689	958	16217	96.715 (+3.962)
Training 4	39813	584	847	15481	97.477 (+3.132)
Testing 1	43765	808	1204	15300	96.706 (+1.462)
Testing 2	42620	1022	1198	19682	96.559 (+2.058)
Testing 3	40994	725	526	15586	97.837 (+1.427)
Testing 4	46766	996	1454	12504	96.030 (+1.968)

Feature Name	Information Gain	Proportion Correct	Split Value
Facility Location First Split	0.310144	0.826932	34.000000
Facility Location Proportion	0.534458	0.919985	0.697995
k-Medians First Split	0.253482	0.813578	14.000000
k-Medians Proportion	0.306648	0.841846	0.876992
Global Merging First Split	0.693778	0.955818	60.000000
Global Merging Proportion	0.721981	0.963856	0.390000
Global Merging Boundary	0.715232	0.959415	0.000000
Boundary Merge First Split	0.690454	0.959213	57.000000
Boundary Merge Proportion	0.691546	0.959466	0.430000
Boundary Merge Boundary	0.675984	0.955342	0.000000

Table 4.6: The results of the random forest when trained using only global features (top). The set of extracted features with the corresponding information gain, proportion correct, and split value (bottom).

shows the full results for the graph cut algorithm for a wide range of considered ratios.

4.5 Post Processing

The final step in the algorithmic process combines the results of the global merging algorithm with the graph cut algorithm. The post processing algorithm emphasizes the invariant that no neuron is completely contained within a volume. Therefore any interior prediction should merge with a neighbor until it also exits the volume at some location. Secondly, neurons should not fully encapsulate others so any neuron that is completely surrounded should merge with its one neighbor.

4.5.1 Merging Graph Cut Results

All predictions should intersect the volume boundary and those that do not should merge with an adjacent prediction. However, it is non-trivial to determine which prediction boundary should collapse. This algorithm presents a simple conservative merging algorithm to improve the results over the global merging algorithm discussed in section 4.3. The graph cut formulation of the problem provides the desired property that every interior supervoxel receives the label of an exterior supervoxel. During post processing, interior predictions are merged with the neighbor that best agrees with the graph cut labeling. This forces a merge of the interior prediction with a boundary prediction while still using the overall global information presented throughout this paper. In actuality, this merging scheme is too aggressive and predictions are only merged if the graph cut labeling extends over the entire boundary between the predictions.

4.5.2 Enclosed Predictions

Neurons should not completely surround neurons, so the algorithm reconsiders any pair of predictions that have this property. This step simply merges the enclosed prediction with the larger prediction. For each dataset, around 10 to 50 predictions are fully enclosed and merged with their neighbor.

4.5.3 Stitching Datasets

Originally, the datasets were split into four equal sizes to allow quicker computation. Thus, the final predictions produced need to be merged together to get a full reconstruction of the SNEMI3D volume. This thesis performed the naive stitching algorithm where a segment in one volume maps to another if and only if a majority of voxels along the boundary agree.

Chapter 5

Results

5.1 Classification Results

Sections 4.2.1, 4.2.2, and 4.4.2 suggest several features for merge classification. Most current state-of-the-art segmentation algorithms such as GALA rely solely on the features presented in section 4.2.1. These features however do not fully leverage the information provided by the affinity maps. Classification rates greatly improve with the introduction of shape descriptors and other global features. Table 5.1 displays the results of four random forest classifiers presented in this paper. Next to the success rate is the improvement over the traditional boundary features. After accumulating all of the features and generating the final random forest classifier, the classification rates improved on average by 2.737% and 1.816% on the training and testing datasets respectively.

5.2 Segmentation Results

Figure 5.2 displays the results of all the global algorithms considered (global merging, k-median, facility location, graph cut) compared to the baseline hierarchical clustering algorithm. These values show the best Rand F-score after varying parameters for

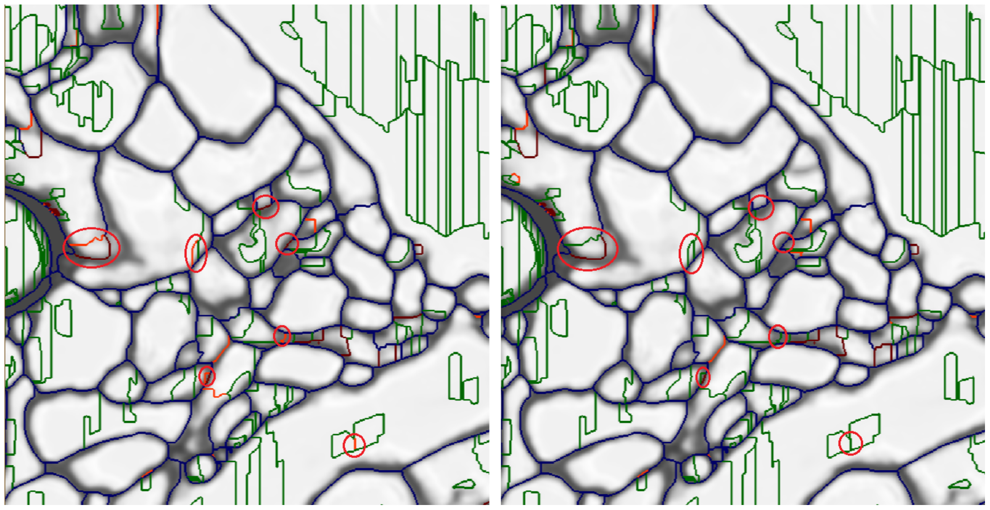


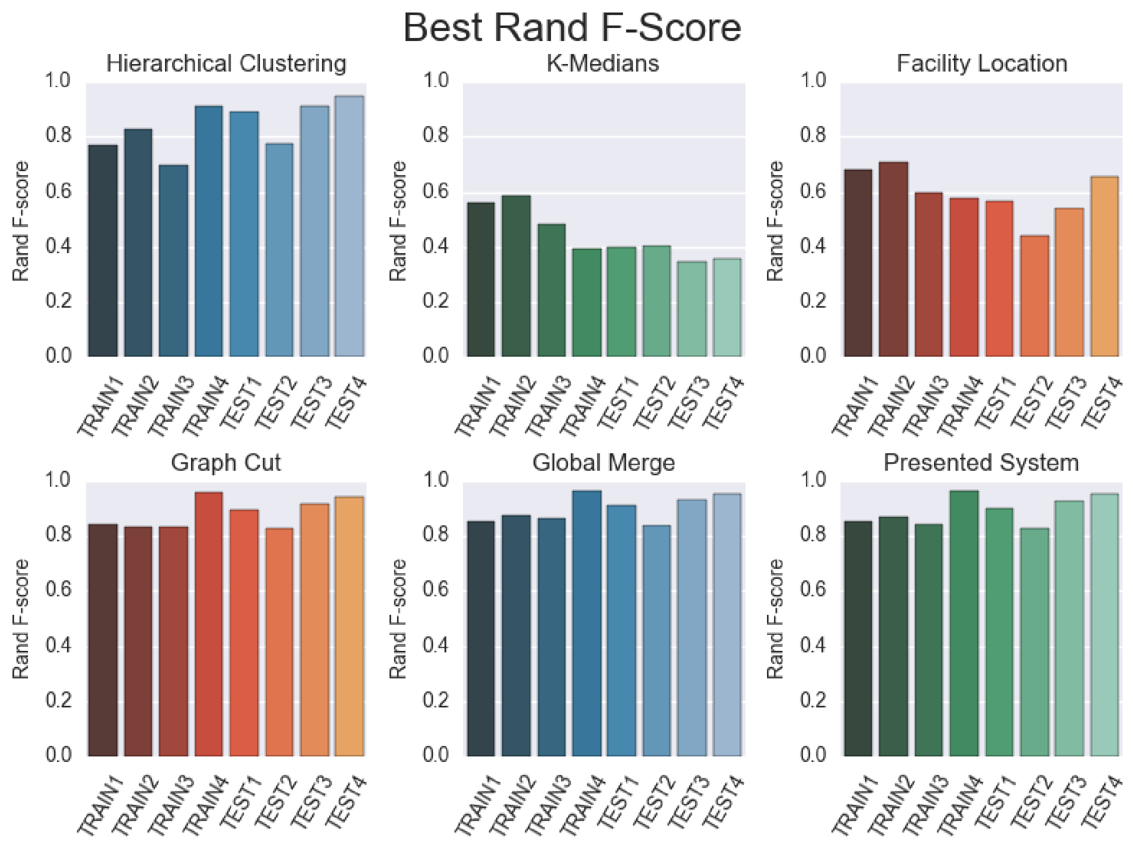
Figure 5.1: Improvements from the original random forest (left) when adding global features (right) are circled in red. Green boundaries represent true positive merges, blue boundaries true negative rejections, red boundaries false positive merges, and orange boundaries false negative rejections.

Dataset	Boundary Features	Shape Descriptors	Global Features	Combined
Training 1	95.927	96.924 (+0.997)	97.838 (+1.911)	97.867 (+1.940)
Training 2	94.818	95.475 (+0.657)	96.701 (+1.883)	96.703 (+1.885)
Training 3	92.751	94.661 (+1.910)	96.715 (+3.964)	96.713 (+3.962)
Training 4	94.345	95.854 (+1.509)	97.477 (+3.132)	97.506 (+3.161)
Testing 1	95.244	95.869 (+0.625)	96.704 (+1.460)	96.779 (+1.535)
Testing 2	94.503	95.496 (+0.993)	96.559 (+2.056)	96.645 (+2.142)
Testing 3	96.403	96.998 (+0.595)	97.837 (+1.434)	97.927 (+1.524)
Testing 4	94.062	95.126 (+1.064)	96.126 (+1.968)	96.126 (+2.064)

Table 5.1: Results of the various random forest classifiers presented in this thesis.

each dataset (except for the presented system which relied on validation to tune parameters).

The full results of this system are presented in the table in figure 5.2 compared to the baseline metrics. After stitching, the Rand F-score improved from 0.7882 to 0.8824 and 0.8761 to 0.8948 on the training and testing datasets respectively. This represents an increase of 11.96% and 2.13% on both datasets. The Rand error decreases by 44.51% and 15.10% on both datasets. Figure 5.3 shows a visual representation of an image slice for the entire algorithm compared to the ground truth.



Dataset	Boundary Mean	K-Medians	Facility Location	Graph Cut	Global Merge	Presented System
TRAIN1	0.7691	0.5617	0.6799	0.8438	0.8530	0.8518
TRAIN2	0.8265	0.5894	0.7096	0.8339	0.8723	0.8671
TRAIN3	0.6986	0.4802	0.6000	0.8338	0.8625	0.8407
TRAIN4	0.9111	0.3917	0.5759	0.9584	0.9609	0.9624
TEST1	0.8926	0.3971	0.5644	0.8960	0.9081	0.9019
TEST2	0.7742	0.4021	0.4418	0.8252	0.8357	0.8288
TEST3	0.9091	0.3467	0.5422	0.9129	0.9287	0.9272
TEST4	0.9484	0.3575	0.6556	0.9420	0.9521	0.9517

Figure 5.2: Bar plots showing the best Rand F-score for the various algorithms (top). Corresponding numerical values for each algorithm (bottom). The presented system does not outperform the global merging algorithm because the global merging algorithm presents the score for the best threshold (not the threshold determined by validation).

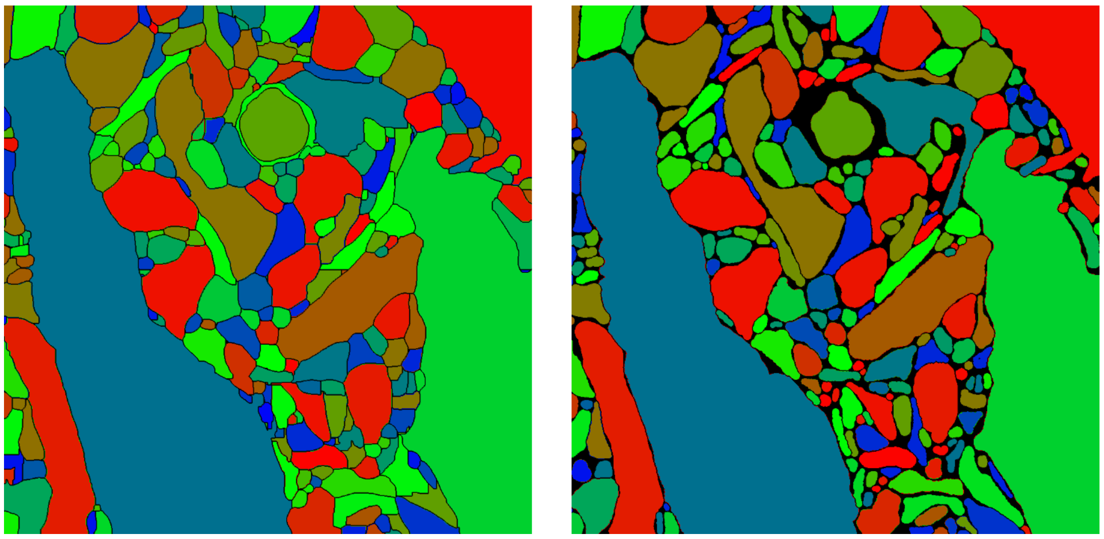


Figure 5.3: A comparison of the ground truth (left) to the output of this system (right).

5.3 SNEMI3D Leader Board

The organizers of the SNEMI3D segmentation challenge maintain a leader board of segmentation submissions. Table 5.2 shows the current results for the SNEMI3D challenge as of the completion of this thesis.

The implementation described in chapter 4 finishes fourth on the current leaderboard. FlyEM is the original GALA submission and Team GALA represents the work of Neal Donnely for his undergraduate thesis [7]. The human values represents the amount of expected difference between two expert labelers on this particular dataset. Although this algorithm does not currently beat "Team GALA", the average Rand F-Score of the four sections contributes a Rand error of 0.097581. The stitching algorithm is rather naive and the Rand error increases to 0.105244 after that step in the process. Future iterations of this algorithm should investigate more sophisticated stitching algorithms to see which methods produce less variance between the unstitched average and the stitched score.

Group Name	**Rand Error**
Human Values	0.059975952
IAL	0.071077332
DIVE	0.091043666
Team GALA	0.100411152
This System	0.10524400
SCI	0.108290875
MIT	0.113610867
Anonymous	0.115012125
FlyEM	0.125041424
rll	0.131110546
Rhoana	0.148351315
shahab	0.166453271
ELEKTRONN	0.204381206
DIVE SCI	0.298128812
SPLab	0.466491124

Table 5.2: The current standing on the SNEMI3D challenge dataset leader board.

Chapter 6

Conclusion and Future Work

This thesis presented global methods for reconstructing neurons in electron microscopy images. First, we explored the feature space currently used by more sophisticated hierarchical agglomeration algorithms. The introduction of shape descriptors produced a noticeable increase in the success of a merge classifier. These shape descriptors leverage additional information gained from the affinity images which are usually discarded. A clustering algorithm that considered the global repercussions of an individual merge transformed the results of the random forest classifier into a segmented image that beats the baseline hierarchical agglomeration techniques. Merging these results with the output of a graph cut segmentation provided a topologically consistent reconstruction of neural images.

6.1 Future Work

Although the merge classifier performs very well with the current boundary and shape descriptors, there is still room for improvement. The random walks are an early attempt at exploiting the entire affinity graph and not just the affinities on the boundaries. However, these random walks merely provide a first glimpse into the possibilities and still rely heavily on initial starting and boundary locations. An ideal feature would measure the similarity between two voxels not by a single path or constrained set of paths but by all possible paths. Fouss et al. explored similarity computations between nodes in a graph using random walks [9]. In particular, they provide simple equations for generating the average commute time between nodes which considers the entire affinity graph. That algorithm requires the pseudo-inverse of the Laplacian of the graph, a computationally expensive matrix to acquire. However, there are alternative approximations which use the highest order eigenvalues, which are cheaper to find. A future iteration of this project should explore the possibilities such features might provide.

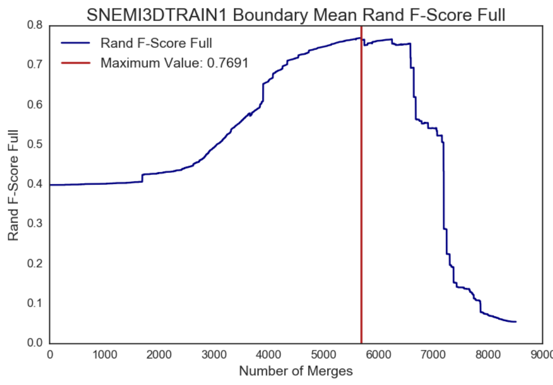
The current features are highly dependent on the quality of the affinities in a given region. Certain sections of the electron microscopy images are noticeably darker or lighter leading to a local variation in affinity not present elsewhere. More robust features would help alleviate the problems caused by these discrepancies in the images. One such feature would be the difference between the maximum and minimum affinity along a shortest path or the shortest path along the derivative image.

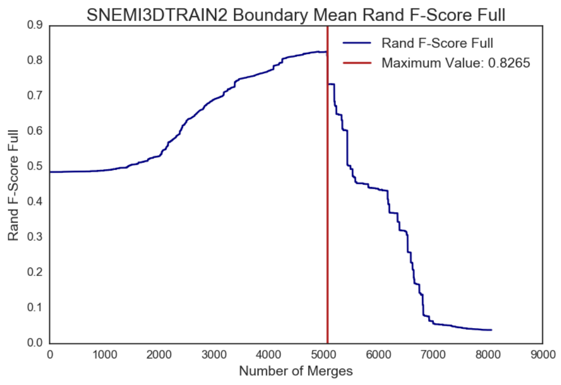
The post-processing algorithmic steps definitely leave room for further exploration. The current conservative methods do not sufficiently solve the problems posed in section 4.5. Although the graph cut algorithm does present overall global structure connecting interior predictions to the boundary, it is not perfect and relies heavily on the success of one algorithm. A more robust method would extract features from the predictions of the global merging algorithm in the same way that features were extracted from the watershed supervoxels to train another merge classifier, similar to the GALA framework. In addition, this classifier should also suggest prediction splits for regions that do not appear to look like neurons.

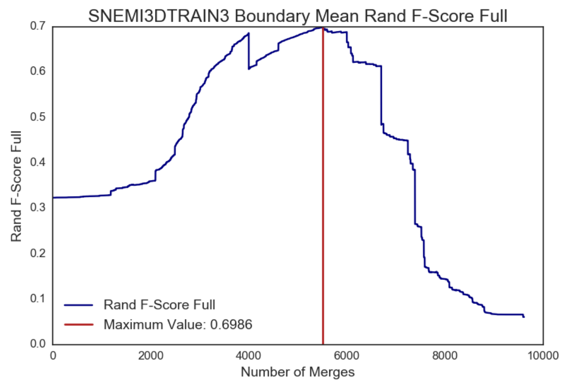
Appendix A

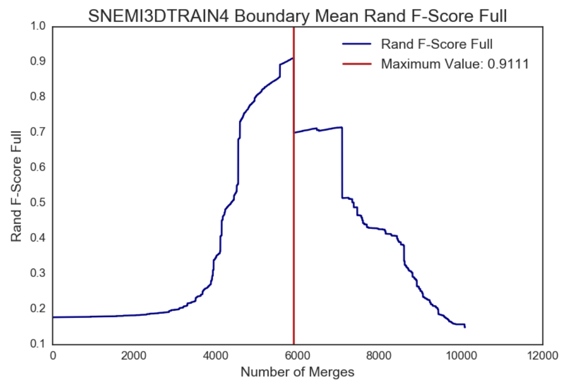
Full Rand F-Score Plots

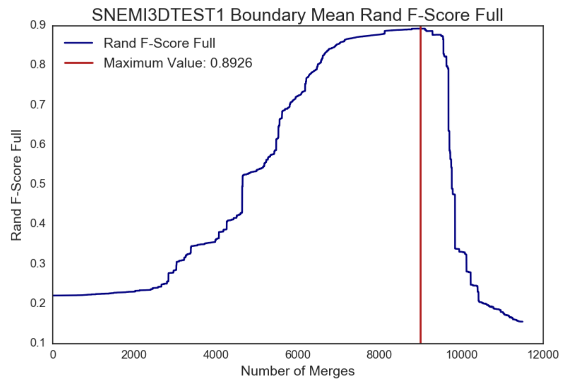
This appendix provides all of the Rand F-score plots for the various algorithms created and employed in this thesis. The first plots show the results of the baseline hierarchical clustering algorithm. The next sets of plots contain the results of the global merging algorithm while varying the vote threshold needed to merge. Next are the plots for the k-median and facility location formulations of the problem with the varying number of cluster and facilities. Lastly, the results of the graph cut algorithm when varying the weighting ratio of data terms to binary terms.

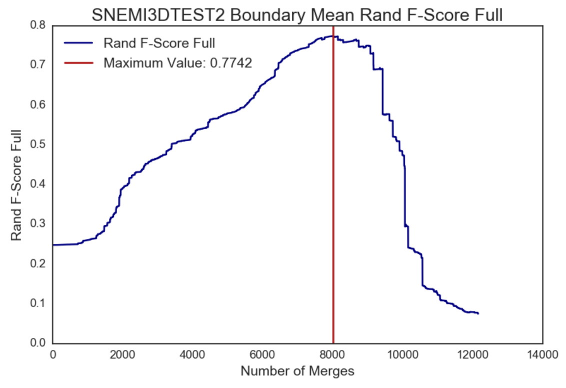


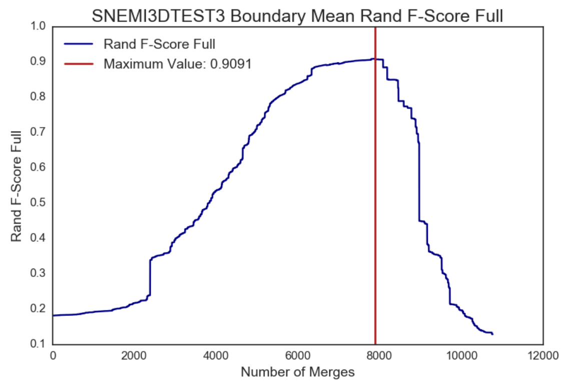


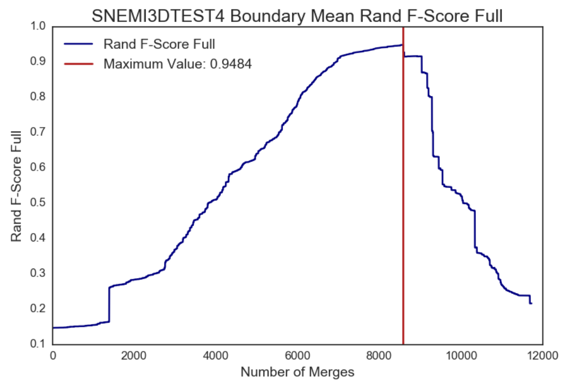

SNEMI3DTRAIN3 Boundary Mean Rand F-Score Full
Rand F-Score Full
Number of Merges
Rand F-Score Full
Maximum Value: 0.6986

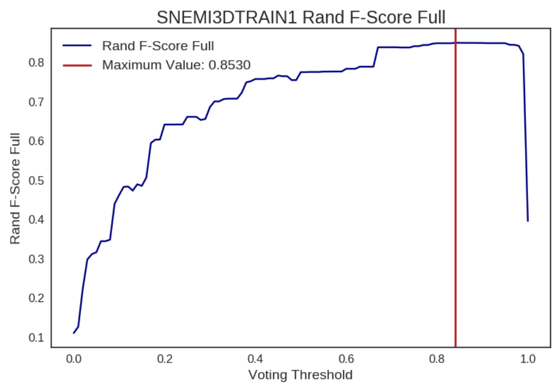

SNEMI3DTRAIN4 Boundary Mean Rand F-Score Full
Rand F-Score Full
Number of Merges
Rand F-Score Full
Maximum Value: 0.9111

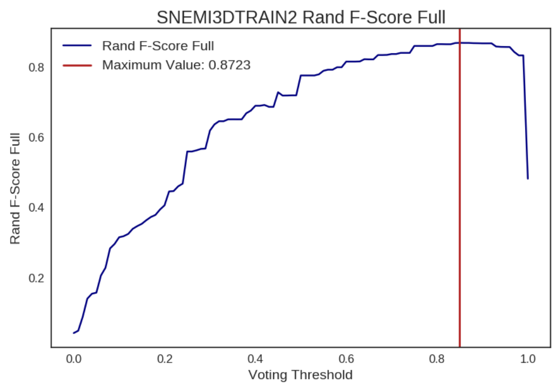

SNEMI3DTEST1 Boundary Mean Rand F-Score Full
Rand F-Score Full
Number of Merges
Rand F-Score Full
Maximum Value: 0.8926

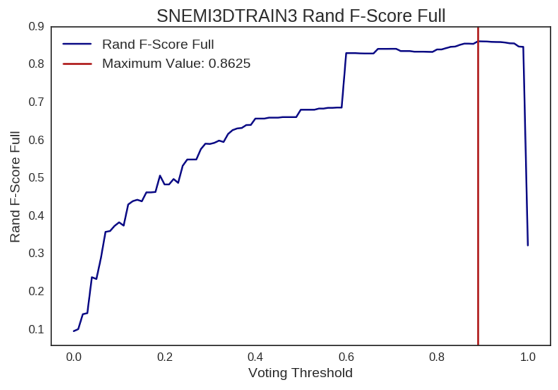

SNEMI3DTEST2 Boundary Mean Rand F-Score Full
Rand F-Score Full
Number of Merges
Rand F-Score Full
Maximum Value: 0.7742

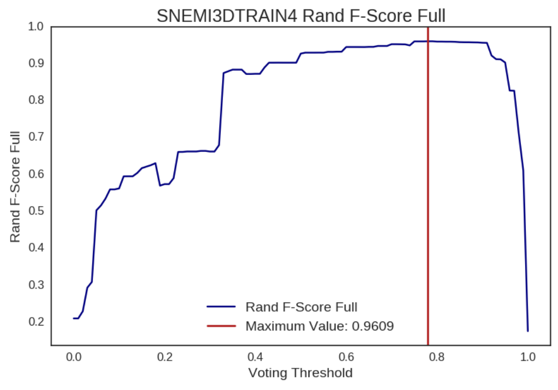

SNEMI3DTEST3 Boundary Mean Rand F-Score Full
Rand F-Score Full
Number of Merges
Rand F-Score Full
Maximum Value: 0.9091

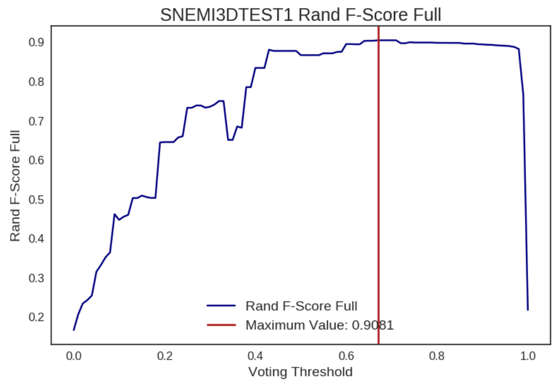

SNEMI3DTEST4 Boundary Mean Rand F-Score Full
Rand F-Score Full
Number of Merges
Rand F-Score Full
Maximum Value: 0.9484

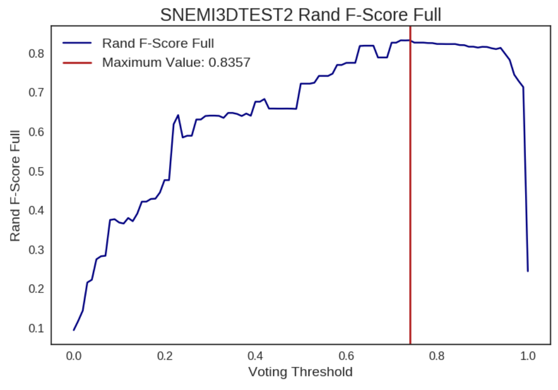

SNEMI3DTRAIN1 Rand F-Score Full
Rand F-Score Full
Voting Threshold
Rand F-Score Full
Maximum Value: 0.8530

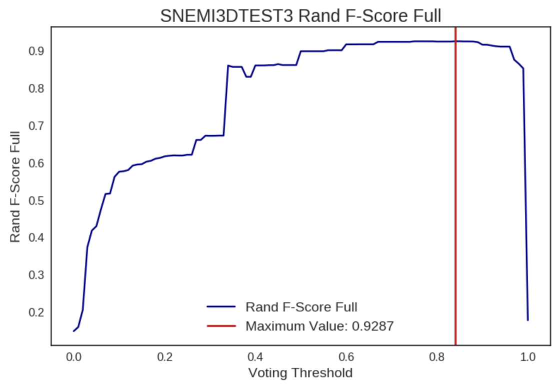

SNEMI3DTRAIN2 Rand F-Score Full
Rand F-Score Full
Voting Threshold
Rand F-Score Full
Maximum Value: 0.8723

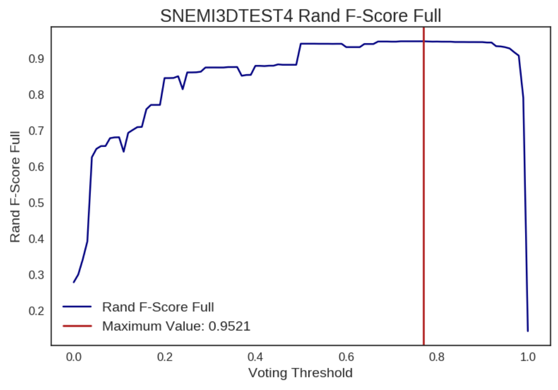

SNEMI3DTRAIN3 Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.8625
Rand F-Score Full
Voting Threshold

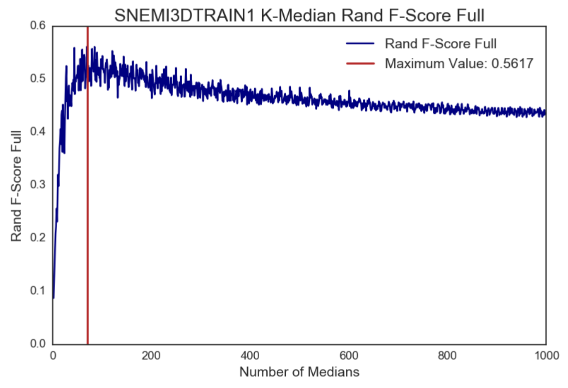

SNEMI3DTRAIN4 Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.9609
Rand F-Score Full
Voting Threshold

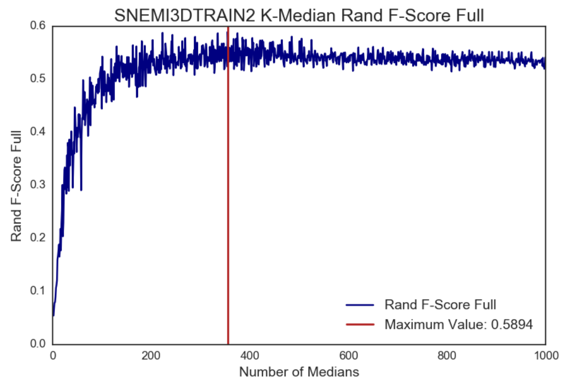

SNEMI3DTEST1 Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.9081
Rand F-Score Full
Voting Threshold

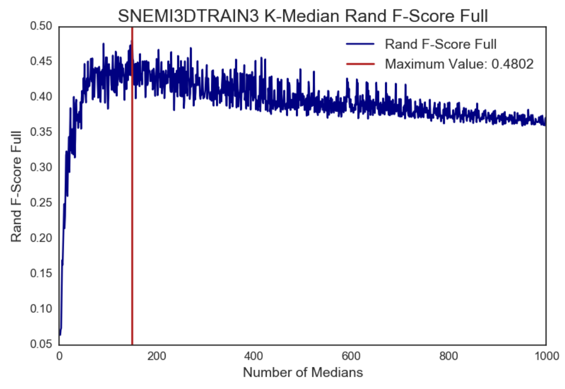
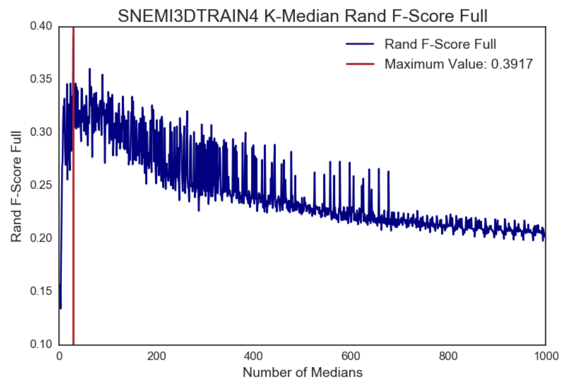
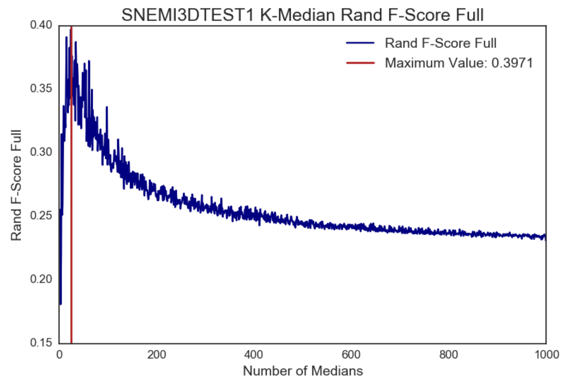
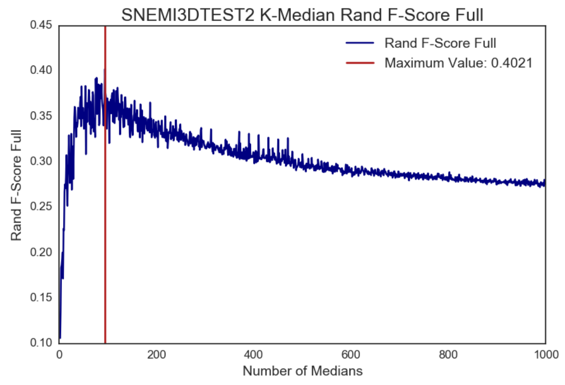
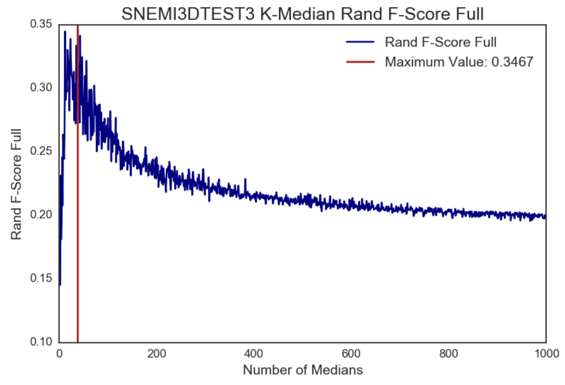
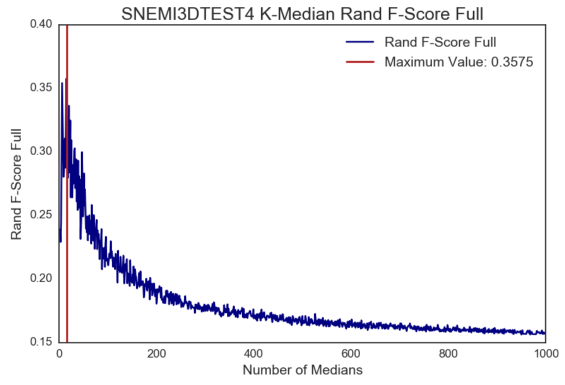
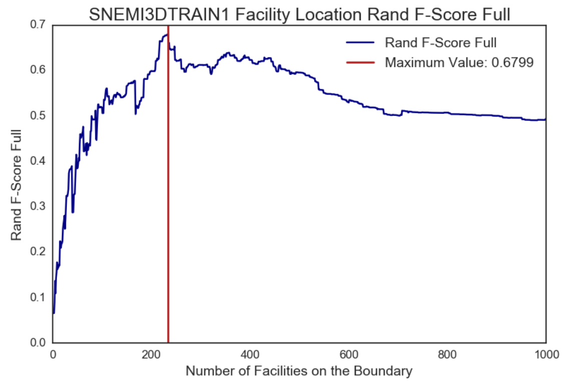
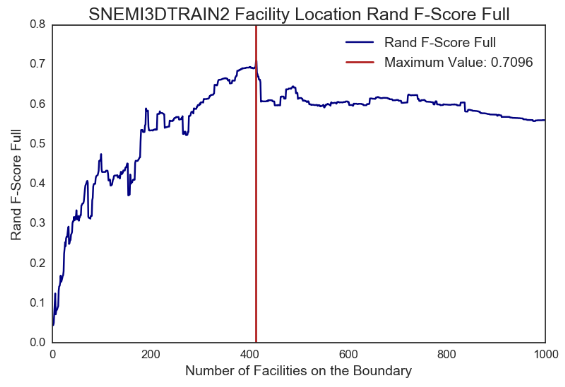

SNEMI3DTEST2 Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.8357
Rand F-Score Full
Voting Threshold

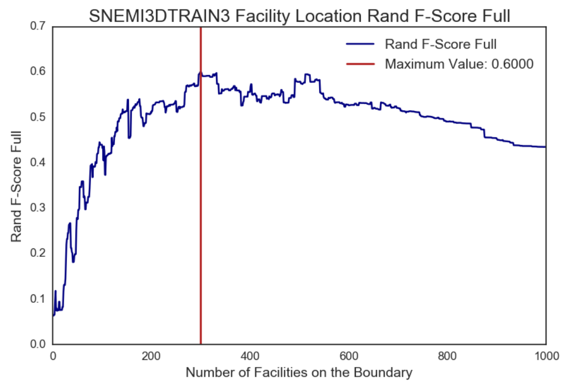

SNEMI3DTEST3 Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.9287
Rand F-Score Full
Voting Threshold

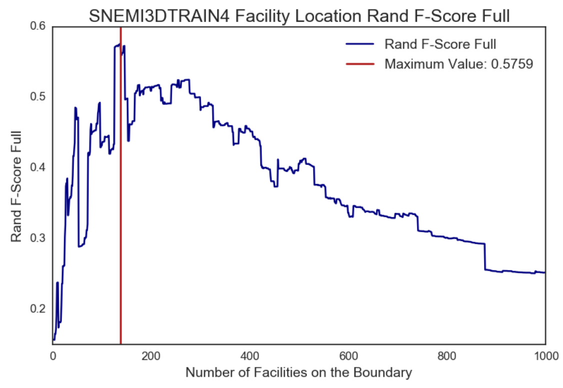

SNEMI3DTEST4 Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.9521
Rand F-Score Full
Voting Threshold

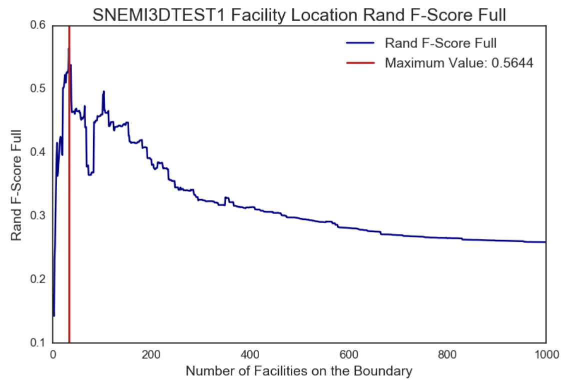

SNEMI3DTRAIN1 K-Median Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.5617
Rand F-Score Full
Number of Medians

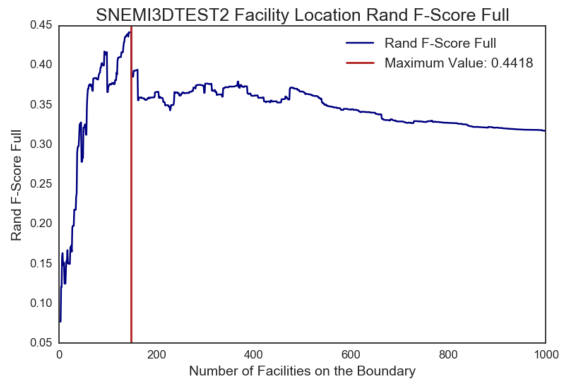

SNEMI3DTRAIN2 K-Median Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.5894
Rand F-Score Full
Number of Medians

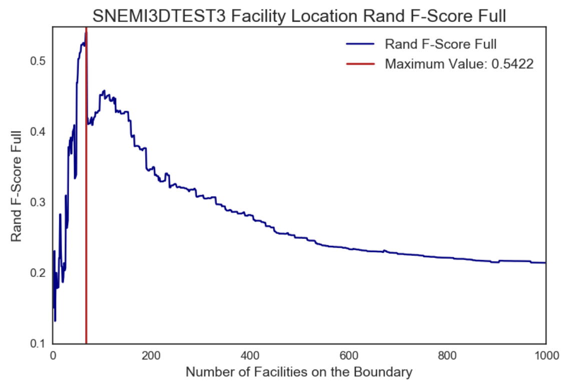

SNEMI3DTRAIN3 K-Median Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.4802
Rand F-Score Full
Number of Medians

SNEMI3DTRAIN4 K-Median Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.3917
Rand F-Score Full
Number of Medians

SNEMI3DTEST1 K-Median Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.3971
Rand F-Score Full
Number of Medians

SNEMI3DTEST2 K-Median Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.4021
Rand F-Score Full
Number of Medians

SNEMI3DTEST3 K-Median Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.3467
Rand F-Score Full
Number of Medians

SNEMI3DTEST4 K-Median Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.3575
Rand F-Score Full
Number of Medians

SNEMI3DTRAIN1 Facility Location Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.6799
Rand F-Score Full
Number of Facilities on the Boundary

SNEMI3DTRAIN2 Facility Location Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.7096
Rand F-Score Full
Number of Facilities on the Boundary

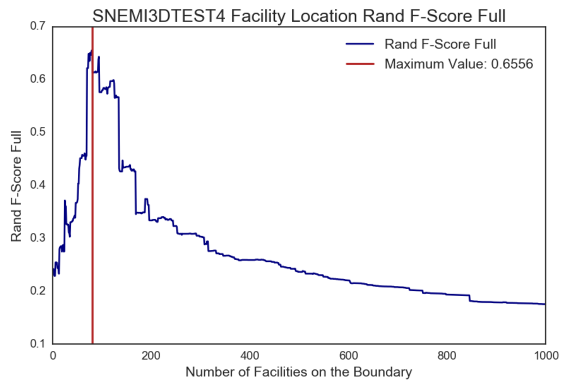

SNEMI3DTRAIN3 Facility Location Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.6000
Rand F-Score Full
Number of Facilities on the Boundary

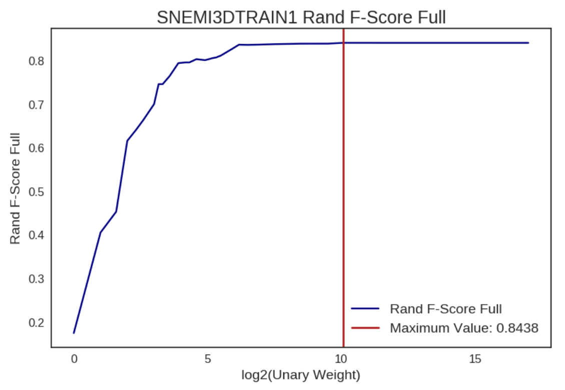

SNEMI3DTRAIN4 Facility Location Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.5759
Rand F-Score Full
Number of Facilities on the Boundary

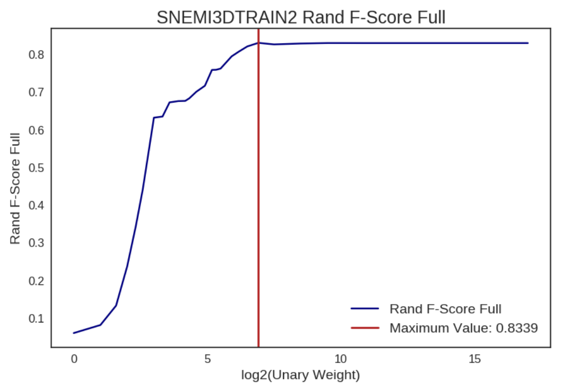

SNEMI3DTEST1 Facility Location Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.5644
Rand F-Score Full
Number of Facilities on the Boundary

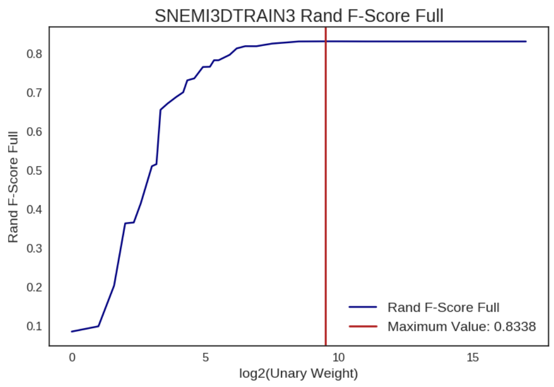

SNEMI3DTEST2 Facility Location Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.4418
Rand F-Score Full
Number of Facilities on the Boundary

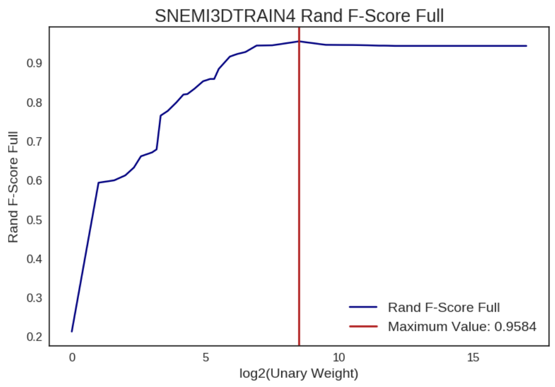

SNEMI3DTEST3 Facility Location Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.5422
Rand F-Score Full
Number of Facilities on the Boundary

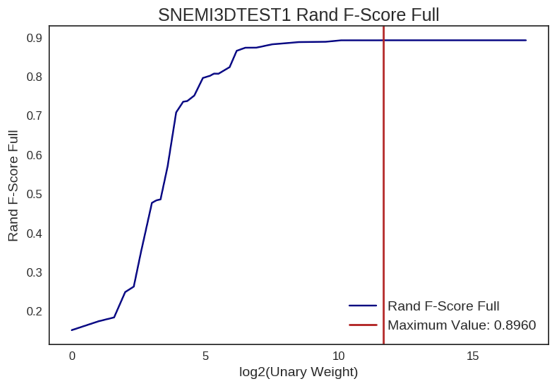

SNEMI3DTEST4 Facility Location Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.6556
Rand F-Score Full
Number of Facilities on the Boundary

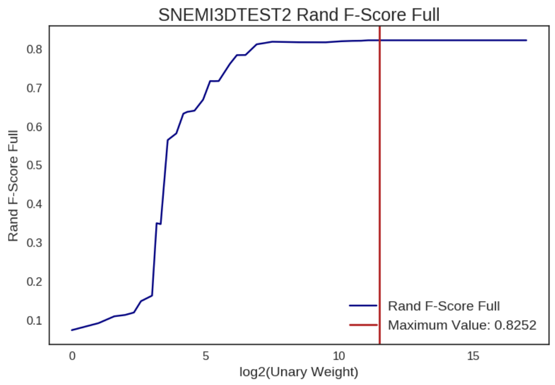

SNEMI3DTRAIN1 Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.8438
Rand F-Score Full
log2(Unary Weight)

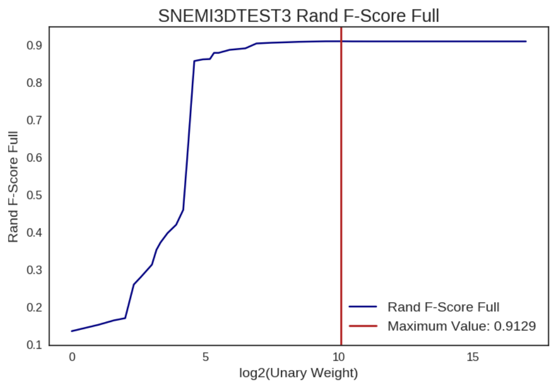

SNEMI3DTRAIN2 Rand F-Score Full
Rand F-Score Full
Maximum Value: 0.8339
Rand F-Score Full
log2(Unary Weight)

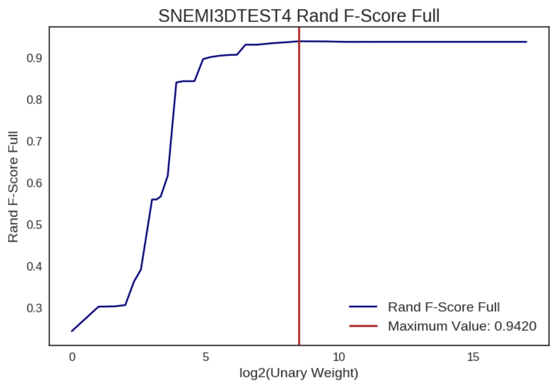

SNEMI3DTRAIN3 Rand F-Score Full
Rand F-Score Full
log2(Unary Weight)
Rand F-Score Full
Maximum Value: 0.8338


SNEMI3DTRAIN4 Rand F-Score Full
Rand F-Score Full
log2(Unary Weight)
Rand F-Score Full
Maximum Value: 0.9584


SNEMI3DTEST1 Rand F-Score Full
Rand F-Score Full
log2(Unary Weight)
Rand F-Score Full
Maximum Value: 0.8960


SNEMI3DTEST2 Rand F-Score Full
Rand F-Score Full
log2(Unary Weight)
Rand F-Score Full
Maximum Value: 0.8252


SNEMI3DTEST3 Rand F-Score Full
Rand F-Score Full
log2(Unary Weight)
Rand F-Score Full
Maximum Value: 0.9129


SNEMI3DTEST4 Rand F-Score Full
Rand F-Score Full
log2(Unary Weight)
Rand F-Score Full
Maximum Value: 0.9420

Bibliography

[1] Frederico AC Azevedo, Ludmila RB Carvalho, Lea T Grinberg, José Marcelo Farfel, Renata EL Ferretti, Renata EP Leite, Roberto Lent, Suzana Herculano-Houzel, et al. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541, 2009.

[2] John A Bogovic, Gary B Huang, and Viren Jain. Learned versus hand-designed feature representations for 3d agglomeration. *arXiv preprint arXiv:1312.6159*, 2013.

[3] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient nd image segmentation. *International journal of computer vision*, 70(2):109–131, 2006.

[4] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[5] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[6] Dan Ciresan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pages 2843–2851, 2012.

[7] Neal Donnelly. Connecting the Connectome: Automating segmentation of neurons from tissue imaging. Princeton University Senior Thesis, 2014. Undergraduate Thesis.

[8] AL Eberle, S Mikula, R Schalek, J Lichtman, ML KNOTHE TATE, and D Zeidler. High-resolution, high-throughput imaging with a multibeam scanning electron microscope. *Journal of microscopy*, 259(2):114–120, 2015.

[9] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *Knowledge and data engineering, ieee transactions on*, 19(3):355–369, 2007.

[10] Jan Funke, Bjoern Andres, Fred A Hamprecht, Albert Cardona, and Matthew Cook. Efficient automatic 3d-reconstruction of branching neurons from em data. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1004–1011. IEEE, 2012.

[11] Kenneth J Hayworth, C Shan Xu, Zhiyuan Lu, Graham W Knott, Richard D Fetter, Juan Carlos Tapia, Jeff W Lichtman, and Harald F Hess. Ultrastructurally smooth thick partitioning and volume stitching for large-scale connectomics. *Nature methods*, 12(4):319–322, 2015.

[12] Suzana Herculano-Houzel, Bruno Mota, and Roberto Lent. Cellular scaling rules for rodent brains. *Proceedings of the National Academy of Sciences*, 103(32):12138–12143, 2006.

[13] Yunfeng Hua, Philip Laserstein, and Moritz Helmstaedter. Large-volume en-bloc staining for electron microscopy-based connectomics. *Nature communications*, 6, 2015.

[14] Viren Jain, Srinivas C. Turaga, K Briggman, Moritz N. Helmstaedter, Winfried Denk, and H. S. Seung. Learning to agglomerate superpixel hierarchies. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 648–656. Curran Associates, Inc., 2011.

[15] Elizabeth Jurrus, Antonio RC Paiva, Shigeki Watanabe, James R Anderson, Bryan W Jones, Ross T Whitaker, Erik M Jorgensen, Robert E Marc, and Tolga Tasdizen. Detection of neuron membranes in electron microscopy images using a serial neural network architecture. *Medical image analysis*, 14(6):770–783, 2010.

[16] Narayanan Kasthuri, Kenneth Jeffrey Hayworth, Daniel Raimund Berger, Richard Lee Schalek, José Angel Conchello, Seymour Knowles-Barley, Dongil Lee, Amelio Vázquez-Reina, Verena Kaynig, Thouis Raymond Jones, et al. Saturated reconstruction of a volume of neocortex. *Cell*, 162(3):648–661, 2015.

[17] Verena Kaynig, Thomas Fuchs, and Joachim M Buhmann. Neuron geometry extraction by perceptual grouping in sstem images. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2902–2909. IEEE, 2010.

[18] Verena Kaynig, Amelio Vazquez-Reina, Seymour Knowles-Barley, Mike Roberts, Thouis R. Jones, Narayanan Kasthuri, Eric Miller, Jeff Lichtman, and Hanspeter Pfister. Large-scale automatic reconstruction of neuronal processes from electron microscopy images. *Medical Image Analysis*, 22(1):77–88, 2015 2015.

[19] Seymour Knowles-Barley, Nancy J Butcher, Ian A Meinertzhagen, and J Douglas Armstrong. Biologically inspired em image alignment and neural reconstruction. *Bioinformatics*, 27(16):2216–2223, 2011.

[20] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.

[21] Masaaki Kuwajima, Josef Spacek, and Kristen M Harris. Beyond counts and shapes: studying pathology of dendritic spines in the context of the surrounding neuropil through serial section electron microscopy. *Neuroscience*, 251:75–89, 2013.

[22] Kisuk Lee, Aleksandar Zlateski, Vishwanathan Ashwin, and H. Sebastian Seung. Recursive training of 2d-3d convolutional networks for neuronal boundary prediction. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3573–3581. Curran Associates, Inc., 2015.

[23] Juan Nunez-Iglesias, Ryan Kennedy, Toufiq Parag, Jianbo Shi, and Dmitri B Chklovskii. Machine learning of hierarchical clustering to segment 2d and 3d images. *PloS one*, 8(8):e71715, 2013.

[24] Peter Penzes, Michael E Cahill, Kelly A Jones, Jon-Eric VanLeeuwen, and Kevin M Woolfrey. Dendritic spine pathology in neuropsychiatric disorders. *Nature neuroscience*, 14(3):285–293, 2011.

[25] HS Seung and L Burnes. Eyewire. *Available a t eyewire. org*, 2012.

[26] Claude E Shannon and Warren Weaver. The mathematical theory of communication. 1949. *Urbana, Univ. Illinois Press.*

[27] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

[28] Shin-ya Takemura, Arjun Bharioke, Zhiyuan Lu, Aljoscha Nern, Shiv Vitaladevuni, Patricia K Rivlin, William T Katz, Donald J Olbris, Stephen M Plaza, Philip Winston, et al. A visual motion detection circuit suggested by drosophila connectomics. *Nature*, 500(7461):175–181, 2013.

[29] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):583–598, 1991.

[30] JG White, E Southgate, JN Thomson, and S Brenner. The structure of the nervous system of the nematode caenorhabditis elegans: the mind of a worm. *Phil. Trans. R. Soc. Lond*, 314:1–340, 1986.

[31] Aleksandar Zlateski, Kisuk Lee, and H Sebastian Seung. Znn-a fast and scalable algorithm for training 3d convolutional networks on multi-core and many-core shared memory machines. *arXiv preprint arXiv:1510.06706*, 2015.