3D RECONSTRUCTION OF BRAIN TISSUE

Hylke Buisman[†]*, Manuel Gomez-Rodriguez*[‡]*, Saeed Hassanpour*[‡]

{hbuisman, manuelgr, saeedhp}@stanford.edu
[†]Department of Computer Science, [‡]Department of Electrical Engineering
Stanford University

ABSTRACT

We consider the problem of identifying neuron nuclei in a stack of 2D images of brain tissue of a rat in order to construct a 3D model of the neuron bodies. Our approach consists of several a combination of image processing techniques, machine learning algorithms and 3D rendering methods. Initially, edge detection (Prewitt filter), non linear diffusion, morphological opening and black and white conversion are used to highlight the neuron bodies in the original 2D images. In the next phase, a supervised learning algorithm is used to decide what part of the 2D images are neuron bodies and to remove the noise. Evaluation of the algorithms showed that SVM and Logistic regression worked particularly well. The performance of both algorithms is analyzed. Finally, a 3D model is generated using VTK.

Index Terms— Machine Learning, 3D reconstruction, brain tissue, neuroscience

1. INTRODUCTION

Recent developments in biomedical methods, like Serial Block-Face Scanning Electron Microscopy, make it possible to obtain high-resolution images of brain tissue. In these methods, a cube of brain tissue is cut in thin layers. In this project, rat brain tissue has been used [1].

Our goal is to detect all the neuron nuclei in every layer of the cube of brain tissue in order to build a 3D model of the neuron nuclei located in a cube of brain tissue, as a first step for a 3D automated tracking of neural activity. This builds on previous work in this field [2]. In order to achieve this aim, we apply several supervised learning approaches (logistic regression, SMO, etc.) combined with image processing techniques and VTK.

In a first stage, the layers are processed in order to make the nuclei more distinctly visible. For each layer, the algorithm outputs a preprocessed image and a mask for every neuron candidate. Thereafter, the relevant features of each nucleus candidate are extracted from the images using the masks, and every nucleus candidate in the training set is labeled using a handy labeling tool. In the next step, the machine learning algorithms are trained on this data and we analyze the performance of every algorithm using cross-validation.

Finally, a 3D model of neuron nuclei is built using the results from these steps. The Visualization Toolkit (VTK), that uses OpenGL, has been chosen to create the 3D model since it significantly speeds up the rendering and the real-time visualization.

This paper is organized in the following way. Section 2 is devoted to image processing techniques used in the preprocessing of the brain tissue layers. Section 3 elaborates on the definition and justification of the machine learning features. Section 4 describes the machine learning algorithms chosen to solve the classification problem and presents some quantitative and graphical performance results. Section 5 elaborates on the 3D rendering and visualization process. Finally, conclusions are discussed in section 6.

2. PREPROCESSING

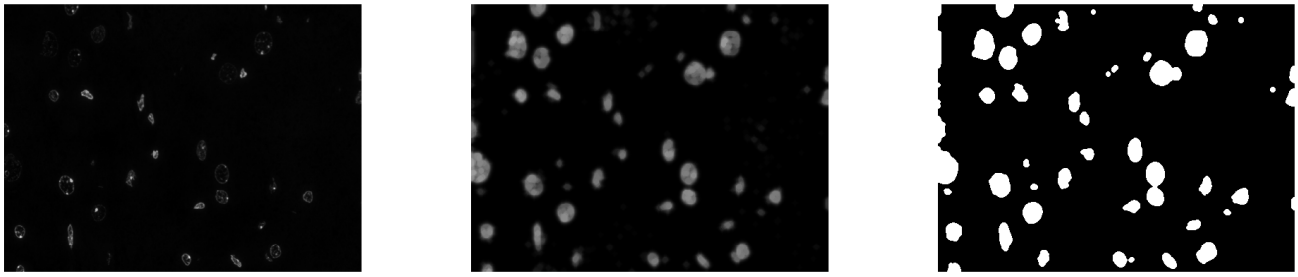
The preprocessing section of the algorithm is used to transform the rather unstructured input images to a form where it is possible to extract relevant features. The output of this section is a preprocessed image, and a list of masks indicating where in the image nuclei are expected to be, in other words a list of *nucleus candidates* is returned.

To achieve this, several image processing steps are applied to the image. It is important to note that each of these steps is tuned in such a way that the preprocessing will output too many neuron candidates. This minimizes the amount of false negatives[1] at the cost of including more false positives. By doing this, the classification process is shifted from the image (pre)processing section to the machine learning section.

2.1. Edge detection

The first step is edge detection: the edges of the nuclei are fairly distinct in the input images, and their insides are of irregular consistency. As a result, significantly more edges are found at the edges of the nuclei and in their insides, than in between the nuclei. Based on experimentation it turned out the Prewitt edge detection works best for this application.

[1] A false negative in this case, means that a part of the image that contains a neuron is not present in the list of neuron candidates.



(a) Original image (b) After the opening (c) Black and white

Fig. 1. Preprocessing steps

2.2. Non-linear diffusion

The edge image roughly indicates where a nucleus is likely to be, but on the wrong scale. A useful output would be a black and white image that indicates areas where nuclei are likely to be. To accomplish this, something similar to a blurring is needed. However, applying a gaussian blur, would result in losing information regarding the borders of the nucleus. For this reason nonlinear diffusion is applied: it preserves the edges, but also blurs edges inside the nuclei. The resulting image is a more reasonable representation of what is likely to be part of a nucleus, and what is not.

A positive side-effect, is that nonlinear diffusion is also known to reduce noise [3]. In this case that means that the resulting image will lose some of its (fine-grained) noise, which makes it easier to work with.

2.3. Morphological opening

At this point the image is still not ready for conversion to a black and white image. Thresholding at this stage would result in some nuclei growing together and there would still be noise present, which would unnecessarily slow down the rest of the algorithm. To resolve this, a morphological opening is applied.

A morphological opening is the combination of applying an *erosion*, and then a *dilation* (also known as the Minkowski subtraction and addition). An opening of set $\mathcal{A}$ with *structuring element* $\mathcal{B}$ is denoted as:

$$\mathcal{A} \circ \mathcal{B} = (\mathcal{A} \ominus \mathcal{B}) \oplus \mathcal{B}$$

where $\ominus$ and $\oplus$ are the erosion and dilation operators. The erosion of two sets can be interpreted as all points where the second set (the structuring element) can fit in the first set (based on some center in B). The dilation is the set containing all additions of points in both sets. Applying the opening operation has the effect of removing all points that are smaller or not of similar shape as the structuring element.

The structuring element was chosen to be a small disc (with radius 4), such that small noise and forms that strongly deviate from a round form (such as lines in the image) disappear.

2.4. Thresholding

Finally the grayscale image can be thresholded to a black and white image. To ensure that the values of the image are between zero and one, the image is first normalized using contrast stretching (in our case this is achieved by dividing it by the maximum value). The threshold is then set to a very low value, to prevent increasing the number of false negatives. The value of the threshold was chosen experimentally.

An overview of the preprocessing steps can be found in Figure 1

3. SELECTION AND EXTRACTION OF FEATURES

In the final output of the preprocessing step, all contiguous areas of white pixels are extracted from the image. Each of these is considered to be a nucleus candidate. For the learning task of classifying these candidates as being an actual nucleus or not, a set of relevant features needs to be selected. These features should provide sufficient information to distinguish noise from actual nuclei.

After analyzing the set of unprocessed and preprocessed images, the following features were selected:

1. Roundness
2. Density
3. Relative density with respect to contiguous layers
4. Area

In the following paragraphs, these features will be motivated and defined.

3.1. Roundness

For this application, the roundness of a nucleus n in a layer l, $r_l(n)$, is defined as

$$r_l(n) = \frac{1}{k}\sum_{i=1}^{k}(d_i - E(d))^2; \quad (1)$$

$$E(d) = \frac{1}{k}\sum_{i=1}^{k}\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} \quad (2)$$

$$d_i = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}; \quad (3)$$

where $\{(x_i, y_i)\}_{i=1}^{k}$ is the set of coordinates of the pixels that belongs to a neuron nucleus n in a layer l and (x_c, y_c) is the center of n in l. A pixel (x_i, y_i) belongs to a neuron nucleus n in a layer l if the value of (x_i, y_i) in the mask l is 1.

Note that $r_l(n)$ will be small for a set of points, for which the variance of the distance between each point and the centroid is small. $r_l(n) = 0$ would mean that n is a disk.

The choice for roundness as a feature is justified by the relatively frequent occurrence of round nuclei. In addition noise tends to be non-round of nature (like lines in the image). These observations make roundness a relevant feature.

3.2. Intensity

The intensity of a neuron nucleus n in a layer l, $D_l(n)$, is defined as

$$D_l(n) = \frac{1}{k}\sum_{i=1}^{k} d_{(x_i,y_i)} \quad (4)$$

where $\{d_{(x_i,y_i)}\}_{i=1}^{k}$ is the set of intensity values of the pixels $\{(x_i, y_i)\}_{i=1}^{k}$ in the preprocessed layer l. Again, A pixel (x_i, y_i) belongs to a neuron nucleus n in a layer l if the value of (x_i, y_i) in the mask l is 1.

$D(n)$ has a lower value for neuron nuclei than for the background of the preprocessed images, being a relevant feature to distinguish neuron nuclei from noise.

3.3. Relative intensity

We define the previous relative intensity, $RD_p(n)$, and next relative intensity, $RD_n(n)$, of a neuron nucleus n in a layer l as

$$RD_p(n) = |\frac{1}{k}\sum_{i=1}^{k} d_p(x_i, y_i) - D_l(n)| \quad (5)$$

$$RD_n(n) = |\frac{1}{k}\sum_{i=1}^{k} d_u(x_i, y_i) - D_l(n)| \quad (6)$$

where $D_l(n)$ is the intensity of the neuron nucleus n in the current layer l, $\{d_p(x_i, y_i)\}_{i=1}^{k}$ is the set of intensity values in the preprocessed layer $l-1$ (previous layer) of the pixels that belong to the neuron nucleus n in the current layer l, and $\{d_n(x_i, y_i)\}_{i=1}^{k}$ is the set of intensity values in the preprocessed layer $l+1$ (next layer) of the pixels that belong to the neuron nucleus n in the current layer l.

Both values have to be close to 0 for the set of pixels of a neuron nucleus because the intensity difference between contiguous layers is not very big. On the other hand, in case of having noise, the intensity of the set of pixels can be completely different between contiguous layers. Then, the previous and next relative intensity can be a relevant feature to classify neuron nuclei. This feature is expected to be useful in modeling the continuity between images.

3.4. Area

The area of a neuron nucleus n in a layer l, A_l, is defined as

$$A_l(n) = k \quad (7)$$

where k is the number of pixels that belong to a neuron nucleus n in a layer l. As stated before, a pixel (x_i, y_i) belongs to a neuron nucleus n in a layer l if the value of (x_i, y_i) in the mask l is 1.

Intuitively, too big or too small values of areas are good reasons to discard a candidate neuron nucleus.

4. MACHINE LEARNING ALGORITHMS

After building the set of nucleus candidates and extracting a full set of features for each of them, machine learning can now be applied. In order to evaluate the performance of many different algorithms, the *Weka* toolkit [4] was used. This machine learning toolkit provides a large set of ready-to-use machine learning algorithms, which can also be readily integrated within the MATLAB environment.

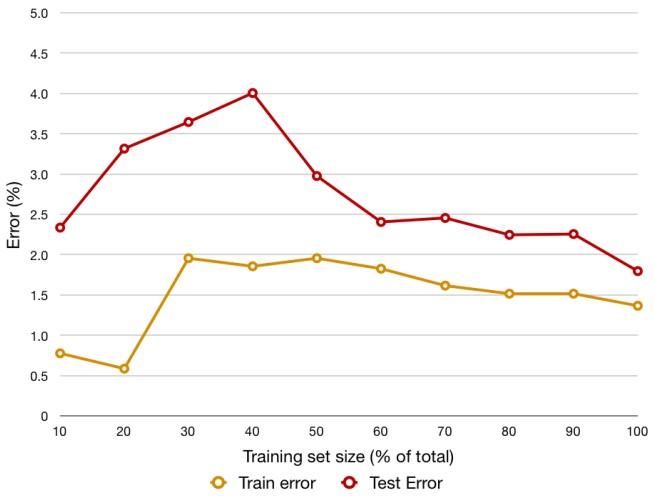
The following methods where applied: decision tree learners, bayesian classifiers, rule based learners, logistic regression and support vector machines.

4.1. Labeling

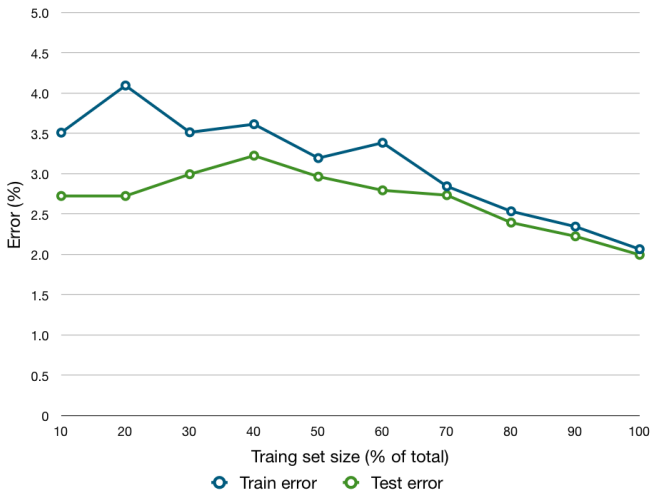
Labeling nucleus candidates for every preprocessed layer to create a training set is a tedious task. In order to accelerate this task we have developed a tool that handily highlights the current, previous and next image. This is done by calculating the convex hull of the mask of every candidate, which is then superimposed on the preprocessed image[2]. In addition to this, labeling judgments were also based on the contents of the raw images within the superimposed mask outline. This method of labeling proved to be more accurate compared to just using the preprocessed images.

Using this method a total of 60 images were labeled, which on average contain 42 nucleus candidates. Thus, a total of about 2500 instances were used for learning.

[2]Using just the preprocessed images for labeling is the method that was described and used for the milestone.



(a) SMO



(b) Logistic

Fig. 2. Supervised learning algorithms performance

4.2. Evaluation

The initial experiments done with WEKA showed that logistic regression (with a ridge estimator) and SMO performed particularly well on this task. Based on this observation, several more experiments were done to find the best choice of parameters. For SMO a polynomial kernel of degree 15 gave best precision. For logistic regression, the ridge parameter was left unchanged, since no significant increase was achieved by changing it. Simple logistic regression performed just as well on the task, however it was significantly slower. In Figure 2 the results of the algorithms can be found. These results where computed with 10-fold cross-validation on the whole data set.

Although the precision of the SMO with degree 15 polynomial kernel performed significantly better, logistic regression can be chosen if time is more important than precision: model building takes 12 seconds for SMO compared to 0.13 seconds for logistic regression. Since logistic regression also outperforms the linear kernel SMO (which exhibits similar speed), logistic regression would be the best choice if time is more important.

To be able to appreciate these results, it is worth mentioning the results using the 'trivial classifier' (sometimes referred to as a zero-rule classifier); this classifier always predicts the majority class. Applying this algorithm gives a precision of 83%. Consequently, the results as shown in Figure 2 show that applying these non-trivial algorithms result in a 15% increase of precision.

Finally, comparing the evolution of the training error and the test error and comparing its values, it is noticeable that SMO has a higher variance. This is an expected result if it is taken into account that a higher order classification space is used in SMO because of having a polynomial kernel.

5. 3D RENDERING

After classifying the nucleus candidates with SMO, and thus removing nearly all further noise, the output is used to generate the 3D model. For the modeling, The Visualization ToolKit (VTK), which is an open source C++ library for 3D computer graphics, was used [5]. VTK uses OpenGL (Open Graphics Library) for basic computer graphics functions and presents higher level functions for 3D visualization and image processing.

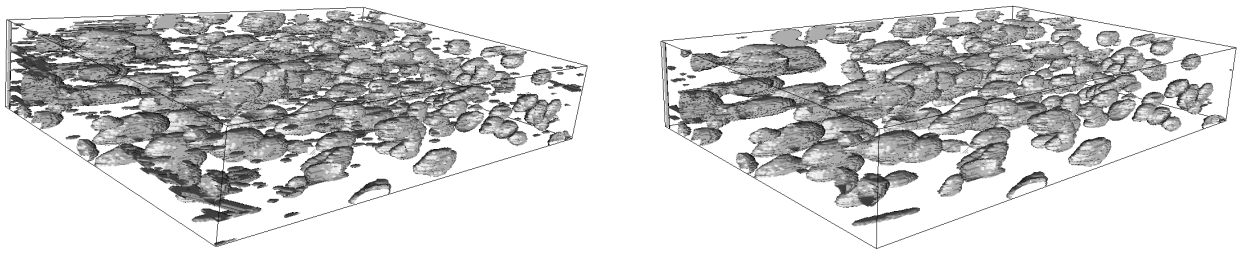
First, an edge detection algorithm is applied, to specify the edges and extract the nuclei surfaces. These surfaces are then covered with polygonal meshes. Finally, the meshes are mapped through the stack to render the 3D model using VTK.

Figure 3 shows two 3D models: one has been generated using the 2D images before applying SMO to reduce the noise and the other was built using the 2D images after applying SMO. These images give a good impression of the improvement the application of machine learning gives: the neuron bodies can easily be distinguished now that the noise has been removed.

6. CONCLUSIONS

The method presented in this paper provides a way for inferring the spatial location and structure of neuron bodies and the spatial relation between them if 2D images of a stack of brain tissue are the only information available [1].

Regarding the overall quantitative performance of the method presented in this paper, machine learning performance has been used as an approximation of it. Based on the quantitative performance analysis presented on the section 4, it is noticeable a significant improvement of the error if a supervised machine learning (approximately 98%) is used instead of using only a 'trivial classifier' (approximately 80%). It is



(a) 3D rendering before applying machine learning

(b) 3D rendering after applying machine learning

Fig. 3. 3D rendering

assumed that labeling using the preprocessed images and the original ones is correct.[3]

It is difficult to measure the accuracy of the preprocessing steps and the 3D model representation. To measure the performance of the preprocessing steps, it would be required to label the data based only on the original images, not the preprocessed ones. This is not manageable, taking into account the vague nature of the original images. To estimate the performance of the 3D model representation, it would be necessary to compare the 3D model with a reference, which is not available.

Focusing on the relationship between test error and amount of training data, the results suggest that increasing the size of the training set, would improve the performance even more. It was not possible to verify the last statement because of the unavailability of more data.

7. ACKNOWLEDGMENTS

The authors thank Assistant Professor Mark Schnitzer for providing us access to the data set of rat brain tissue and Assistant Professor Tom Clandinin for giving us expert information regarding this data set.

8. REFERENCES

[1] K.D. Micheva and S.J. Smith, "Array Tomography: A New Tool for Imaging the Molecular Architecture and Ultrastructure of Neural Circuits," *Neuron*, vol. 55, no. 1, pp. 25–36, 2007.

[2] J. H. Macke, N. Maack, R. Gupta, W. Denk, B. Schlkopf, and A. Borst, "Contour-propagation algorithms for semi-automated reconstruction of neural processes," *Journal of Neuroscience Methods*, vol. Epub ahead, pp. 1–18, 08 2007.

[3] P. Mrazek, "Nonlinear Diffusion for Image Filtering and Monotonicity Enhancement," *Prague: Czech Technical University*, 2001.

[4] I.H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S.J. Cunningham, "Weka: Practical Machine Learning Tools and Techniques with Java Implementations," *ICONIP/ANZIIS/ANNES*, pp. 192–196, 1999.

[5] W. Schroeder, K. Martin, and B. Lorensen, "The Visualization Toolkit An Object-Oriented Approach To 3D Graphics, Kitware," *Inc. publishers*, vol. 2, 2004.

[3] The preprocess section has been designed to maximize the amount of candidate neuron bodies. Then, false negatives are minimized by the preprocess itself and false positives are reduced by the expertise of the labeler