Image Segmentation of Histology Slides

Austin Adams
CSU Long Beach
1250 Bellflower Blvd.
Long Beach, CA 90840
austin.adams314@gmail.com

KC Skubic
CSU Long Beach
1250 Bellflower Blvd.
Long Beach, CA 90840
kskubic@yahoo.com

LeighAnn Van Deventer
CSU Long Beach
1250 Bellflower Blvd.
Long Beach, CA 90840
lkvandeventer@gmail.com

G.D. Young
CSU Long Beach
1250 Bellflower Blvd.
Long Beach, CA 90840
gdyoung1@gmail.com

May 20, 2011

Abstract

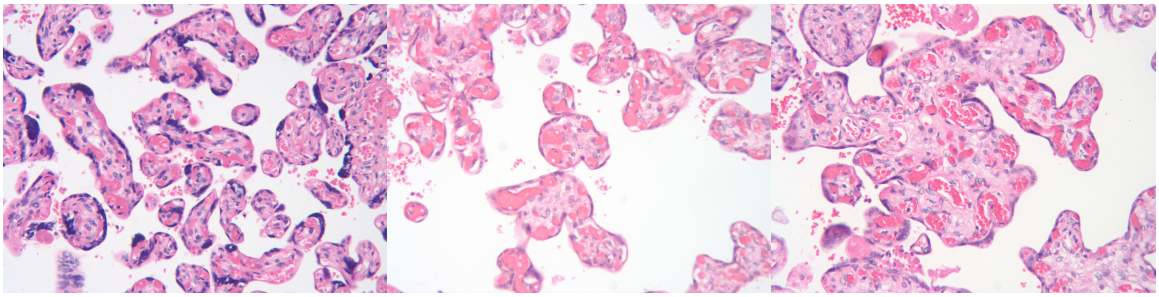
Image segmentation is the process of dividing an image into meaningful sections. In examining histology slides from a placenta various segmentation methods were used to segment the slide into blob regions and vessel regions. These methods included K-means clustering, Chan-Vese, histogram, edge detection, and watershed. Results showed that blob regions are easier to segment than vessel regions. In conclusion, while image segmentation is certainly possible for histology slides, it will require future work to do so with any acceptable degree of accuracy.

1 Introduction

The world of image processing is an enormous world with endless possibilities. One small section of this field includes image segmentation; partitioning of an image. The main reason for doing this, of course, is to simplify the representation of that image into something easier to analyze. Our group decided to explore this field and its many capabilities.

Given the concept of placenta analysis, and a set of placenta histology slides, we designed our project. Our goal was to find both an automated and also accurate method for segmenting these microscopic images and their corresponding blobs. In other words, we wanted to take the microscopic images and extract not only the villi, but also the corresponding vessels within. In order to complete our goal we examined various segmentation techniques and applied them

to our data set, hand-sketched the villi and vessel outlines of our data set, and compared the methods to determine accuracy. We had high hopes that, with a simplified and accurate representation of the images, we could provide useful data for medical analysis.



(a) Histology Slide1 (b) Histology Slide2 (c) Histology Slide3

Figure 1: Our Histology Data Set

2 Research Methods

As the majority of us were neophytes to image processing, we began with an implementation of the algorithm described in *Quantifying Clinically Significant features of Placental Histology Images: A Method*, by Morten Andersen, David Belanger, Radina Droumeva, Jenny Li, Gilbert Moss, Gabriela Pala (hereafter, *Quantifying Clinically Significant Features*) [1]. This proved useful on multiple fronts - from building a vocabulary of of Matlab image processing functions, gaining a comfort level with deploying the appropriate method for a required task, all the way to developing a more thorough understanding of the data set and the larger placental analytics project. Specifically, beginning to differentiate the term 'blobs' from 'vessels' and becoming aware of a hypothesized connection between health of the fetus (and mother) and vasculature development in the placenta (Placental Ratio, et al). Debugging the code was occasionally problematic but ultimately successful and we learned an important lesson confirmed by Radina Droumeva in correspondence - "You are absolutely right - lesson number one in image processing - you need to carefully choose and adapt methods for application-specific requirements".

In this case, the reference algorithm was built around performing segmentation, conducting morphological operations, and computing statistics on relatively low resolution images (compared to our data set). We concluded that both the specific segmentation technique as well as the morphological operations was inappropriate for our data set. As you can see in Figure 2, much of the biologically useful information in the placental cross section photos has been decimated by a faithful reproduction of the algorithm.

We will talk later about our concerns for the utilization of a Mahalanobis Distance measure in the Kmeans segmentation in our Kmeans Mahalanobis Segmentation section. For now, a visual inspection of the post processed images shows that the structuring element used in the morphological operations was too large to retain significant information of our high resolution

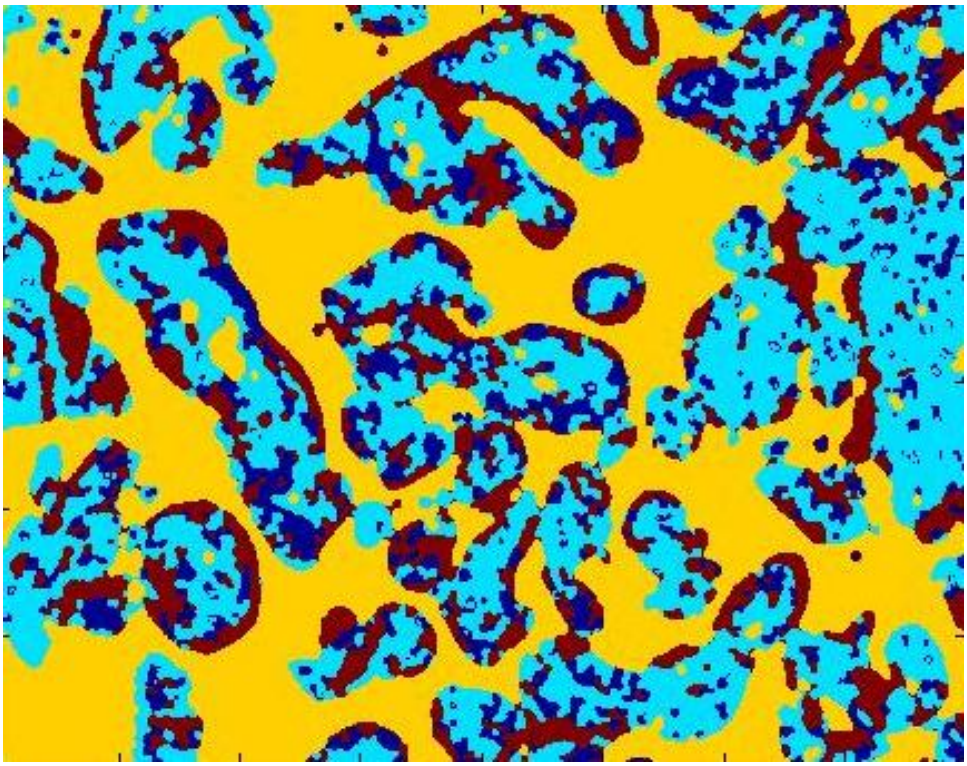


Figure 2: Resulting Histology Image

data, which intuitively makes sense, given the relatively large amount of detail (contrast of features) on our data set versus the data set for which the reference algorithm was built.

We used several different methods of segmentation, most of which we got the inspiration for from Wikipedia. As for the specifics of how each method works we found in Each will be explained in turn. Whenever morphological operations are discussed, we are using them as defined in González and Woods [2].

2.1 K-Means Clustering

2.1.1 Eucliean Distance

In our (non-exhaustive) survey of segmentation, we began with Matlab's built-in K-means segmentation using a Euclidean distance measure, specifying only the number of cluster centers. We originally experimented with 4 cluster centers, deferring to the assertion of *Quantifying Clinically Significant Features*, but decided after a closer visual inspection of the data set that the 'Blue' cluster in a 4 cluster segmentation should actually be split into a 'Blue' cluster AND a 'Light Purple' cluster, for a total of 5 cluster centers (Blue, Light Purple, Pink, Red and White) [1]. Increasing the number of cluster centers beyond 5 did not improve the segmentation results. We only lightly applied the Morphological Operations of Opening (erosion then dilation) and Closing (dilation then erosion) on the segmented images (in K-means Segmentations and Chan Vese, a structuring element disk of radius 1 pixel, on an image of 1200X1600 pixels!).

We ran Matlab's Kmeans algorithm on images that we preprocessed via a transformation to Lab color space. What we ultimately came to understand was that the Lab Corridoring technique (Figure 3) we developed was simply a form of intensity thresholding common to all image processing techniques, and indeed we found that Mathwork's File Exchange has several interesting thresholding algorithms with convenient GUI's that are used in the medical research community [4].

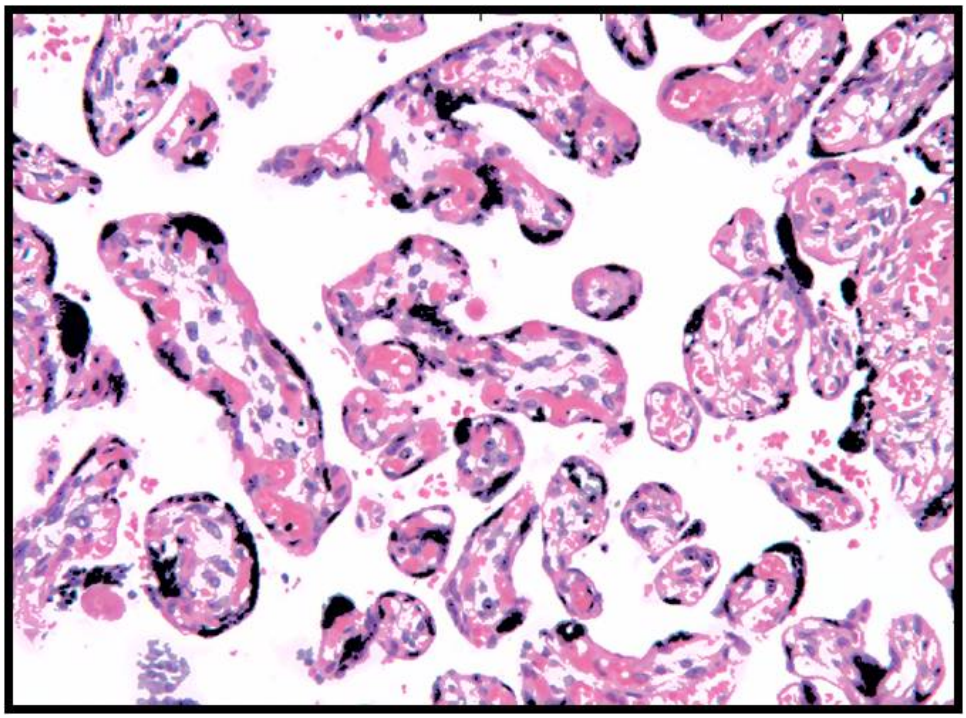
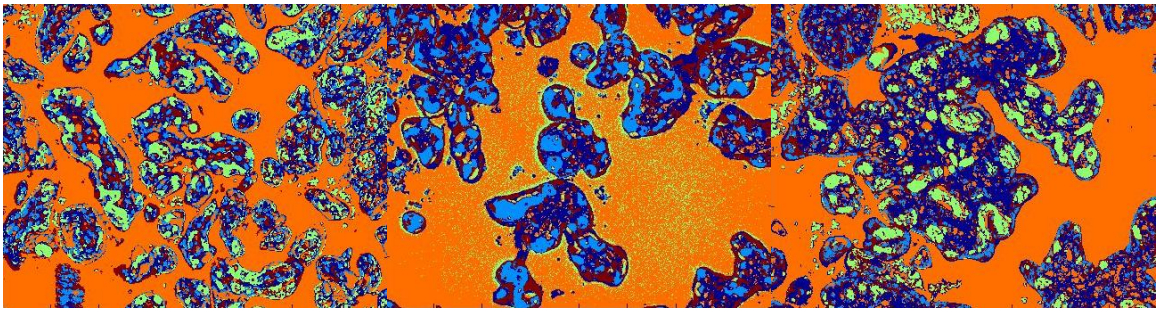


Figure 3: Lab Corridoring Image

Ultimately, the K-means Euclidean Distance Segmentation using 5 cluster centers was not sensitive to the inclusion of the Lab preprocessing, although the K-means Segmentation with 4 cluster centers was sensitive to its inclusion. The K-means Euclidean Distance performed well at segmenting the vessels, as well as finding the blobs. Figure 4

The 5 clusters total segmentation results appear satisfactory in visual inspection (Figure 4). A visualization of the final step of the algorithm for Hist 1 showing retained vessels after morphological operations in retained blobs after morphological operations reveals the effectiveness of the algorithm. (Figure 5)

One idea for better results from the K-means Euclidean method is to segment the image, then perform a 'heavy' morphological erosion (with a large structuring element). We hypothesize that this is very likely to separate the only minimally connected blobs. The loss of area of the blobs, as well as an accurate vessel count per blob could potentially be recovered by 1) identifying each connected component via Matlab's 'bwlabel' function 2) compute the area of each blob component plus a 'buffer' around the perimeter of each blob equal to the size of the original structuring element. 3) find and quantify the vessels in the larger area (blob plus



(a) Histology Slide1 (b) Histology Slide2 (c) Histology Slide3

Figure 4: Euclidean Method Results

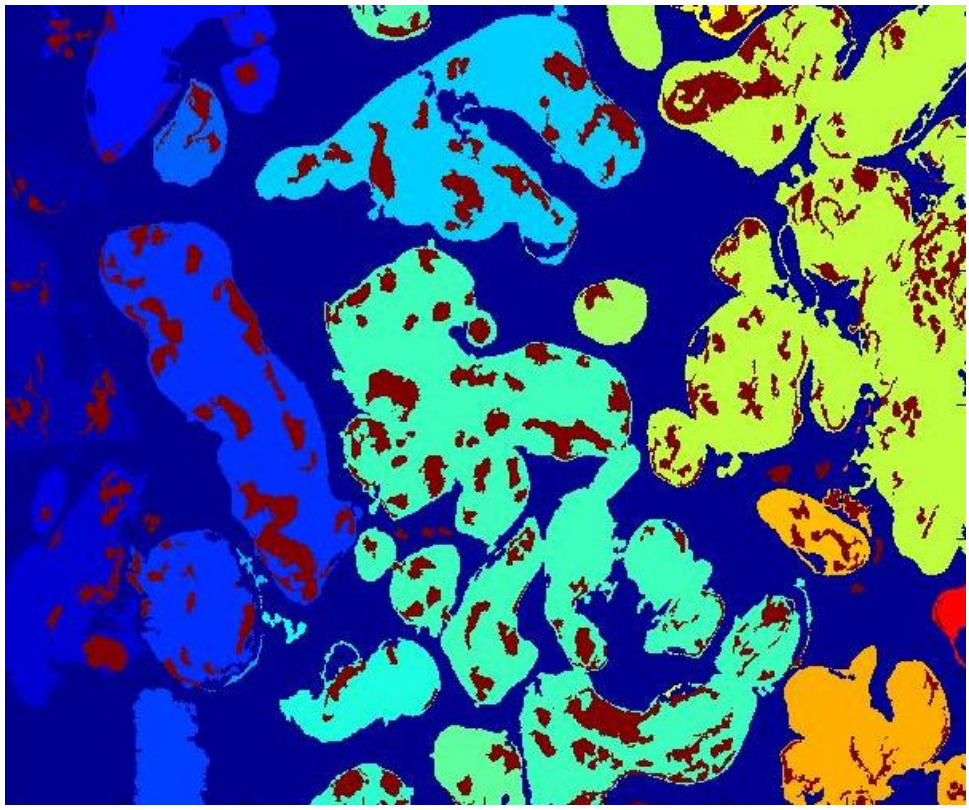


Figure 5: Euclidean Final Result

buffer). Likely, there would also need to be a 4th step - subtracting the areas of overlap (double counting) induced by the 'buffer'.

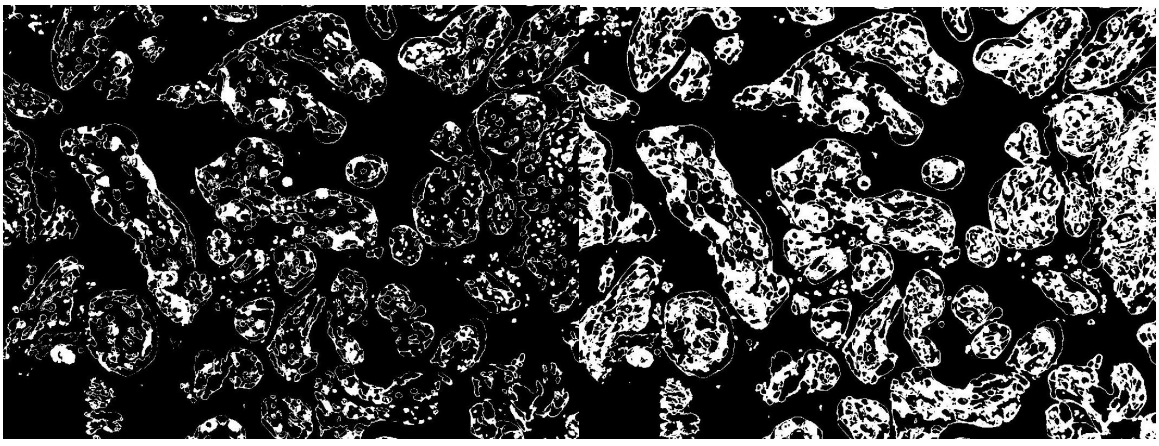
2.1.2 Mahalanobis Distance

After exhausting the K-means Euclidean Segmentation, we returned to the core philosophical idea in *Quantifying Clinically Significant Features* - that better K-means segmentation results

would occur from the utilization of a Mahalanobis distance measure in place of a Euclidean distance measure [1]. This method effectively weights distance in color by the variance in color per color channel in the given image. For example, in RGB color space, if there is little contrast in the Green channel, the distance between two pixels is effectively made larger. The reason the idea was appealing was that it could potentially be more effective in separating pink from red in the data set.

We implemented K-means with Mahalanobis distance and with 4 absolute color standards as cluster centers - a faithful reproduction of the segmentation technique in *Quantifying Clinically Significant Features*, stripped of the morphological operations that prevented drawing any conclusions about the segmentation technique itself[1].

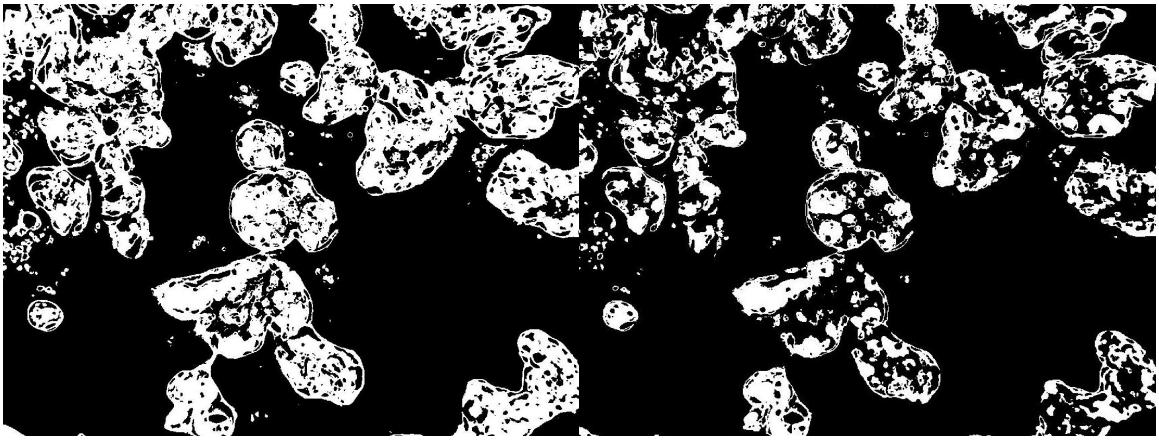
We discovered there are problems with utilizing the Mahalanobis measure, and it did not perform well at segmenting the vessels. The flaw in the algorithm is inherent in utilizing the variance weighted distance measure itself - the variance used was unique to each image. In our algorithm, the color variance of each image was computed, with the distance to an absolute color standard weighted by the variance in each color channel. Since each image has its own variance structure, the SAME COLOR on two different images can (and does) get sent to different cluster centers. A good way to see this dynamic is to run the algorithm with 2 different sets of absolute color markers, chosen by picking representative pixel values for each color marker on a given image (so a run of the algorithm off of Hist1 representative color values for Red, Pink, Blue and White and a separate run using representative pixel values for Hist2). The results shown in Figure 6 show the sensitivity to small changes in 'initial conditions' - not a feature of a robust algorithm. As you can see in Figure 7, the same pixel in a given image (with the same variance structure, since the image itself is not changing between the 2 calibrations) is sent to different cluster centers, frustrating an attempt to automate vessel detection. We experimented with both RGB and Lab color spaces, and the results were invariant to the transform.



(a) Calibration w/ Hist1 (b) Calibration w/ Hist2

Figure 6: Histology Slide 1 with Different Calibrations

We continue to believe there is merit in the Mahalanobis distance measure for segmentation. However, in our algorithm, we believe the wrong population of pixels was used to calculate



(a) Calibration w/ Hist1 (b) Calibration w/ Hist2

Figure 7: Histology Slide 2 with Different Calibrations

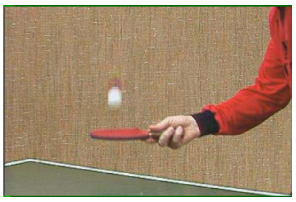
variance. We hypothesize that the best variance to calculate is for the entire population of pixels of each color channel for the entire set of images. This could be accomplished in a preprocessing step before performing the segmentation, although the speed gains from the Mahalanobis-based segmentation could be compromised (diagonalized covariance matrix). Another possible method to improve results would be to use either a 'median' covariance matrix, applied to all the images, or an extreme covariance matrix applied to all the images. At the very least, the variance of variance could be reported as a likely quite useful statistic of the reliability of the segmentation results. While we believe this further work could improve the segmentation results, this method will still suffer from the potentially fatal flaw described above for all discrete methods - it does not take advantage of the abundance of information by evaluating neighborhood properties of each pixel.

2.2 Chan-Vese

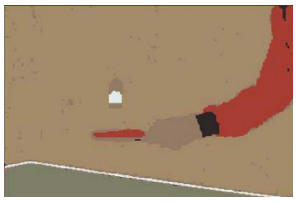
In order to explore the power of continuous methods, late in the semester we explored an implementation of a Chan Vese Active Contours algorithm. Based on the Mumford Shah Energy Functional, the Chan Vese algorithm attempts to minimize the difference between the algorithm's 'mask' of the image (think of drawing a faithful reproduction cartoon on top of the original image, with boundaries between features clearly marked). It achieves this by minimizing the value of the following integral [3]: $E(f,\Gamma) = \mu^2 \iint_R (f-g)^2 dxdy + \int_R \int_\Gamma \|\nabla f\|^2 dxdy + v|\Gamma|$

f is our representation of the image, g is the image itself. Γ is the set of boundaries between the R Regions of the image that the algorithm identifies by iterating over small steps in time. The first integral term gives the best 'closeness of fit', as the better the 'cartoon' representation is a faithful reproduction of the original image, the smaller the value of that term. The second integral term forces the 'best' segmentation by loading all of the gradient of the image onto region boundaries, thereby segmenting regions based on the neighborhood property of the gradient at each given pixel. Please see Zoltan Kato for a more detailed discussion [3].

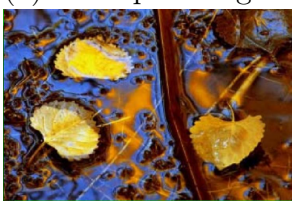
Cartoon Image Example [3]



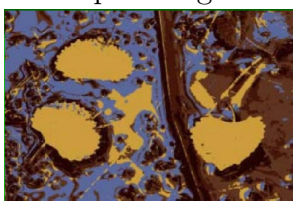
(a) Example Image 1



(b) Example Image 1 After

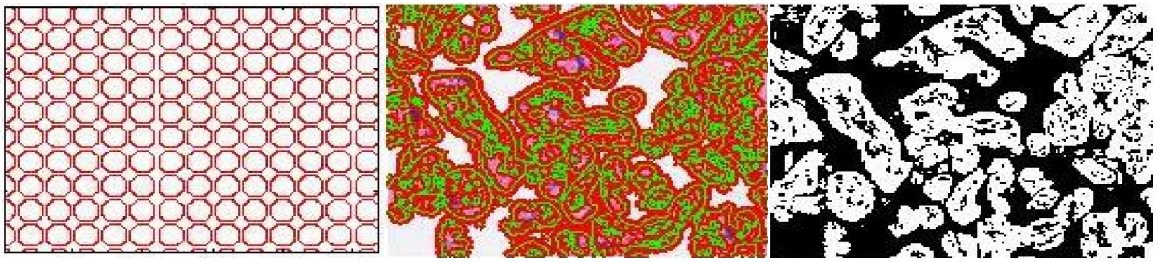


(c) Example Image 2



(d) Example Image 2 After

Unfortunately, we ran out of time before we could fully implement a multi-counters algorithm necessary to extract all of the information we needed from the color images. But we were able to implement the method on a grayscaledversion of our images in order to find the blobs. The Chan Vese Segmentation did a better job of separating minimally connected blobs, Figure 8.



(a) Initial Contour (b) 138 Iterations (c) Chen Vese Blobs

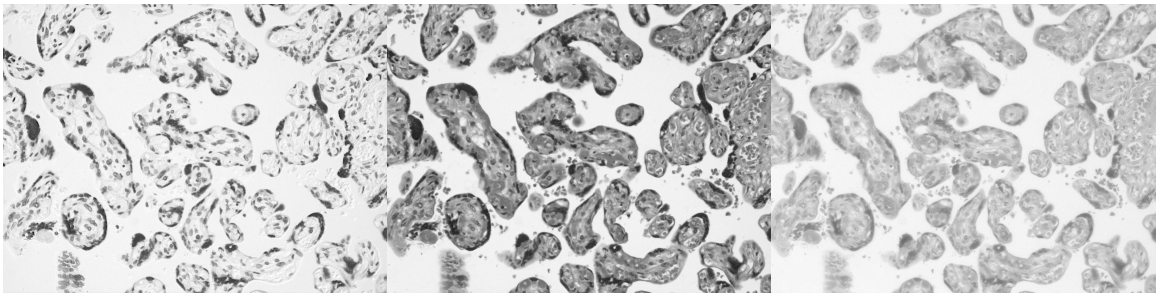
Figure 8: Chen Vese Process

As is evident on Hist 1, the method does a better job at correctly separating only slightly touching or closely neighboring blobs. The full power of this method can surely be brought to bear on this problem, as a quick google search can attest. For the Chan Vese method Results, Vessels from the KMeans Euclidean Segmentation were used.

2.3 Edge Detection

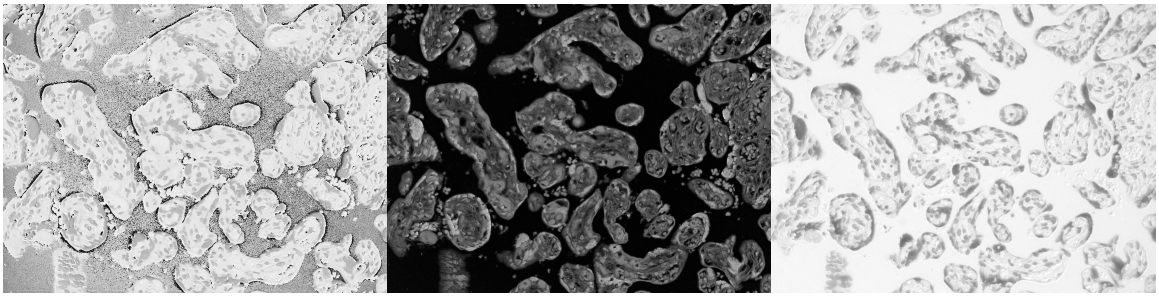
This method uses Canny edge detection to find the edges of the blobs, and then fills in the edges to form regions.

The original image is in RGB. Each of these colors can be seen in Figure 9. None has high contrast though, so we converted to HSV which can be seen in Figure 10. We noticed that the Saturation Channel (Figure 10b) looked to clearly separate the background from the blobs, so we ran the edge detection on just that channel.



(a) Red Channel (b) Green Channel (c) Blue Channel

Figure 9: The three channels of a Histology Slide



(a) Hue Channel (b) Saturation Channel (c) Value Channel

Figure 10: The adjusted channels of the Histology Slide

The Canny edge detection works by finding places in the image where there is a sudden change in pixel intensity value. At each one of these locations a direction is assigned to it so similar lines can be joined together. The result of just the edge detection can be seen in Figure 11.

Notice that these are very sparse regions, since the edge detection only picks up edges and not full regions. Thus we need to fill in the regions using some morphological operations. First, to close all the blobs so that when we fill the image we don't fill everything we use an operation called morphological closing. This works by first expanding all the edges a set amount, hopefully connecting them, and then thins them back to the same amount they were before. This can be seen in Figure 12 We then fill in the image. This can be thought of as flooding each blob with white unless it can bleed out into the background. This leaves all the blobs mostly filled in as seen in Figure 13

To help split the regions into distinct blobs, we run one more morpholgical operation called opening. This is the opposite of closing, as it first thins all the edges, hopefully separating regions, and then fills the edges back a little bit more smoothly. This can be seen in Figure 14.

The above is what the algorithm segEdge.m performs on the image. The full algorithm can be seen in Appendix A. After we have run this code, it is time to label the different blobs and then overlay the vessels on top of them using other methods. See blobLabel.m in Appendix C

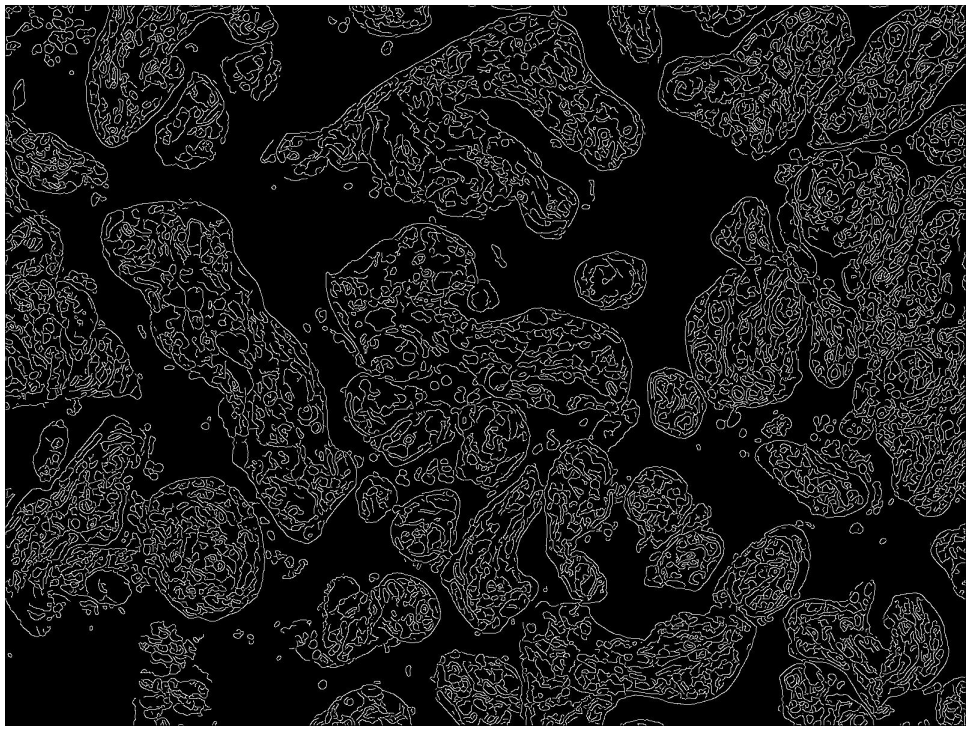


Figure 11: Canny Edge Detection on Saturation Channel

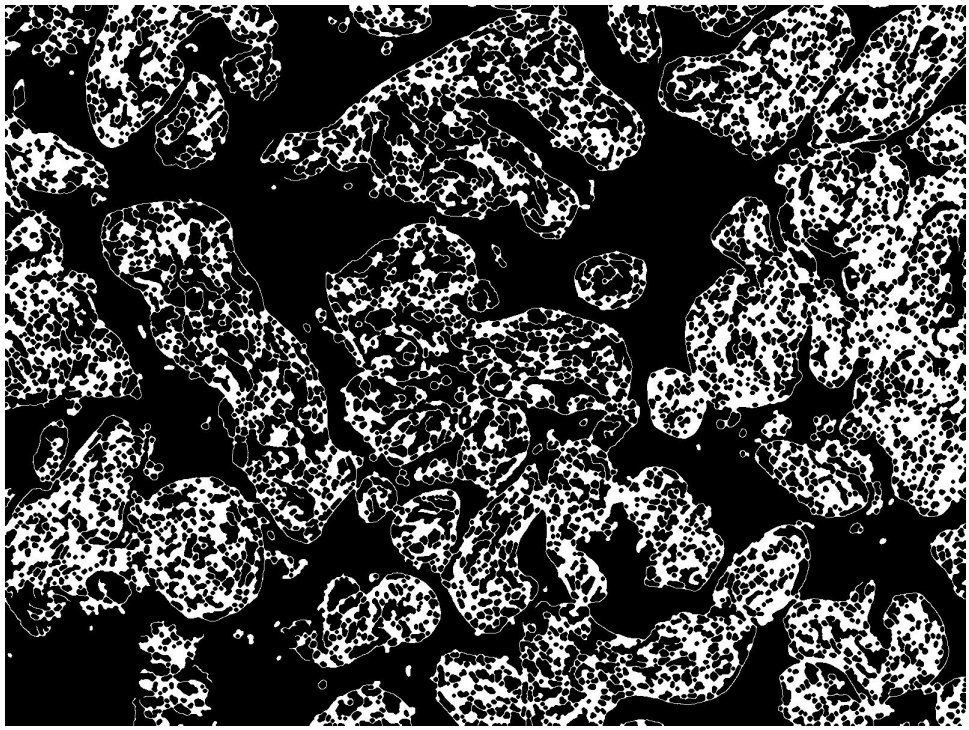


Figure 12: Edges after a closing operation

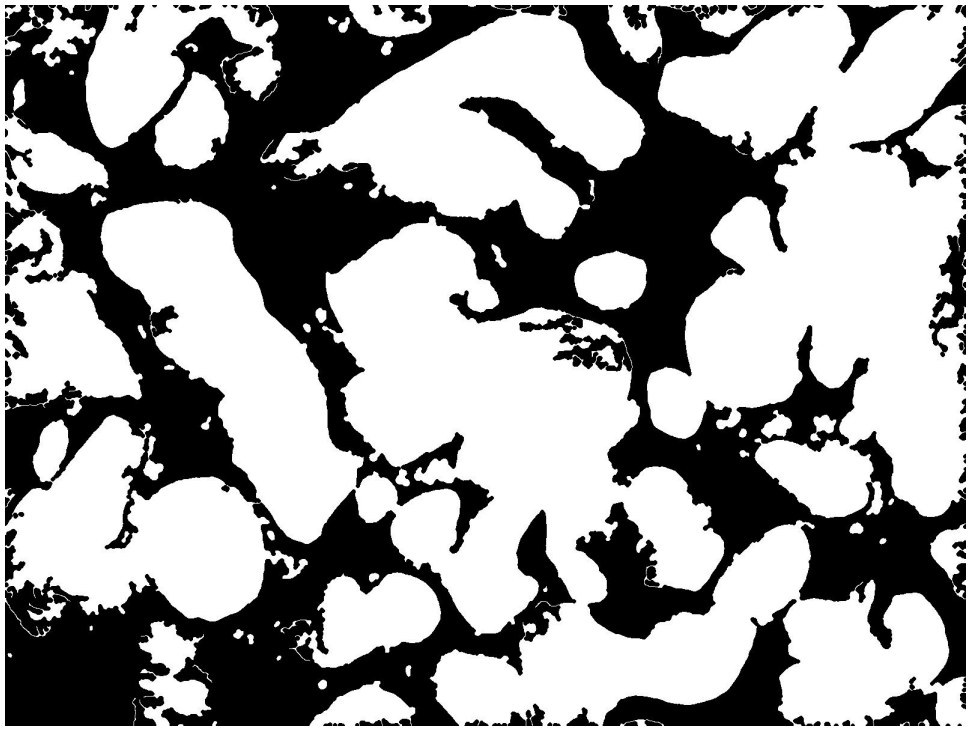


Figure 13: Filled Blobs

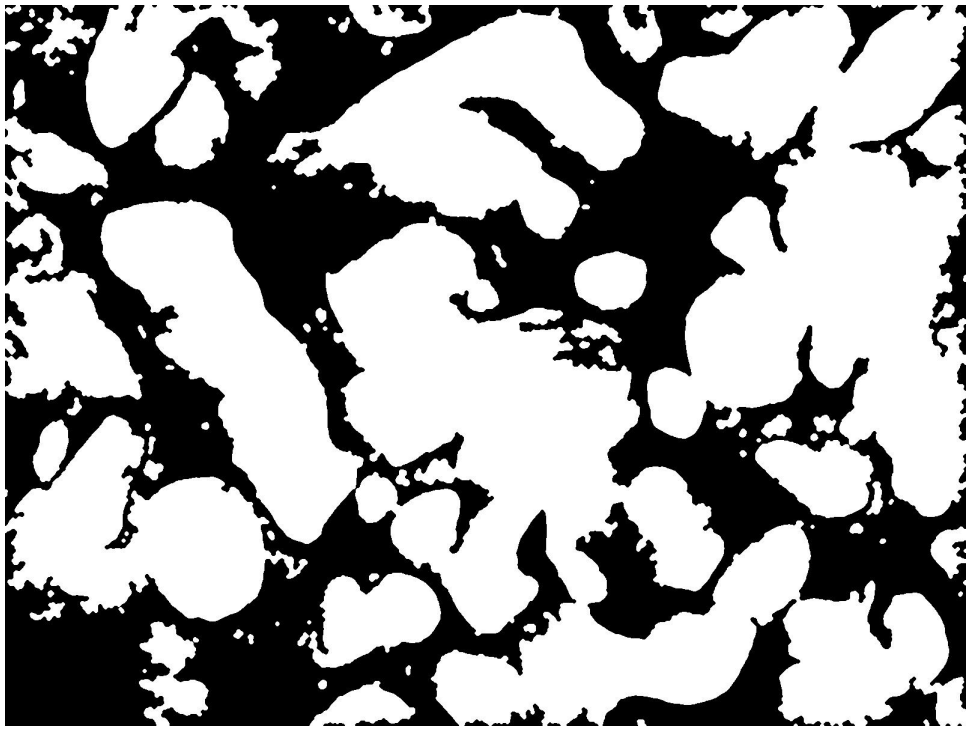


Figure 14: Opened Regions

for how the labeling was performed, and read the Histogram section for how the vessels were found.

2.4 Histogram

2.4.1 Blob Detection

The Histogram based method is one of the simplest in idea. The entire method calls everything a blob that is in a certain range of pixel intensity values.

To begin, we once again start by using only the Saturation channel from Figure 10b. Since all of the background is really dark, it is easy to apply a histogram to the image and keep only pixel values with higher intensity than the background. The Saturation channel had pixel values from 0 to 1, 0 being black, so we chose to keep all pixel values greater than 0.2. All of these kept values were set to 1, and so we created the binary image seen in Figure 15.

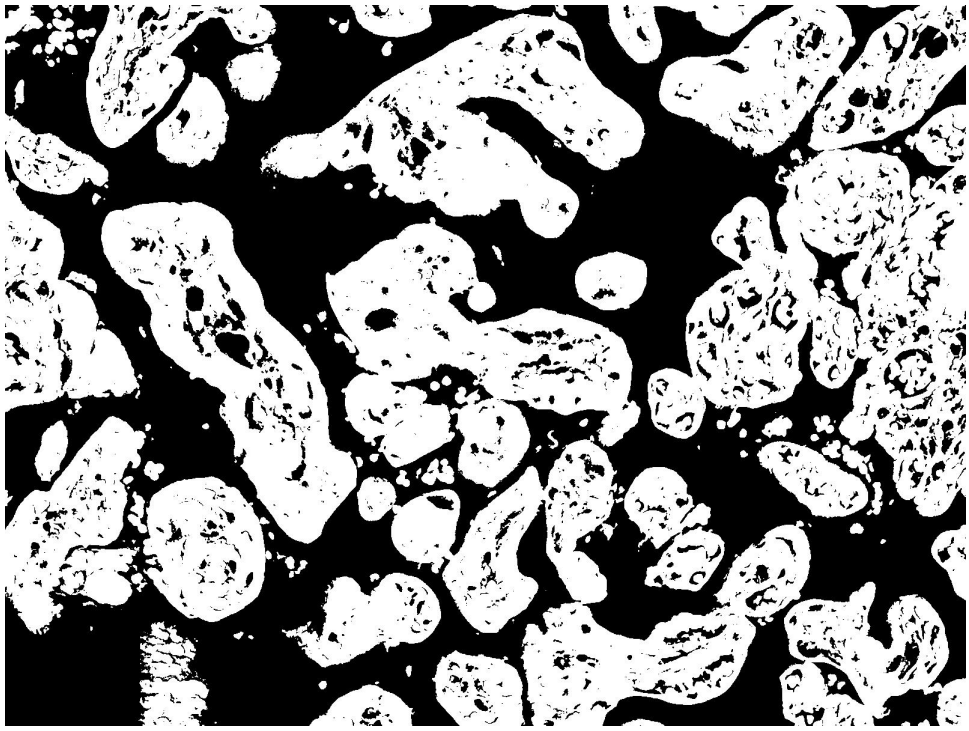


Figure 15: Histogram Method Applied to Saturation Channel

From here, all we had to do was clean up the image a little bit to try and fill the blobs in a little better. To start, I applied a morphological closing to make sure all the blobs were closed regions. The closing can be seen in Figure 16.

Following up the closing, we filled in all the holes in the blobs which can be seen in Figure 17.

Finally for the Histogram method on the blobs, we used morphological opening to try and separate the blobs from each other. This can be seen in Figure 18.

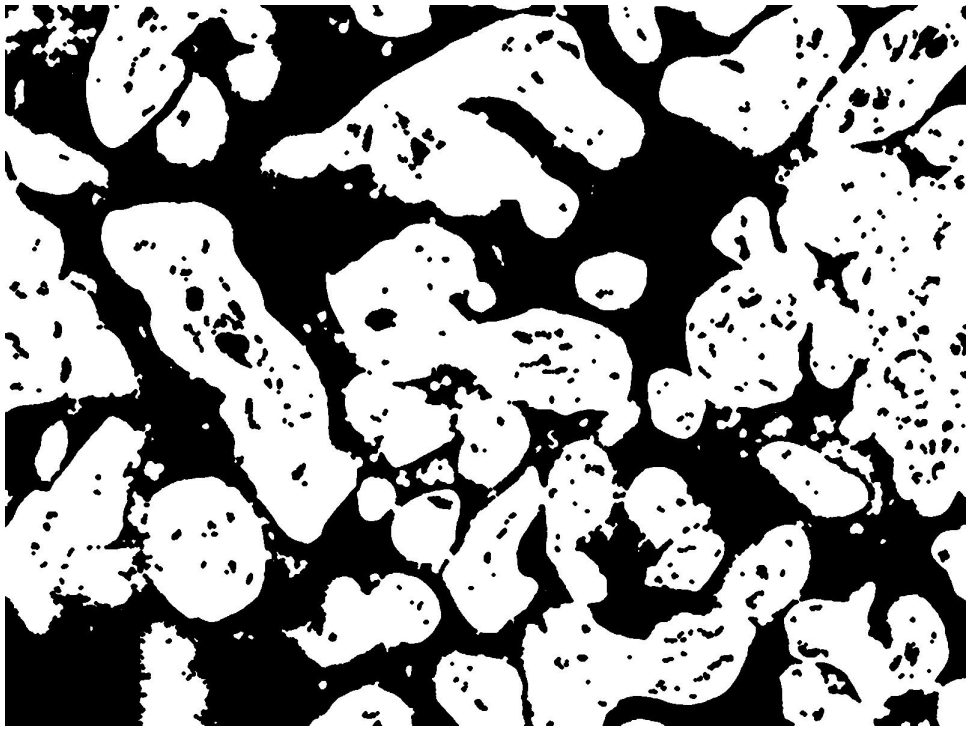


Figure 16: Histogram Method After Closing

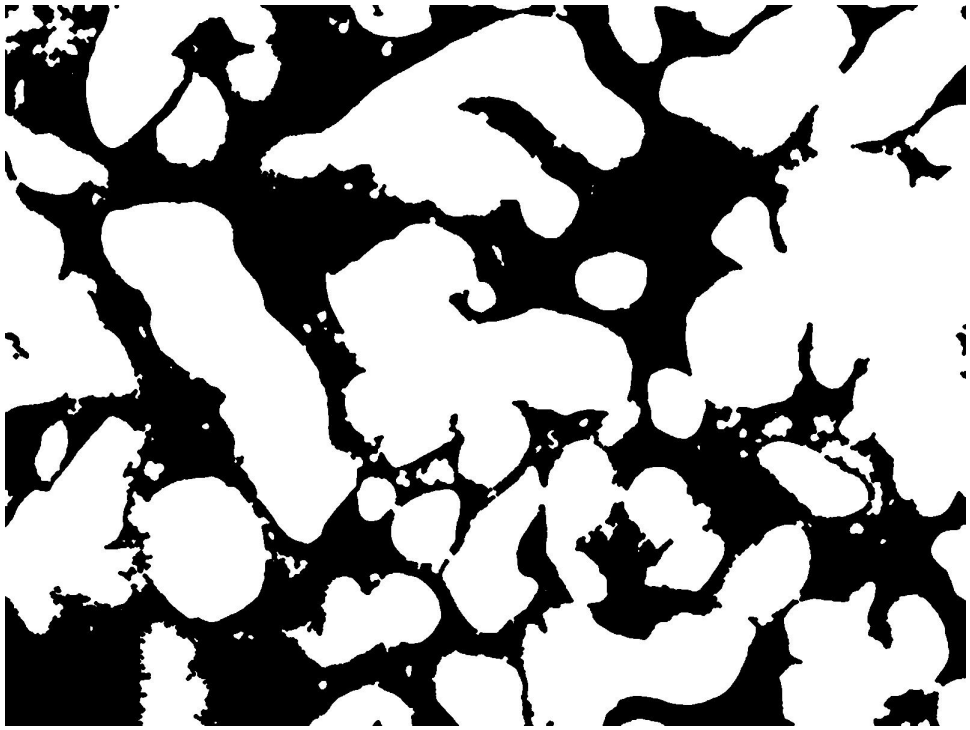


Figure 17: Histogram Method Filled Blobs

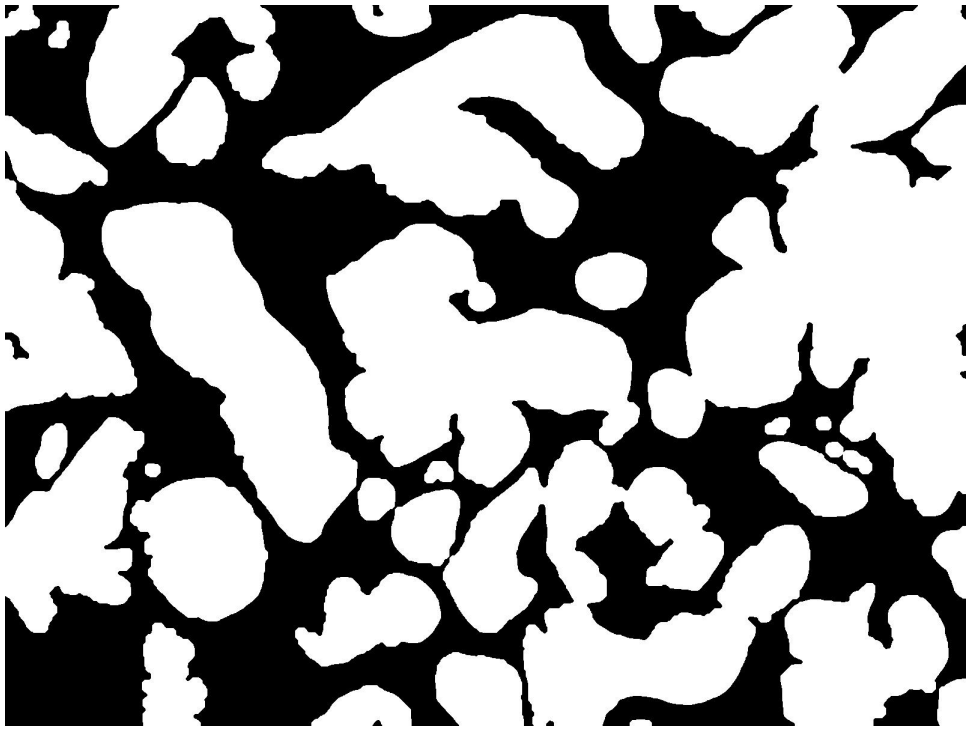


Figure 18: Histogram Method Opened Regions

This is how the algorithm segHist.m runs. The full code can be seen in Appendix B. After getting the binary image of the blobs, we labeled each and then used the Histogram method again to find the vessels in each blob. The labeling algorithm can be seen in Appendix C, and the vessel detection follows.

2.4.2 Vessel Detection

Because of how simple the Histogram Method is to use based on color, we decided to use it to find the vessels in each blob as well. After labeling the blobs, we looked at the vessels in the original image and found them to have a specific color given by a range of values in the red and green channels seen in Figures 9a and 9b. The red and green channels have pixel values between 0 and 255, and the specific red of the vessels was given by values between 210 and 235 in the red channel, and less than 150 in the green channel. All of the pixels meeting these criteria were labeled as vessels, and then we overlaid them on top of the labeled blobs. This overlay looks like what we see in Figure 19. We used this to label vessels for the Edge Detection, Histogram, and Watershed based methods. The code can be seen in Appendix D.

2.5 Watershed

The Watershed Method we used is a further refinement of the Histogram Method. That means we start the method directly on Figure 18. To begin we find the distance of each pixel in the blobs from the nearest background pixel. By distance we mean measure of length of the

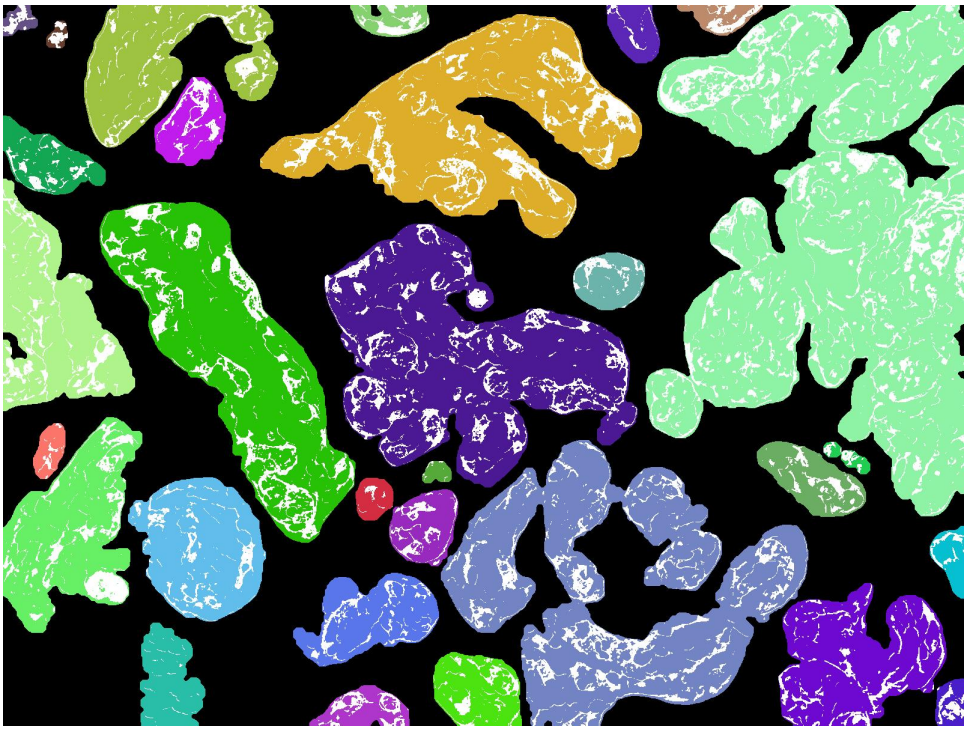


Figure 19: Histogram Method to Find Vessels

smallest line segment connecting each blob pixel to a background pixel. That results in an image like Figure 20.

The Watershed method works by finding which white pixel every blob pixel in Figure 20 would flow toward. This can be thought of as the white pixels being the lowest part of the image, and when you drop water on any pixel in a blob you measure where that drop of water ends up. Every new region that pixels flow to is labeled as different from each other. After finding these regions, we filled them in and called them blobs. This results in the Watershed blob image seen in Figure 21. We cleaned up the very thin line segments left over using morphological opening resulting in the final Watershed image in Figure 22.

The Watershed method code can be seen in Appendix E. Like the other methods, to finish the Watershed method we labeled each blob and overlaid the vessels on top of the blobs using blobLabel.m and redVessels.m.

3 Results and Conclusions

We used hand-traced images of the original histology slides to verify the accuracy of our results.

The segmented blobs with vessels can be seen in Figures 23 - 25.

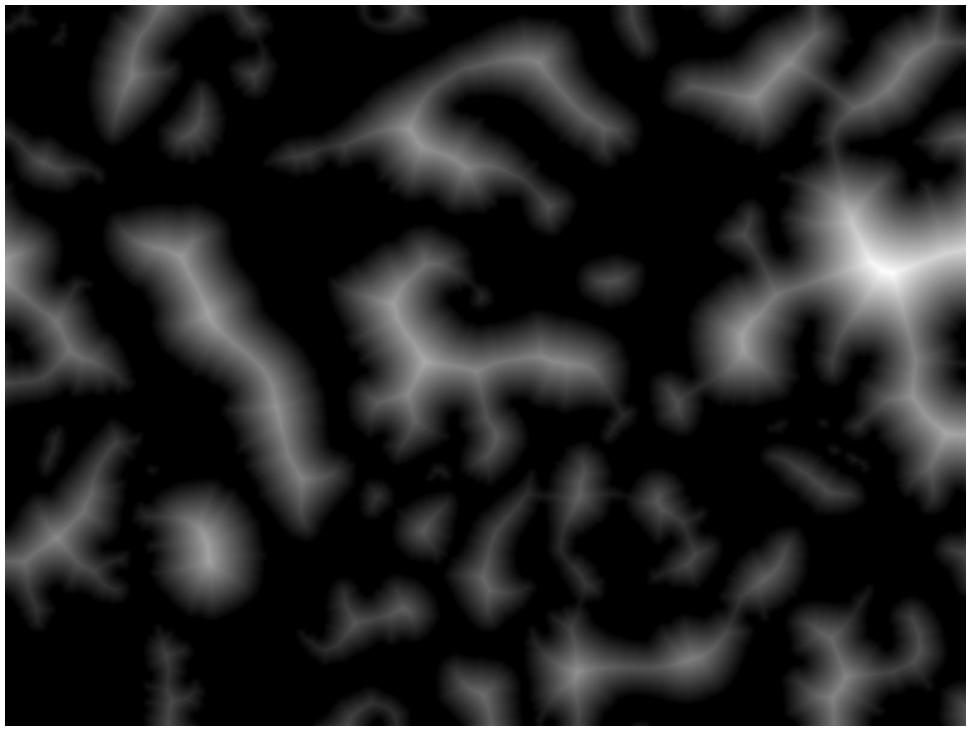


Figure 20: Distance of Blob Pixels to Background

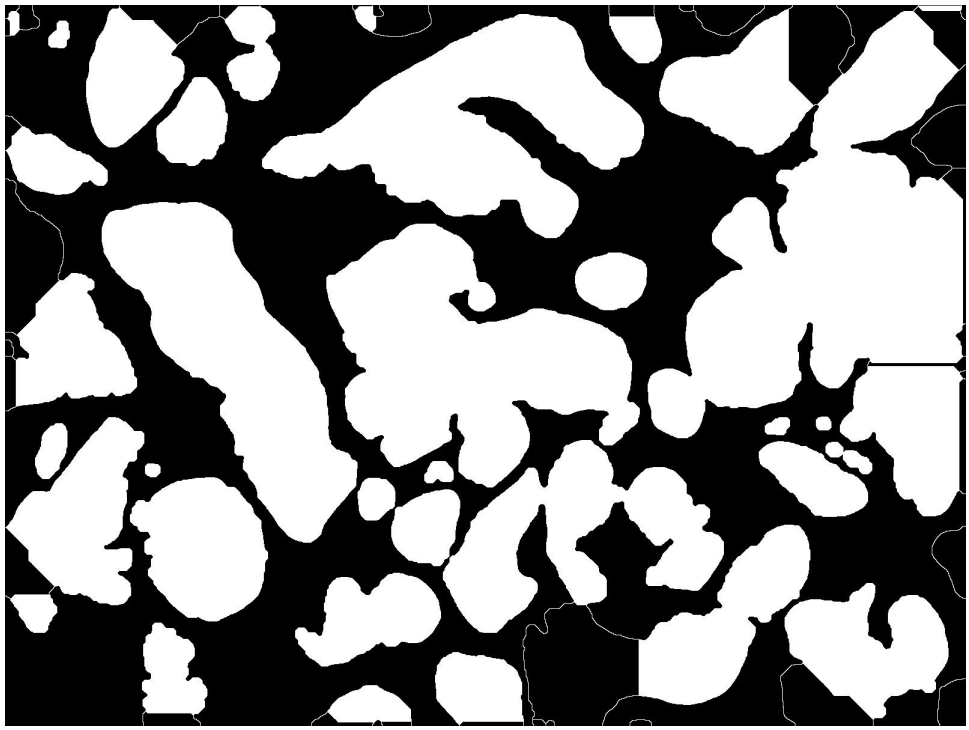


Figure 21: Watershed Method Initial Regions

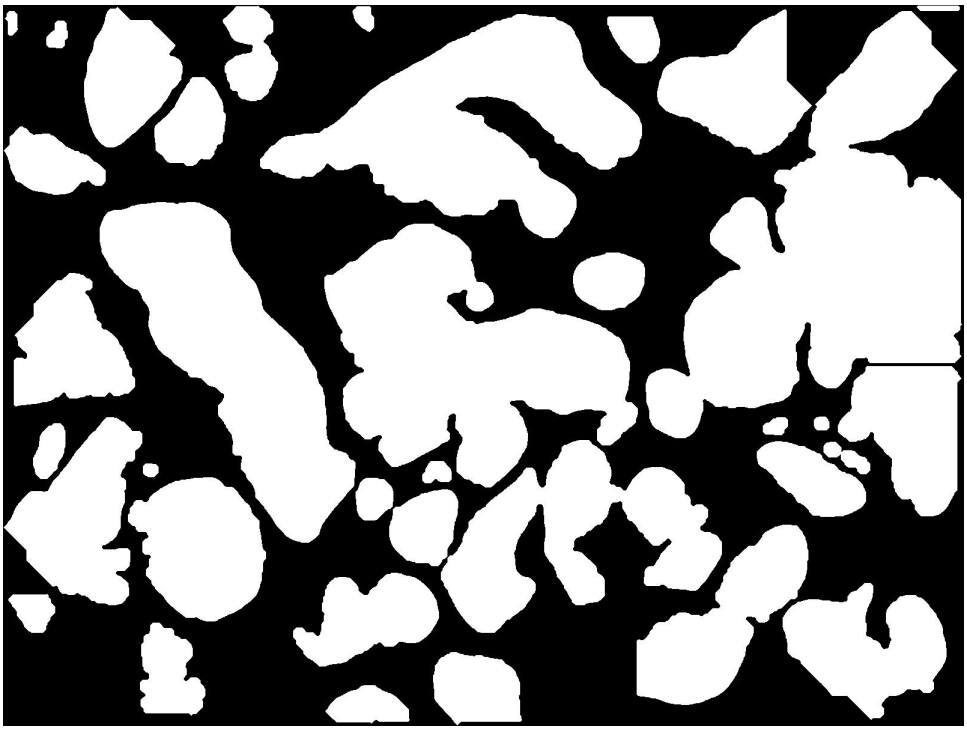


Figure 22: Watershed Method Final Binary Image

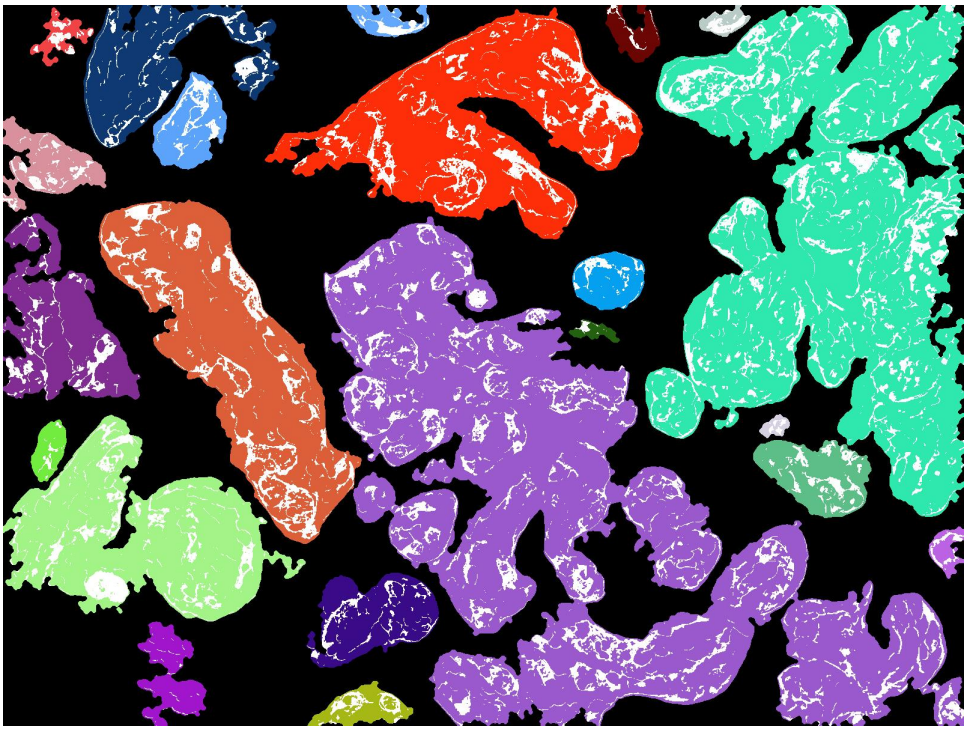


Figure 23: Edge Detection Method Final Result

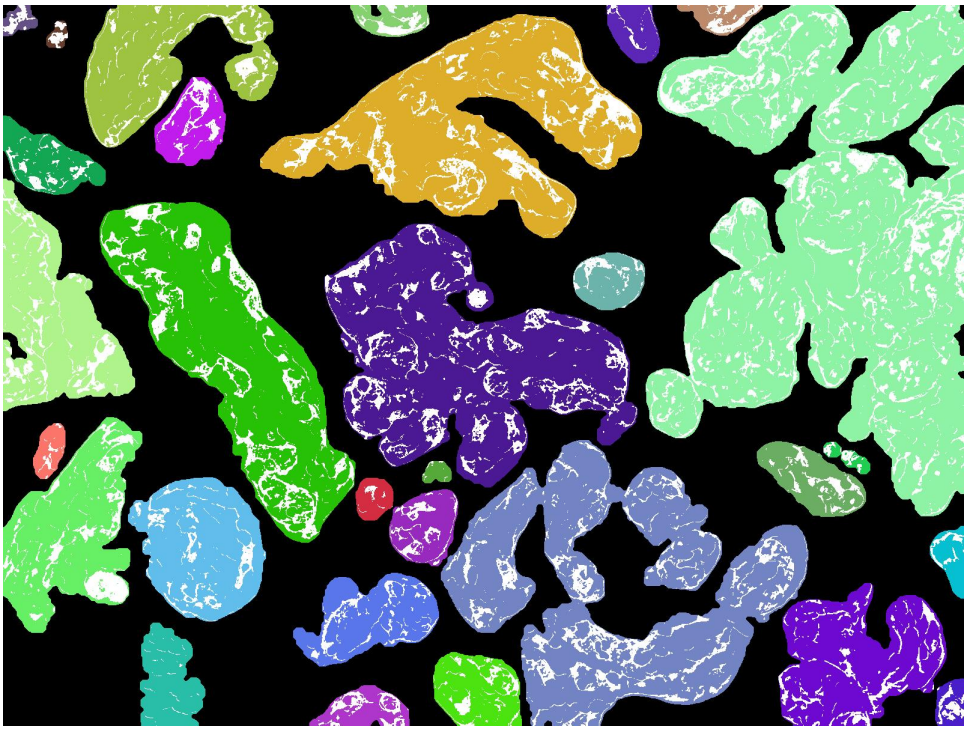


Figure 24: Histogram Method Final Result

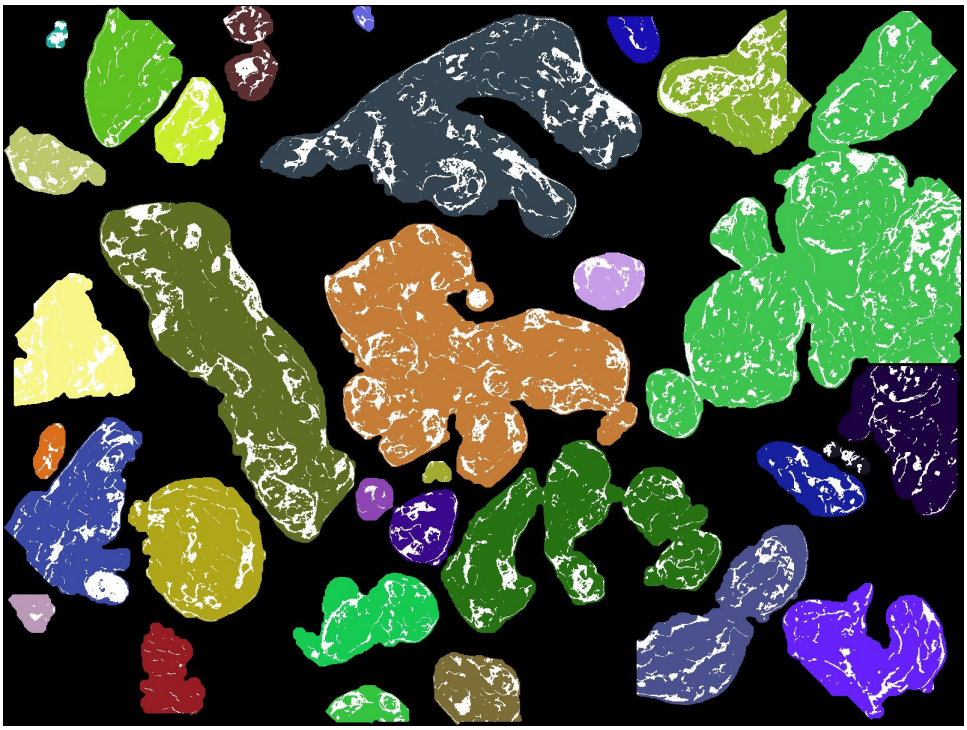


Figure 25: Watershed Method Final Result

3.1 Hand-Tracings

We hand-traced the outline of each of the blobs in the slides, and then scanned the traced images as seen in Figure 26. Using Adobe Illustrator, we extracted the outlines and made them black and set the background to white so that we could turn it into a binary image in Matlab. Once in Matlab, we ran the traced images through our function fillAREA in Appendix C. This function converts the image to a binary image, fills and labels each of the blobs, and then calculates the total and median areas of all the blobs, outputting the results.

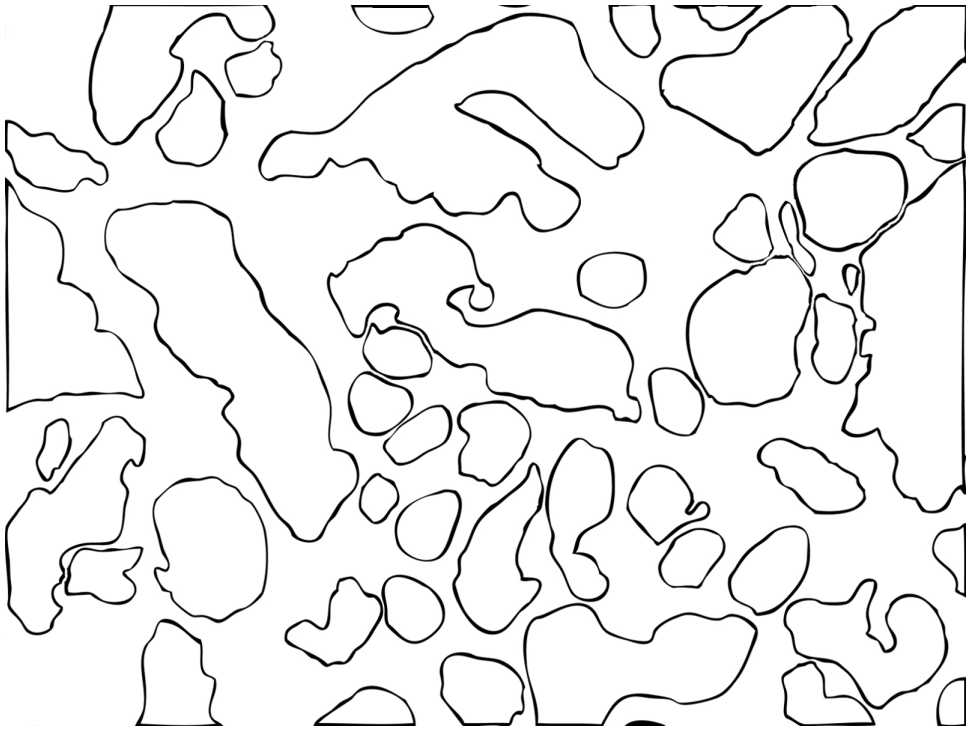


Figure 26: Hand-traced Image of Blob Outlines

We hand-traced all the vessels in the slides and scanned them as well as seen in Figure 27. In Matlab, we combined the image of the blob outlines with the image of the vessel outlines and ran a code similar to fillAREA. This code calculated the total vessel area of all the vessels in the slide and then calculated the median area of all the vessels. We now had ideal results to compare each of our methods to see which was most efficient. We calculated the total blob and vessel areas as well as the median blob and vessel areas on each our processed images and compared our results to the hand-traced results.

We had three images we traced by hand and then compared our methods to. The relative errors of our methods can be seen in Table 1.

Looking at Table 1 we can see that finding the total blob area was pretty accurate for several methods. However, due to the large variation in each individual blob, finding an accurate representation of the median blob area was at best 45.77% accurate. Although the best accuracy for the total vessel area was 36.86%, this may have been due to the segmentation only picking up specific colors of the vessels and not the fulll structure. When we traced the vessels by

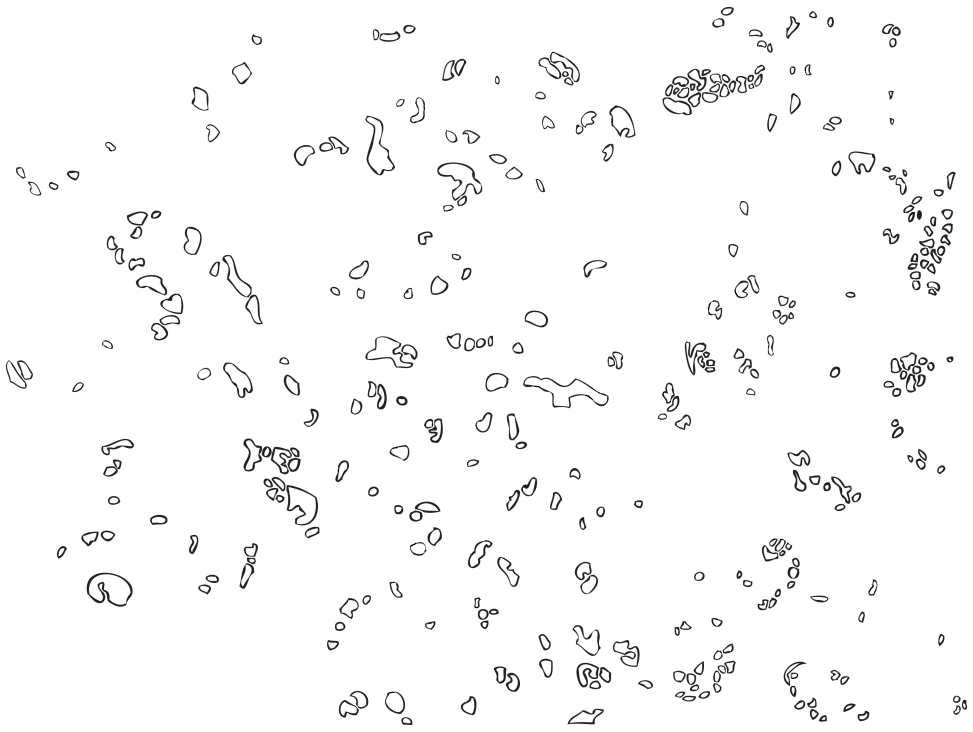


Figure 27: Hand-traced Image of Vessel Outlines

Relative Error Comparison				
Method	Total Blob Area	Median Blob Area	Total Vessel Area	Median Vessel Area
KM Euclid	22.77%	n/a	53.17%	n/a
KM Mahal RGB	24.71%	n/a	143.52%	n/a
KM Mahal Lab	23.97%	n/a	39.47%	n/a
Chan-Vese	25.77%	n/a	34.66%	n/a
Edge Detection	1.05%	45.77%	39.72%	39.94%
Histogram	1.93%	47.51%	36.86%	24.81%
Watershed	18.78%	50.32%	46.22%	45.38%

Table 1: Relative Errors of Our Methods

hand we enclosed all the shapes with the correct color in them, but our computer methods only picked out the specific color. This led to our median vessel area per blob also being only 24.81% accurate at best.

4 Future Work

In the future we would like to improve the accuracy of our methods and come up with valid way to segment complicated medical images. We believe the accuracy of the vessel segmentation could be improved by filling in the vessel areas we found, as well as smoothing them out with

some morphological operations. Combinations of our various methods could be tried together to get improved results.

5 Acknowledgements

We would like to thank the following for their help in our project:

- Dr. Salafia
- Dr. Chang

References

[1] Morten Anderson, David Belanger, Radina Droumeva, Jenny Li, Gilbert Moss, and Gabriela Palau. Quantifying clinically significant features of placental histology images: a method. Technical report, RUC, Denmark Harvard University, USA Simon Fraser University, Canada Tecnologico de Monterrey, Mexico, 2008.

[2] Rafael C. Gonzlez and Richard Eugene Woods. *Digital Image Processing*. Pearson/Prentice Hall, 2008.

[3] Zoltan Kato. Mumford-shah energy functional. PhD Course on Variational and Level Set Methods in Image Processings.

[4] Brett Shoelson. Introduction to data analysis in matlab for life scientists. Webinar. MathWorks.

A segEdge.m

```
%%Segmentation based on Canny Edge method
%By Austin Adams

%Input:X
%   X=m by n by 3 image

%Output:BW
%   BW=binary image, 1==blobs, 0==background


function[BW]=segEdge(X)

%Use the saturation channel as it has high contrast
HSVim=rgb2hsv(X);
```

```
Sat=HSVim(:,:,2);

%Find the edges using Canny method
e2=edge(Sat,'canny');

%Try and connect all the regions
se=strel('disk',4);
BW=e2==1;
bwconnected=imclose(BW,se);
BW=imfill(bwconnected,'holes');

%Remove any tiny extra pieces to hopefully split into distinct regions

BW=imopen(BW,se);
```

B segHist.m

```
%% Thresholding segmentation idea
%Written by Austin Adams

%Input:X
%   X=input image (m by n by 3)

%Output:Y
%   Y=Individual cells


function [Y]=segHist(X)

%Find the Saturation Channel
HSVim=rgb2hsv(X);
Sat=HSVim(:,:,2);

%Main Step of Histogram Method
Y=Sat>.2;

%Clean up Image
se1=strel('disk',4);
se2=strel('disk',10);

%I chose a smaller disk to close with since we just wanted to fill in gaps,
%not join all the blobs into one huge blob
closed=imclose(Y,se1);
```

```
filled=imfill(closed,'holes');
opened=imopen(filled,se2);

Y=opened;
```

C blobLabel.m

```
%%Labeling of blobs
%Written by Austin Adams

%Input:bw,thresh
%   bw=binary image we want to label
%   thresh=smallest amount of pixels we still want to call a blob

%Output:L,n,cmap
%   L=labeled image
%   n=number of blobs
%   cmap=an appropriate colormap

function[L,n,cmap]=blobLabel(bw,thresh)
[l,n]=bwlabel(bw);
sizes=1:n;
for i=1:n
    sizes(i)=length(find(l==i));
end
keepers=find(sizes>thresh);
L=zeros(size(bw));
n=length(keepers);
for j=1:n
    blobj=l==keepers(j);
    L=L+j*blobj;
end
cmap=[0 0 0;rand(n,3)];
```

D redVessels.m

```
%%Find vessels in blobs
%Written by Austin Adams


%Input:X,L,n,cmap
%   X=original image
```

```
%   L=image split into blob regions
%   n=number of blobs

%Output:L2,cmap2
%   L2=updated L with a new color identifying vessels
%   cmap2=updated color map

function[L2,cmap2]=redVessels(X,L,n,cmap)
red=X(:,:,1);
green=X(:,:,2);
hist=red>210 & red<235 & green<150;
vess=zeros(size(L));
for i=1:n
    vess(L==i & hist==1)=1;
end
L2=L;
L2(vess==1)=n+1;
cmap2=[cmap;1 1 1];
```

E segWater.m

```
%%Segmentation based on Watershed Method
%By Austin Adams

%Input:X
%   X=m by n by 3 image in double format

%Output:BW


function[BW]=segWater(X)
Y=segHist(X);

%Apply the Watershed method
D=bwdist(~Y);
D=-D;
D(~Y)=-Inf;
L=watershed(D);

%Use the connecting curves as regions
BW= imfill(L==0,'holes');

%Clean up the random line segments in the image
```

```
se=strel('disk',2);
BW=imopen(BW,se);
```

F blobMeasures.m

```
%Measures for blobs
%Written by Austin Adams

%Input:
%   L1=all blobs of original image,0=background,1-n=blobs
%   n=number of blobs
%   L2=blobs of original image with vessesls on top, n+1=vessels

%Output:
%   tBlob=total blob area
%   mBlob=median blob area
%   tVess=total vessel area in blobs
%   mVess=median vessel area in blobs

function[tBlob,mBlob,tVess,mVess]=blobMeasures(L1,n,L2)

blob=1:n;
vess=1:n;
for i=1:n
    blob(i)= sum(sum(L1==i));
    vess(i)= sum(sum(L1==i & L2==n+1));
end

tBlob=sum(blob);
mBlob=median(blob);
tVess=sum(vess);
mVess=median(vess);
```

G EuclidMaster.m

```
% Is the master m file that calls all other functions in order
% to segment, clean up and compute stats on the blobs and vessels
% type in the directory where the images are located in line 9

clear all
dirname = 'E:\Images'; % type the path to the directory of the images
imdir = dir(dirname);
```

```
%process each image
nI = length(imdir);
StatFile = cell(nI-2,1);

%start at 3 because the first two elt's of the directory are . and ..
for i = 3:nI
    j=i-2;
disp(['processing image ' num2str(i-2) ' of ' num2str(nI-2) ]);
fname = imdir(i).name;
I=imread([dirname '\' fname],'tiff');
Imod = LabProcess(I);
[ TotalSeg, Vessels, Blobs ] = KMeansEuclid(Imod,j);

%Need Blob processing
[ B BB BBB BBBB] = BlobProcess(Blobs,j);
%Vessel processing
[ V VV VVV VVVV] = VesselProcess(Vessels,j);
[L, n]= bwlabel(BBBB);
[LL,nn]=bwlabel(VVVV);
L;
LL;
n;
nn;
% [ Stats ] = Stats(L, LL, n, nn);
% Stats(j)={Stats};

Stat = Stats(L, LL, n, nn, j);
StatFile(j)={Stat};

end



function [Imod] = LabProcess(I)
%Performs an intensity thresholding in Lab space, in the L channel
% I=double(I);

R=I(:,:,1);
G=I(:,:,2);
B=I(:,:,3);
[m n d]=size(I);

[L a b]= RGB2Lab(R,G,B);
```

```
L(L>89)=100;
L(L<70)=0;
[R1 G1 B1]=Lab2RGB(L,a,b);
Imod=zeros(m,n);
Imod(:,:,1)=R1;
Imod(:,:,2)=G1;
Imod(:,:,3)=B1;
% imagesc(Imod)

end




function [ pixel_labels, Vessels, Blobs ] = KMeansEuclid(Imod,j)
%  performs a kmeans euclidean clustering in Lab color space
%  is called from Euclidmaster.m

he = Imod;
cform = makecform('srgb2lab');
lab_he = applycform(he,cform);

ab = double(lab_he(:,:,2:3));
[m n d]=size(ab);
nrows = m;
ncols = n;
ab = reshape(ab,nrows*ncols,2);

nColors = 5;
% repeat the clustering 3 times to avoid local minima
[cluster_idx cluster_center] = kmeans(ab,nColors,'distance','sqEuclidean','Replicates',3


pixel_labels = reshape(cluster_idx,nrows,ncols);
% imshow(pixel_labels,[]), title(sprintf('Hist%d Total Segmentation',j));

segmented_images = cell(1,5);
    [m n d]=size(he);

for k = 1:nColors
    color=zeros(m,n,1);
    color(pixel_labels == k) = k;
    segmented_images{k} = color;
```

```
end

BB=zeros(nColors,2);
CC=zeros(nColors,2);
for k = 1:nColors
    Q=segmented_images{k};
    J=find(Q,1);
    R=Imod(:,:,1);
    G=Imod(:,:,2);
    B=Imod(:,:,3);

%     RedDist=abs((R(J)-0.91))+abs(G(J)-0.435)+abs(B(J)-0.66);
    RedDist=G(J)+ B(J)-R(J);
    BB(k,2)=k;
    BB(k,1)=RedDist;
    WhiteValue=G(J)+B(J);
    CC(k,1)=k;
    CC(k,2)=WhiteValue;


end

% blue red pink white
% color_markers =  [151 114 184; 232 111 168; 243 155 196; 247 242 243];
BBB=BB(:,1);
%
QQ=min(min(BBB));
[Row Column]=find(BB==QQ);
%
% % QQ = sortrows(B);
VesselIdx=Row;

CCC=CC(:,2);
QQQ=max(max(CCC));
[Row Column]=find(CCC==QQQ);
BlobIdx=Row;

Vessels=segmented_images{VesselIdx};
Blobs=segmented_images{BlobIdx};
```

```
end



function [ J JJ JJJ JJJJ] = BlobProcess(X,j)
%Performs morphological operations on Blobs

RealBlobs=~X;
RealVessels=logical(RealBlobs);
[L,n,cmap]=blobLabel(RealVessels,2000);
figure
subplot(1,2,1)
imshow(X), title(sprintf('Hist%d Blobs Raw',j));
subplot(1,2,2)
imshow(L), title(sprintf('Hist%d Blobs Above 2000 Area',j));

J=L;
se=strel('disk',1);
JJ=imopen(J,se);
JJJ=imfill(JJ,'holes');
JJJJ=imclose(JJJ,se);
JJJJ=logical(JJJJ);

% figure
% subplot(2,2,1)
% imshow(X),title(sprintf('Hist%d Blobs Raw',j));
% subplot(2,2,2)
% imshow(JJ),title(sprintf('Hist%d Blobs ImOpened',j));
% subplot(2,2,3)
% imshow(JJJ),title(sprintf('Hist%d Blobs ImFilled',j));
% subplot(2,2,4)
% imshow(JJJJ),title(sprintf('Hist%d Blobs ImClosed',j));
end


function [ J JJ JJJ JJJJ] = VesselProcess(X,j)
%Morphological Operations on Vessels

RealVessels=logical(X);
[L,n,cmap]=blobLabel(RealVessels,200);
figure
subplot(1,2,1)
```

```
imshow(X), title(sprintf('Hist%d Vessels Raw',j));
subplot(1,2,2)
imshow(L), title(sprintf('Hist%d Vessels Above 200 Area',j));

J=L;
se=strel('disk',1);
JJ=imopen(J,se);
JJJ=imfill(JJ,'holes');
JJJJ=imclose(JJJ,se);
JJJJ=logical(JJJJ);

% figure
% subplot(2,2,1)
% imshow(X),title(sprintf('Hist%d Vessels Raw',j));
% subplot(2,2,2)
% imshow(JJ),title(sprintf('Hist%d Vessels ImOpened',j));
% subplot(2,2,3)
% imshow(JJJ),title(sprintf('Hist%d Vessels ImFilled',j));
% subplot(2,2,4)
% imshow(JJJJ),title(sprintf('Hist%d Vessels ImClosed',j));
end



function [ Statistics ] = Stats( L, LL, n, nn, j)
%Computes the stats on Blobs and Vessels
%L is bwlabeled Blobs, LL is bwlabeled Vessels, n is total number of blobs
%nn is total number of vessels
% stats computes areas of blobs. for each blob, it find the nonzero
% elements in vessels using nnz.  So total number of blobs, area of each
% blob, total blob area (from nnz on the entire blob image), total vessel
% area (nnz on input vessel page).

TotalBlobArea=nnz(L)
TotalVesselArea=nnz(LL);
TotalNumberofVessels=nn;
NumberofBlobs=n;
Statistics=zeros(n,8);

for i = 1:n
    B=L;
    V=LL;
    B(L~=i)=0;
```

```
    B=logical(B);
    IndividualBlobArea=nnz(B);
    V(L~=i)=0;
    V=logical(V);
    VesselAreaPerBlob=nnz(V);
    [LLL nnn]=bwlabel(V);
    NumberofVessels=nnn;
    Statistics(i,1)=i;
    Statistics(i,2)=IndividualBlobArea;
    Statistics(i,3)=TotalBlobArea;
    Statistics(i,4)=NumberofVessels;
    Statistics(i,5)=VesselAreaPerBlob;
    Statistics(i,6)=TotalVesselArea;
    Statistics(i,7)=TotalVesselArea;
    Statistics(i,8)=j;
%     %we need stats of Total Vessel Area, Average Vessel Area per blob,
%     %total blob area, and average blob area
%     %%need cell structure in program calling this one with cells for each
%     %Hist slide, each cell will have a matrix
%
%
%
%
%
%
end
```

H MahalMaster.m

```
% This m file calls all other functions necessary to run a Kmeans
% segmentation using Mahalanobis Distance, then perform morphological
% operations and compute the relevant Histology stats.
% All you need to do is change the directory path of the images (images
% should be in their own folder.
clear all
dirname = 'E:\Images'; % type the path to the directory of the images
imdir = dir(dirname);
%process each image
nI = length(imdir);
StatFile = cell(nI-2,1);

%start at 3 because the first two elt's of the directory are . and ..
for i = 3:nI
```

```
    j=i-2;
disp(['processing image ' num2str(i-2) ' of ' num2str(nI-2) ]);
fname = imdir(i).name;
I=imread([dirname '\' fname],'tiff');
class(I)
[TotalSeg, Vessels, Blobs]=KMeansMahal(I,j);



%Need Blob processing
[ B BB BBB BBBB] = BlobProcess(Blobs,j);
%Vessel processing
[ V VV VVV VVVV] = VesselProcess(Vessels,j);
[L, n]= bwlabel(BBBB);
[LL,nn]=bwlabel(VVVV);
L;
LL;
n;
nn;
% [ Stats ] = Stats(L, LL, n, nn);
% Stats(j)={Stats};

Stat = Stats(L, LL, n, nn, j);
StatFile(j)={Stat};

end



function [ TotalSeg, Vessels, Blobs] = KMeansMahal(I,j)
%
% If input image matrix is class uint8, then you need color markers between
% 0 and 255.

% blue red pink white
color_markers = [158 93 176; 245 159 187; 248 176 213; 254 254 254];
% Hist3 color_markers =  [90 57 159; 247 126 171; 248 176 213; 250 246 247];
% Hist2 color_markers =  [158 93 176; 245 159 187; 248 176 213; 254 254 254];
% Hist1 color_markers =  [151 114 184; 232 111 168; 243 155 196; 247 242 243];


[TotalSeg] = Kmeans(I,color_markers);
j
```

```
% pixel_labels = reshape(cluster_idx,nrows,ncols);
% imshow(pixel_labels,[]), title(sprintf('Hist%d Total Segmentation',j));
nColors=size(color_markers,1);
segmented_images = cell(1,nColors);
    [m n d]=size(I);


    BB=zeros(nColors,2);
CC=zeros(nColors,2);

for k = 1:nColors
    color=zeros(m,n,1);
    color(TotalSeg == k) = k;
    segmented_images{k} = color;

end

if j==1
        Vessels=segmented_images{2};
        Blobs=segmented_images{4};
        figure
        imshow(Vessels), title(sprintf('Hist%d Vessels',j));
        figure
        imshow(Blobs), title(sprintf('Hist%d Blobs',j));

elseif j==2
        Vessels=segmented_images{2};
        Blobs=segmented_images{4};
        figure
        imshow(Vessels), title(sprintf('Hist%d Vessels',j));
        figure
        imshow(Blobs), title(sprintf('Hist%d Blobs',j));

else
        Vessels=segmented_images{2};
        Blobs=segmented_images{4};
        figure
        imshow(Vessels), title(sprintf('Hist%d Vessels',j));
        figure
        imshow(Blobs), title(sprintf('Hist%d Blobs',j));
end
```

```
end


function [TotalSeg] = Kmeans(rgb,color_markers)
% performs the Kmeans Calculation, computing the covariance matrix and
% minimizing distance in each color (rgb) divided by the variance in each
% color
%separate the color components
r = double(rgb(:,:,1));
g = double(rgb(:,:,2));
b = double(rgb(:,:,3));

% segmented_images=cell(4,1);

%compute covariance matrix
%change the resize factor for faster computation
ims = imresize(rgb,1);
rs = double(ims(:,:,1)); gs = double(ims(:,:,2)); bs = double(ims(:,:,3));
C = covmatrix([rs(:)'; gs(:)'; bs(:)']') ;


%diagonalize C to speed up computation of inner product
[u v] = eig(C); Ceval = diag(v); Cevec = u; T = Cevec';
w1 = Ceval(1); w2 = Ceval(2); w3 = Ceval(3); %eigen-basis
nC = size(color_markers,1);
distance = zeros(size(r,1),size(r,2),nC);

% distance2=zeros(size(r,1),size(r,2),1);

%loop over the color markers, computing each pixel's distance
for k = 1:nC
rd = r - color_markers(k,1); gd = g - color_markers(k,2); bd = b - color_markers(k,3);
%explicit transformation to eigenbasis to preserve
%vectorization
v = cat(3,T(1,1)*rd + T(1,2)*gd+ T(1,3)*bd,...
T(2,1)*rd + T(2,2)*gd+ T(2,3)*bd,...
T(3,1)*rd + T(3,2)*gd+ T(3,3)*bd);

%inner product (after diagonalization)

distance(:,:,k) = w1*(v(:,:,1).^2)+w2*(v(:,:,2).^2)+w3*(v(:,:,3).^2);
end
```

```
% %find the vessels segment experiment
% for k = 1:nC
% rd2 = r - color_markers(k,1);
% %explicit transformation to eigenbasis to preserve
% %vectorization
% v = cat(3,T(1,1)*rd2,T(2,1)*rd2,T(3,1)*rd2);
%
% %inner product (after diagonalization)
%
% distance2(:,:,k) = w1*(v(:,:,1).^2)+w2*(v(:,:,2).^2)+w3*(v(:,:,3).^2);
% end
```

I ChenVeseMaster.m

```
%make sure your path to images is correct
%change method in chen vese call on line 21

clear all
dirname = 'E:\Images'; % type the path to the directory of the images
imdir = dir(dirname);
%process each image
nI = length(imdir);
StatFile = cell(nI-2,1);

%start at 3 because the first two elt's of the directory are . and ..
for i = 3:nI
    j=i-2;
disp(['processing image ' num2str(i-2) ' of ' num2str(nI-2) ]);
fname = imdir(i).name;
I=imread([dirname '\' fname],'tiff');

Imod = LabProcess(I);
[ TotalSeg, Vessels, EucBlobs ] = KMeansEuclid(Imod,j);
Blob=chenvese(I,'whole',500,0.1,'multiphase');
[m n d]=size(Blob)
Blobs=imresize(Blob, [1200 1600]);
[m1 n1 d1]=size(Blobs)
figure
imshow(Blobs), title(sprintf('Hist%d Chen Vese Blobs',j));

%Need Blob processing
[ B BB BBB BBBB] = BlobProcess(Blobs,j);
```

```
%Vessel processing
[ V VV VVV VVVV] = VesselProcess(Vessels,j);
[L, n]= bwlabel(BBBB);
[LL,nn]=bwlabel(VVVV);
L;
LL;
n;
nn;
% [ Stats ] = Stats(L, LL, n, nn);
% Stats(j)={Stats};

Stat = Stats(L, LL, n, nn, j);
StatFile(j)={Stat};

end
```