UNIVERSITY OF CALIFORNIA, SAN DIEGO

WebStem: Supervision Tool to Improve Unsupervised Landmark Based Registration of Brainstem Sections

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

in

Computer Science

by

Idan Izhaki

Committee in charge:

Professor Yoav Freund, Chair
Professor Sanjoy Dasgupta
Professor David Kleinfeld

2015

The Thesis of Idan Izhaki is approved and is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2015

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank Professor Yoav Freund for his support as the chair of my committee. His guidance proved to be invaluable to achieve these results.

I would also like to thank Mr. Yuncong Chen, without whom my research would have no doubt be impossible. It is his support that helped me in immeasurable ways, and his help through drafts, inputs and many long nights.

Chapter 2.2 is a summarizations of a paper Mr. Yuncong Chen, a UCSD Computer Science PhD student, prepared for submission and publication of the material, and he is the co-author of this section. The thesis author was the primary investigator and author of this material.

VITA

2005-2007	Information Security, IDF, Israel
2007-2011	Bachelor of Science, Software Engineer in Computer Science, Technion, Haifa, Israel
2010-2013	Software Engineer, Intel, Haifa, Israel
2013-2015	Master of Science, Computer Science, University of California, San Diego
Current	Machine Learning Engineer, Yelp, San Francisco

PUBLICATIONS

"Active tracking and pursuit under different levels of occlusion" Mach. Vis. Appl. 25(1): 173-184 (2014)

ABSTRACT OF THE THESIS

WebStem: Supervision Tool to Improve Unsupervised Landmark Based Registration of Brainstem Sections

by

Idan Izhaki

Master of Science in Computer Science

University of California, San Diego, 2015

Professor Yoav Freund, Chair

Some of the most fundamental tools for research in mouse brain studies are brightfield and fluorescent sections of whole brain sections. The current method of extracting this information, however, requires many hours of human labor. There is a need to automate this process and eliminate or reduce the human interaction aspect. With digital imaging and sectioning becoming more reliable, such a tool can be a ”digital brain atlas”.

Tools, such as the ”digital brain atlas”, also need to be easily compatible with other tools. Outputting or inputting from one tool or another should require as little

human interaction as possible. A brain-wide map annotated with cell types and tract tracing data would allow automatic registration of image stacks to a common coordinate system. Currently, registration of slices requires manual identification of landmarks for every slice.

This thesis builds upon the process introduced by ”Texture landmark detecting in mouse brain images using significance-based boosting”, by Mr. Yuncong Chen. In his paper, he introduced a workflow as a pipeline that analyzed mouse brain textures, segmented them into sections, and classified sections into landmarks using unsupervised or semi-supervised ways. The result of this paper achieves its goal, but there is room for improvement. Since the digital brain atlas is meant to be able to interface with other programs, it needs to obtain high accuracy and reliability.

The paper will explain briefly the concepts and methodologies the unsupervised pipeline implements, will introduce a web-app called WebStem to supervise some of the output the pipeline generates, and will show improvements in accuracy thanks to the supervision in three different ways. The new results will serve as the basis for reliable registration and atlas building.

Keywords: landmark detection, atlas building, mouse brain, registration, automated annotation, triplets, web-app, WebStem, supervise

Introduction

There is a lot of useful data about the mouse brain on the Internet. The format is easy and usable for research purpose that Neuroinformatics placed. The data is mostly images (scanned sections, CT, and MRI results), and in some models, the data is neuron behavior and maps of genes that are ”turned on” in different brain landmarks. This shared data enables researchers to piggyback off of one another's studies and discover more.

Engineers and computer scientists help brain researchers to share and compare data by writing more software and using the technology advancements. For instance, software can now analyze whether MRIs of Alzheimer's patients with different brain sizes and shapes have similar brain features. Does brain structure predispose to bipolar disorder? This question, and many others, may one day be answered by computer programs that re-analyze images of past patients, instead of studying new ones.

One of the main free mice brain atlases on the internet researchers can mine and use is **Allen Mouse Brain Atlas:** with 21,000 genes expressed on the mouse brain. The brains are sliced up by researchers and stained where genes are expressed. Researchers can query photos or 3D models of the brain for their gene of interest [1].

There are other databases for human brain atlases as well, like **Allen Human Cortex Atlas** and **Whole Brain Atlas**, but since the thesis focuses on research purposes only, and deals with mouse brain sections, we will not discuss them any further.

This paper briefly describes a pipeline process for the automatic detection of landmarks in histology images of mouse brain sections based on Yuncong Chen's paper.

Then, it focuses on the inaccuracies the pipeline resulted and how to overcome them. The goal is to facilitate image registration and atlas generation. We aim to create an accurate digital atlas for the mouse brainstem by virtually aligning images of nissl and fluorescent-stained brain sections. The atlas should incorporate cell type, tract tracing and other physiological data.

Most brain atlases are intended to be viewed by human researchers, but the atlas aimed to be built was intended to be used by a computer. This can be done by representing the landmarks in a data-structure which facilitate the alignment of a whole stack of sections to a common coordinate system. The novelty of the thesis described here is the use of mostly unsupervised learning to find regions with distinct texture and clear boundaries, and immensely improve accuracy using supervision, which assures highly accurate results. Our secondary goal is also to use the least man-hours as possible for supervision work, as the man-hours are costly and scarce.

Figure 1 is a collection of typical nissl-stained mouse brain sections images. Although the brainstem does not have salient edges like the cortex, it contains many compact neuron clusters (nuclei) and striated regions (fiber tracts). Both types of structure have distinct texture that can be detected and modeled.

Unsupervised learning can be achieved in many ways, but we would like to make sure we need the least man-hours of labelers, researchers who know how to map the different cells to their landmarks manually, and still bring great accuracy improvement to current output. We decided to use a technique of supervision based on other indirectly related concepts, usually used for embedding. The paper will show three different ways to improve the unsupervised output accuracy, focusing on building an atlas-guided annotation, landmark-based registration and atlas generation into a single iterative and partially interactive framework.

The remaining of this paper is organized as the follows. Section 1 describes the

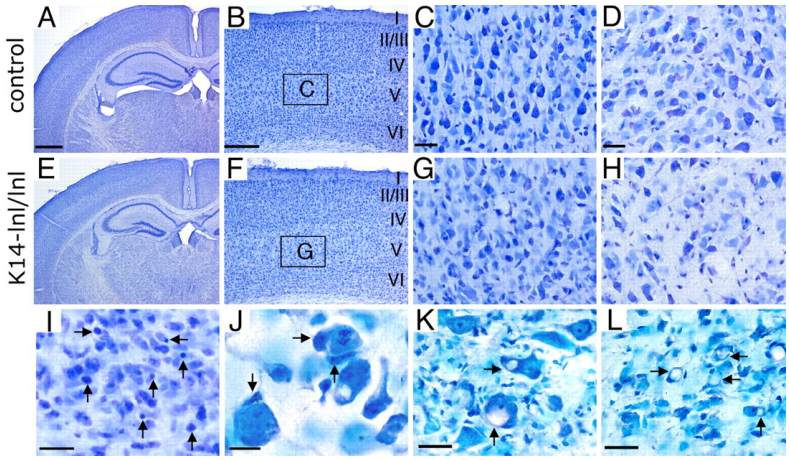

Figure 1. Mouse brainstep sections in different zoom levels. Arrows point the fiber tracts and nuclei that identify thhe types of cell, and can be analyzed.

literature review needed to enable the supervised and unsupervised learning. Section 2 describes the pipeline to generate the atlas eventually, starting from using Gabor filters, through segmenting the different regions, and mapping the regions across sections, to the evolution of supervised learning, end results and impact. Notice that each stage on section 2 outputs data as an input to the next stage. Section 3 describes the experiments made and results. Section 4 is our conclusion and future work.

Chapter 1

Literature Review

This thesis explores the semi-supervised detection of different regions and types of cells, as ”landmarks” in the brainstem, done using histology images of mouse brain sections. In order to create an accurate digital atlas of the different landmarks extracted from these sections, it requires researching in the fields of computer vision and machine learning. Considering the natural transition of textures in mouse brain stem, and the fact that the image are not completely aligned, traditional segmentation using a Gabor filters (as a pyramid of kernels) is not enough for classification of the different landmarks, and requires more complicated estimations using distance function. Furthermore, some cases are yet too difficult to accurately train, which requires supervision. Since the man-hours for supervision is precious, we would like to explore ways to minimize the amount of work needed for supervision.

1.1 Gabor Filter

In computer vision world, we can define a texture pattern as a vector of properties (e.g., colors, contours, tiling, corners geometry), similarly to asking a child to describe a texture. There are many ways to extract data into a properties vector for such a patch of an image. One way is using Gaussian Markov random fields [2], and relying on sufficient statistics [3] notion to estimate these parameters. Another common way is

using Gabor filters bank instead of spatial Gaussian ones, constructing vectors of features by classifying the response of an image to different Gabor kernels with different wavelets parameters. That method is also called a steerable pyramid of texture extraction (using Gabor filters).

One of the papers analyzing Gabor filters [4] explores the usage of this filter, and how well each kernel convolution responds to a pattern. A two dimensional Gabor function $g(x,y)$ and its Fourier transform $G(u,v)$ can be written as follows:

$$g(x,y;\sigma_x,\sigma_y,\mu_0,\phi) = cos(2\pi\mu_0 x+\phi)\cdot exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)\right]$$

where σ_x,σ_y are the standard deviation of the Gaussian envelope (defining the width of wavelets), and μ_0 defines the sinusodial factor. ϕ defines the phase offset (to shift the wave, usually set to 0). x and y are the pixel coordinates of a predefined patch.

$$G(u,v) = 2\pi\sigma_x\sigma_y\cdot\left(exp\left\{-\frac{1}{2}\left[\frac{(u-u_0)^2}{\sigma_u^2}+\frac{v^2}{\sigma_v^2}\right]\right\}+exp\left\{-\frac{1}{2}\left[\frac{(u+u_0)^2}{\sigma_u^2}+\frac{v^2}{\sigma_v^2}\right]\right\}\right)$$

where $\sigma_u = 1/2\pi\sigma_x$ and $\sigma_v = 1/2\pi\sigma_y$. The values of $g(x,y)$ and $G(u,v)$ depend on all of these parameters, but the most relevant ones are σ - the ”width”, or frequency, of the wavelet and $\theta = 2\pi\mu_0 x$ - the directionality of the wavelet. Theoretically, by applying a large amount of different different kernels on an patch of image, into a 1-dimensional vector, we will get a unique representation for each texture. Using clustering algorithms like K-Means would enable us to learn the different families of textures, and is a tool for texture analysis and segmentation.

Figure 1.1 is an example of a single wavelet rotated in 15% and 10 pixels wavelet-wide, on a 50 pixels window size. The wider the window gets, the weaker the wavelets gradually become. The results (or response) image is simply convolution of the image

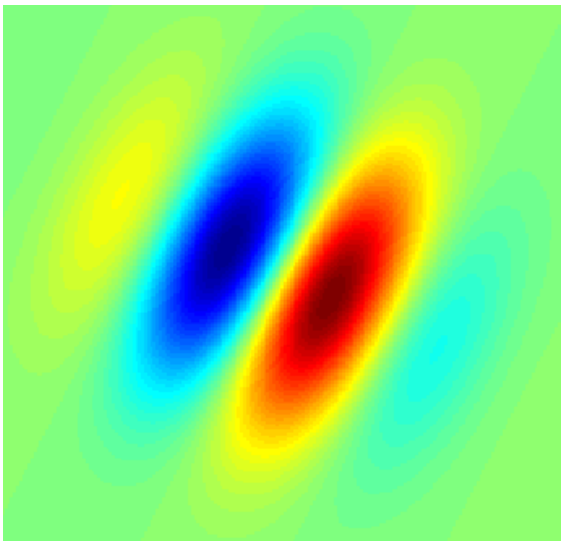

Figure 1.1. A two-dimensional Gabor kernel ($\theta = 15$, $\sigma_x = \sigma_y = 10px$)

with the Gabor kernel. This type of convolution reacts strongly to similar intensity patterns in the image patch. On this example, there will be strongest response on a repeating long straight lines tilted in 15 and are $10px$ wide pattern, and weaker on any other pattern.

Figure 1.2 demonstrates the kernel reaction to different parameters values (filter bank), as different angles θ, widths σ, offsets from centers x, y, and more. As Fitzgibbon's paper [5] focuses only on learning patterns using Gabor kernels, we know that it is highly common to use at most of these parameters as variables in order to learn the different patterns, and usually the rotation and width parameters are the main game changers.

Paper [6] tries to distinguish and find the correlation between different patterns. Figure 1.3 is an example for 9 different kernel responses; we expect to observe different responses to each one. Figure 1.4 exemplifies the mixture of these different patterns and how the kernel responds to each region.

Since the response of a single kernel is not enough to distinguish between patterns, it is necessary to process the outputs into a single data-structure as mentioned before, and

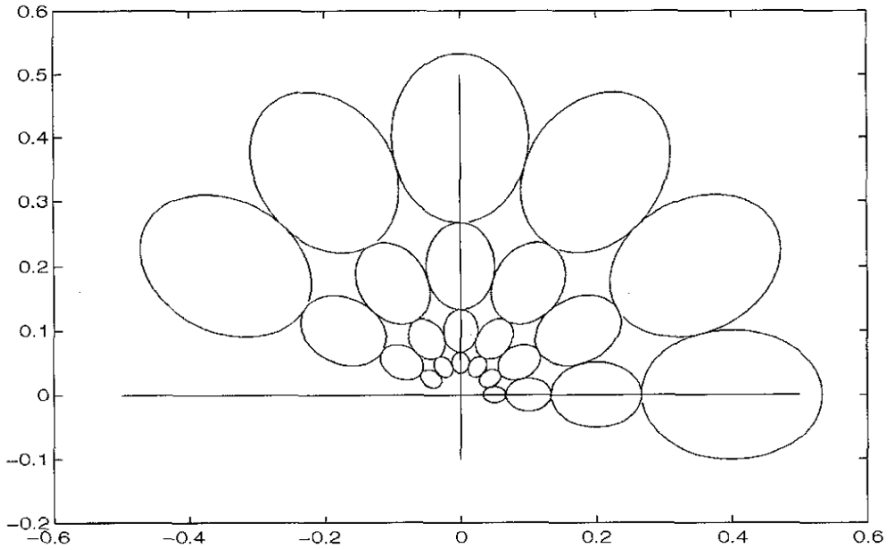

Figure 1.2. The contours indicate the half-peak magnitude of the filter responses in the Gabor filter dictionary.

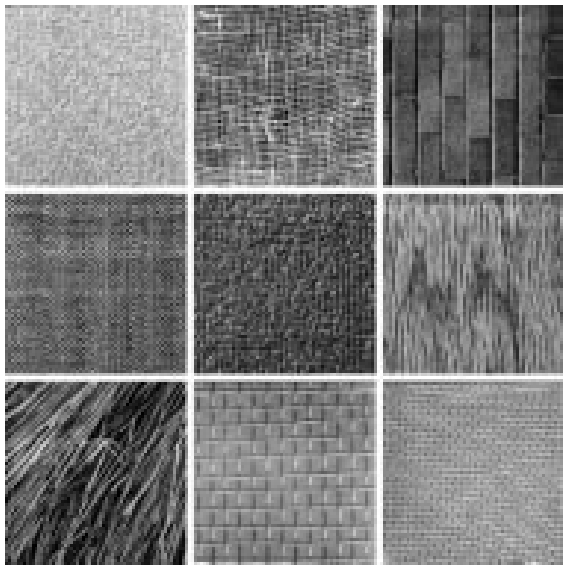

Figure 1.3. Nine different patterns for Gabor filters testing.

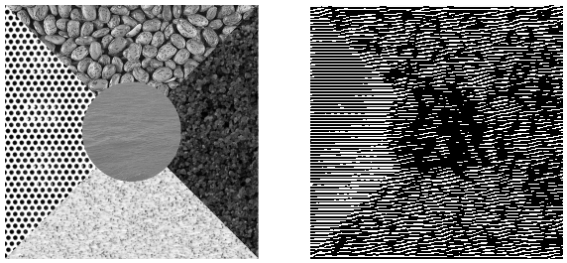

Figure 1.4. Before and after response to a single Gabor kernel application.

use a classifier or some clustering algorithms to really differentiate between them. The output data-structure is simply a features vector generated by ”the steerable pyramid” technique, and can be reduces into a smaller vector we will call a ”textons” histogram.

Much of the any other work on texture analysis and synthesis can be classified according to what type of texture model is used. Some of the successful texture models include reaction diffusion [7, 8], frequency domain [9], fractal [10, 11], and statistical/random field [12, 13, 2, 14, 15, 16, 17, 18] models. Some (e.g., [14]) have used hybrid models that include a deterministic (or periodic) component and a stochastic component. Surprisingly enough, despite of all this work, the principle sources of texture maps in computer graphics are still scanned images and hand-drawn textures.

1.2 SLIC Segmentation

”Superpixel algorithms group pixels into perceptually meaningful atomic regions which can be used to replace the rigid structure of the pixel grid...” [19] (figure 1.5). It provides a convenient simple methodology to reduce complexity of consequent computations by capturing features in some non-rigid grid. Their simplicity made them a key building block of many computer vision algorithms, such as top scoring multiclass object segmentation entries to the PASCAL VOC Challenge.

They capture image redundancy, provide a convenient primitive from which to compute image features, and greatly reduce the complexity of subsequent image

Figure 1.5. SLIC segments Images into superpixels of size 64, 256, and 1,024 pixels (approximately).

processing tasks. They have become key building blocks of many computer vision algorithms, such as top scoring multiclass object segmentation entries to the PASCAL VOC Challenge [20, 21, 22], depth estimation [23], segmentation [24], body model estimation [25], and object localization [20].

The paper "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods" [19] tries to propose a new method for generating superpixels. The method should be faster than any other method, more memory efficient, gives cleaner well defined boundaries, and should improve the performance of any existing segmentation algorithm when combined.

By defining the color data of a pixel in RGB space as $[r_k, g_, b_k]$, the physical distance between cluster centers as Euclidean distance $d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_k)^2}$, and the distance between colors as $d_s = \sqrt{(r_j - r_i)^2 + (g_j - g_i)^2 + (b_j - b_i)^2}$, s.t. the normalized distance function is $D = \sqrt{(d_c/N_c)^2 + (d_s/N_s)^2}$ (N_c and N_s are color and spatial proximities). The algorithm formula is defined as follows:

/* Initialization */

* Initialize cluster centers $C_k = [r_k, g_k, b_k, x_k, yk]^T$

```
  by sampling pixels at regular grid steps S.
* Move cluster centers to the lowest gradient position
  in a 3×3 neighborhood.
* Set label l(i) = −1 for each pixel i.
* Set distance d(i) = ∞ for each pixel i.

* repeat
  /* Assignment */
  for each cluster center C_k do
    for each pixel i in a 2S×2S region around C_k do
          Compute the distance D between C_k and i.
          if D < d(i) then
            set d(i) = D
            set l(i) = k
          end if
        end for
  end for
  /* Update */
  Compute new cluster centers.
  Compute residual error E.
until E ≤ threshold
```

Figure 1.6 compares various methods segmentation results.

SLIC algorithm is important for us as the unsupervised pipeline uses it to extract superpixels, collect them into boundaries (different groups of cell on each sections). These boundaries are then mapped to landmarks. The ”WebStem” web-app will use these

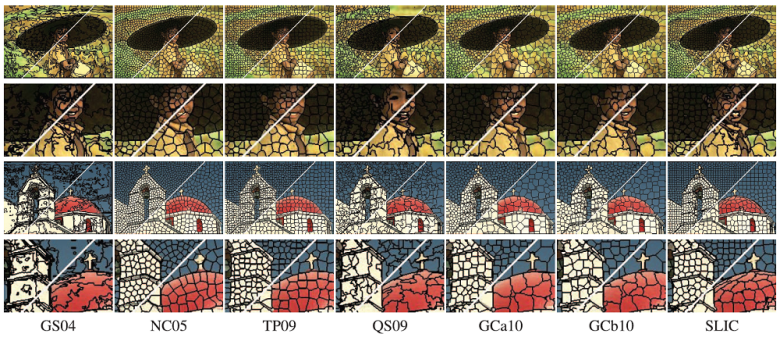

Figure 1.6. Visual comparison of superpixels produced by various methods. The average superpixel size in the upper left of each image is 100 pixels and is 300 in the lower right. Alternating rows show each segmented image followed by a detail of the center of each image.

superpixels and let the labeler to distinguish between different landmarks superpixels. Successful distinction would mean a different pattern, which we hope to be able to capture and improve classification accuracy that way.

1.3 Classifiers

Classifiers are extremely useful in artificial intelligence world as they do not require manual labeling, and unlike segmentation, can be tuned to care only about regions that are significant (similarly to anomaly detection).

Famous tools that can be wrapped, modified and used as unsupervised or semi-supervised classifiers are K-Means, Fuzzy-C-Means, Adaptive K-Means, SVM, Perceptron, AdaBoost and many more. Each algorithm has its own weaknesses and strengths, as sensitivity to noisy data and outliers, performance, consistency, etc.

The unsupervised pipeline uses K-Means, Perceptron and experimented with AdaBoost to classify the different textures. The analysis of the supervised data will be done using SVM (Support Vector Machine), which computes a hyperplane in $\mathbb{R}^n$ in a relatively high accuracy, and will enable us to capture the different patterns in textons histogram the professional human eye was able to capture.

1.4 Triplets for Supervised Classification

There is a growing interest in collecting human similarity comparisons of the form Is a more similar to b than to c?. Such comparisons can provide constraints of the form $d(a,b) < d(a,c)$, where $d(x,y)$ is some distance function between x and y (figure 1.7). By collecting these constraints as triplets, and collecting them from human labelers, researchers can learn the structure of data inputs without real analysis of them. For example, the authors of McFee paper [26] learned music genres from triplet comparisons themselves with no other annotations. Human similarity comparisons are useful in computer vision for creating perceptually-based embeddings - capturing a lower dimension information about data without really analyzing the object itself, but mostly by getting feedback from a labeler. In Agarwal paper [27], the authors created a two dimensional embedding. One axis represented brightness and the other axis represented glossiness of objects. On [28] the paper focuses on creating perceptual embeddings from images of food.

Even though embedding is not important for improving the accuracy of the unsupervised classifier from the pipeline code, we are interested in the concept of Is a more similar to b or c? to collect the data from the labeler. On that paper, they improved the triplets technique and asked labelers Mark k images that are most similar to a reference image, show on figure 1.8. Displaying the images as a grid of size n, a human can generate $k \cdot (n-k)$ triplets per task, or "a page". This kind of query allows researchers to collect more triplets on every screen. It also allows labelers to avoid having to wait for multiple screens to load, especially in cases where one or more of the images in the queried triplets does not change. This concept also allows labelers to benefit from the parallelism in the low-level human visual system [29].

Since many of these observations involve man-hours, the right way of measuring

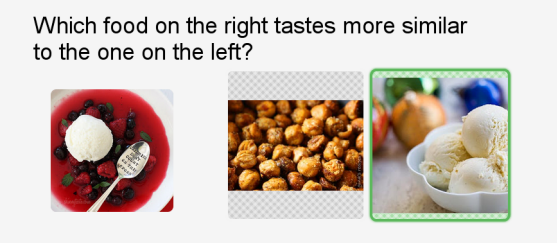

Figure 1.7. Basic concept: match one out of two pictures to a single reference.

Figure 1.8. Advanced concept: match all pictures who to a single or more references.

supervision quality is with respect to human cost rather than accuracy alone. The human cost is related to the time it takes labelers to complete a task, and the pay rate of a completed task. Some authors [30, 31] already used these ideas in their work but did not quantify the improvement.

The results for paper [28] experiment were as described on figure 1.9. That figure explains that the data given from a labeler with the new enhancement does impact his accuracy, but normalizing by the time and money saved by doing it is more profitable than asking the labeler to match many triplets alone.

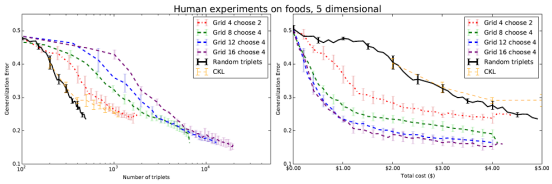

Figure 1.9. Results of human experiments on some food dataset. Left graph: Triplet generalization error when viewed with respect to the total number of triplets. Right: The same metric when viewed with respect to the total cost of constructing each embedding. The left graph implies that a randomly-sampled embedding appears to converge faster. However, when quality is viewed with respect to cost, they found that an embedding generated using a 16-choose-4 grid cost 0.75, while an embedding with random triplets of similar quality costs 5.00. It is clear that the grid UI saves them money; in this case, by over a factor of 6.

Chapter 2

Approach

2.1 Overview

The goal of the thesis is to describe a method for generating a brain atlas for scientific purposes by automatic detection of landmarks in histology of mouse brain sections, and improve accuracy using supervision. Since the input images of nissl and fluorescent-stained brain sections are not aligned (can be manually aligned), and there are partial tears and missing sections, we expect the pipeline not to rely on unnecessary assumptions of the brain structure and sections alignment.

The digital atlas of the mouse brainstem should incorporate cell types, tract tracing and other physiological data. The atlas is made to be used by a computer rather than be viewed by a person, meaning to represent all landmarks in a data-structure that algorithms can use to automatically align a whole stack to a common coordinates system.

The novelty of the work described here is the smart and efficient use of supervised learning correct mismatched of boundaries to landmarks made by the unsupervised pipeline algorithms. A smart use of supervision results a better accuracy with minimum man-hours of the labelers. This will also reduce the manual work needed to identify reliable landmarks.

Figure 2.1 is a typical image of nissl-stained mouse brainstem section. The

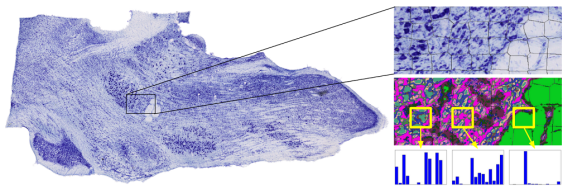

Figure 2.1. Left: A typical image of mouse brainstem section. Right top: part of the original image with superpixels overlaid. Middle: texton map of the same area. Bottom: texton histograms of three superpixels with different textures.

brainstem does not have salient edges like the cortex, but it contains many compact neuron clusters (nuclei) and striated regions (fiber tracts). Both types have distinct textures that can be detected and modeled.

The unsupervised learning first stage is to create a library of landmark detectors based on texture and shapes. The website development, or "WebStem" web-app, receives a feedback from labelers to improve the unsupervised learning. They can be applied to identify landmarks from new images or updated ones by incorporating new data. This is our main focus and goal; integrating atlas-guided annotation, landmark-based registration and atlas generation into a single iterative framework with a smart interactive supervision system.

This section on the paper is organized the follows. Section 2.1 mentions the related work for the unsupervised pipeline, as a background to the algorithms there. Section 2.2 describes the full unsupervised pipeline flow from brainstem input images data-set to an atlas data-structure with all landmarks and their relevant data. Each subsection is a stage in the pipeline, starting from processing the images using Gabor filters, computing the superpixels, computing the segment regions (boundaries), and map them to landmarks across sections. Section 2.4 uses supervision techniques to find missing landmark boundaries, correct mismatching or ambiguity, using the "WebStem" Web-App, SVM binary classifiers and other techniques.

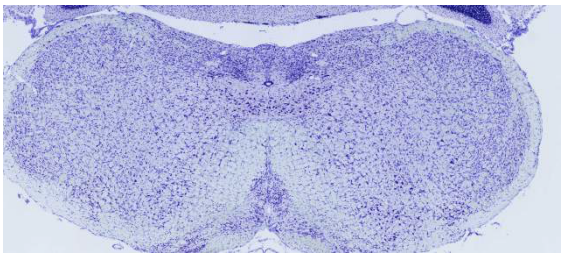

Figure 2.2. Main sections of brain-stem all steps in pipeline illustrated on.

2.2 The Unsupervised Pipeline

We called this whole section ”The Unsupervised Pipeline”, as it is assembled of significant stage that process some input (e.g., raw images, convoluted images, texton histograms), and outputs some new data to next pipeline stage, until getting to final result. Each subsection is a step in the pipeline. All steps are illustrated on the slice from figure 2.2.

2.2.1 Unsupervised Related Work

Parts of this sections are based on Yuncong Chen’s paper [32] in order to explain the inputs, process and results of the supervision part. Existing work on automatically reconstructing 3D volume from histology section include the Allen Mouse Brain Atlas [33, 34] and the Waxholm Space [35], both of which use intensity-based methods such block-matching [36, 37] and mutual information maximization. Landmark-based methods take advantage of details in the histology image using descriptors such as SIFT [38] and binarized gradient orientation histogram [39].

2.2.2 Texture Model - Using Histograms of Gabor Textons

We can classify textures into two types, deterministic textures and stochastic textures. Deterministic textures can usually be defined by a set of primitives and a placement rule (as ”Penrose tiling” [40]), while stochastic textures do not have easily

identifiable primitives (e.g. bark, granite, sand). The brain-stem images have a mixture of these two structures, since different cell types create different shapes, but they also transition naturally from one cell type to another.

This paper will try to analyze the brain-stem images as a synthesis of deterministic textures, and will improve accuracy impacted by this assumption using supervision. Our approach is motivated by research on human texture perception. It is apparent that texture discrimination of two textures are often difficult to discriminate when they produce a similar distribution of responses in a bank of (orientation and spatial-frequency selective) linear filters, but in a case it is hard for the naked human eye to distinguish between them, we do not expect our classifiers to do better.

Computational efficiency is one of the advantages of this approach compared with many of the previous texture analysis/synthesis systems. The algorithm involves a sequence of simple image processing operations: convolution (of Gabor kernels), subsampling (of shifted windows), histogramming (texton histograms), and other basic nonlinear transformations. These operations are fast, simple to implement and abundant implementations exist as open sources.

We filtered images using Gabor filters bank of 9 orientation angles and 11 scale values, resulting in 99 dimensional feature vectors. We reduced the data by quantizing these feature vectors into 100 clusters using K-Means [41]. Clusters with close centroids are then merged, and the final cluster centroids are the textons themselves. In our experiments, we eventually received 14 textons. We further reduced the data by describing texture at the level of superpixels. For each superpixel, the histogram of the textons it contains results the texture representation itself. Figure 2.3 is the result images of figure 2.2 section with different Gabor kernels: rotations and frequencies.

We define h_i to be the texton histogram of the i-th superpixel. Superpixels are the patches obtained by the SLIC algorithm. Since different sections may be oriented

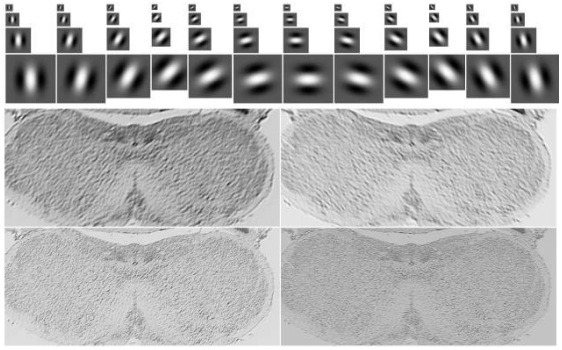

Figure 2.3. Main sections of brain-stem all steps in pipeline illustrated on.

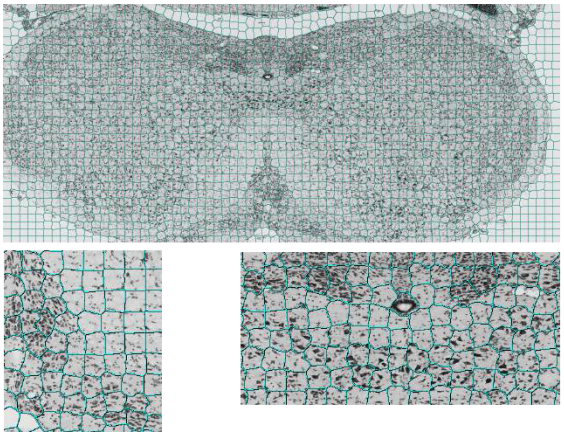

Figure 2.4. SLIC segmentation result on section.

at different angles, we need the features to be rotation-invariant. Therefore, before running K-Means, we need to compute the energy distribution over orientations for each Gabor feature vector, and shift all feature vectors such that the modes of the directional energy distributions are aligned. This way, the texture is decoupled from directionality. Figure 2.4 is the segmentation result over SLIC.

2.2.3 Computer Significance of Superpixels

In order to generate superpixel descriptors, it is necessary to combine the segmentation data from SLIC algorithm with the computed significance of superpixels. There are couple of approaches to compute the significance, but we decided to combine three to give the best results. The result $F(S)$ is a linear combination of the three and evaluates

the scoring function for superpixel S. We also define $T(S)$ as the surrounding superpixels set of superpixel S.

First term is a result of Chi-Sqaured test, or χ^2 *test* [42] that results $p-value$ as a quantitative measure of distinctiveness of superpixels (”expected”) to their surrounding ones sampled from the same distribution (”observed”):

$$\chi^2 = \sum \frac{(observed - expected)^2}{expected}$$

Smaller $p-value$ means a more significant region. All comparisons are made on histogram values of superpixels. To be conservative, the largest $p-value$ among all tests is used first. The $p-value$ is Chi-Squared test applied on two independent histograms.

Second term is a texture homogeneity within a region. It is computed as the mean $p-value$ of Chi-Squared tests between all superpixels in the region. Our motivation was to give a better score to a region consistent of similar superpixels over a ”noisy” region. This can also be caused because of many small tears in the samples.

Third term is a region compactness estimation. Most region landmarks have compact shapes and therefore a higher value should mean it is more likely to be part of a landmark. Compactness can be measured using *isoperimetric quotient* [43], defined as the enclosing area divided by the square of circumference. $|S|$ defines the number of pixels in superpixel S, and $|T(S)|$ defines the number of pixels in the region itself:

$$F^{comp}(S) = \frac{|T(S)|}{|S|^2}$$

The overall significance function is defined as the linear combination: $F(S) = w_1 \cdot F^{cont}(S) + w_2 \cdot F^{coh}(S) + w_3 \cdot F^{comp}(S)$, with weights chosen empirically.

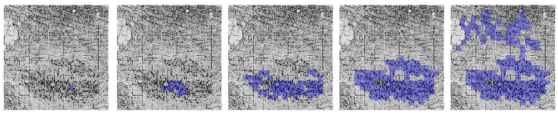

Figure 2.5. Regions during growing, from left to right, at iteration 1, 10, 50, 96 (most significant), 150 (last iteration).

2.2.4 Growing Regions and Clustering to Find Significant Regions

In order to find candidate regions as ”boundaries”, and even candidate regions in landmark, we performed region growing algorithm from every superpixel. By starting from a seed superpixel, we greedily grow by adding all superpixels with χ^2 distance greater than some threshold, between their texton histograms. The threshold is again was chosen empirically. Growing stops when the region reaches 10% of total area or none of the surrounding superpixels are within the constraints. Eventually, the region computed with highest significance score $F(S)$ is returned, and the superpixels corresponding to it are masked out. Figure 2.5 shows examples of such a growing to convergence.

We call the set of superpixels that grows from a seed superpixel k a ”boundary” cluster of seed k, or S_k. In case that some boundary has distinct different patterns of textures, we can rerun the algorithm on that boundary with a more strict threshold. Such a scenario can happen on inhomogeneous random regions, as on figure 2.6. We concluded that the denser a boundary cluster is, the more distinctive the region it represents.

Another enhancement made was grouping boundaries using Jaccard index as pairwise distance and hierarchical clustering. From each group whose size is large enough, we selected the proposal with the highest significance score to be the representative of the group. All representatives are ranked by their significance score.

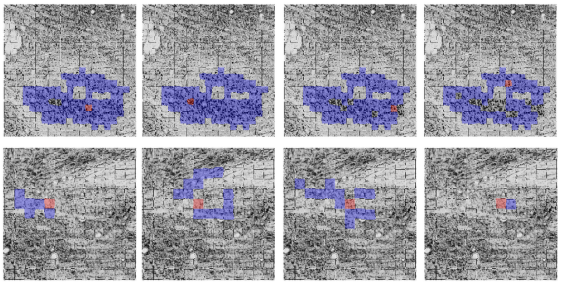

Figure 2.6. Top row: the region proposals of four different seeds (in red) in the facial motor nucleus. They are very similar. Bottom row: the proposals of four neighboring seeds in an inhomogeneous region. They are very random.

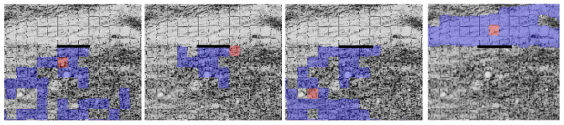

Figure 2.7. The three region proposals in an inhomogeneous area are not consistent as a whole, but they all agree on a robust boundary segment (highlighted).

2.2.5 Identify Robust Boundaries by Region Consensus

When a region gradually transitions into neighboring textures on one side, yet has a clear boundary on the other side, we might experience incorrect results. This method may not capture some inhomogeneous regions, even though the open boundary is a perfect landmark. Figure 2.7 shows such an example. Notice the amount of variation between the proposals from the seeds in such a region, yet they agree on the clear boundary. This motivated a consensus-based approach for evaluating boundary robustness: each region proposal ”votes” for the segments on its boundary.

A boundary segment between two superpixels is represented by an ordered tuple $(i; j)$; i is the interior superpixel and j is the exterior superpixel. δS_k denotes the set of segments on the boundary of a region proposal S_k. A vote of a superpixel to a region depends on how contrasty the segment is, and defined by the distance of texture histograms between the average histogram of the region, and the histogram of the exterior superpixel. The set of superpixels that vote for a segment is called the *supporterset*

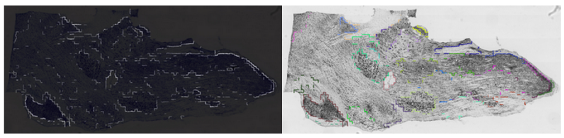

Figure 2.8. Left: thresholded boundary vote map. Right: grouped boundary segments

of the segment, and denote it by $R(i;j) = k : (i;j) \in \delta S_k$. The total score received by a segment $(i;j)$ is then the sum of Chi-squared test over all *supporter* superpixels:

$$b(i;j) = \sum_{k \in R(i;j)} \chi^2(h_{S_k}; h_j)$$

We discarded segments whose vote is lower than some threshold. Figure 2.8 shows a vote map. Since we need big enough boundaries to define landmarks, we combine segments with similar *supporter sets* into a single group, using hierarchical clustering and Jaccard index between *supporter sets* and pairwise similarity. At the end, each set represents a boundary, and all boundaries are ranked by the total vote received by all its superpixels.

2.2.6 Matching Landmarks from Different Sections

Since the final goal of the pipeline is to find landmarks - correlating boundaries across sections, we needed to merge coinciding boundaries into a single set of open and closed boundaries. The correspondence between boundaries from different sections is done using a distance function. The functions is a weighted linear combination of four different aspects, with weights chosen empirically:

$$D(B_1, B_2) = w^{int} D^{int}(B_1, B_2) + w^{shape} D^{shape}(B_1, B_2)$$

$$+ w^{ext} D^{ext}(B_1, B_2) + w^{loc} D^{loc}(B1, B2)$$

First term is the difference of interior textures (”int”). A region landmark S’s

interior texture is the average of superpixels texton histograms h_S. Similarly, for a boundary B, the interior texture is the average texton histograms of the union of all superpixels that are in the supporters sets.

Second term estimates the similarity of boundary shapes, by reducing the boundary to a set of center points of the segment, and the sum of computed distances. The "shape distance" between two point sets can be computed using *shape context descriptors* [44]. The shape context descriptors characterize the organization of other points around each point using a histogram. Two sets of points are matched by finding the minimum bipartite matching, where the edge weights are the Chi-Square distances between shape context descriptors. The Hungarian algorithm [45] is used to find the minimum matching. The average cost of this matching, is used as the second term.

Third term is computed after the matching is done. It computes the total distance between exterior textures of all matched segments:

$$D^{ext}(B_1,B_2) = \sum_{(i,j),(p,q)\in M(B_1,B_2)} \chi^2(h_j,h_q)$$

Fourth term measures the "spatial proximity". It computes the thresholded Euclidean distance between the center of mass and the center-points sets of all boundaries:

$$D^{loc}(B_1,B_2) = max(0,||m_{B_1},m_{B_2}||_2 - l)$$

where m_{B_1} and m_{B_2} correspond to boundaries B_1 and B_2 as their centers of mass, and l is a tolerance value, allowing the position deviation not to be penalized (we set it to 1mm empirically). Using the linear combination of distance functions, we can compute for any two sets of landmarks their pairwise distance detected from any different section. We match two landmarks iff they are the closest landmark to each other.

2.3 Supervision to Improve Registration and Classification

The process of unsupervised registration and classification has worked well to some extent, but since the goal is to completely automate the digital atlas creation, and since it is intended to be used by a computer, supervised methods are necessary to improve accuracy. This subsection describes the process to achieve that.

2.3.1 Goals and Priorities

In order to create a reliable digital atlas, the boundaries matched between every two adjacent sections have to be correct. It seems that either the dimension reduction to textons or the equal handling of the different texton bins (using Chi-Square distance) is not ideal for landmarks classification and causes some error rate. We would like to tune and improve these inaccuracies by using additional supervision trained classifiers to prevent stages in pipeline from making these mistakes. All of these, while still using the least expensive man-hours as possible for the task.

Figure 2.9 is an classification example where each boundary in one section is nearly perfectly matched to its adjacent section boundary.

On the other hand, boundaries 1, 2, 3, 4 on figure 2.10 did not match well, which raises the question of what caused the mismatch and how can we solve it.

2.3.2 Focusing Supervision by Exploration

Our main goal is to use the same outputs that we got from the unsupervised pipeline, and possibly without changing the pipeline at all, to improve landmarks mapping accuracy. We believe this can be done by better understanding the different texton histogram bin values, which can be done using supervision. Each superpixel is a collection of pixels defining some segment, obtained using SLIC algorithm. We define a

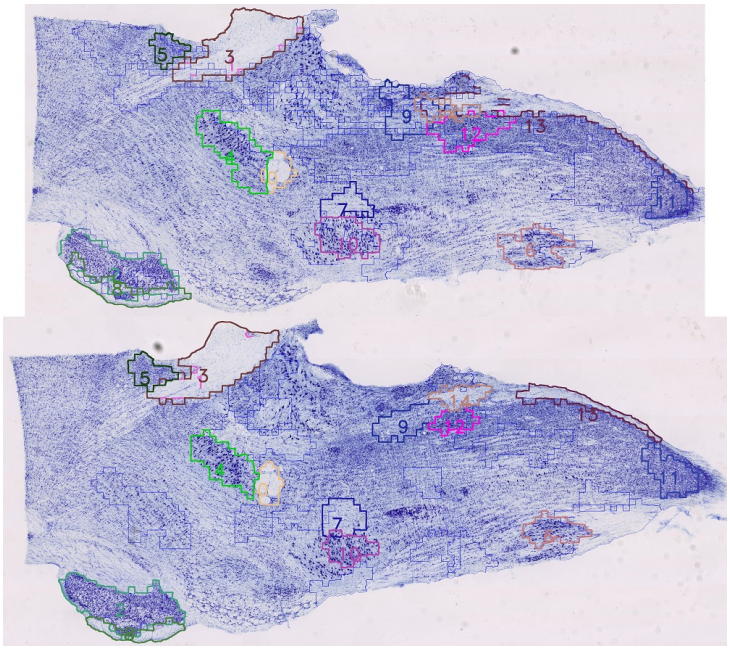

Figure 2.9. Sections 10-11; Boundaries 0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14 have all matched in a good accuracy.

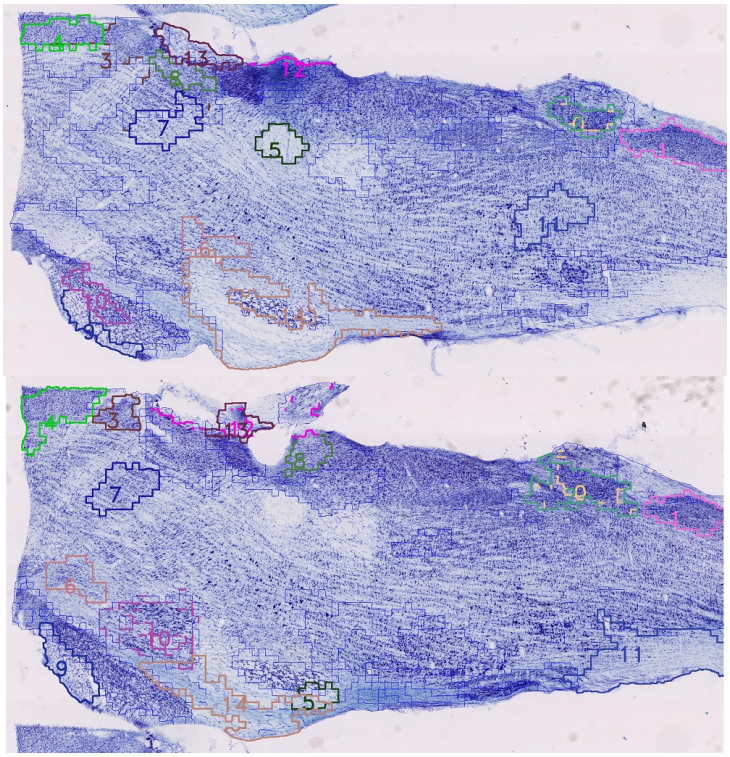

Figure 2.10. Sections 23-24; Boundaries 5, 6, 10, 11, 12, 13 mismatched

super-superpixel as a cluster of one or more superpixels - using the greedy "Snake" algorithm [46] and some pre-defined thresholds. By sampling three super-superpixels from boundaries A and B that incorrectly matched, and let a "labeler" (a person with knowledge in brainstem cells) to decide whether the third super-superpixel is more similar to A or B, we can collect that data and hopefully infer regarding the texton histogram bins. These comparisons are asking the labeler to provide constraints of the form $d(a,b) < d(a,c)$, where $d(x,y)$ represents some perceptual distance between $x \in A$ and $y \in B$. We will refer to these constraints as triplets. By collecting these triplets from labelers, as explained in the literature review section, we can use this information to train classifiers. For example, the authors of McFee [26] were able to learn music genres from triplet comparisons alone with no other annotations (an embedding task). Specifically in computer vision, human similarity comparisons are useful for creating perceptually-based supervision. On the next steps we focus on creating perceptual supervision of images from superpixels and super-superpixels.

2.3.3 Trial 1: Compare One Superpixel at a Time

This stage explores the basic concept of comparing a superpixel, either from boundary A or B to either boundary. The labeler decides which one it is more similar to, and this data is stored as a triplet in a log file of triplets: ($< SuperPixel_i >$, $< Matched\ Boundary >$, $< Other\ Boundary >$). It let the labeler to decide which boundary it is more similar to. Each superpixel is rotated using the accumulated rotations information of each pixel of the first piplipne stage - and takes the rotation value that appears most of the times per superpixel.

The struggle of balancing between the amount of triplets we collect and the man-hours always exist. Generating triplets provides information about some cases, but may take a long time to collect enough data to cover all cases. Letting the labeler to

label hundreds of pages of triplets is an inefficient utilization of his time, and required a better solution. Trying to give a single sample of each pair of mismatched boundaries is insufficient as well. Since the SLIC [19] algorithm collects the data up to a threshold, sometimes the two given samples are similar enough and cannot provide enough or any data about the boundary. We did not have a well enough defined way to find a representing super-pixel too.

At this point we tried taking super-superpixels, "patches", to increase the probability to match right representative patch. By choosing a center superpixel, SP, and probing the surrounding 8 superpixels with histogram distance Th smaller than some threshold, we can collect superpixel similar to each other into a single patch:

$$Patch(i) = SP_i \cup \bigcup_{j \in adj(i,8)} SP_j$$

Even with this method, there were thousands pages generated.

2.3.4 Trial 2: Compare Many Super-Superpixels to Two References

Traditionally, a task designed to collect triplets would show labelers three images, labeled a, b, c. The labeler is then asked to select either image b or image c, whichever looks more similar to image a (see figure 2.11 for an example). Although this is the most direct design to collect triplets, it is potentially inefficient, as the labelers time is expensive. Instead, we chose to investigate triplets collected from a grid of images.

In the grid format, instead of showing references a and b, and ask to compare a single image to each reference. The images are displayed on a grid of n images. The labeler is then asked to mark the most similar images to reference a and then to reference b. This layout allows us to collect k_a images that are more similar to reference a, k_b images that are reference b. We assume that sometimes the task of matching is too hard

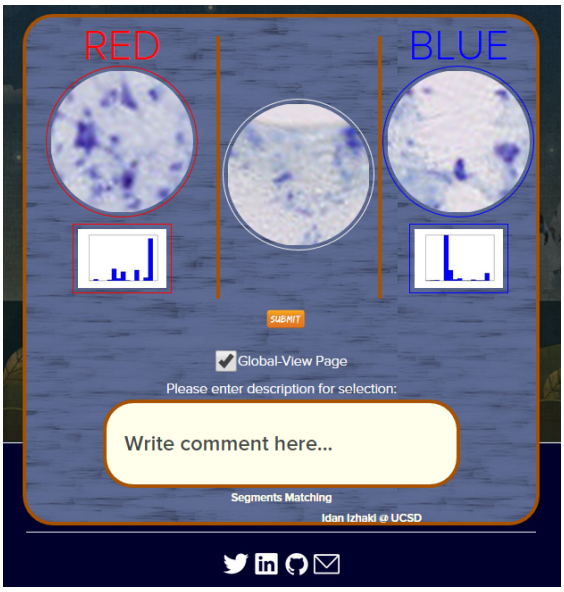

Figure 2.11. 2 Reference Superpixels and a Single To Match

with the information on the a superpixel image, and in such a case we do not expect the labeler to match the image to either of the references. Again, our goal is to do well as closest as possible to a labeler work, but do not expect to do any better than him.

We are not the first to realize that a grid is more efficient for collecting triplets. Such techniques were also used by Wah [30], but we believe we are the first to investigate more thoroughly the effectiveness of triplets collected with a grid on natural and sometimes very similar textures, and to improve performance on unsupervised results, rather than using for embedding.

The reason we decided to use super-superpixels, or ”patches”, was because we wanted to increase the amount of images matched to references. The more matches we have, the better we can train the different classifiers. We started trialing with different sizes of grid, but found 32 (8x4) grids to be the best balance between showing all information on the screen and still receiving enough information of each page. We used exactly two references, as we believe it is the most effective combination. The user can leave patches unmarked, as in ”indecisive”. In such a case we want to say that sometimes

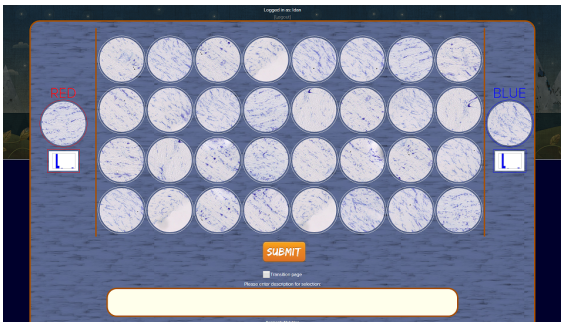

Figure 2.12. Grid of 32 images that allows multiple selection of images to any of red/blue references.

the labeler himself cannot identify which patch is more similar to what. We do not expect the computer to be able to mark it then as well. We will decide whether to add it to the boundary or not by using the snake algorithm with a predefined threshold. Every time we show a new page, we shuffle the images within it. That allows to eliminate labeling that is a result of surrounding images acting as a background to that image, or because of distance to either reference images.

Figure 2.12 is an example for a page. Left reference is called ”red” reference and right reference is called ”blue” reference. Clicking on either the ”red” or ”blue” references allows marking on unmarking the images in the center grid, with the same color. A comment box is located at the bottom allows the labeler to comment why he believes the reasons for the selection may be useful or important to us. Any action the labeler does, as selecting a reference, marking or unmarking any image, etc., is logged to unable us a better understand dilemmas, and maybe remove them from training, as they were inconclusive labeling.

Trial 3: Transition Page with Patches Marked

Another addition to the WebApp is a global view page (figure 2.13), showing the two section images, the upper one correlates to the red patch, the bottom one to blue patch, and all the 32 patches circled. We believed it could give the labeler some better

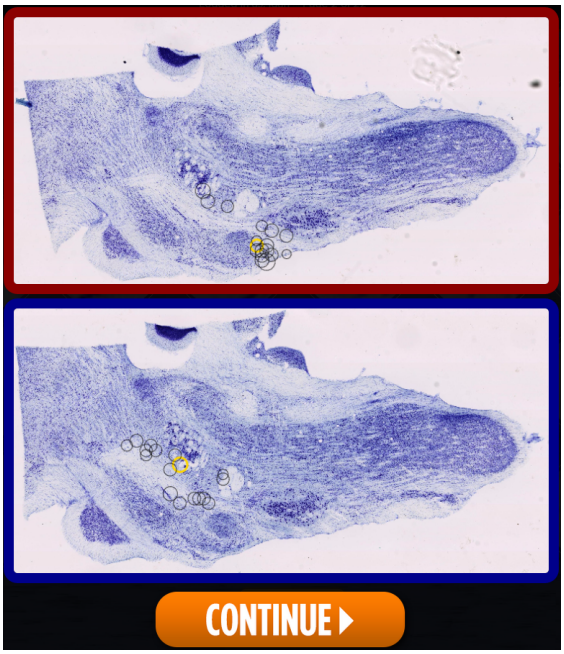

Figure 2.13. Global view with all patches circled. The two golden circles are the reference patches.

understanding of the origins of the two different patches, and thus make better more accurate decisions. We also made sure that the two references we show, are the two patches from each boundary that has the highest histogram distance, that way we make sure the references are not too similar which makes the labeling process confusing.

At this point we relied on two wrong assumptions - we want to make it easier for the labeler to label by showing him extra information, and that the different classifiers can still get to same accuracy as the labeler's by looking on a match smaller windows at that time (superpixel vs. super-superpixel).

2.3.5 Trial 4: Compare 32 Superpixels to Two References

We concluded that the super-superpixels approach is wrong, as it gives the labeler data that might give him a better understanding of the surrounding area than the algorithms that process the superpixels, and therefore a better accuracy in labeling the patches. It also

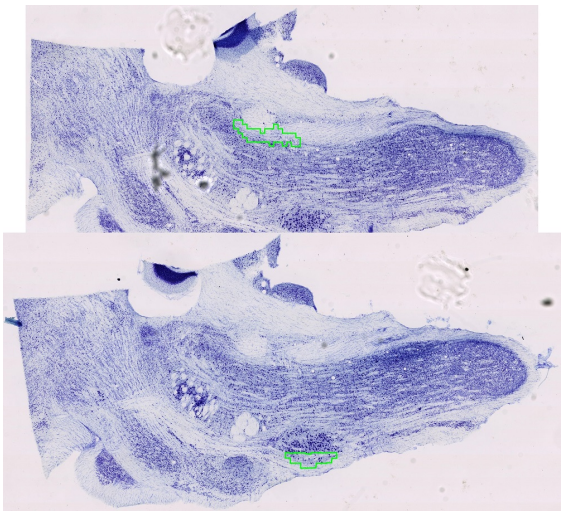

Figure 2.14. Before each page, the mismatched boundaries that related to patches are shown.

means that we are trying to train a classifier with data which is not as comprehensive - the single superpixel data. The first change was to remove the patches and replace them back by superpixels. The second change was to remove the ”global-view” window and show the labeling results only after that the labeler submits his prediction. The third change was to show before each page the two references boundaries without the superpixels in them (figure 2.14). The second change was just as an entertaining / motivational tool for the labeler, to see how well he did on labeling, and maybe be more cautious about mislabeling on next pages. The third tool was meant to give the labeler some idea of which brain-stem regions are being probed, some of them maybe very similar to each other for a good reason.

This change has given us the best classifier training results. Using all the inputs collected from the WebApp, we were able to compare the accuracy of the unsupervised classifier output to the new supervised classifier output.

2.3.6 Using SVM on Supervised Labeling

Collecting the intersection of all the correct labeling that all labelers resulted, we created a binary labeler for each landmark. This is done by using Support Vector Machine, SVM, that results for each superpixel, the value ”1” in case it was classified as a part of the landmark, or ” -1” otherwise:

$$D = \{(x_i, y_i) | x_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}_i^n = 1$$

Assuming that the ”blue” reference is corresponding to ”boundary-A” and the ”red” reference is corresponding to ”boundary-B”, the triplets information is processed as follows:

- ”Blue” reference is marked as 1 on boundary-A classifier, and ”red” reference is marked as -1 on boundary-B classifier.
- ”Red” reference is marked as 1 on boundary-B classifier, and ”blue” reference is marked as -1 on boundary-A classifier.
- Each ”blue” superpixel labeled correctly to its boundary, marked as 1 on boundary-A and -1 on boundary-B.
- Each ”red” superpixel labeled correctly to its boundary, marked as 1 on boundary-B and -1 on boundary-A.

The input x_i is the texton histogram value of superpixel i, and y_i is either 1 or -1, as described above.

After having all classifiers trained, we need to redo the labeling process, with a new distance function. Each classifier corresponding to a specific landmark i is called

Cl_{l_i}. The new function is a weighted linear combination of five different aspects similarly to before, with weights chosen empirically:

$$D(B_1,B_2) = w^{int}D^{int}(B_1,B_2) + w^{shape}D^{shape}(B_1,B_2)$$

$$+w^{ext}D^{ext}(B_1,B_2) + w^{loc}D^{loc}(B1,B2)$$

$$+w^{text}D^{text}(B_1,B_2)$$

The term $D^{text}(B_1,B_2)$ is the probability of two boundaries to overlap. It is computed by counting the amount of superpixels on each boundary, from both sections, that receive "1":

$$D^{text}(B_1,B2) = \begin{cases} 1, & \text{if } \exists i \text{ s.t., } \frac{\sum_{b_1 \in B_1, b_2 \in B_2, Cl_{l_i}(b_1)=1, Cl_{l_i}(b_2)=1} 1}{|B_1|+|B_2|} > \alpha \\ \text{-1}, & \text{otherwise} \end{cases}$$

where α is some threshold empirically defined, meaning that the percentage of superpixels that were determined as of the same landmark by some classifier i is at least α (percent). Since the value of D^{text} is a constant, we discovered that high values of w^{text} are more effective than low ones, as $w^{text} = w^{int} + w^{shape} + w^{ext} + w^{loc}$.

At this point we have classifiers that can predict mismatching boundaries to landmark (because of either ambiguity or outliers noise), or even add boundaries to each section, that were not discovered by the unsupervised pipeline approach.

Chapter 3

Experiments

This chapter will cover the experiments results, by explaining the inputs, the platforms and technologies used, explain the previous experiments results, and the improvements supervision had. The ”WebStem” web-app changed a couple of times to fit and achieve optimal results, and to enable easier integration into the pipeline code.

3.1 Input Data

The input data is assembled of four different nissl-stained brain sections: RS141 (figures 3.1), RS140, RS139, and RS58. Each brain sample is composed of $\sim$ 30 slices; referred as ”sections”. We expect to find for each sample $\sim$ 20 definitive different cell clusters; referred as ”landmarks”. At this point, we will only process sample RS141 for training, but similar pipeline should be relevant for any given mouse brainstem sample.

Each image in original size is about 100 Megabytes and 280 Megapixels, but the JPEG2000 [47] format enables us to easily extract 5 different resolutions of the image,

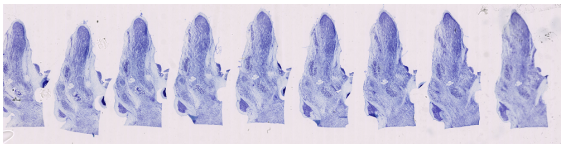

Figure 3.1. RS141 nissl-stained brain sample; a couple of adjacent sections side by side.

and save them in TIFF format, a lossless compressed format. The given sections are not aligned, and there are tears and stretches in most of them. Since the mouse brain is small, the slices are comparably thick, meaning the transition from one slice to another is sometimes significant. These properties may emphasize the complexity of the task.

3.2 Target Platform and Technologies

The chosen platform for this work is the San Diego Supercomputer Center "Gordon" computer cluster. Large graph problems, data mining, de novo genome assembly, database applications, and quantum chemistry are some of the fields of research benefiting from Gordon's unique architecture. The system is characterized by:

- 1,024 dual-socket Intel Sandy Bridge nodes, each with 64 GB DDR3-1333 memory.
- Over 300 TB of high performance Intel flash memory SSDs via 64 dual-socket Intel Westmere I/O nodes.
- Large memory supernodes capable of presenting over 2 TB of cache coherent memory.
- Dual rail QDR InfiniBand network (up to 5.0 Gigabit/second internally).
- Data Oasis high performance parallel file system with over 4 PB capacity and sustained rates of 100 GB/s.

The combination of multiple processors, a lot of RAM and high speed Oasis file system enables running all algorithms in parallel, and load all data into memory. Since each section photo is hundreds of Megabytes, some of the algorithm take hundreds of Gigabytes of RAM. The total data stored on disks (input, pipeline, result) is 1.4 Terabytes of data. Gordon environment assign static IP address to each cluster node, which enables running

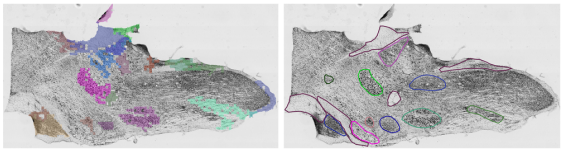

Figure 3.2. Left: top 20 landmarks detected by the algorithm. Right: human labeling for recognizable nuclei and white matter.

iPython notebook and ”WebStem” on any node, and connect to them from anyplace in the world fast, and mostly reliable.

All code runs on either Python (and Scipy), NodeJS as a back-end web-app, HTML, CSS, Javascript, etc., for front-end.

3.3 Comparison to Human Labeling

We test our algorithm on a series of 30 sections images of a nissl-stained mouse brainstem. The images are scanned at 2 microns per pixel, showing individual neuronal cell bodies. Both types of landmarks are detected on all images. For each image we use the top 20 closed boundaries and the top 10 open boundaries. Figure 3.2 shows the landmarks detected from one image. Also shown is a version created by a human labeler who annotated for nuclei and fiber tracts with the help of a printed atlas. Most significant structures from the human labeling are detected by the algorithm.

3.4 Robustness of Unsupervised Matching

The landmark matching algorithm is applied to all 30 pairs of consecutive images. In order to test the robustness of matching under large displacement, we remove the spatial proximity term from the landmark distance function, leaving only the texture and shape terms. A human evaluator (”labeler”) then judges whether a matching is correct, partially correct, or incorrect. Among all 166 matchings returned by the algorithm, 106 are correct (63%), 34 are partially correct (22%) and 26 are wrong (15%). One example

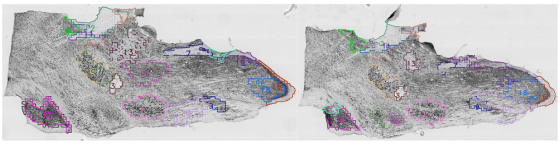

Figure 3.3. Landmark matching example. Matched landmarks are marked with the same color and number. Note that matchings are found based only on texture and shape. Matching 13 is made possible by modeling both open boundaries and close boundaries. Landmarks such as 12 and 9 show considerable shape change, but are still matched.

is shown in figure 3.3. This demonstrates the effectiveness of texture modeling. Even though some landmarks change shape significantly, the algorithm still finds the correct matchings.

3.5 The Supervision Process

The errors made by the unsupervised pipelines is mainly caused because of 3 reasons:

- Boundaries from different adjacent sections mapped incorrectly to same landmark.
- Boundaries from different non-adjacent sections mapped incorrect to same landmark, and some of the boundaries between these sections mapped correctly.
- Boundaries Some clear boundaries were filtered out as non-boundaries (segmentation stage), missing possible mapping to correct landmarks.

We will show strategies to improve accuracy for all three scenarios. This requires manual work and analysis of the different boundary matching results, and all of those found - were covered as part of the data here for completeness. We will also show that good results can be achieved using either mainly true position or mainly true negative sampling. Iterative mode can also be applied in tough cases.

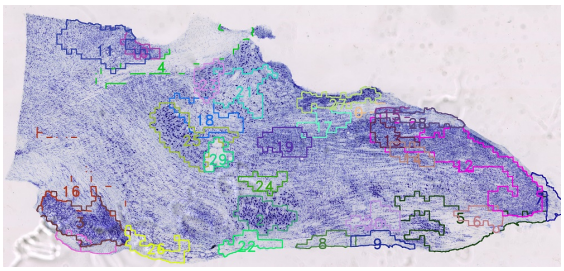

Figure 3.4. All boundaries detected for section 9.

3.6 True Positives and Negatives to Improve Mismatch

One of the most common cases of mismatching, is when adjacent sections don't match boundaries to same landmark when they should have. Since matching non-adjacent section uses clique to assure matching is correct (related to implementation details of the unsupervised pipeline), the odds that two or more mistakes will occur for some landmark is low, and such matches are mostly filtered out. Adjacent sections don't have that privilege.

Figure 3.4 exemplifies that sections 9, boundary 10 match to section 10, boundary 3, but section 11, boundary 17, does not match to same landmark as it should have, as demonstrated on figures 3.5 and 3.6.

In that case, we have more positive examples for boundaries to match to the landmark, than negative ones. We start by generating a page in the "WebStem" web-app, comparing a "red" reference from 9:10, 10:3 to a "blue" reference from 11:17 (11:17 refers to section 11, boundary 17). Since these example boundaries are relatively small, with no many superpixels, we generated only 2 pages, one for 9 against 11, the other is 10 against 11. Nevertheless, we do have cases where the boundaries are big enough to generate multiple pages for each tuple. From our experiments there is no real need to generate more than a single page per matched boundaries example, the contribution is negligible compare to the labeler work-hours doubtfully needed.

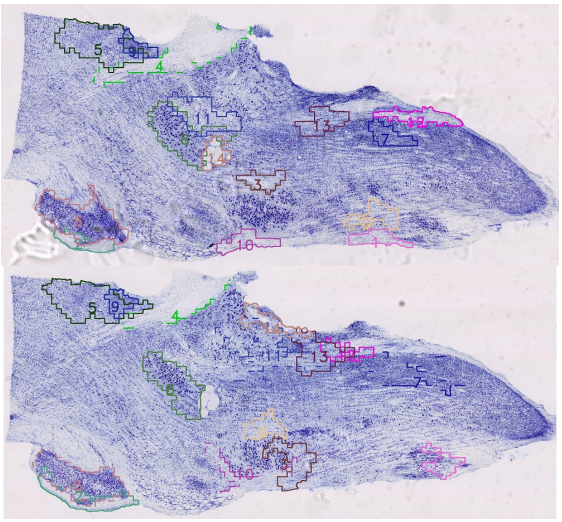

Figure 3.5. Section 9 boundaries relatively to section 10.

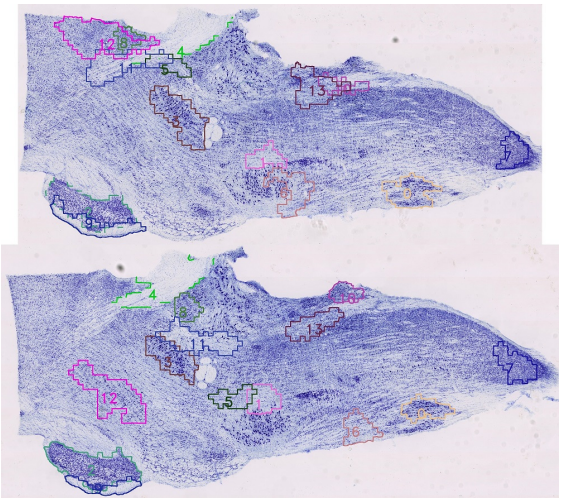

Figure 3.6. Section 10 boundaries relatively to section 11.

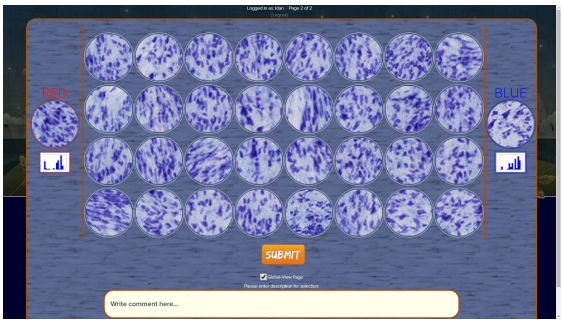

Figure 3.7. A sample Web-App page for combination of section 9 with section 11.

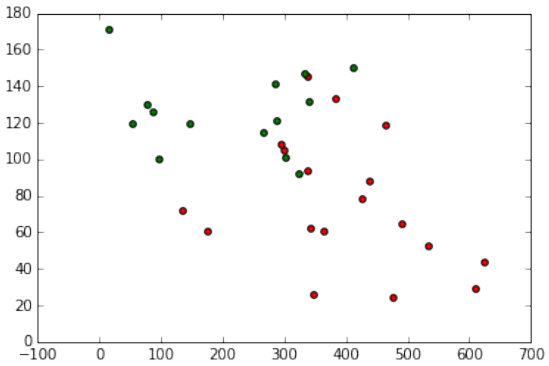

Figure 3.8. Graphs shows separation of histogram for random two bins. The pattern is the same for any combination of bins.

Table 3.1. Top 5 boundaries ranked for section 11 on the using the supervision data)

Ranking	Boundary	Score
1	17	12.173
2	13	7.946
3	27	5.360
4	25	3.479
5	21	2.741

On figure 3.7 it may seem at the beginning that the comparison is extremely difficult and impossible to solve in the context of superpixel itself. After careful examination, we see that the blobs in the red image superpixels have similar color to the blue image, but more stretched and overlapping rather than evenly spread. The texton histograms seems distinctive as well. Figure 3.8 visually proofs that indeed for some 2 bins of the histograms, the separation is high. That pattern was separability was similar for any other chosen two bins. This type of separation seems promising, SVM be able to classify well on it.

Running SVM on the textons histograms and taking only the boundaries with positive average prediction score results table 3.1

First of all, we can see that boundary 17 reacted to the landmark we hopped for. In addition, all boundaries but 17 on the pipeline results cancel out from the

Table 3.2. Top pipeline results to match section 11 with 10 four boundary 17. All Cancel out but the correct one.)

Ranking	Boundary	Score	Weighted
1	9	1.300	inf
2	15	2.444	inf
3	17	2.642	-9.531
4	5	2.807	inf

supervision output, as shown on table 3.2. That tells us the highly scored (low distance values) boundaries, were simply outliers faking as similar boundaries, and not a result of ambiguity or similarity to another boundary.

These results prove us that combining the estimations from the unsupervised pipeline algorithms with the supervision information and SVM training, we can eliminate sometimes all outliers, and have a single definite boundary which is the correct one as well.

3.7 True Positives to Improve No-Match

In some cases, a boundary is discovered by the segmentation algorithms, but the classifiers are unable to match that boundary to any other boundary, even though it is clear to the professional eye this should not be the case. Sometimes the boundary segmentation is not generated at all.

Such boundaries are usually extremely diversified in their patterns, and 14 bins of textons are probably not enough to match them with a high enough confidence. In such a case, we can use supervision to choose matched boundaries of same landmark, let the labeler to distinguish between patterns within the landmark itself and some other different looking landmarks, expecting the texture to be unique enough to be identified on any section afterwards.

Section 0, boundary 26 (figure 3.9), and section 1, boundary 12 (3.10) are an

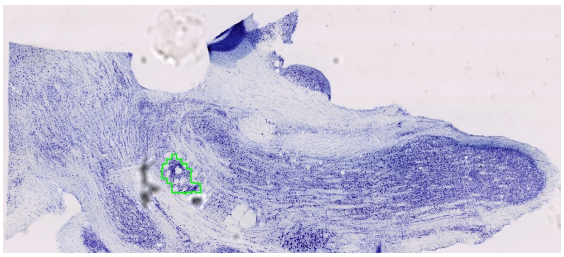

Figure 3.9. Section 0, boundary 26 as part of a landmark.

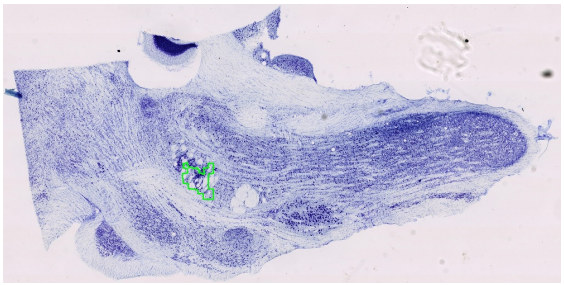

Figure 3.10. Section 1, boundary 12 as part of the same landmark.

example for that. It seems clear to expect a boundary from section 2 to match, but there is no such boundary segmentation at all.

Running the labeler output on the SVM results a binary classifier for any superpixel on the screen. Since it is a very unique pattern, we believe that running that classifier on section 2 would result positive classification for that missing part of landmark. And indeed, clustering positive superpixels, and allow neighbors up to a threshold to join the cluster, and new boundary to be identified. Figure 3.11 are the original boundaries found the pipeline algorithms results, figure 3.12 is the new boundary generated by using supervision and clustering.

3.8 Iterative Mode for Outliers

As the pipeline algorithms tries to match boundaries to landmarks, it relies a lot on a distance functions that returns for a pair of boundaries from different sections, a real number value. The lower the value, the closer they are to each other. This enables ranking

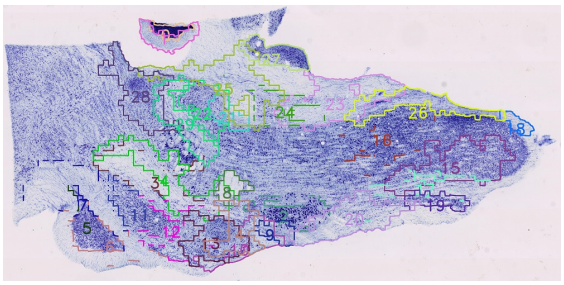

Figure 3.11. Original boundaries generated for section 2.

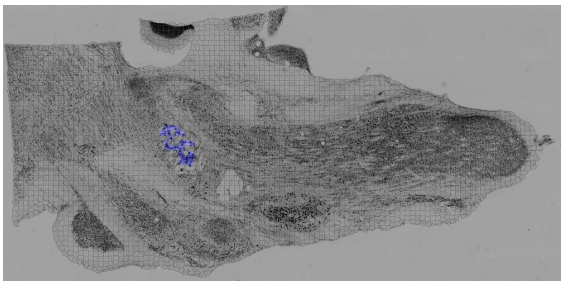

Figure 3.12. New boundary created using semi-supervision for section 2.

sections similarities, and iteratively that way to match and proceed to next non-matched highest ranked pairs. Table 3.3 the ranking for boundary 28 on section 27, comparing to top 8 matched boundaries from section 27. Figure 3.13 is the boundaries identified of section 26 and section 27.

It is easy to see that boundary 3 on section 26 has the highest ranking and therefore

Table 3.3. Top 8 ranks and boundaries between sections 26 and 27 for boundary 28 (lower is better)

Ranking	Boundary	Score
1	3	0.706
2	21	0.785
3	16	0.812
4	18	0.832
5	17	0.832
6	19	0.897
7	0	0.975
8	5	1.071

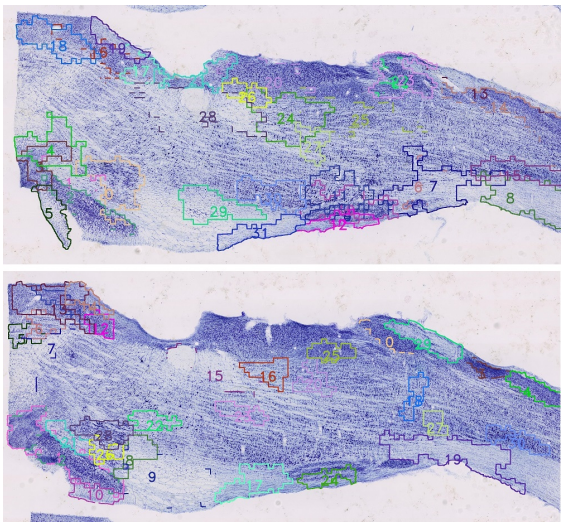

Figure 3.13. Section 26 and 27 detected boundaries.

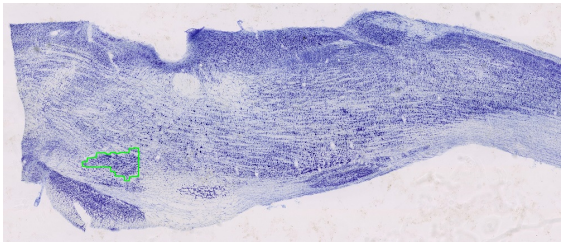

Figure 3.14. Section 25 and boundary 22.

matched, but is definitely not on the same landmark. Sections 25 on figure 3.14 and 28 on figure 3.15 matched boundaries 22 and 1 correctly to the same landmark as 28.

At that stage, we collected all the superpixels of each boundary, and generated 3 web pages (figure 3.16, 25 against 26, 27 against 26, and 28 against 26). These are three positive cases against a single negative one. We let the labelers to match the right superpixels to references, as before, and processed the outcome.

Running SVM on the textons histogram and taking only the boundaries whose average suprepixel score is non-negative, table 3.4 are the generated results. The last column is the weighted value against the pipeline output (pipline minus SVM value for non-filtered out values).

This computation gave us two candidates. Between boundaries 10 and 0, it is

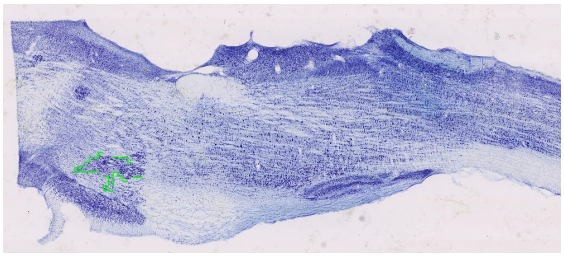

Figure 3.15. Section 28 and boundary 1.

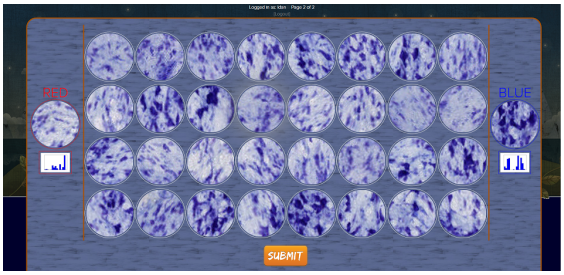

Figure 3.16. Section 27 again section 26.

Table 3.4. Top 7 SVM results for boundary 26, using feedback from labeler (higher is better)

Ranking	Boundary	Score	Weighted
1	30	1.818	inf
2	11	0.913	inf
3	15	0.693	inf
4	5	0.547	0.524
5	0	0.294	0.681
6	10	0.274	inf
7	2	0.253	inf

easy to see that boundary 0 is the correct one to match to. It is expected to happen as SVM trained on two different boundaries, and outliers sometimes might match to one of the boundaries because that some of the superpixels got extremely high score, and raised the average. For that reason, we need to use hierarchy matching - continue by elimination of either 5 or 0. We created stack of pages, this time against boundary 5.

Running SVM on this algorithm gave 100% accuracy on the training input, meaning boundary 5 was filtered out. That tells us that boundary 5 and the landmark are still separable, and therefore can be filtered out. We are left only with boundary 0 as the correct one to join to the landmark.

Running the same procedure but this time assuming that boundary 0 is incorrect did not give 100% accuracy on training input. The average score on boundary 0 remained positive, meaning there is high dependency between boundary 0 and the landmark - we cannot filter it out and naturally can drop the other candidate - 5.

We can conclude from this experiment that some cases, related either to how well the labeler managed to match the superpixels to boundaries, or related to overfitting the SVM algorithm and accepting outliers may happen, but iterative run on the candidates should eliminate the conflict. By the fact, from all the (similar) experiments we made, most of the time one iteration was enough, but never needed more than two.

Chapter 4

Conclusion

This work has shown us that it is possible, using a professional feedback from labelers through a web-app interface, to improve unsupervised segmentation and classification of brainstem sections into a digital landmarks atlas.

The semi-supervision capabilities contributed in three ways:

- Improve accuracy of landmarks by removing incorrect boundaries from the landmark set, caused by outliers, and find the correct one to replace with, using existing information from unsupervised data,
- Find new boundaries the unsupervised method could not find, by using the textons and find the important superpixels to define this texture,
- Resolve ambiguity of multiple good possible candidates using iterative interleaving run of supervised and unsupervised algorithm stages.

It is clear that the unsupervised pipeline data is performing well and gives accurate results many of the times, but the fact the landmarks are need for a digital atlas, it is expected to perform almost flawlessly. The supervision improves significantly the accuracy; interleaving iterative mode of unsupervised and supervised runs would probably always lead to best results.

In addition, the concept of ”WebStem” web-app using a grid of images seems to work well. The easy-to-use interface, the grid layout, the comparison to two references, the pre-processing filtering and prefetching - all make the web-app fast, reliable, and saves a lot of time to the labeler comparing to primitive methods of collecting triplets. Instead of manually labeling each and every superpixel on a canvas, or comparing endless amounts of single superpixels to two references, we collect sometime dozes of valuable samples from each screen.

As an optional future work, an on-the-fly samples loader web-app engine was suggested. The engine will start from showing some scenario of mismatched boundaries, and will keep querying with more samples over the two boundaries until the engine gets confident enough about the sufficiency of the data logged. That way, we can keep querying more samples on tougher scenarios, or finish easy scenarios faster. This would increase results accuracy and would require even less time labeling all cases.

Bibliography

[1] A. I. for Brain Science, "Mouse brain sections database." [Online]. Available: http://help.brain-map.org/display/mousebrain/Allen+Mouse+Brain+Atlas.

[2] R. Chellappa and S. Chatterjee, "Classification of textures using gaussian markov random fields," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, pp. 959–963, aug 1985.

[3] W. authors and references, "Sufficient statistics." [Online]. Available: http://en.wikipedia.org/wiki/Sufficient_statistic.

[4] B. Manjunath and W. Ma, "Texture features for browsing and retrieval of image data," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, no. 8, pp. 837–842, 1996.

[5] A. W. Fitzgibbon, *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06) : [June 17-22, 2006, New York, NY*. Los Alamitos, Calif: IEEE Computer Society, 2006.

[6] S. Grigorescu, N. Petkov, and P. Kruizinga, "Comparison of texture features based on gabor filters," *IEEE Trans. on Image Process.*, vol. 11, pp. 1160–1167, oct 2002.

[7] G. Turk, "Generating textures on arbitrary surfaces using reaction-diffusion," *ACM SIGGRAPH Computer Graphics*, vol. 25, pp. 289–298, jul 1991.

[8] A. Witkin and M. Kass, "Reaction-diffusion textures," *ACM SIGGRAPH Computer Graphics*, vol. 25, pp. 299–308, jul 1991.

[9] J. Marques, *Pattern recognition and image analysis second Iberian conference, IbPRIA 2005, Estoril, Portugal, June 7-9, 2005 : proceedings*. Berlin New York: Springer, 2005.

[10] A. Fournier, D. Fussell, and L. Carpenter, "Computer rendering of stochastic models," *Commun. ACM*, vol. 25, pp. 371–384, jun 1982.

[11] J. P. Lewis, "Generalized stochastic subdivision," *ACM Trans. Graph.*, vol. 6, pp. 167–190, jul 1987.

[12] C. Bennis and A. Gagalowicz, "2-d macroscopic texture synthesis," *Computer Graphics Forum*, vol. 8, pp. 291–300, dec 1989.

[13] R. Chellappa and R. Kashyap, "Texture synthesis using 2-d noncausal autoregressive models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, pp. 194–203, feb 1985.

[14] J. Francos, A. Meiri, and B. Porat, "A unified texture model based on a 2-d wold-like decomposition," *IEEE Trans. Signal Process.*, vol. 41, no. 8, pp. 2665–2678, 1993.

[15] A. Gagalowicz, "Texture modelling applications," *The Visual Computer*, vol. 3, pp. 186–200, dec 1987.

[16] B. Burns, *Percepts, concepts, and categories the representation and processing of information*. Amsterdam New York: North-Holland, 1992.

[17] T. Malzbender and S. Spach, "A context sensitive texture nib," in *Communicating with Virtual Worlds*, pp. 151–163, Springer Japan, 1993.

[18] K. Popat and R. W. Picard, "Novel cluster-based probability model for texture synthesis, classification, and compression," in *Visual Communications and Image Processing 93* (B. G. Haskell and H.-M. Hang, eds.), SPIE, oct 1993.

[19] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 2274–2282, nov 2012.

[20] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *2009 IEEE 12th International Conference on Computer Vision*, IEEE, sep 2009.

[21] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes, "Layered object detection for multi-class segmentation," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2010.

[22] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller, "Multi-class segmentation with relative location prior," *Int J Comput Vis*, vol. 80, pp. 300–316, may 2008.

[23] C. L. Zitnick and S. B. Kang, "Stereo for image-based rendering using image over-segmentation," *Int J Comput Vis*, vol. 75, pp. 49–65, feb 2007.

[24] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum, "Lazy snapping," in *ACM SIGGRAPH 2004 Papers on - SIGGRAPH 04*, ACM Press, 2004.

[25] G. Mori, "Guiding model search using segmentation," in *Tenth IEEE International Conference on Computer Vision (ICCV05) Volume 1*, IEEE, 2005.

[26] B. McFee, *More like this machine learning approaches to music similarity*. La Jolla: University of California, San Diego, 2012.

[27] J. Wills, S. Agarwal, D. Kriegman, and S. Belongie, "Toward a perceptual space for gloss," *ACM Transactions on Graphics*, vol. 28, no. 4, pp. 1–15, 2009.

[28] M. J. Wilber, I. S. Kwak, and S. J. Belongie, "Cost-effective hits for relative similarity comparisons," in *Proceedings of the Seconf AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2014, November 2-4, 2014, Pittsburgh, Pennsylvania, USA*, 2014.

[29] J. M. Wolfe, "Guided search 2.0 a revised model of visual search," *Psychonomic Bulletin & Review*, vol. 1, pp. 202–238, jun 1994.

[30] C. Wah, G. V. Horn, S. Branson, S. Maji, P. Perona, and S. Belongie, "Similarity comparisons for interactive fine-grained categorization," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2014.

[31] O. Tamuz, C. Liu, S. Belongie, O. Shamir, and A. T. Kalai, "Adaptively learning the crowd kernel," *CoRR*, vol. abs/1105.1033, 2011.

[32] Y. Chen, "Texture landmark detection in mouse brain images using signicance-based boosting (paper)," 2014.

[33] A. P. refer to link for full list: http://dx.doi.org/10.1038/nature05453, "Genome-wide atlas of gene expression in the adult mouse brain," *Nature*, vol. 445, pp. 168–176, dec 2006.

[34] P. A. Yushkevich, B. B. Avants, L. Ng, M. Hawrylycz, P. D. Burstein, H. Zhang, and J. C. Gee, "3d mouse brain reconstruction from histology using a coarse-to-fine approach," in *Biomedical Image Registration*, pp. 230–237, Springer Science + Business Media, 2006.

[35] G. A. Johnson, A. Badea, J. Brandenburg, G. Cofer, B. Fubara, S. Liu, and J. Nissanov, "Waxholm space: An image-based reference for coordinating mouse brain research," *NeuroImage*, vol. 53, pp. 365–372, nov 2010.

[36] S. Ourselin, A. Roche, G. Subsol, X. Pennec, and N. Ayache, "Reconstructing a 3d structure from serial histological sections," *Image and Vision Computing*, vol. 19, pp. 25–31, jan 2001.

[37] N. Roberts, D. Magee, Y. Song, K. Brabazon, M. Shires, D. Crellin, N. M. Orsi, R. Quirke, P. Quirke, and D. Treanor, "Toward routine use of 3d histopathology as a research tool," *The American Journal of Pathology*, vol. 180, pp. 1835–1842, may 2012.

[38] S. Sun, N. D. Magalhães, and N. D. Cahill, "Nearly rigid descriptor-based matching for volume reconstruction from histological sections," in *Medical Imaging 2012: Image Processing* (D. R. Haynor and S. Ourselin, eds.), SPIE, feb 2012.

[39] U. Kurkure, Y. H. Le, N. Paragios, J. P. Carson, T. Ju, and I. A. Kakadiaris, "Landmark/image-based deformable registration of gene expression data," in *CVPR 2011*, IEEE, jun 2011.

[40] W. authors and references, "Penrose tiling explained." [Online]. Available: http://en.wikipedia.org/wiki/Penrose_tiling.

[41] W. authors and references, "K-means clustering explained." [Online]. Available: http://en.wikipedia.org/wiki/K-means_clustering.

[42] W. authors and references, "Chi-squared test explained." [Online]. Available: http://en.wikipedia.org/wiki/Chi-squared_test.

[43] W. authors and references, "Isoperimetic quotient explained." [Online]. Available: http://mathworld.wolfram.com/IsoperimetricQuotient.html.

[44] S. Belongie, J. Malik, and J. Puzicha, "Shape context: A new descriptor for shape matching and object recognition," in *In NIPS*, pp. 831–837, 2000.

[45] W. authors and references, "The hungarian algorithm explained." [Online]. Available: http://en.wikipedia.org/wiki/Hungarian_algorithm.

[46] W. authors and references, "The greedy snake algorithm explained." [Online]. Available: http://en.wikipedia.org/wiki/Active_contour_model.

[47] D. Taubman, *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Boston, MA: Springer US, 2002.