University of Nevada, Reno

Retinal Vessel Segmentation using Tensor Voting

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science and Engineering

by

Daniel Farmer

Dr. George Bebis, Advisor

Dr. Mircea Nicolescu, Co-Advisor

May, 2015

THE GRADUATE SCHOOL

We recommend that the thesis

prepared under our supervision by

DANIEL FARMER

Entitled

Retinal Vessel Segmentation Using Tensor Voting

be accepted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

George Bebis, Ph D., Advisor

Mircea Nicolescu, Ph D., Co-Advisor

Ilya Zaliapin, Ph D., Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

May, 2015

Abstract

Medical imaging studies generate tremendous amounts of data that are reviewed manually by physicians every day. Medical image segmentation aims to automate the process of extracting (segmenting) "interesting" structures from background structures in the images, saving physicians time and opening the door to more sophisticated analysis such as automatically correlating studies over time. This work focuses on segmenting blood vessels (in particular the retinal vasculature), a task that requires integrating both local and global properties of the vasculature to produce good quality segmentations. We use the Tensor Voting framework as it naturally groups structures together based on the consensus of locally voting segments. We investigate several ways of encoding the image data as tensors and compare our results quantitatively with a publically available hand-labeled data set. We demonstrate competitive performance versus previously published techniques.

Acknowledgements

I would like to acknowledge a few people that have played pivotal roles in my graduate education. My advisors, Dr. George Bebis and Dr. Mircea Nicolescu, have taught me much and have been extremely patient with me while I tried to balance working and studying. I would also like to thank Dr. Ilya Zaliapin for taking time out to serve on my thesis committee.

I would also like to thank Dr. Fabien Scalzo of the Neurovascular Research Core at the University of Californa, Los Angeles for suggesting this research topic.

Finally, I need to thank my wife, Ashlee, for her patience and support throughout this process and for taking care of our boys while I had to work on my research.

Contents

List of Tables

List of Figures

Introduction

Few areas of image processing have the kind of impact that medical image processing does. From performing reconstructions from MRI and CT scans to contrast enhancement of X-rays to techniques aimed at allowing more automated diagnoses by physicians, advancements in medical image processing have the potential to save lives and to save medical facilities time and money by realizing improved efficiency in delivery of care.

One area of particular difficulty is the process of segmenting blood vessels from medical images. As a brief review, the human vascular system is composed of arteries which carry oxygenated blood from the heart out the the extremities; capillaries, where gases and nutrients are exchanged; and veins, which carry deoxygenated blood back to the heart and lungs.

Imaging of the vasculature is carried out for many reasons: to detect blood clots in veins, to diagnose stenosis (narrowing) of arteries, to monitor cerebral vessels for post-operative vasospasm, and for early detection of atheroscelrosis (plaque build up on the walls of the arteries), and to support many other diagnoses. These imaging studies are carried out using a wide-variety of imaging modalities as well: ultrasound, fluoroscopy, computed tomography (CT) angiography (using X-rays), magnetic resonance (MR) angiography, and in rare cases even visible light can be used. In particular that is the case with retinal fundus imaging, the application area of this work.

Many typical approaches to segmentation rely on texture information or the fact that objects of interest (like cars and people) form closed contours in images. This is not the case with blood vessels. Blood vessels do not necessarily follow simple paths, but instead can be “tortuous” as clinicians describe them. In addition the size (or

caliber) of vessels varies tremendously, from millimeter wide capillaries to the 300mm aorta. The changes in size serve an important physiological purpose: to maintain sufficient blood pressure to perfuse the tissues in the body, but they make the task of automatically finding vessels difficult.

Our particular interests are, long-term, in the cerebro-vasculature (the brain's blood supply), and more immediately (in this work) segmenting blood vessels from retinal fundus images. Retinal fundus photography uses visible light to image the interior of the eye, allowing the physician to inspect the vasculature, the optic disc, and the retina. There are also several kinds of lesions that a physician might be interested such as hard exudates (See Figure 1) which are lipids deposited in the eye from leaking capillaries. The patient in this image most likely has diabetes and/or hypertension (high blood pressure) We are not focused on detecting these other structures in this work, but they do provide additional challenges for our vessel segmentation procedure because along with the blood vessels they are also "salient" objects that we must not confuse with vessels.

Compared with other imaging modalities retinal fundus images have readily available benchmark data to enable us to validate the direction of our research before moving on to more challenging types of images such as computed tomography (CT) angiograms of the cerebro-vasculature.

Retinal fundus images are essentially visible light photographs of the eye taken through a microscope lens. As a result they present several challenges to effectively analyze (refer to Figure 1 again as it exhibits nearly all of these problems)

1. Aperture effects from the lens (the sharp circular border at the edge of the eye pixels)

Figure 1: Challenging DRIVE image showing vignetting from poor lighting, the presence of exudates (the yellow spots), and central vessel reflex.

2. Uneven illumination from the light source on the camera

3. Pathology (hard exudates, red lesions, etc.)

4. Extravascular structures (the dark macula in the center of the image, the bright optic disc).

5. The irregular structure of the vessels (for example, try to image the convex hull that contains all of the vessel pixels in Figure 1)

Our specific approach to medical image segmentation will use the ideas of perceptual grouping as implemented in the Tensor Voting framework [10]. Perceptual grouping refers to the goal of comprehending higher level objects from their lower level parts (at the lowest level, from the pixels in the image). The Tensor Voting

framework refers to the work done in Gerard Medioni's group at USC to encode geometric knowledge about the scene using tensor representations and then propagating that information throughout the image using a voting process. In this work our contributions are an encoding scheme for segmenting thick vessel segments (rather than extracting only the border of an object) by merging information from morphological operations and edge gradient information. We have additionally implemented a software framework in Python for easily performing grid search of the parameter space for voting. We will discuss Tensor Voting in detail in the following sections, but for now let us say that we will use image processing techniques to deal with the first and second problems above and we believe that Tensor Voting can compensate for the others.

Background

Due to the existence of two standard datasets (with labeled ground truth) retinal vessel segmentation is a very popular area of research with many publications to compare with. These works can be divided in many ways, but one of the most simple divisions is to compare learning-based (supervised) approaches to unsupervised techniques. Supervised techniques are those that require training data with labeled ground truth data so that a mapping can be learned from the feature inputs (such as edges strength, color intensity, etc.) to the classification output (in this case vessel or not-vessel).

Unsupervised methods

All classification systems perform image processing to extract features that we hope will make it easier for the software to correctly label each pixel as vessel or not vessel, however unsupervised systems (like this work) do not require additional training data to develop the classification criteria to assign the label. Sometimes this is simply because the decision criteria is implicit or hand-tuned (e.g., a hand chosen threshold value).

One of the earliest approaches published used matched filters at 12 orientations to approximate linear segments of blood vessels at all orientations [4] unlike the other papers, this was implemented in hardware and takes a very classical signal processing approach to extracting vessels, its strengths are its simplicity. The method (along with hand-tuned parameters) builds a relatively simple model that still produces qualitatively good results. It does fail to account for illumination conditions both uneven illumination, and the central vessel reflex.

Zana and Klein applied sophisticated morphological processing based on a Gaussian model of (the profile of) blood vessels these included a sum of directional top-hat filters and what they call geodesic reconstruction [25] this amounts to a sum of top hat filters then filtering noisy segments by curvature (using the Laplacian).

[7] dealt with the uneven illumination and irregular vessel structures using an adaptive thresholding scheme with a verification procedure to reject non-vessel points. The verification procedure is where the prior information about the shape characteristics of the blood vessels are introduced. Specifically, for each chosen threshold they calculate the distance transform they then "prune" the result to find the vessel centerlines using critera based on angle, width, contrast and a minimum size. They then

reconstruct the final image by taking the logical OR of all of the processed images at each threshold value. Finally they post-process the images using non-maximal suppression by calculating the gradient magnitude and orientation using the Sobel operator and then for each orientation removing the pixels that do not have the greatest gradient magnitude.

[11] implemented another variant on morphological reconstruction. They preprocess the image to account for irregular illumination by subtracting the resulting image from convolving with a large arithmetic mean kernel, then enhance small vessels by adding the strongest result of convolving the image with line detecting features at 0, 45, 90, and 135 degrees at each pixel. Like [7] they then extract center lines, in this case with 4 Difference of Offset Gaussian filters which are then connected by region growing and filtered by a validation procedure. The authors then construct a multi-scale representation using the morphological top-hat filter (albeit a modified version, to try to suppress noise) using circular structuring elements with radii from 1 to 8 pixels. Finally the authors perform another round of vessel filling using a so-called "Double Threshold operator" and post-process by considering pixels with 6 out of 8 connected neighbors that are vessel pixels to also be labeled as vessel pixels.

The unsupervised approaches are very similar to each other in that they each attack the known difficulties with these images and attempt to use the geometry of the vessels to guide the labeling of pixels. Every paper so far has applied operations to compensate for uneven illumination and/or aperture effects, find the center-line of the vessels and then, having identified the most easily labeled vessel pixels they then extend out to try to label the rest of the vessel (this is another challenge of this data set, it is not sufficient to generate thin line boundaries, but one must instead densely label the vessel).

An important disadvantage of the proposed methods is that most of them take ad-hoc approaches to trying to find the paths of the vessels, as a result they have many thresholds and parameters that must be tuned (and in general which hurt the prospects that these systems could ultimately be deployed in the real world with varying hardware and environmental conditions). Additionally we have seen that the methods above assume an explicit model (e.g., Gaussian intensity profile) for the shape of the vessels. We hope to show in this thesis that with very simple feature extraction and the Tensor Voting framework that we have implemented a system that is conceptually similar to these related works, but which naturally and elegantly expresses these constraints on shape while being model-free with respect to the shape of the vessels (and which can easily be extended to 3D data).

Supervised learning methods

In contrast with the unsupervised techniques, supervised learning methods require labeled ground truth data and pre-training to adapt the system to the task at hand, in this case vessel pixel segmentation. Supervised methods tend to follow the same pattern: the problem is formulated as a binary classification task (vessel vs not vessel). Image features are hand-engineered and then a machine learning classifier is trained to map from those features (such as gradient information, interest point descriptors, responses to image processing filters like Gabor wavelets, etc.) to either a probability that that pixel is a vessel or directly to the binary classification (vessel / not vessel).

For example, [5] computes features based on pixel responses to Gaussian Derivative filters, then applies an interesting result to "rotate" feature vectors via multiplication with a linear operator. They use this idea to achieve rotation invariance

(since the blood vessels appear at all orientations). With feature vectors in hand they then apply Support Vector Machines (SVM) as its classifier. One downfall of their approach is that by not explicitly modelling the vessels, but instead trying to learn their features they developed a classifier that also produced a strong result from the optic disc giving them many false positive detections across the testing data set.

[21] introduced the DRIVE database to test their system which started with ridge detection based on the curvature (using the Hessian) to find the center-lines. These lines are then grouped together into convex sets by an region growing process that looks in a neighborhood around the pixel and check (via the eigenvectors) that the direction of adjacent lines are similar and that they are not on parallel lines. Interestingly this is like an ad-hoc test for the perceptual grouping employed by tensor voting. The authors then extract 18 different features based on color and various properties of the detected ridge lines and convex sets. These features are fed into a k-NN classifier (with k = 101 in their case). Their system produces excellent results with an area under the curve score of 0.95 however the disadvantages of using a k-NN classifier are classification time (at the time of publication it took 15 minutes to classify an image on a 1 Ghz Pentium computer), additionally k-NN classifiers require large amounts of memory because their "training" regime simply consists of memorizing all of the training data. This was not a problem for them, however they mention that handling certain failure cases could be improved with more training data, however this would directly increase the memory requirements of the system since there is no "compression."

[19] implements their own preprocessing to deal with camera aperture issues by extending the border of the image (by replicating pixel values), they invert the green channel and then compute the Wavelet transform using 2D Gabor wavelet because

of their good properties for localizing details. The Gabor wavelet can be steered and so they compute the transform at from 0 degrees to 180 degrees in steps of 10 degrees and for each pixel select the maximum Wavelet response as that pixels feature value. They pass the pixel intensity value and maximum Wavelet response into two classifiers, a Gaussian mixture model classifer and through logistic regression.

[20] is more closely related to our work in the sense that his features are simple (matched filters) and he trains a conditional probability density function using histograms to use as the decision criteria for assigning a likelihood ratio to the vessel vs non-vessel classification problem.

With a few exceptions the supervised learning approaches produce outstanding results. Various combinations of features and learning have all produced results around 0.95 area under the ROC curve (our metric of choice because it is widely reported by others). However, supervised approaches require labeled data to train with, and depending on their features may need to be retrained for example, for systems with different camera Field of View (FOV), or other changes in the input data versus the training data. We will come back to this point in the conclusion.

The Tensor Voting Framework

Tensor voting is an algorithm for discovering "salient" structures in multi-dimensional data. In our case this is image data (in 2D) but it has been applied in higher dimensional spaces for tasks like stereo reconstruction (3D, [13]), motion analysis (4D, [15]), and even epipolar geometry (estimating the fundamental matrix, 8D [22]) and manifold learning [14]. The English word salient means noticeable or important, in the context of Tensor Voting this is still the case, but in particular we make this

concrete by defining it in terms of two principles defined by Gestalt psychology [24]: proximity and good continuation. We try to extract important (salient) structures in the image (globally) by looking locally for agreement among tensors about the structures that they are a part of.

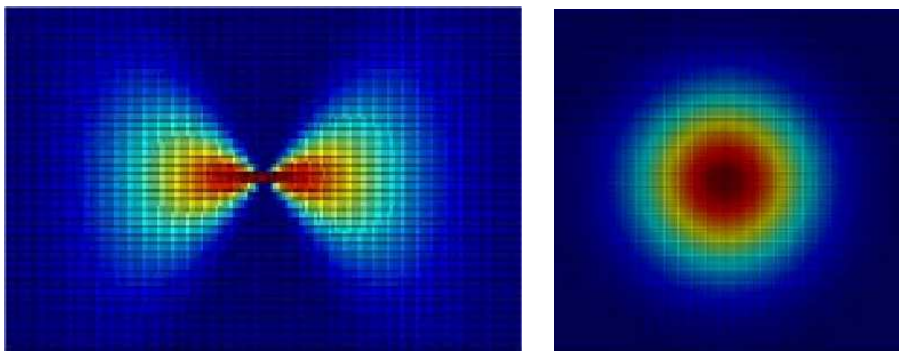
Tensor Representation

Tensor Voting consists of two parts: the tensor representation of the data and the voting procedure for propagating information about orientation and saliency. In the general case each tensor in tensor voting is represented by a $D \times D$ symmetric positive semi-definite matrix. If we consider the 2D case then these matrices can be interpreted visually as ellipses whose major and minor axes are given by the eigenvalues, λ_1, λ_2, of the 2×2 matrix and whose directions are similarly encoded in the eigenvectors, e_1, e_2 of the matrix. We consider these in sorted order such that $\lambda_1 \geq \lambda_2$ and e_1 and e_2 are the respective eigenvectors of λ_1 and λ_2. This describes a "generic tensor." The tensor can be represented as a matrix S (equation 1)

$$S = \lambda_1 \cdot e_1 e_1^T + \lambda_2 \cdot e_2 e_2^T \tag{1}$$

There are two "fundamental" tensors in tensor voting that represent the two types of features that exist in 2D data (geometrically speaking): curves and points.

The first is the *ball tensor*, the tensor representation of a point. It is represented by the $D \times D$ identity matrix with $\lambda_1 = \lambda_2 = 1$. Ball tensors encode complete uncertainty about the orientation of the structure that the tensor is located on. The



(a) Stick voting field (b) Ball voting field

Figure 2: The fundamental tensors from [8]

other end of the spectrum is the *stick tensor*. This tensor is represented by a tensor with e_1 = the normal vector of the curve (and e_2 is the tangent), $\lambda_1 = 1$ and $\lambda_2 = 0$. It has only one non-zero eigenvalue and it's normal direction is in the direction of the associated eigenvector (it represents absolute certainty about the orientation of the structure it is on). Between these two extremes are the generic tensors that we began with, they may encode a preferred orientation with one eigenvalue larger than the other (and therefore the orientation described by the associated eigenvector) and additionally the salience of a particular generic tensor (in 2D) is given by the difference of the eigenvalues $(\lambda_1 - \lambda_2)$.

In the most simple use of tensor voting we encode our points as ball tensors or if we have edge information we can encode them as stick tensors.

Vote Propagation

The voting portion of tensor voting refers to the way that salient structures are extracted by local analysis. Voting happens in a pairwise fashion between a receiver tensor and each of its neighbors within a neighborhood determined by the only free

parameter in the algorithm: σ. Each neighbor "votes" with its orientation. The vote is weighted by the distance and curvature ρ between the voter and the votee. This is given by the so-called "Saliency Decay Function" (Equation 2)

$$VS(\overrightarrow{d}) = \exp(-\frac{|\overrightarrow{d}|^2 + c \cdot \rho^2}{\sigma^2}) \tag{2}$$

$$V_{ball}(\overrightarrow{d}) = \int_0^{2\pi} R_\theta V_{stick}(R_\theta^{-1}\overrightarrow{d}) R_\theta^T d\theta \tag{3}$$

We can try to get an intuitive feel for equation 2 by first noticing that it appears to be a Gaussian. So we know that the strength of votes decay exponentially with distance between voter and receiver.

d is the distance from voter to receiver. ρ is the curvature between voter and receiver. If we consider just the voting point and think back to basic calculus, we know that the tangent line is the line that best approximates a curve at a point. With tensor voting we extend this to consider the osculating circle, which is the *circle* that best approxmates the curve at a point (because we want the best smooth path connecting points). c is a constant that controls the interaction of distance and curvature, in our C++ implementation of tensor voting $c = \frac{(\sigma-1)\times -log(0.1)}{\frac{\pi}{4}^2}$. Refer back to Figure 2a once more and notice there is one other key difference, the field is truncated at the 45 degree angles giving it a barbell type of shape. This is another measure to try to ensure smooth continuation (and additionally a performance optimization since perpendicular votes would have been penalized due to the curvature anyway). This

is the stick voting case.

On the other hand the ball tensor vote is conceptually the integration of a rotating stick tensor (Equation 3). It is isotropic and so essentially votes strictly based on distance to it's neighbors. It is good for estimating density primarily.

Each tensor "collects votes" via tensor addition (i.e., element-wise addition) resulting in a generic tensor as described above. Consider Figure 3: each receiver (Q, Q', Q'') receives votes with the orientation that they should have according to P with saliency strength proportional to the distance and curvature from P. If P is on a consistent structure then the Q tensors will receive many consistent votes from P and P's neighbors and they will all agree that they are on the same structure. On the other hand, if P is a noisy segment it will still contribute the same vote, but presumably neighbor tensors that share a common structure with the Q's will overwhelm the information received from the noisy voter.

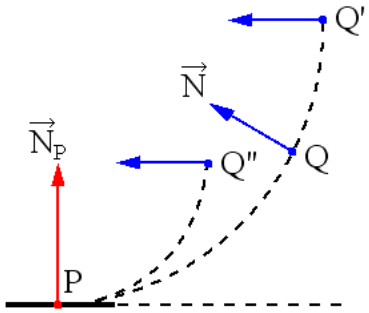


Figure 3: Collecting votes, the blue arrows denote the orientation of the Q's according to P

In this work we also follow the work of [8] in implementing iterative removal of low salience segments to improve the results of tensor voting. Given a number of iterations, i, and a threshold value T_s we encode our tensors and save a copy then perform each round of tensor voting. After each iteration we calculate the curve

salience value, $\lambda_1 - \lambda_2$, for all tensors then remove those tensors that fall below T_s from the originally encoded tensors and then continue to the next iteration (voting again from the filtered original tensor values). Through this process we progressively improve our confidence in the tensors that exist on consistent structures (i.e., blood vessels in our case).

Typical image processing techniques might look at something like the image gradient as computed by the Sobel operator to find simple salient features (i.e., edges) in the image. The gradient of the image contains one piece of geometric information: the orientation of the vector that is normal (perpendicular) to the curve at that point. Tensor voting extends things from this vector representation to a more general tensor (in 2D this is a matrix). Now continuing the example above we encode not only the orientation of the gradient in the normal direction, but we also encode the tangent information as well as the magnitude at each point (tensor). This gives us our tensor representation, it also extends to arbitrary dimensional spaces (N-D) but for clarity of exposition we will not explore that further and instead refer the interested reader to [14]. Without belaboring the point it might be worthwhile to contextualize this one more time: a scalar encodes magnitude, a vector encodes magnitude and direction, our tensor representation encodes direction (normal and tangent), magnitude and salience (or this can be considered something like confidence). So we can see how this is a richer representation.

Proposed Methodology

Preprocessing

We begin by preprocessing our image (see Figure 10a for the raw image). We extract the green channel based on the empirical observation that it provides the best contrast and least noise of the three color channels (Figure 4). Additionally reducing the problem to a grayscale image more closely mimics the types of images encountered in medical imaging (which is typically not done with visible light and therefore color information is usually not available).

(a) The red channel (b) The green channel (c) The blue channel

Figure 4: Viewing each color channel separately clearly shows the best contrast in the green channel.

We then perform morphological closing ($\bullet$) with a circular structuring element (SE) to model the uneven illumination. We will denote the green channel image as G see Figure 5. Finally we subtract the green channel image from the result of morphological closing, this is known as the black top-hat transform (Equation 4) and is illustrated in Figure 6a. Although Figure 6a looks promising there is unfortunately still quite a bit of noise left behind, this is more evident when the image contrast is

enhanced (Figure 6b)

$$TH(G) = (G \bullet SE) - G \tag{4}$$

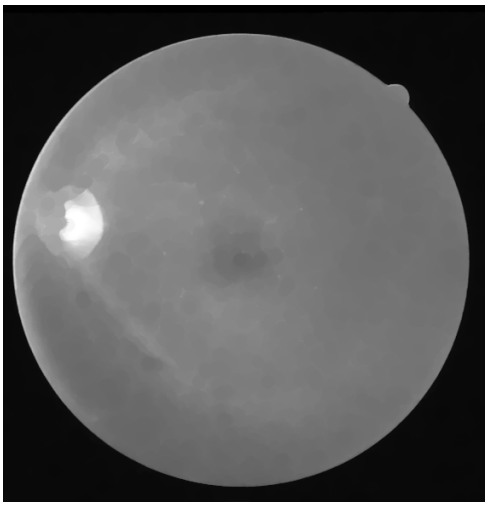
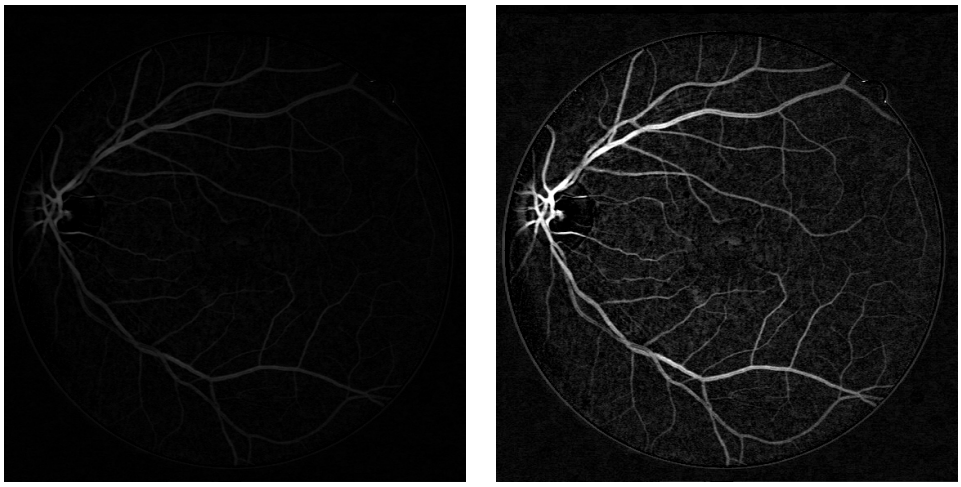


Figure 5: Grayscale morphological closing produces a good model of the background.

The result of the black top-hat transform gives us the intensity values that we use to initialize our ball tensors for the interior vessel pixels. As we will explain in Section we can actually perform tensor voting directly at this stage, however given the amount of noise it is helpful to guide the process more by using the strong edge pixels of the vessel walls and encoding them as stick tensors. This shows one of the strengths of tensor voting, that we are able to use one unified framework to encode both of these types of information. As an optimization we also threshold the intensity values of the black top-hat image to remove excessively noisy segments. This is merely an attempt to prevent encoding and voting on tensors that were a priori unlikely to belong to a vessel.



(a) The result of the black top-hat transform.

(b) Contrast enhanced result of the black top-hat.

Figure 6: Black top-hat results

We use a combination of the Sobel gradient operator and the Canny edge detector [3] to extract edge information from the image. In principle this could be done in one step with a modified Canny procedure (because the Canny algorithm computes this same information during the course of generating its output) but for simplicity of implementation we perform the steps separately. The Canny edge detector gives us better detections of the thinnest end vessels due to the use of hysteresis (completing edges).

We convolve the image with directional derivative filters (Sobel's kernel) to estimate the gradient of the image in the x and y directions, G_x, G_y. We then follow the well known procedure to compute the gradient magnitude (Equation 5) image (See Figure 7 for an example) and calculate the orientation of the edges (Equation 6).

$$\|\nabla G\| = \sqrt{G_x^2 + G_y^2} \tag{5}$$

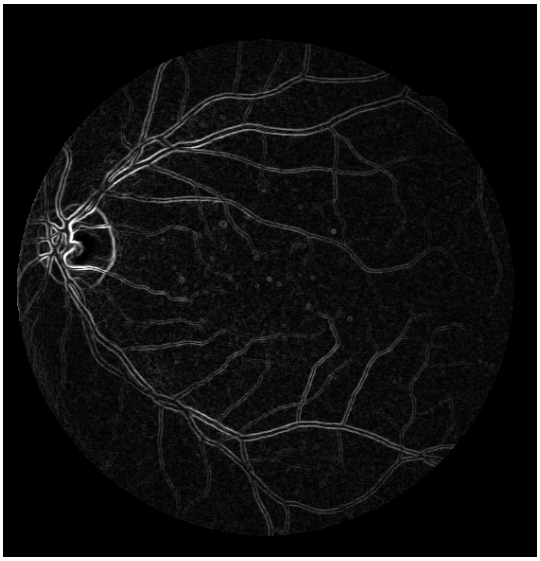


Figure 7: Gradient magnitude edges of the retinal fundus image

$$\Theta = arctan(\frac{G_y}{G_x}) \tag{6}$$

Finally we use the gradient magnitude to decide which edges are strong enough to be encoded as stick tensors and then create their tensor representation with $e_1 = [cos(\Theta), sin(\Theta)]$, e_2 should be orthogonal, since we are in 2D this is simply $e_2 = [-sin(\Theta), cos(\Theta)]$. All of these operations are carried out in an element-wise fashion across the whole image.

Our contribution here is to steer the voting process by using ball tensors on the interior of the vessel (to find the densest regions of vessel pixels) with stick tensors on the vessel walls to guide the process using better estimation of the vessel orientation.

Figure 8: A schematic representation of decomposing an elliptical generic tensor into a ball component and a stick component

Applying and Interpreting Tensor Voting

With the image input cleaned up and encoded as tensors we now apply tensor voting on the input (stick and ball) tensors, and the output of the process are generic tensors as described previously.

The fundamental idea is that any generic tensor can be decomposed into its stick and ball components. We depict it graphically in Figure 8.

More formally Equation 7 shows the first term $\lambda_1 - \lambda_2$ (the stick salience) multiplied by the outer product of e_1, which is the normal vector to the curve, so this is our stick component. The 2nd term incorporates both eigenvectors and is scaled by λ_2, the ball salience.

$$T = (\lambda_1 - \lambda_2)e_1e_1^T + \lambda_2(e_1e_1^T + e_2e_2^T) \tag{7}$$

In 2D tensor voting, as mentioned previously, we get two eigenvalues when we decompose a generic tensor. Analzying these values gives us three cases:

1. $\lambda_1 - \lambda_2 > \lambda_2$ - High stick saliency, probably on a curve.
2. $\lambda_1 \approx \lambda_2 > 0$ - Uncertain direction but high saliency implies a junction
3. $\lambda_1 \approx \lambda_2 \approx 0$ - Low saliency overall implies this is noise (an outlier)

In the following section we will review the results of our experiments and when we display result images we are visualizing the stick saliency as a gray-level intensity value.

Iterative Voting

To further improve results of the system we have implemented iterative voting following the work of [8]. For 2D figures this idea extends back to [23]. As before we pre-process the images and encode the edge pixels as stick tensors and non-edge pixels (i.e., the center of the vessels, noisy segments, the optic disc) as ball tensors. Our idea now is that after a round of tensor voting tensors that are likely to be on a common structure will have increased their salience more than isolated noisy segments. Rather than directly (and optimally) try to threshold the salience map at this stage, instead we can choose a very conservative threshold value and eliminate tensors that were not sufficiently salient after the first round of voting.

At each iteration, i, of voting we calculate the saliency map for T_i then threshold low salience segments, we then remove the low salience tensors from the original tensor list (i.e., T_0) and then vote again using the filtered version of T_0. Over time there is a greater and greater separation between the salience of noisy segments and vessel segments as demonstrated in the results section.

Implementation

In this section we will briefly review the implementation of the software. The foundation of the system is the original Tensor Voting application from Dr. Medioni's

lab, courtesy of Dr. Mircea Nicolescu. This application is a generic C++ library that includes the implementation of tensor voting and eigendecomposition as well as a Microsoft Foundation Class (MFC) based Windows application for the user interface.

For our purposes, knowing that we would be running many experiments, we modified this application to remove the GUI (making it portable to Linux in the process). We also conducted profiling of the code and optimized some IO code to improve the speed by approximately 20%.

To facilitate faster prototyping we then implemented a Python based wrapper around the Tensor Voting C++ code. Over the course of this research this system evolved considerably, from initially being a script that wrote out a serialized form of the tensors (hence the improvements to IO mentioned before), to a cross-platform GUI application using Qt (Figure 9).

In its final form we implemented a Cython [2] binding directly between the Tensor Voting library and Python, which allowed us to directly communicate using shared memory rather than files and eliminated the need for messing text file parsing. We also returned to the script format, but implemented a Pipeline approach with facilities for grid search of parameters inspired by the scikit-learn project [17]. This is the software that has produced the tables of results in the next section.

The use of the pipeline simply requires code like

```
X = np.load('drive_images.npy')
image_pipe = Pipeline([('tophat', TophatCannyPreprocessor())
                      ,('tenslist', ImageTenslistAdapter(5.0))
                      ,('vote', IterativeTensorVotingTransformer(9.0, 4, 3))
                      ])
```

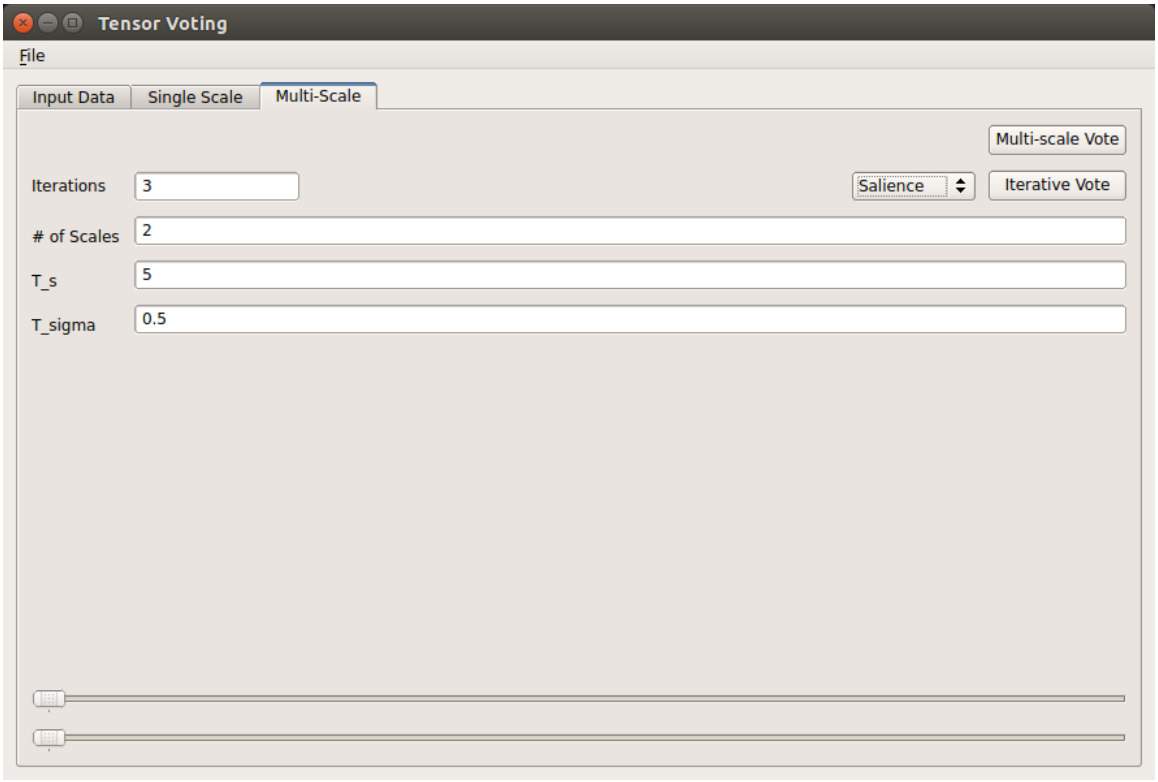


Figure 9: Screenshot of the GUI incarnation of this work

```
voting_result = image_pipe.transform(X)
```

The code for the pipeline clearly mirrors the approach proposed in this paper, making it very easy to swap components while maintaining an uniform interface in between. There are Preprocessor objects that correspond to each strategy described in this thesis, the ImageTenslistAdapter carries out the transformation from pixels to tensors (via the gradient and intensity methodology described previously), and finally there are TensorVotingTransformers which carry out either single scale, or iterative tensor voting on the tensors encoded by the adapter.

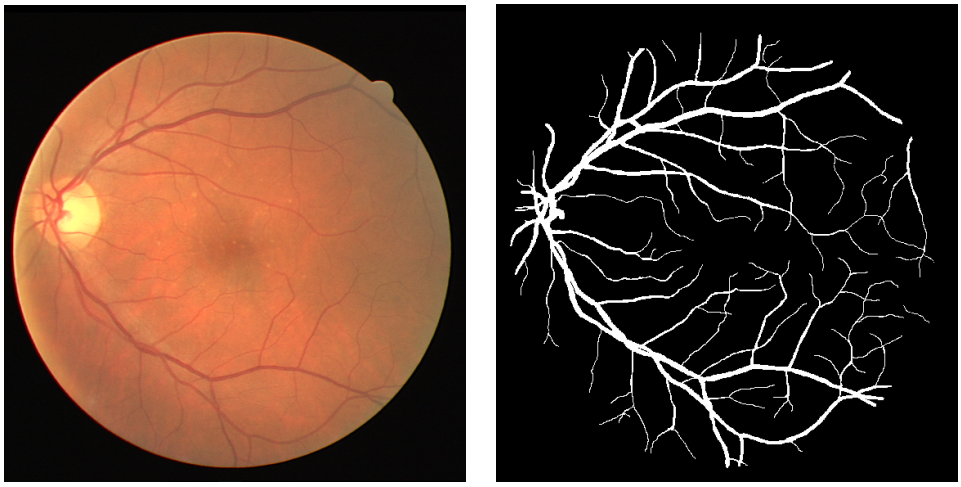
Each step is named in the above code ('tophat', 'tenslist', 'vote) because the system uses introspection to allow the programmer to specify ranges of parameters for each component by name, then it will compute and log all of the results.

Experimental procedure and results

Retinal Fundus Imaging

There are two ground truth data sets that are used by most researchers in the field: the DRIVE data set and the STARE [6] dataset. The anatomic variability described above manifests itself in the images of the vessels as tremendous variability in size ranging from approximately 1 to 12 pixels in diameter.

There are 40 images in the DRIVE data set split into 20 training images and 20 tests images. Figure 10 shows an example image with its corresponding ground truth labeling.



(a) Color fundus image (b) Hand labeled ground truth

Figure 10: An example DRIVE image and its corresponding groundtruth segmentation

> The DRIVE (Digital Retinal Images for Vessel Extraction) database, consists of a total of 40 color fundus photographs. [...] The photographs were obtained from a diabetic retinopathy screening program in The Netherlands. The screening population consisted of 453 subjects between 31 to 86 years of age. Each image has been JPEG compressed, which is common practice in screening programs. Of the 40 images in the database, 7 contain pathology, namely exudates, hemorrhages and pigment epithelium changes. [...] The images were acquired using a Canon CR5 nonmydriatic 3CCD camera with a 45 degree field of view (FOV). Each image is captured using 8 bits per color plane at 768×584 pixels. The FOV of each image is circular with a diameter of approximately 540 pixels. [16]

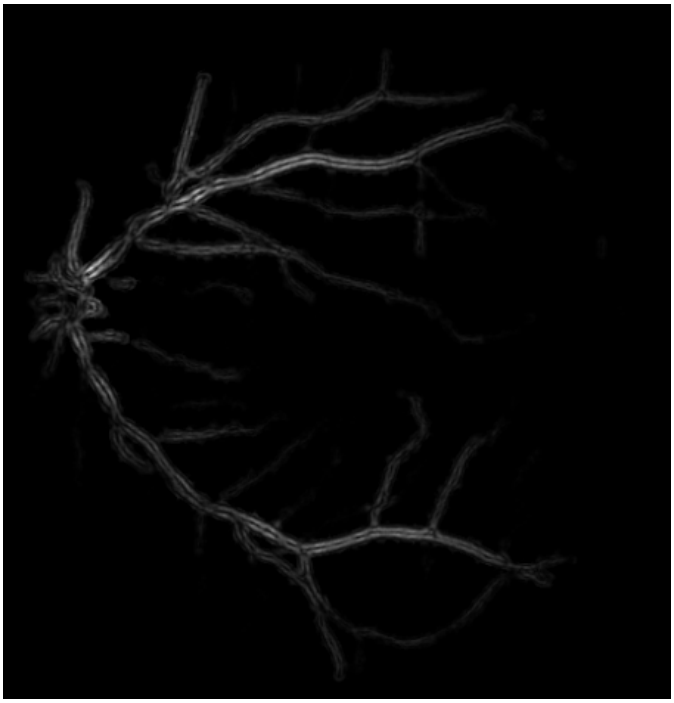


Figure 11: DRIVE image after encoding Sobel edges and voting

Tensor Voting with only stick tensors

Initially we attempted using edge detectors like the Sobel operator and the Canny edge detector by themselves to initialize voting. Although these work well, they miss the interior of the vessels as shown in Figure 11 (which are segmented in the ground truth data as seen in 10b).

Compared to the input edge data (Figure 7) we can see however that much of the noise has been suppressed and the optic disc that is visible on the left of Figure 7 (the semi-circular structure) is not present here. Unfortunately the center of the vessels and the smallest end vessels are missing.

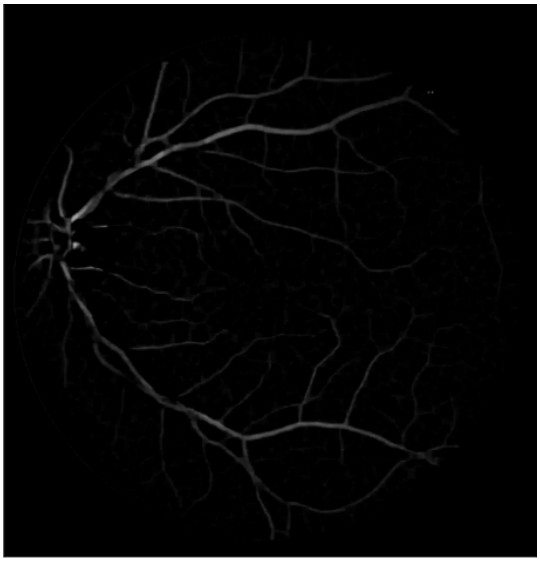


Figure 12: DRIVE image after encoding vessel with ball tensors

Tensor Voting with ball tensors

To accomodate for the lost interior vessel pixels we attempted to vote with ball tensors throughout, using the intensity of the black top-hat image (discussed in the previous section) to determine the initial saliency.

Table 1 shows the results of our experiments using ball tensors and varying the free parameter σ to determine the best size of the voting neighborhood given this choice on initialization. We also were searching for the best implementation and optimization specific parameters. In each experiment the computational cost increased so in this experiment we chose the structuring element size, in the following we selected the threshold.

The effect of σ in this case is modest.

Table 1: Results of varying σ with ball tensor voting on the training set

Vote σ	Gradient threshold	Structuring element size	AUC
2	8	(13, 13)	0.859953487
2.5	8	(13, 13)	0.861561565
3	8	(13, 13)	0.863176824
3.5	8	(13, 13)	0.864529811
4	8	(13, 13)	0.865610557
4.5	8	(13, 13)	0.866483513
5	8	(13, 13)	0.867199552
5.5	8	(13, 13)	0.867765332
6	8	(13, 13)	0.868193213
6.5	8	(13, 13)	0.868542593
2	9	(13, 13)	0.84080648
2.5	9	(13, 13)	0.841689584
3	9	(13, 13)	0.84257189
3.5	9	(13, 13)	0.843354123
4	9	(13, 13)	0.843968725
4.5	9	(13, 13)	0.844496999
5	9	(13, 13)	0.844890078
5.5	9	(13, 13)	0.845232286
6	9	(13, 13)	0.845486168
6.5	9	(13, 13)	0.845705778
2	8	(14, 14)	0.860024819
2.5	8	(14, 14)	0.86159816

Table 1: Results of varying σ with ball tensor voting on the training set

Vote σ	Gradient threshold	Structuring element size	AUC
3	8	(14, 14)	0.863187343
3.5	8	(14, 14)	0.864540403
4	8	(14, 14)	0.865618034
4.5	8	(14, 14)	0.866508101
5	8	(14, 14)	0.867191459
5.5	8	(14, 14)	0.867767472
6	8	(14, 14)	0.86820653
6.5	8	(14, 14)	0.868529446
2	9	(14, 14)	0.840779716
2.5	9	(14, 14)	0.841722306
3	9	(14, 14)	0.842557567
3.5	9	(14, 14)	0.843342227
4	9	(14, 14)	0.843988393
4.5	9	(14, 14)	0.84449337
5	9	(14, 14)	0.844895704
5.5	9	(14, 14)	0.845231426
6	9	(14, 14)	0.845497979
6.5	9	(14, 14)	0.845700744

This produces substantially better results as can be seen in Figure 12, but tends to lose very thin end vessels (which are so small as to almost only consist of edges).

Voting with stick and ball tensors

In this experiment we use the complete method described in the proposed methods section except the use of iteration. Based on the previous experiment we've restricted the size of the structuring element parameter to 13 and increased the upper limit on σ.

We can see that encoding the stick information has made a significant difference in the results

Table 2: Detailed parameter search of proposed method on the training set

Vote σ	**Gradient threshold**	**AUC**
2	3	0.779993435
2.5	3	0.787726112
3	3	0.802415061
3.5	3	0.816458224
4	3	0.831090158
4.5	3	0.847334387
5	3	0.858788663
5.5	3	0.869152561
6	3	0.877471899
6.5	3	0.883786157
7	3	0.889147886
7.5	3	0.893259303
8	3	0.896504815
8.5	3	0.899216099
9	3	0.90110371

Table 2: Detailed parameter search of proposed method on the training set

Vote σ	**Gradient threshold**	**AUC**
9.5	3	0.902659276
2	4	0.824114309
2.5	4	0.833179657
3	4	0.844802392
3.5	4	0.857753232
4	4	0.869488711
4.5	4	0.880683633
5	4	0.889160185
5.5	4	0.896714781
6	4	0.902551969
6.5	4	0.907135678
7	4	0.910642017
7.5	4	0.913471751
8	4	0.915498243
8.5	4	0.917291697
9	4	0.918542644
9.5	4	0.919340945
2	5	0.856961697
2.5	5	0.865134271
3	5	0.875293609
3.5	5	0.885512018
4	5	0.893472027

Table 2: Detailed parameter search of proposed method on the training set

Vote σ	**Gradient threshold**	**AUC**
4.5	5	0.901241274
5	5	0.907207318
5.5	5	0.912370725
6	5	0.916081636
6.5	5	0.918875711
7	5	0.921123327
7.5	5	0.922775713
8	5	0.924030631
8.5	5	0.925026027
9	5	0.9256225
9.5	5	0.926208098

Iterative Voting

Informed by the previous experiment we narrowed the space of parameters for the grid search in this experiment. We observe rapid improvement in classification accuracy initially however it does plateau quickly.

Table 3: The effect of varying σ on iterative voting results on the training set.

vote sigma	AUC
2	0.871572725
2.5	0.917430881

Table 3: The effect of varying σ on iterative voting results on the training set.

vote sigma	AUC
3	0.932194443
3.5	0.937527454
4	0.939023867
4.5	0.939707897
5	0.939891583
5.5	0.94009257
6	0.940165766
6.5	0.940230886
7	0.940236311

The results reported above demonstrate the effect of varying parameters on the training set of images for DRIVE. Choosing the best parameters from our experiments there and applying them to the test set yields an area under the ROC curve of 0.93. Figure 13 shows the corresponding ROC curve. Figure 14 shows an example binary segmentation made by thresholding the saliency image. Figures 15 shows two example "soft" segmentations, or probability maps for vessel salience (likelihood).

Comparison with other methods

The following performance numbers are taken from their respective papers, only Soares provides access to his source code so that it is possible to repeat the experiments.

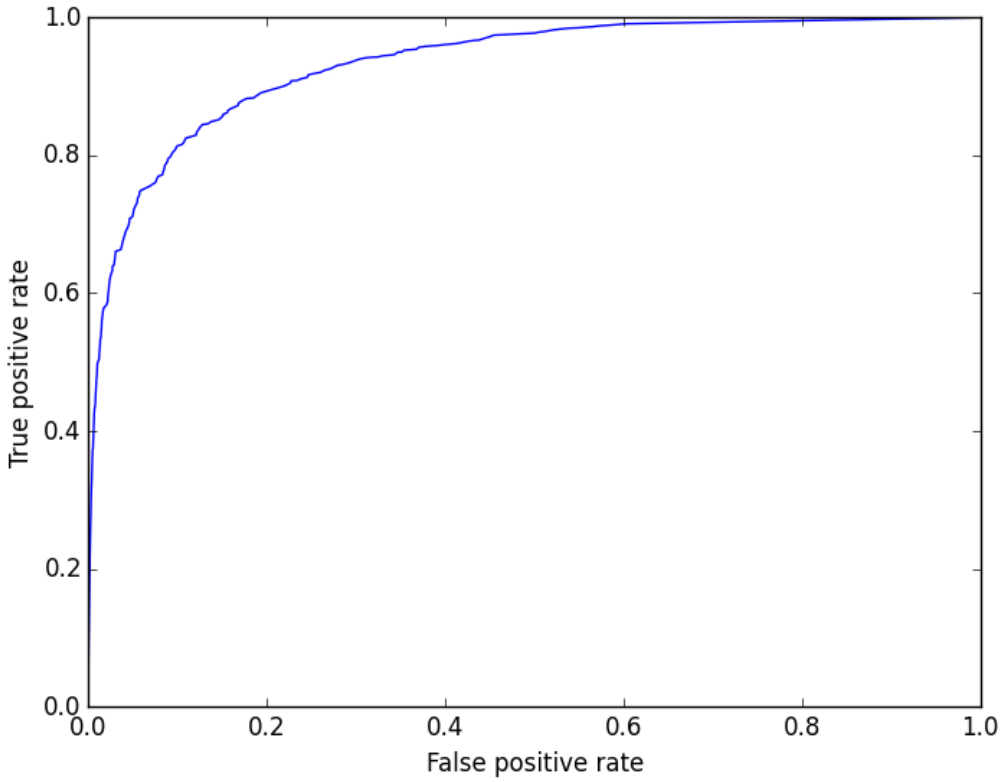


Figure 13: Receiver Operator Curve for the complete test set. AUC = 0.93

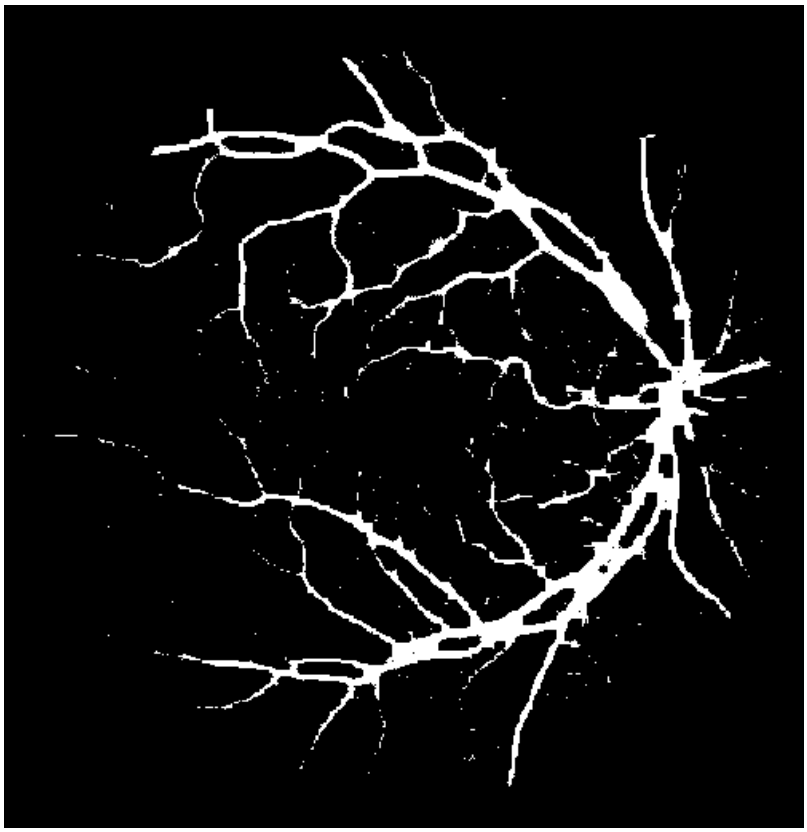


Figure 14: Binary segmentation by thresholding salience image

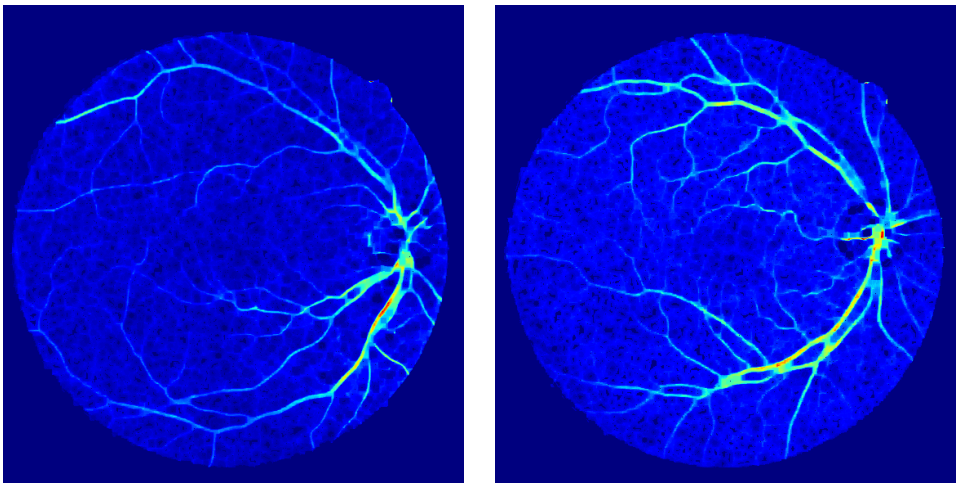


Figure 15: Example soft segmentations on the test set

Table 4: Comparison of our method vs other published approaches

Paper	**Approach**	**Area Under ROC Curve**
Chaudhuri	Unsupervised	0.79
Zana & Klein	Unsupervised	0.89
This work	Unsupervised	0.93
Ricci & Perfetti	Supervised	0.95
Soares	Supervised	0.96

The learning (discriminative) approaches are clearly able to outperform the unsupervised methods, however they require training. Training does not just refer to the CPU time required to train the model, but also refers to obtaining sample data for that particular sensor, and then labeling it by hand. For example, for the DRIVE database the authors say, "it took an observer 2 hours on average to label a single image" [21] That means to obtain a similar amount of training data would require 40 hours of manual labeling effort by trained medical staff, and depending on the machine learning technique this may have to be repeated for new equipment (new sensors, new Field of View of cameras, etc) depending on how well the classifier generalizes. Additionally because our method has only one critical parameter (the size of the neighborhood for tensor voting) it would be easy to create a semi-automatic application where a clinician could interactively select the best setting to produce the best accuracy (on a per-image basis rather than across the entire data set).

Two published papers ([9], [18]) have conducted the experiment alluded to above, cross-training their supervised systems on the DRIVE database and testing on STARE and vice versa. As we expected they show noticeable drops in performance, in fact

[18] falls from AUC of 0.949 to 0.927, slightly below the performance we achieve in this work. [9] seems to have a more robust classifier, but they also report a drop in AUC from 0.959 to 0.945 (when trained on STARE and tested on DRIVE). These examples clearly show the benefits of a model free, unsupervised technique.

Future Work

For future work it may be possible to add a classifier on top of the Tensor Voting output to further improve delineation of vessel vs non-vessel pixels. This should be straight forward to experiment with due to shared pipeline architecture with scikit-learn, which has many classifiers already such as Random Forests, Support Vector Machines, Naive Bayes, etc.

Additionally because Tensor Voting naturally extends to higher dimensions it will be interesting to perform further experiments on 3D (such as 3D CT angiograms) and 4D data sets such as 4D MRI. Our choice for pre-processing can theoretically scale up as well (i.e., Sobel can be extended to find gradients in 3D) so this could be an interesting avenue of research.

However the current implementation, despite being well implemented C++ code is prohibitively slow to compute with dense data sets.In the future we should consider some combination of developing a multi-core or GPU implementation of tensor voting and possibly working with more sparse data (although this is quite difficult to do with medical imaging because the accuracy requirements are so high). We investigated the feasibility of updating the code to run on the Graphics Processing Unit (GPU) for significantly greater parallelism, but ultimately abandoned this attempt. The reason for this is that GPUs require more predictable memory accesses but the current

implementation uses Approximate Nearest Neighbor (ANN, [1]) search to determine the neighborhood based on σ. Additionally the fundamental data structure, the "tenslist" is implemented as a linked-list. This is a perfectly good representation on the CPU, but pointer chasing on the GPU is an anti-pattern. In the future it would be more straightforward to write a clean room implementation starting with arrays. There has been previous work on GPU implementation [12], but it implemented on top of the frame buffer objects, this could almost certainly be improved today using CUDA or OpenCL.

Conclusion

We have detailed a unique initialization strategy for segmenting thick tubular structures (i.e., blood vessels). Using our software framework we performed extensive searches of the parameter space to arrive at an iterative 2D tensor voting process and subsequent tensor analysis for the task of retinal vessel segmentation. We achieve results that are competitive with the state-of-the-art supervised techniques and particularly among the best for unsupervised approaches. Tensor Voting naturally incorporates the types of geometric verification that have been implemented in an ad-hoc way by other unsupervised approaches.

Bibliography

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 891–923, 1998.

[2] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, "Cython: The best of both worlds," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 31–39, 2011.

[3] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 6, pp. 679–698, 1986.

[4] S. Chaudhuri, S. Chatterjee, N. Katz, M. Nelson, and M. Goldbaum, "Detection of blood vessels in retinal images using two-dimensional matched filters," *IEEE Transactions on medical imaging*, vol. 8, no. 3, pp. 263–269, 1989.

[5] G. González, F. Fleurety, and P. Fua, "Learning rotational features for filament detection," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 2009, pp. 1582–1589.

[6] A. Hoover, V. Kouznetsova, and M. Goldbaum, "Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response," *Medical Imaging, IEEE Transactions on*, vol. 19, no. 3, pp. 203–210, 2000.

[7] X. Jiang and D. Mojon, "Adaptive local thresholding by verification-based multithreshold probing with application to vessel detection in retinal images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 1, pp. 131–137, 2003.

[8] L. Loss, G. Bebis, M. Nicolescu, and A. Skurikhin, "An iterative multi-scale tensor voting scheme for perceptual grouping of natural shapes in cluttered backgrounds," *Computer Vision and Image Understanding*, vol. 113, no. 1, pp. 126–149, 2009.

[9] D. Marín, A. Aquino, M. E. Gegúndez-Arias, and J. M. Bravo, "A new supervised method for blood vessel segmentation in retinal images by using gray-level and moment invariants-based features," *Medical Imaging, IEEE Transactions on*, vol. 30, no. 1, pp. 146–158, 2011.

[10] G. Medioni, M.-S. Lee, and C.-K. Tang, *A computational framework for segmentation and grouping*. Elsevier, 2000.

[11] A. M. Mendonca and A. Campilho, "Segmentation of retinal blood vessels by combining the detection of centerlines and morphological reconstruction," *Medical Imaging, IEEE Transactions on*, vol. 25, no. 9, pp. 1200–1213, 2006.

[12] C. Min and G. Medioni, "Tensor voting accelerated by graphics processing units (gpu)," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3. IEEE, 2006, pp. 1103–1106.

[13] P. Mordohai and G. Medioni, "Stereo using monocular cues within the tensor voting framework," *European Conference on Computer Vision, Lecture Notes in Computer Science*, vol. 3024, pp. 588–601, 2004.

[14] ——, "Dimensionality estimation, manifold learning and function approximation using tensor voting," *The Journal of Machine Learning Research*, vol. 11, pp. 411–450, 2010.

[15] M. Nicolescu and G. Medioni, "A voting-based computational framework for visual motion analysis and interpretation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 739–752, 2005.

[16] M. Niemeijer, J. Staal, B. van Ginneken, M. Loog, and M. D. Abramoff, "Comparative study of retinal vessel segmentation methods on a new publicly available database," in *Medical Imaging 2004*. International Society for Optics and Photonics, 2004, pp. 648–656.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] E. Ricci and R. Perfetti, "Retinal blood vessel segmentation using line operators and support vector classification," *Medical Imaging, IEEE Transactions on*, vol. 26, no. 10, pp. 1357–1365, 2007.

[19] J. Soares, J. Leandro, and R. M. Cesar, "Retinal vessel segmentation using the 2-D Gabor wavelet and supervised classification," *Medical Imaging*, 2006.

[20] M. Sofka and C. V. Stewart, "Retinal vessel centerline extraction using multi-scale matched filters, confidence and edge measures," *Medical Imaging, IEEE Transactions on*, vol. 25, no. 12, pp. 1531–1546, 2006.

[21] J. Staal, M. D. Abràmoff, M. Niemeijer, M. A. Viergever, and B. van Ginneken, "Ridge-based vessel segmentation in color images of the retina," *Medical Imaging, IEEE Transactions on*, vol. 23, no. 4, pp. 501–509, 2004.

[22] C.-K. Tang, G. Medioni, and M.-S. Lee, "N-dimensional tensor voting and application to epipolar geometry estimation," *IEEE Trans. PAMI*, vol. 23, no. 8, pp. 829–844, 2001.

[23] S. Ullman and A. Sha'ashua, "Structural saliency: The detection of globally salient structures using a locally connected network," Tech. Rep., 1988.

[24] M. Wertheimer, "Untersuchungen zur lehre von der gestalt (english translation)," *II. Psychologische Forrschung*, vol. 4, pp. 301–350, 1929.

[25] F. Zana and J.-C. Klein, "Segmentation of vessel-like patterns using mathematical morphology and curvature evaluation," *Image Processing, IEEE Transactions on*, vol. 10, no. 7, pp. 1010–1019, 2001.