Detecting and modelling Ferritin and vessel boundaries in EM images

Nicolae Mariuta - rqt629@alumni.ku.dk

Main Supervisor: Jon Sporring

Co-Supervisor: Casper Hempel

Department of Computer Science
University of Copenhagen

Master's Thesis

August 2016

Abstract

This thesis focuses on detecting and measuring the intensity function of the ferritin molecules that are located in blood vessels from the brain. For this purpose, I analyzed a set of 10 images obtained using TEM(Transmission Electron Microscope). The main topics that are discussed are: noise filtering, ferritin molecules destection, blood vessels' boundary detection and analysis of the distribution of ferritin molecules inside the blood vessels.

For noise filtering, I show that the detection of ferritin molecules is slightly improved by using Median or Wiener filters for reducing the grainy noise produced by the TEM images.

I demonstrate that Blob Detection using Laplacian of Gaussian and Template Matching with Squared Difference are effective in highlighting the ferritin molecules. Using these transformations together with Mahalanobid Distance Classifier, I obtained 0.98 sensitivity and 0.88 specificity for detection on the data that I selected for training and testing. By focusing the detection on the interior of blood vessels and removing the small connected components that are detected, the ferritin molecules are efficiently identified in all images from the dataset.

The dependency between the location of ferritin molecules within the blood vessels and their density function is important to calculate, so automatic boundary detection could be useful for the study. I try to use the parameters provided from computing Gradient Magnitude and Hessian Matrix but the accuracy of detection was too low. I built detection models based on pixel classification with SVM(Support Vector Machine) using different types of input data but the accuracy of predictions was too low to obtain automatic boundary detection. As a result, I had to annotate the boundaries of the blood vessel.

Finally, for analyzing the distribution of ferritin molecules, I select to compute inhomogeneous K function, L function and pair correlation function. Each of these measurements prove graphically that there is clustering within the ferritin molecules in the blood vessels.

Table of contents

Chapter 1

Introduction

In this thesis I present the work I did on analyzing Electronic Microscope (EM) images of blood vessels with the purpose of detecting and analyzing spatial distribution of the Ferritin molecules that are usually located inside of the blood vessels, close to boundary.

The project is part of a more complex research project conducted by Casper Hempel for Rigshospitalet that targets the sugar crystal structures which are located on the walls of the blood vessels. The Ferritin molecules are attached to this sugar structure and it might be useful for the research to find if there is any clustering within these particles.

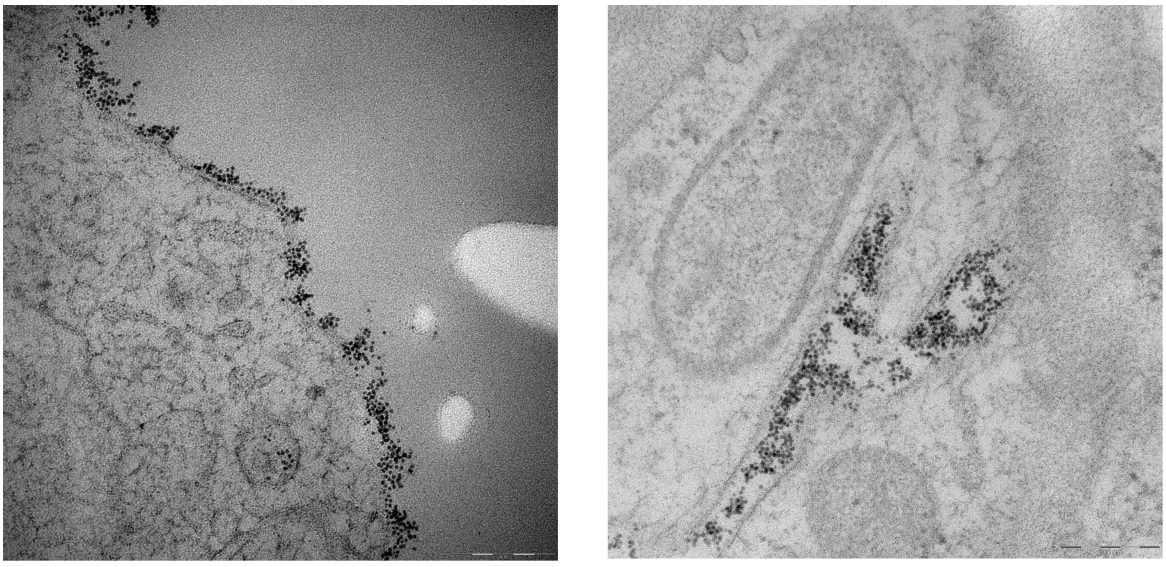


Fig. 1.1 Two typical images from the dataset that I worked with: part of a thick blood vessel in the left image and a thinner one to the right. The Ferritin molecules are the black, spherical particles inside the vessels.

Casper Hempel provided me with the dataset from Appendix A containing 10 images with the size of 2048x2048 pixels and Figure 1.1 shows two examples of images that I worked with. These images were obtained using a Transmission Electron Microscope (TEM), Philips CM100. Contrast substances were used to highlight the Ferritin molecules for the microscopy, which caused that the sugar crystals structure and other particles to not appear in the images.

1.1 Objectives

In order to analyze the Ferritin molecules intensity distribution, I worked on four objectives that also represent the main chapters of this Thesis:

- **Noise analysis and filtering:** the method used to acquire the images caused them to be very noisy, so I started by trying to understand the type of noise and then I used several methods to reduce its impact on further image analysis.
- **Ferritin molecules detection:** this part of the thesis describes the techniques that I used to highlight, model and detect the Ferritin molecules using Image Processing algorithms followed by Machine Learning techniques.
- **Blood vessels' boundary detection:** the boundaries are very unclear in the available images and this objective is not mandatory but it helps with analyzing the distribution of ferritin molecules and also further research at Rigshospitalet.
- **Measuring Ferritin molecules intensity function:** The last step is to model the detection of the molecules and use Spatial Point Statistics methods to find out if there is clustering in their distribution.

For each chapter I will analyze the results of the approaches that I relied on and the thesis ends with a conclusion presenting a summary of these results and suggestions for future work on the project.

1.2 Research methods

In order to deal with the objectives described previously, Image Processing and Machine Learning techniques were used to analyze the images that were provided.

I used Matlab to analyze and visualize the results of the different algorithms that I experimented. For some parts where computation duration was an issue, I also tried Python packages for the option of using CUDA in order to obtain a lot faster computations.

As a start, I considered some work done by my supervisor, Jon Sporring, in some attempts to detect and model the Ferritin molecules that were also written in Matlab together with the support of the R's spatstat library for applying Spatial Point Statistics.

Using guidance from Casper Hempel, I selected patches from the available images and annotated them manually in order to build the data used for training and analyzing the performance of the detection models.

Chapter 2

Noise Analysis and Filtering

2.1 Motivation

All of the images from the dataset seem to be degraded because to the method that was used to generate them and removing or at least reducing the noise might improve the quality of the detection that I want to make. There are actually two reasons why the quality of the images from the dataset is quite low:

- There is random grainy noise on all surface of the images and it is, the most likely, generated by the TEM when the electrons bombard the piece of tissue that is analyzed. The Figure Figure 2.1 shows how this noise make the shape of ferritin spots hard to detect and it also makes hard to distinguish the molecules that are very close to each other.
- When the images are generated, the TEM bombards with electrons a sample with a specific thickness. If the electrons ray does not fall perpendicularly on the surface of the sample, then the quality of the resulting image is altered and that is why in the images from the dataset, the boundary of the vessel is more visible in some areas and almost impossible to distinguish it from the tissue in other parts of the blood vessel.

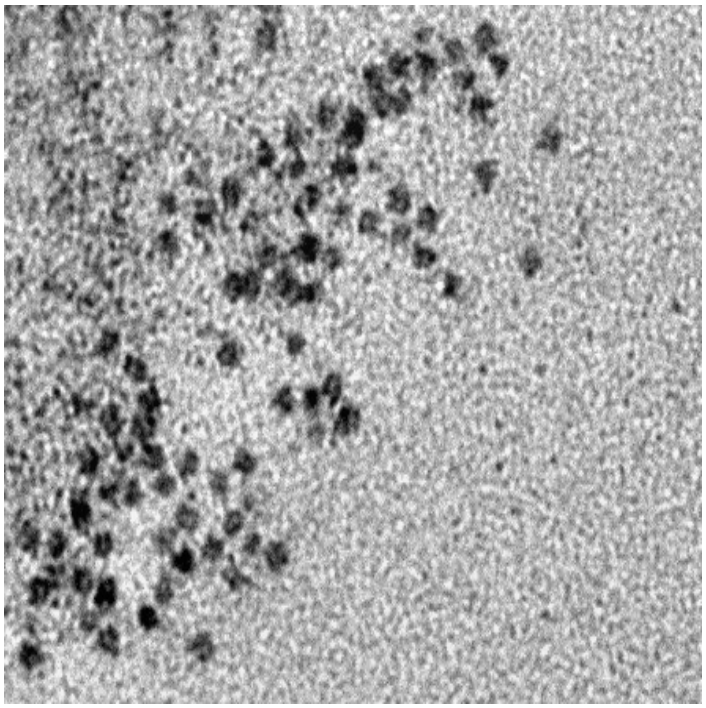


Fig. 2.1 Part from Image A.7, focused on some ferritin molecules located close to the boundary of the blood vessel, that shows how noise affects the resulting images from TEM.

There is not much that can be done about the second reason why the images are degraded but there are certainly different methods that can reduce the noise generated by the microscope. In order to deal with this noise, I used two approaches:

- **Noise signal analysis:** with the purpose of describing, isolating and removing the noisy signal that is generated by the TEM.
- **Image filtering:** testing different types of filters together or separately in order to make the images more clear and improve the performance of the Ferritin molecules detection. In this chapter I will describe the filters that I tested but their influence on detection will be shown in Chapter 3.

2.2 Noise Signal Analysis

As already mentioned, there appears to be an uniform signal generated by the TEM all over the image and I considered that it might be possible to obtain its parameters and remove it from the images.

I noticed that in the interior of the blood vessels, where there are no ferritin molecules, there are no other particles visible to the microscopy procedure that was used, so there is only background color and noise signal in those areas. So in order to analyze the noise, I cut six patches with size 256x256 pixels from six different images in the dataset in areas where there are no visible molecules or artifacts. The Appendix B shows how I chose the patches and having the size of the images power of 2, it is easier to make further computations.

The best way to analyze the signals in these patches is to visualize them frequency domain, so in the next stage Fast Fourier Transform was computed for each of the patches. Followed by the logarithm for the absolute values obtained from the transform, the results are easier to visualize as they are shown in Figure 2.2.

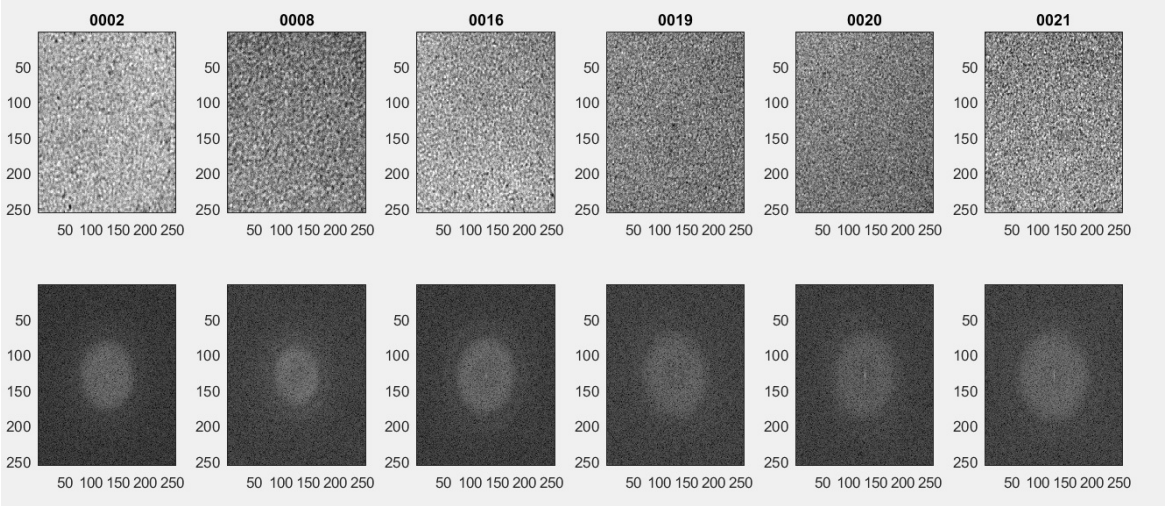


Fig. 2.2 Plot showing the noise patches that I obtained together with their corresponding logarithm of the Fourier Transform.

In order to make a better comparison of the results, I extracted the pixel values from the middle horizontal slice for each transform of the noise patches and plotting them together resulted in Figure 2.3.

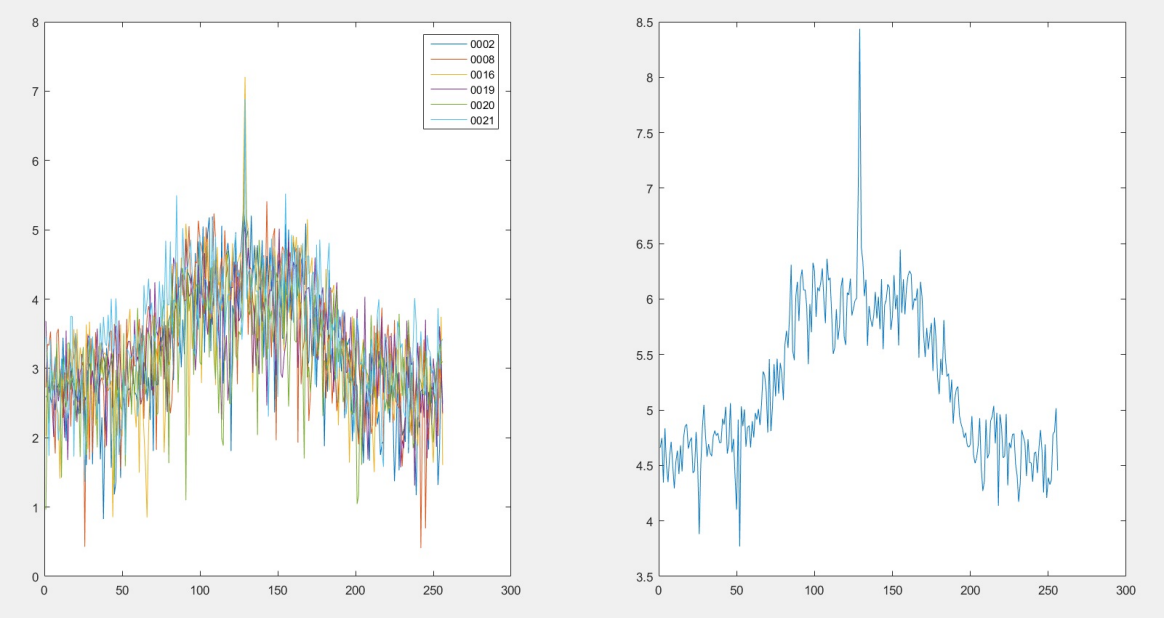


Fig. 2.3 Plot for the middle horizontal slice from the transform of each noise patch. On the right is the average of these slices.

It appears that the frequency characteristics for noise samples are quite different for each image. This might mean that each image has different characteristics of the noise and the signal generated by the electrons of the microscope is just random. By this analysis, I consider that it might be needed to find noise characteristics for each image in order to remove it. Finding manually patches of noise might be tough for large number of images and I did this analysis hoping to find a general procedure for dealing with the noise generated by TEM. And there are also images where density of tissue and ferritin particles did not allow to find any 256x256 patch with noise and background only.

However, I also tried for an image to calculate the Fourier Transform for both: the original image and the noise patch. Next, I subtracted them hoping that this will remove the noise. But from Figure 2.4 appears that most of the important information disappears by doing this.

Perhaps there are some other computations that can be done using the noise samples but from the experiments that I tried, I concluded that it might consume too much time to completely separate the noise from the useful information in the images, which I considered that it might not be necessary for the final objective of the project, so I decided to try some more basic filters in order to reduce the noise and improve the quality of the analysis.

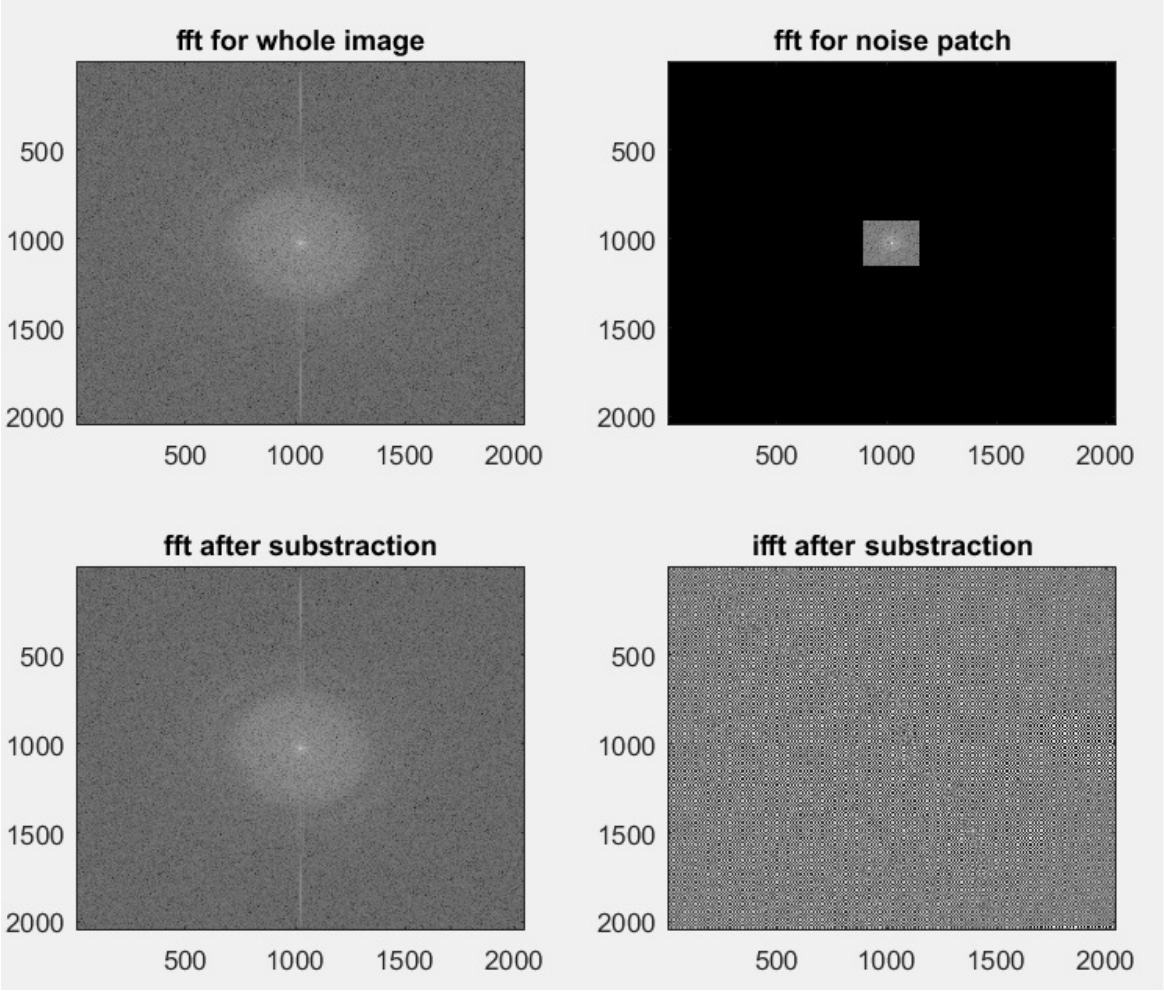


Fig. 2.4 One attempt to remove the noise in the image. fft means Fast Fourier Transform and IFFT is the Inverse Fourier Transform

2.3 Image Filtering

In this section I will present several techniques that I tried with the purpose of reducing the noise and improving the quality of detection.

The analysis for the influence of these algorithms on the detection of ferritin molecules is shown in Chapter 3.

2.3.1 Bias Correction

Some images from the dataset, like Figure A.6 or Figure A.7, use to have a variation in the background color: it is lighter in the center and darker around the corners. This issue may have an impact on the detection and it can be solved using bias correction.

I used a function implemented by Jon Sporring that fits a 2 dimensional second degree polynomial and returns the bias field of the image. By substracting the bias field from the image, I obtain the corrected image as in Figure 2.5.

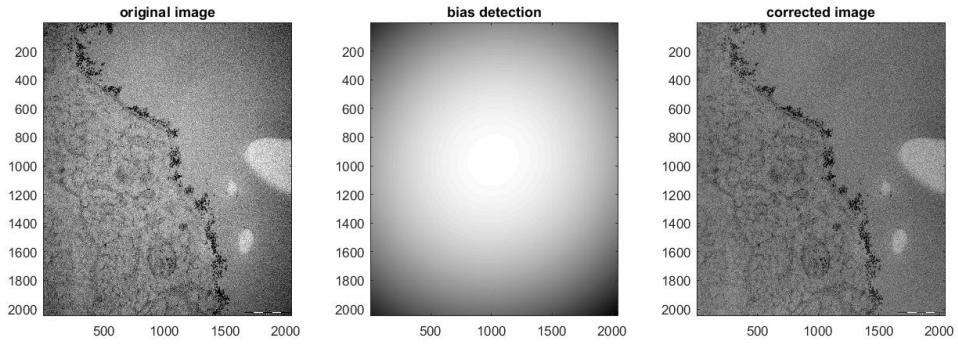


Fig. 2.5 The result of applying the bias correction to one image from the dataset: the result from the right is obtained after subtracting the bias field from the original image.

The result shows that the background color becomes uniform, so the bias field should no longer have an influence on the ferritin molecules and vessel boundaries detection. Following this conclusion, I decided to apply bias correction to the images before any operation I do for the rest of the research.

2.3.2 Histogram Equalization and Median Filter

I assumed that the noise from the dataset I work with is a common issue in the electronic microscopy so I did some literature research to find what are the more common techniques to deal with it. I found in the article of Naresh Marturi [5] that the noise of the images obtained using EM can be reduced by using Histogram Equalization followed by Median Filter.

I tried this algorithm and I applied median filters with different sizes and the results from Figure 2.6 show that the ferritin molecules become more visible and with higher contrast from the background after applying the procedure described in Naresh Marturi [5] for using median filter with sizes 3x3 or 5x5. For larger filters, the image becomes blurred and the molecules might become harder to separate.

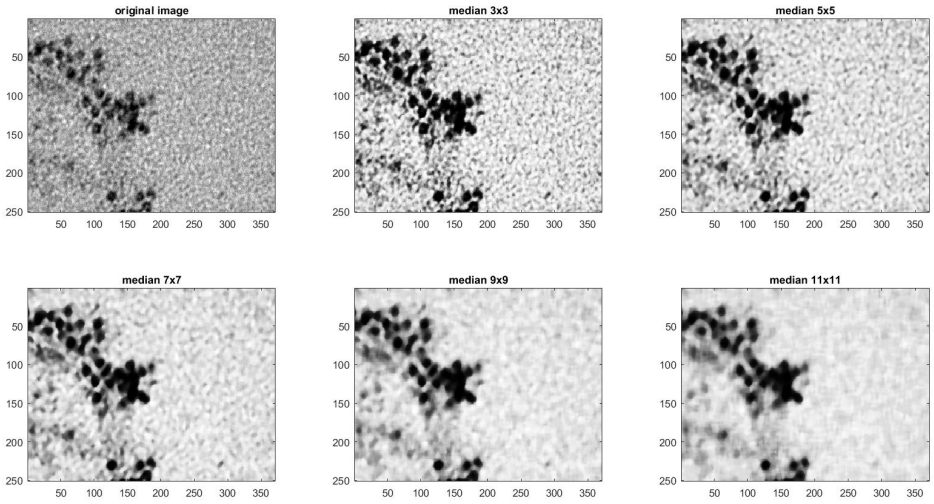


Fig. 2.6 Zoomed part of an image from dataset after applying histogram equalization and median filter with different sizes.

The results show that median filter does not remove the noise from the background but it increases the contrast of the ferritin molecules, which might increase the accuracy of analyzing these images. Mostly for experimental purpose I also tried histogram equalization together with average filter but the result from Figure 2.7 show that it has a worse effect.

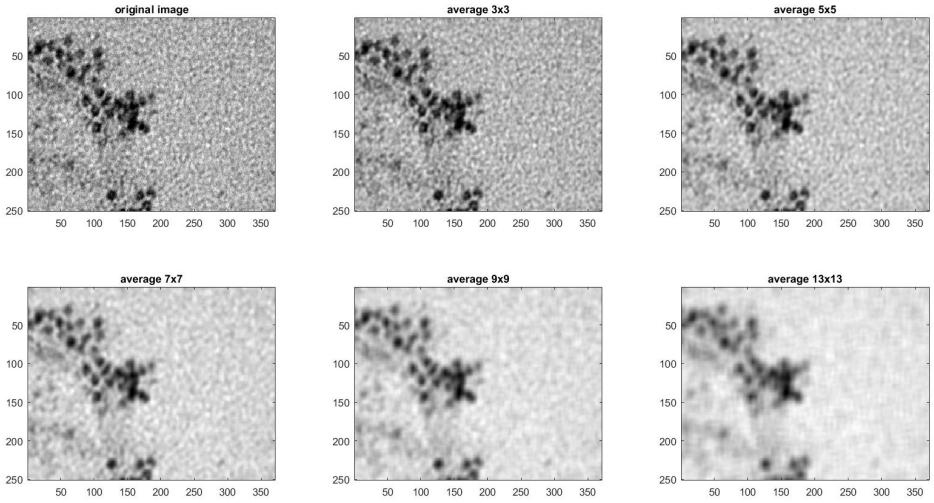


Fig. 2.7 Zoomed part of an image from dataset after applying histogram equalization and average filter with different sizes.

2.3.3 Histogram Equalization and Wiener Filter

While searching for alternatives, I found in the article by Himmat S. Kushwaha [3] that Wiener filter is also a popular choice because it incorporates prior knowledge about the noise embedded in the signal and also the spectral density of the object being imaged.

I applied the Wiener filter after histogram equalization in order to check the effects and I obtained the results from Figure 2.8.

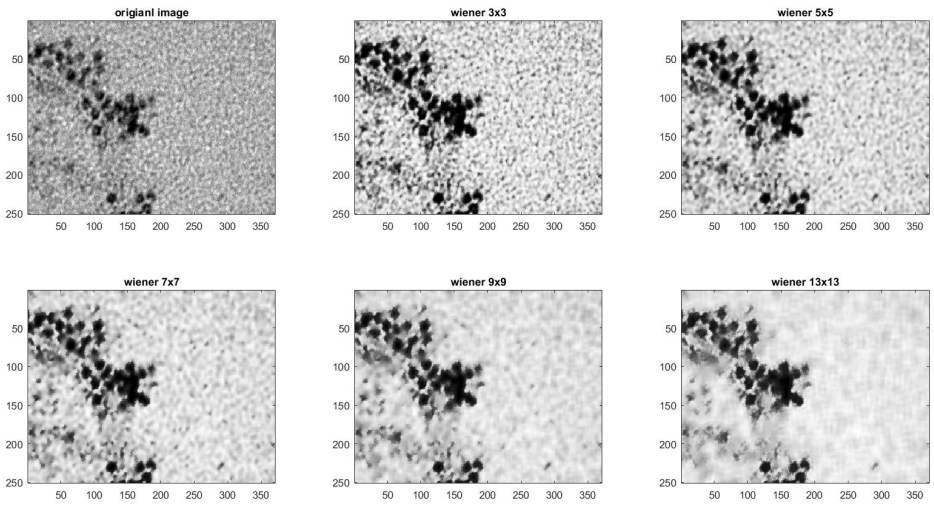


Fig. 2.8 Zoomed part of an image from dataset after applying histogram equalization and wiener filter with different sizes.

Just as for median filter, the ferritin molecules are less blurred and their shape is easier to distinguish from the background. For larger filter sizes the molecules appear to be degraded but the grainy noise seems to be reduced.

2.3.4 BM3D filtering

I also decided to try BM3D (Image Denoising by Sparse 3D transform-domain collborative filtering) with software provided by Tampere University of Technology (http://www.cs.tut.fi/~foi/GCF-BM3D/).

This type of filter works by creating 3d arrays by stacking together similar image neighborhoods and then effective collaborative filtering is realized as shrink-age in transform domain by applying multidimensional linear transform to the groups as it is described by Kostadin Dabov and Karen Egiazarian [4]. BM3D filter produces results like in Figure 2.9.

The grainy noise seems to be removed but the ferriting molecules and the vessels boundaries shapes appear to be degraded. Also, the resulting texture of the images is quite different from what I expect the images to look like without noise.

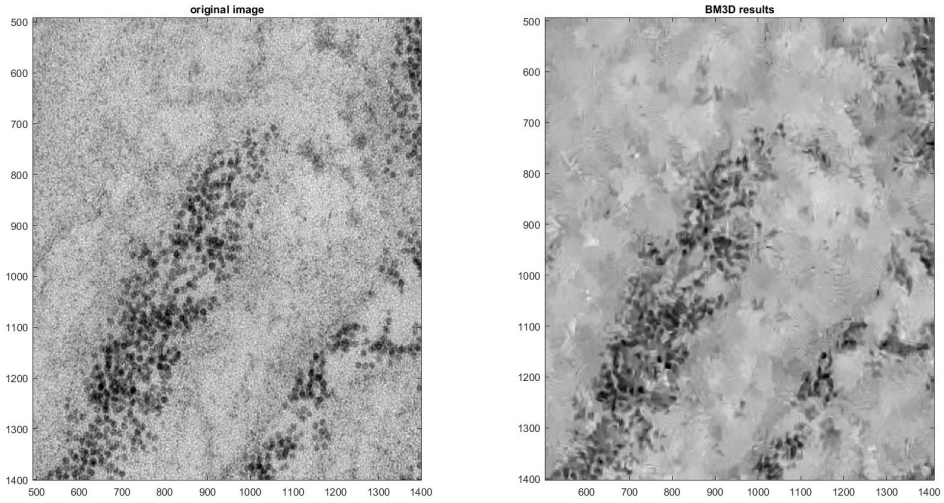


Fig. 2.9 Zoomed part of an image from dataset before and after applying the BM3D filter.

2.4 Filtering Conclusions

After the experiments I did for the Noise analysis and filtering I conclude that just Fourier Transform analysis is not enough to obtain a description of the noise signal that can enable to remove it from the images.

However, this might not be necessary since I shown that different types of filters improve the quality of the images but I will present better estimation for the performance of the median, wiener and BM3D filter after defining the detection algorithms and try them without or together with each of these filters. The resulting accuracy should give a better overview of their effect.

Chapter 3

Ferritin Molecules Detection

3.1 Overview

In this chapter I present the work I did on achieving the detection of the Ferritin molecules in the images from the dataset. The main approach is to apply Image Processing to highlight them and then to build models for binary pixel classification which allows to identify the pixels that are part of ferritin in a manner that enables to identify the location of each molecule in the images.

This task is quite challenging and as Figure 3.1 also shows, there are several reasons:

- The images are very noisy and the ferritin molecules are quite blurred. But as I shown in Chapter 2, some filters might reduce this issue.
- In several places, the ferritin molecules are very close to each other and actually overlapped which will make a challenge to identify each molecule in some clusters.
- Some ferritin molecules are very clear while others are very blurred. Even for a human eye it is very hard to be sure if some shapes are ferritin or not. For the estimation and optimization of my classification methods, I annotated manually parts of the images for building training and test sets, and it was quite hard to decide which pixels to select.
- There are boundary elements and parts of the tissue that look very much like a ferritin molecule but here it is where boundary detection might help very much: by enabling to only considering the area inside the blood vessel for detection. There are also some elements in the middle of the blood vessel that I could not be sure if they are ferritin or not.

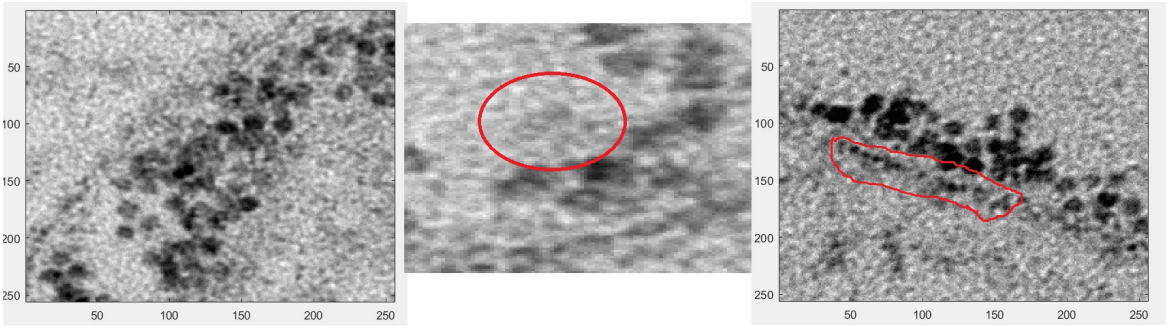


Fig. 3.1 Patches from images of dataset showing the challanges for detecting the ferritin spots: to the left there is a cluster of molecules that are overlapped, the patch in the middle highlights an area where it might be a ferritin but it is very hard to see and the patch to the right shows an area where the boundary presents elements that are very similar to the ferritin molecules.

In order to achieve the purpose and overcome the issues offered by the quality of the dataset, I started by using two techniques to highlight the particles of interest: Template Matching and Blob Detection. After analyzing the results obtained from using these two image transformations, I will present different Machine Learning methods that were used in order to find an optimal algorithm for classifying the pixels from the images into ferritin or non-ferritin.

3.2 Ferritin molecules highlighting

I start by presenting each of the two methods that I mentioned in the overview: Blob Detection and Squared Difference. After this, I will analyze and compare their performances. In the last stage, I will also describe how I decided to use them together.

3.2.1 Blob detection

The Blob Detection (Wikipedia [7]) consists of using convolution with Laplacian of Gaussian. By just convolving the Laplacian of Gaussian with different values for the scale of the Gaussian(sigma), I obtained the results from Figure 3.2. Low value for sigma results in having too many particles that are not ferritin to be highlighted while higher scales remove more and more smaller particles until some of the ferritin molecules are no longer highlighted and the pixels from particles diffuse with each other making harder to identify every molecule.

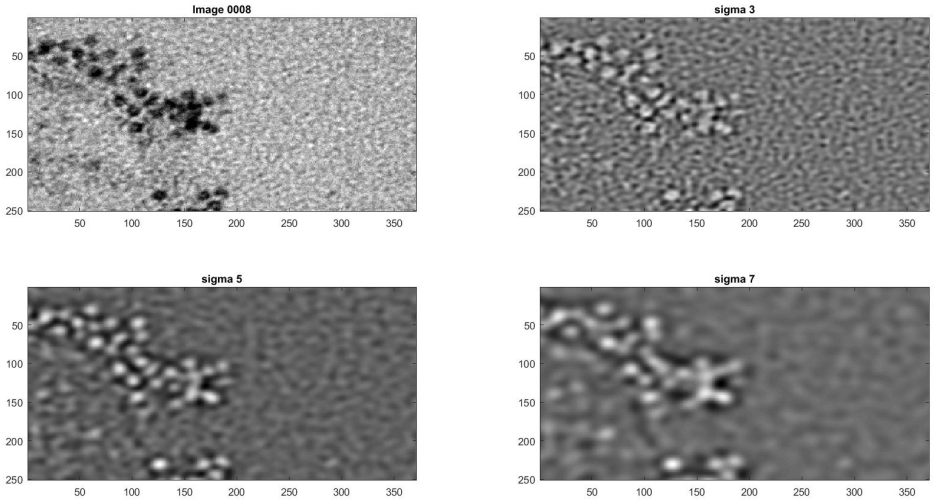


Fig. 3.2 Zoomed area of an image before and after applying Laplacian of Gaussian with different sizes.

The transformation shows that different sizes of ferritin molecules and other particles are detected by blob detection with different sizes of the Gaussian and this property was used by Jon Sporring into a more complex algorithm that is trying to detect the ferritin molecules using different values for sigma and by running his code, I obtain the results from Figure 3.3.

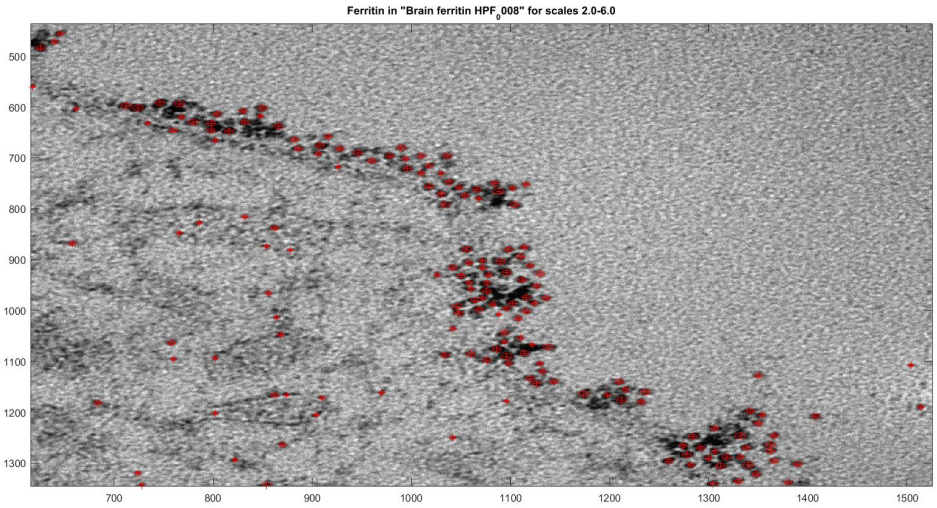


Fig. 3.3 The results of the algorithm developed by Jon Sporring to find the ferritin molecules buy applying blob detection with values of sigma between 2 and 6.

By visualizing the results obtained by Jon, I concluded that blob detection works quite well for detecting the ferritin molecules with the drawback that there are many particles in the tissue or vessel boundary that are wrongly identified and there is only one object identified in places where the molecules overlap.

However, the blob detection shows potential to be useful for Ferritin molecules detection so I will present its performance in Section 3.2.3.

3.2.2 Template Matching

As an alternative to blob detection, I also considered the Template Matching using sum of Squared Difference (Wikipedia [11]).

The first step of the procedure was to obtain a template that will be used as convolution filter. I chose the ferritin from Figure 3.4 spot that is quite visible and has a more regular shape.

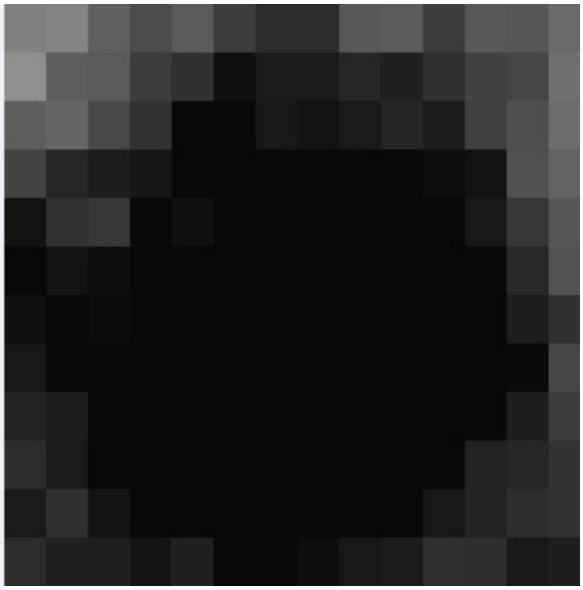


Fig. 3.4 The ferritin molecule that I decided to use for applying Template Maching.

The next step is to convolve the template as a filter and at each pixel in the image, the Squared Difference is computed. Considering that f is the whole image and that g is the template, I used the following calculation in order to make it easier to implement in Matlab:

$$\begin{aligned} TM : ||f, g_{ij}||^2 = (f - g_{ij})^T \cdot (f - g_{ij}) &= f^t \cdot f + g_{ij}^t \cdot g_{ij} - 2f^t \cdot g_{ij} \\ &= f \cdot f + g_{ij} \cdot g_{ij} - 2f \cdot g_{ij} \end{aligned} \tag{3.1}$$

where i,j are the coordinates of the convolving window.

I also considered standard convolution using the template that I defined and Figure 3.5 presents the results for using both approaches. Both highlight the ferritin spots but Squared Difference shows a result that makes a lot easier to identify each molecule while convolution makes the image to be too blurred to be used for future analysis.

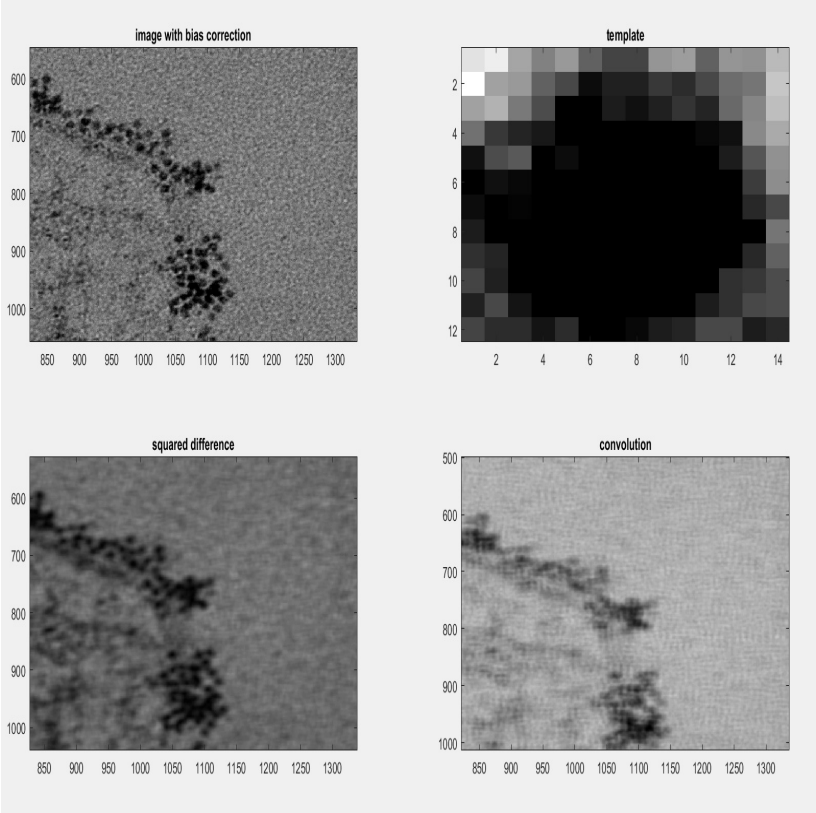


Fig. 3.5 Results of applying Template Matching using Squared Difference in comparison to convolution using the same template.

In order to get an overview of the performance, I applied a threshold with different values on the image obtained through Squared Difference which turned the image into binary and I obtained the result from Figure 3.6. I tried the same procedure for the result obtained from Convolution with the template and I obtained Figure 3.7

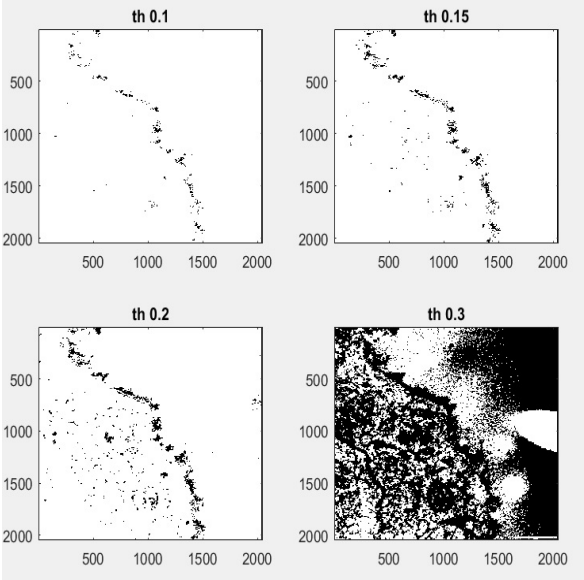


Fig. 3.6 Results of applying Squared Difference with the template and threshold with different values for the image in Figure A.6 .

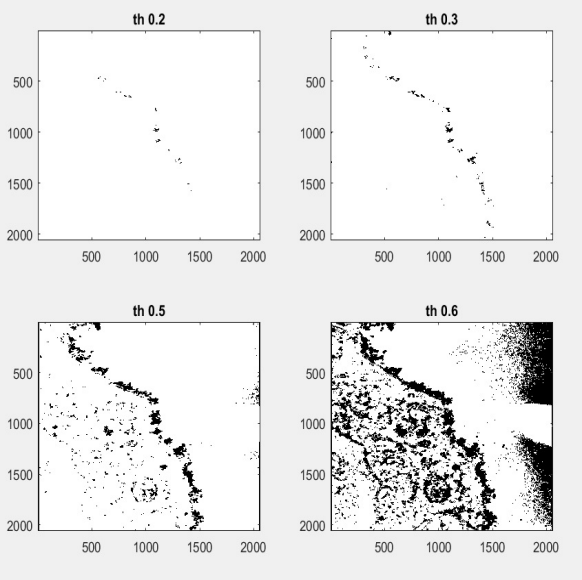


Fig. 3.7 Results of applying convolution with the template and threshold with different values for the image in Figure A.6 .

It appears that Squared Difference has lower values of Threshold for which there are some visible particles in the result and for the normal convolution there are a lot more other particles visible for the threshold values at which ferritin spots appear.

As a conclusion, Template Matching using Squared Difference is very effective for highlighting the Ferritin molecules and, unlike the Blob detection, it has lower effect on the other elements in the images.

3.2.3 Comparison between Blob Detection and Template Matching

Since both methods have shown to be very effective in highlighting the ferritin molecules with different drawbacks, I decided to compare their efficiency.

Before starting, I chose a crop from an image in the dataset and performed two types of annotations as Figure 3.8 shows: in the first annotation I marked all pixels that I considered to be inside of a Ferritin molecule and for the second annotation I only marked the center of each blob as accurate as I appreciated (I only annotated one pixel for each blob but in the figure I dilated it to make more visible). The first type of annotation is probably the most correct to use but at this stage I was interested to get an overview on how accurate are the two types of detection that I described previously and having two different approaches on analyzing the performance was very effective.

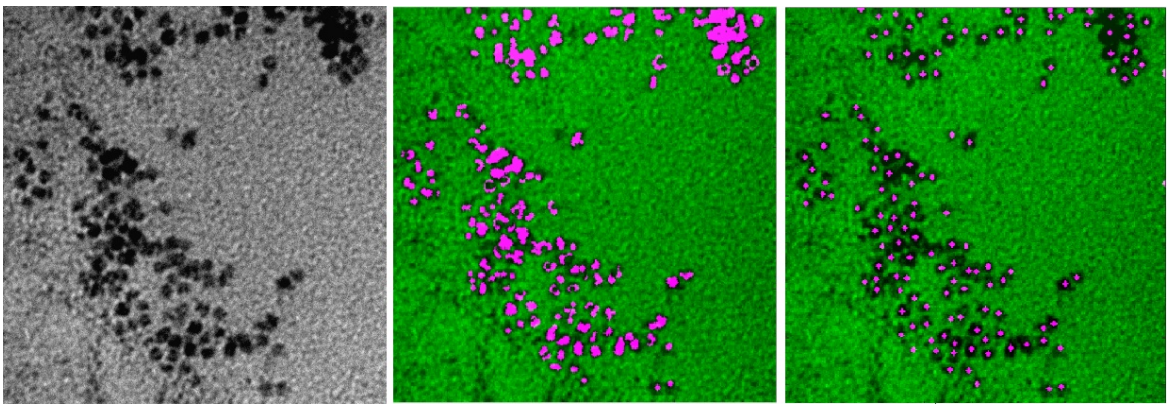


Fig. 3.8 The crop from Figure A.7 that I chose to use for analyzing the algorithms to the left. The images from the middle and right show with pink color the annotations that I made in order to test the performance of my algorithms.

Since most of the pixels in the images are not in Ferritin molecule, just calculating accuracy will not give a good measure for the quality of detection so I decided to calculate sensitivity and specificity. Sensitivity measures the proportion of positives that are correctly identified while specificity measures the proportion of negatives that are correctly identified.

For my case, the positives are the pixels in the ferritin molecules and the negatives are the other pixels.

By applying Template Matching with Squared Difference followed by threshold, I obtain the results shown in Figure 3.9 and I started by comparing to the annotation where I only marked one pixel from the center of each blob that is Ferritin molecule.

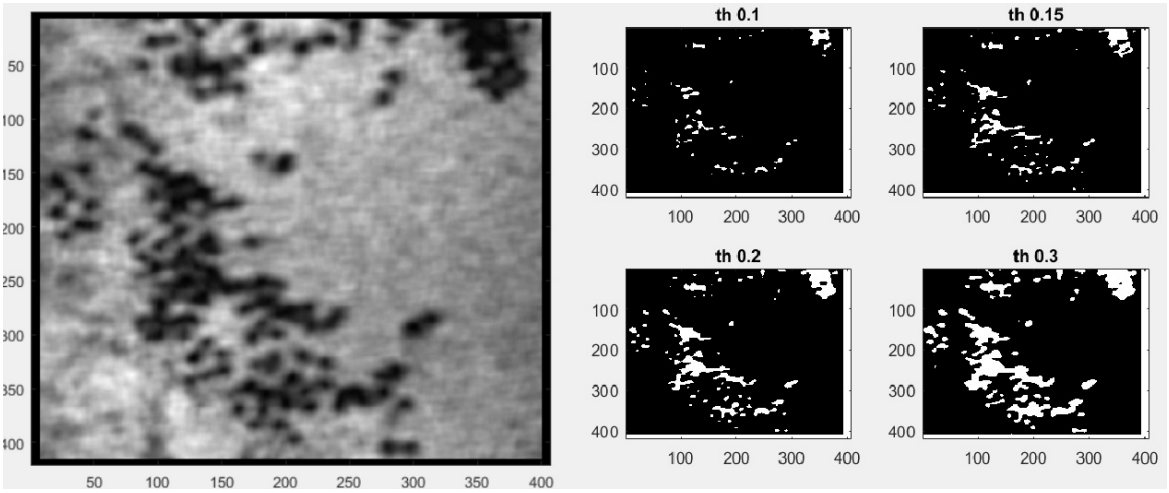


Fig. 3.9 The results of using Template Matching at described in 3.2.2 and then applying threshold with different values.

For this annotation it was not possible to also evaluate specificity since not all pixels from the molecules are marked, so I only calculated sensitivity which means that I calculate the proportion of annotation that overlaps with the detection obtained through Template Matching and threshold. I obtained the following values:

- 0.4362 for threshold value 0.1;
- 0.5937 for the threshold value 0.15;
- 0.7852 for the threshold value 0.2;
- 0.9597 for the threshold value 0.3;

and as 3.10 shows, most of the ferritin centers are covered by the detection but it is important to also show false positives are resulting.

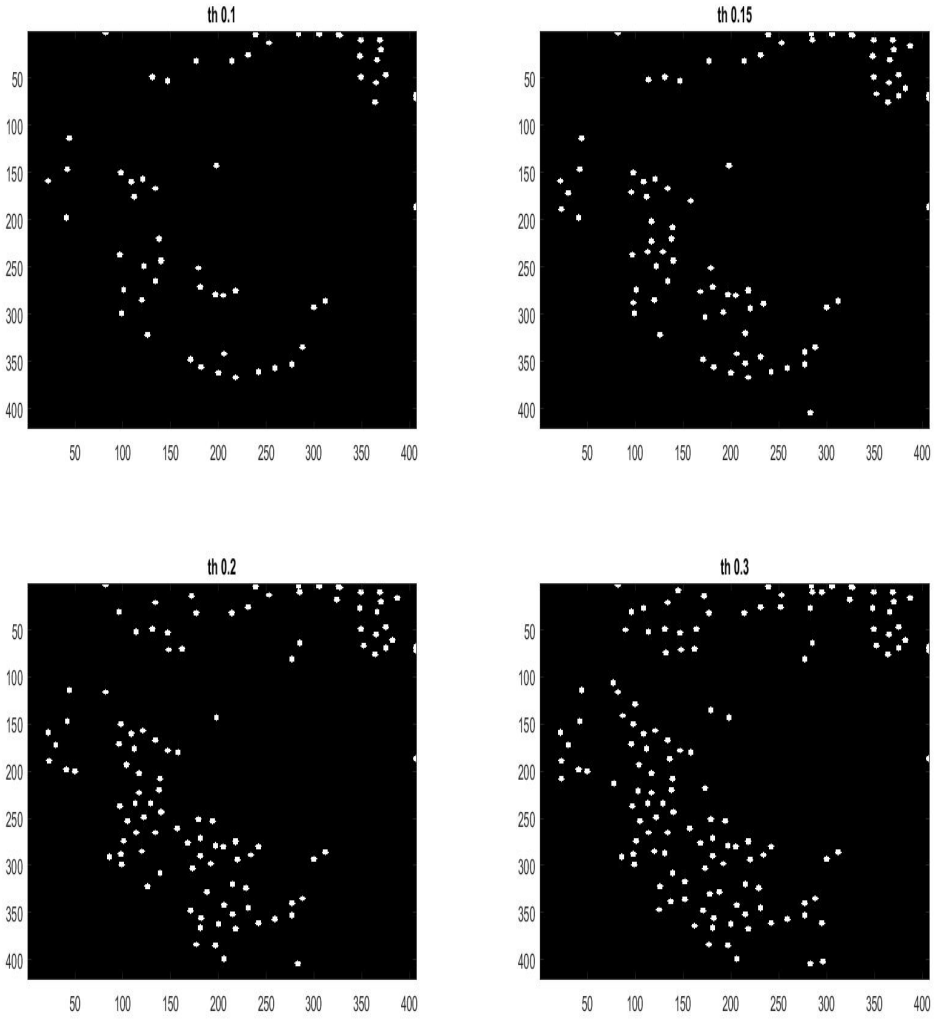


Fig. 3.10 The annotation of Ferriting molecules centers and the overlapping with the results of Template Matching and threshold from Figure 3.9

I also used the annotation of all pixels inside the ferritin molecules and the comparison to the Template Matching resulted in Figure 3.11. In this case I calculated both: sensitivity and specificity. I obtained:

- sensitivity 0.2821 and specificity 0.9319 for threshold 0.1;
- sensitivity 0.4535 and specificity 0.9203 for threshold 0.15;
- sensitivity 0.6094 and specificity 0.9055 for threshold 0.2;
- sensitivity 0.8468 and specificity 0.8693 for threshold 0.3.

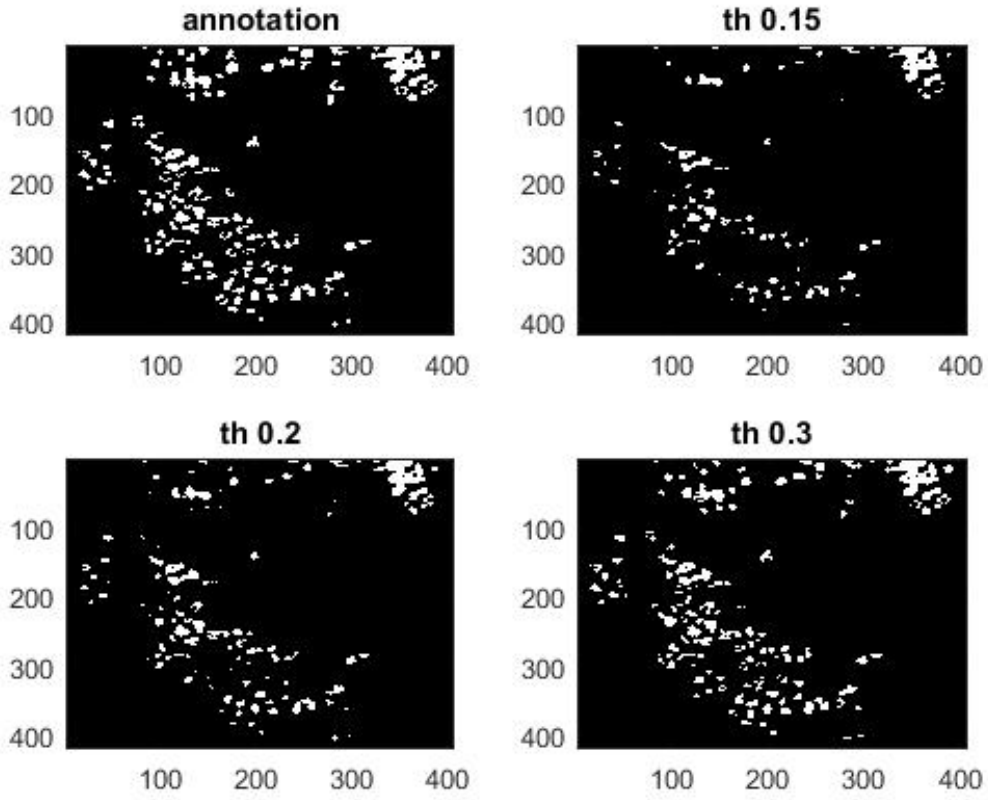


Fig. 3.11 The overlapping between the annotation of all pixels from the interior of Ferritin molecule and the result of Template Matching and threshold from Figure 3.9

Both methods that I used to analyze the performance of Template Matching with Squared Difference for detection of ferritin show a good efficiency. Most of the molecule centers are detected and even if there is not a detection for the most of the surface of ferritin, it might be better for identifying each molecule for the purpose of applying later spatial statistics.

The specificity parameter show that there are not many pixels outside the ferritin that are highlighted by Template Matching in the area where the molecules are located. I shown in Section 3.2.2 that for 0.3 threshold there are a lot of pixels that appear especially in tissue area but I can ignore that area when I make the detection.

And now, for the blob detection, I loaded the resulting blobs from the result of Jon Sporring's implementation. The data obtained by his algorithm contains the coordinates for the center of each blob and its radius. I only used the coordinates for the center of the blobs and applied dilatation with sphere structure of size 3 to an image that contains one pixel for each blob center and also for the annotation where I only marked the center of the ferritin molecules. By overlapping the annotations and blob detection I obtain the result from Figure 3.12. I did not make any calculations or other analysis because the figure shows very clearly that most of the molecules that I marked are detected by Jon's implementation but there are quite many false positives so the specificity is a lot lower than for the detection obtained by Template Matching despite the fact that the crop that I made does not contain much tissue.

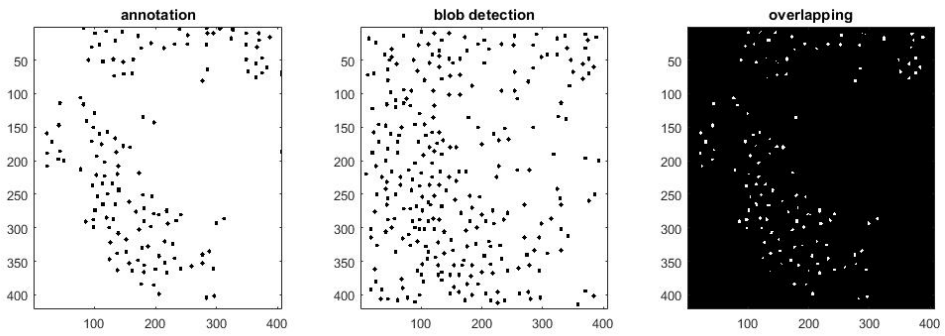


Fig. 3.12 Analyzing the result of blob detection against he dot annotation and the resulting overlapping

As a conclusion for the comparison, I would say that blob detection provides a better sensitivity and lower specificity than template matching together with a threshold value that does not obtain a too low specificity. However, both methods have proven to be quite good for highlighting the Ferritin molecules and there is potential for using them together for a pixel classification and the next section describes this approach.

3.2.4 Blob Detection together with Template Matching

I considered that used together, the two image transformations described in this section might obtain more efficient detection so in this section I illustrate this approach.

In the first stage, I considered the same crop as in the previous chapter together with the annotation of all pixels from the Ferritin molecules. Next, I applied Template Matching with the template from Figure 3.4 and Blob Detection obtaining the data shown in Figure 3.13.

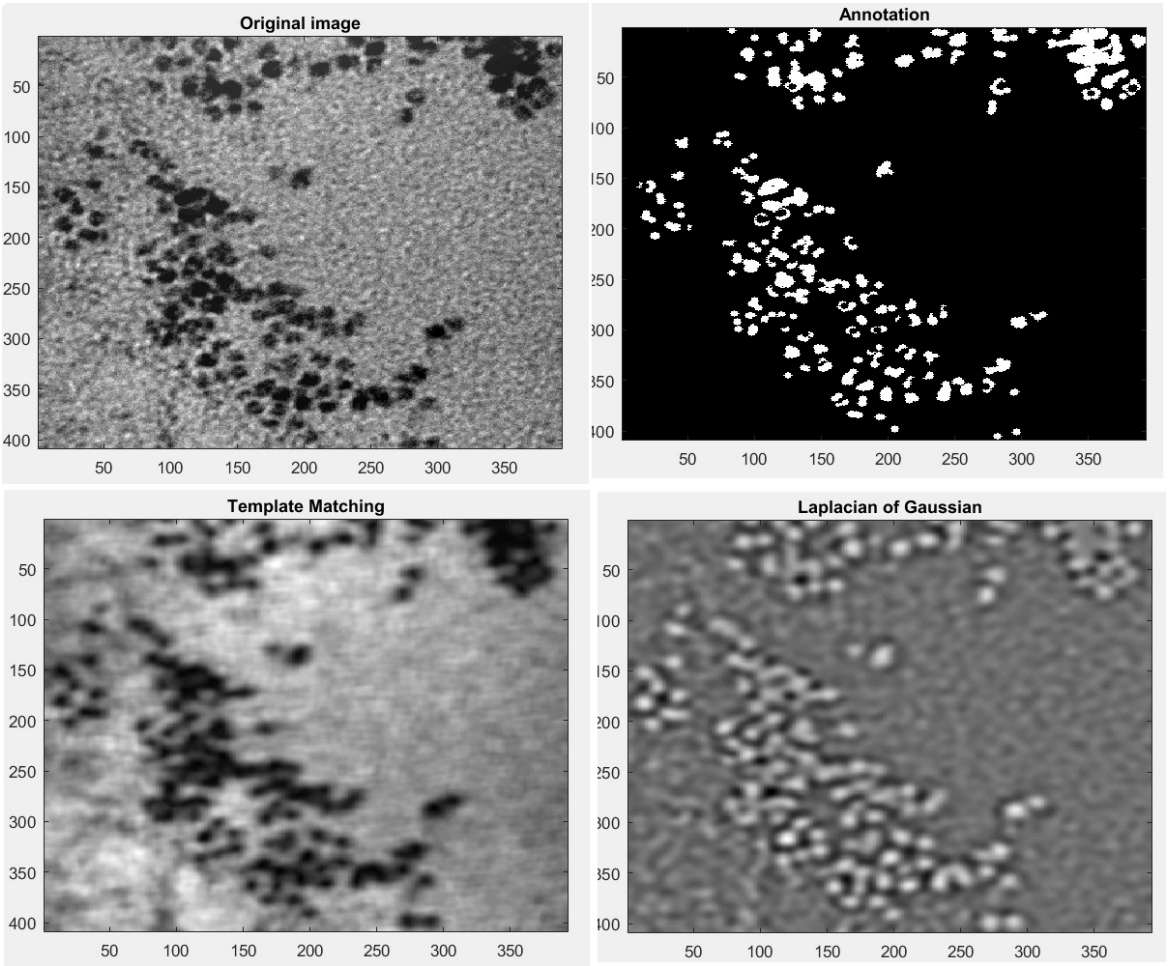


Fig. 3.13 Crop from Figure 3.9 together with the annotation of Ferritin, result for applying Template Matching with Squared Difference and the result for convolving with Laplacian of Gaussian having size sigma = 5

By using these transformations, I wanted to show if there is a correlation between the values of pixels obtained from both. So, I plotted together, for each pixel, the value obtained from Template Matching and the value obtained from Blob Detection, labeled as ferritin or not-ferritin and I obtained the graph from Figure 3.14.

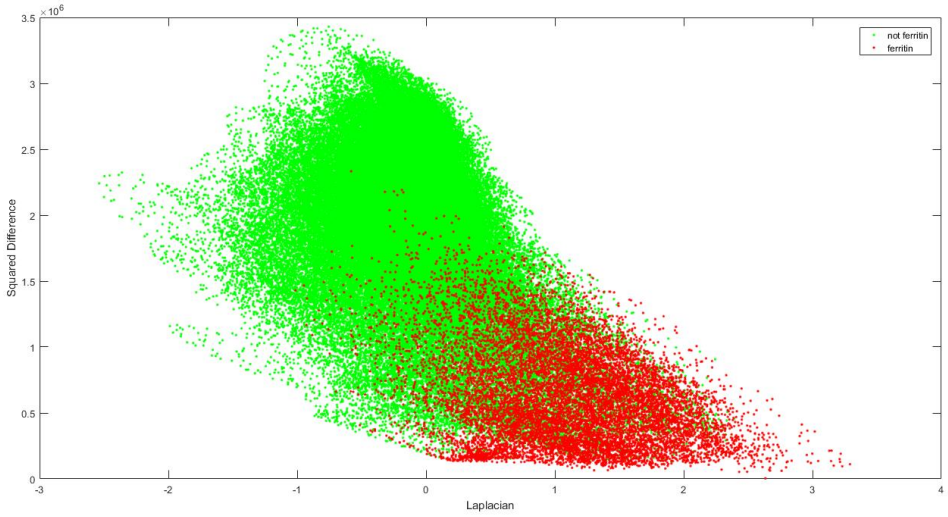


Fig. 3.14 Graph showing together the values for each pixel obtained from Laplacian of Gaussian and Squared Difference. Pixels annotated to be in Ferritin molecule are shown with red.

The graph clearly shows that most of the ferritin pixels are grouped at higher values for Laplacian of Gaussian and lower values for Template Matching. This is expected, since from Figure 3.13 can be noticed that Laplacian makes the interior of blobs whiter(pixels with value close to maximum) while Template Matching makes the interior of Ferritin Molecule to be dark (pixel values closer to 0). There is quite much overlapping between the 2 categories of pixels but there is clearly the possibility to obtain good detection of pixels that are ferritin by using a linear separator between the two categories.

Based on this observation I concluded that binary linear classification should provide quite good accuracy in identifying the pixels that are ferritin. Another observation that I can make is that not annotating the edges of the ferritin molecule might make the classifier to show more compact shapes for detection and that will enable easier splitting of grouped particles.

In order to prove the observations made with Figure 3.14 I show that Blob Detection and Template Matching enable implementing a very efficient detection of ferritin based on Mahalanobis Distance. Mahalanobis distance is a measure of distance between a point and a distribution as described in Wikipedia [9].

The detection relies on using the values of pixels obtained from the two types of transformations for performing binary classification into ferritin or non-ferritin. Before calculating Mahalanobis distance, I normalized the values obtained for the pixels since the values resulting from Squared Difference are a lot larger than the values obtained from Blob Detection.

For building the model, mean and covariance for pixels marked as ferritin and non-ferritin are calculated separately. Then, for each pixel, the distance to each of these two distributions is computed. The pixel is assigned to the category whose distribution is closer. Figure 3.15shows how pixels are split by the algorithm that I described.

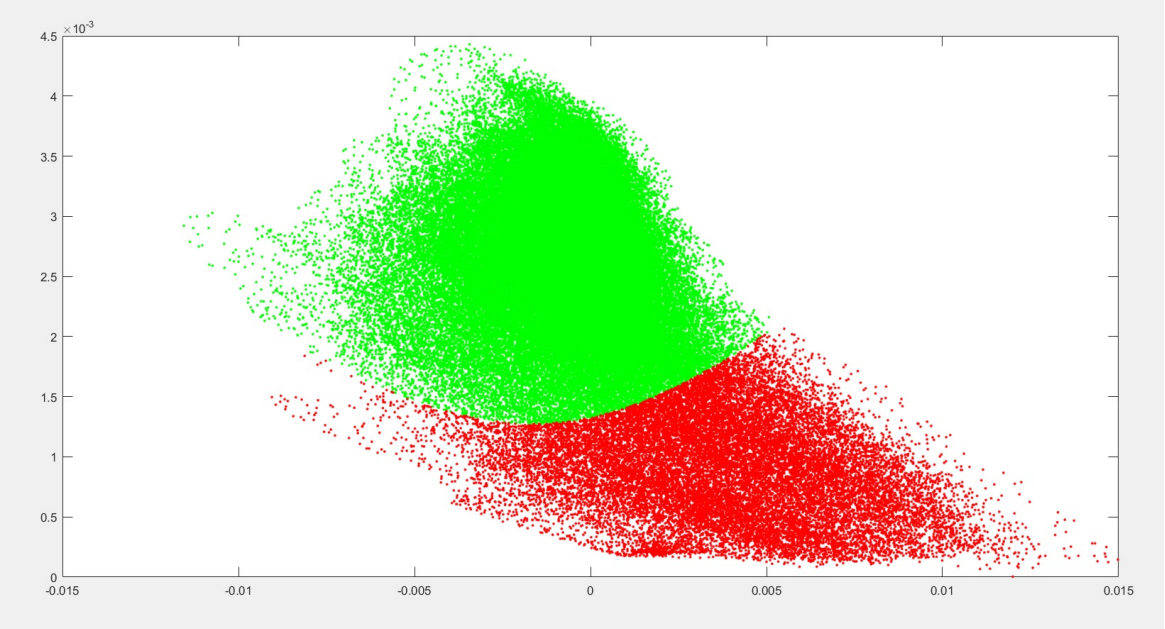


Fig. 3.15 Graph showing how Mahalanobis distance classification splits the patterns from Figure 3.14.

For this classification, I obtained the efficiency and I obtained sensitivity 0.9295 and specificity 0.9129 and Figure 3.16 also supports the fast that the detection is very efficient using Mahalanobis Distance.

The results appear to be very good since the sensitivity is a lot higher than when I considered the blob detection and template matching separately. The specificity is also very good with very few pixels outside the ferritin that are considered. And most of those pixels are in the tissue area so the specificity could be increased if I focus the detection of the interior of blood vessels.

In order to evaluate even better the performance of Mahalanobis distance, I also considered a patch from another image (Figure 3.17) for a test. By applying same technique as before, I calculated the Mahalanobis distance of the pixels towards the distribution of ferritin and non-ferritin pixels that was obtained for the train data and I applied the classification. In this case, it resulted sensitivity 0.9118 and specificity 0.9400.

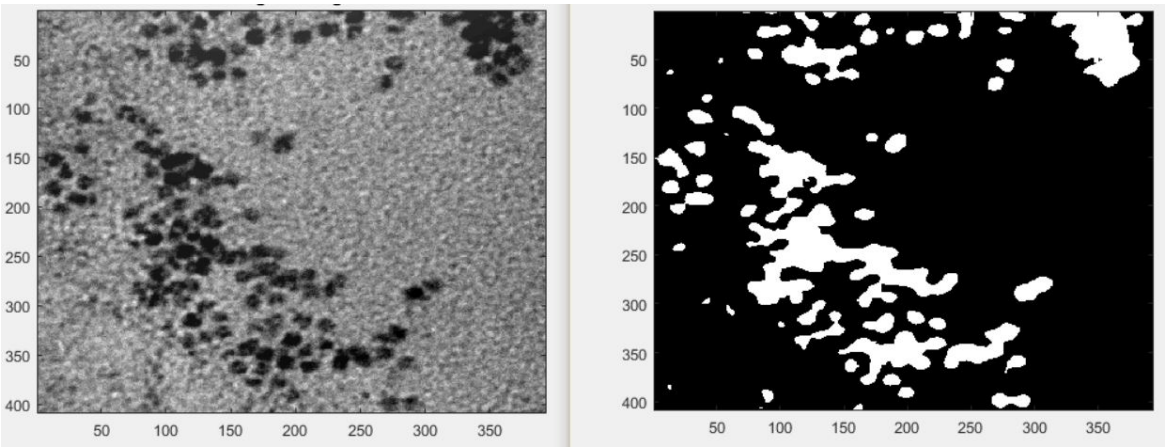


Fig. 3.16 Result for Mahalanobis distance classification on the patch whose pixels were used as train data.

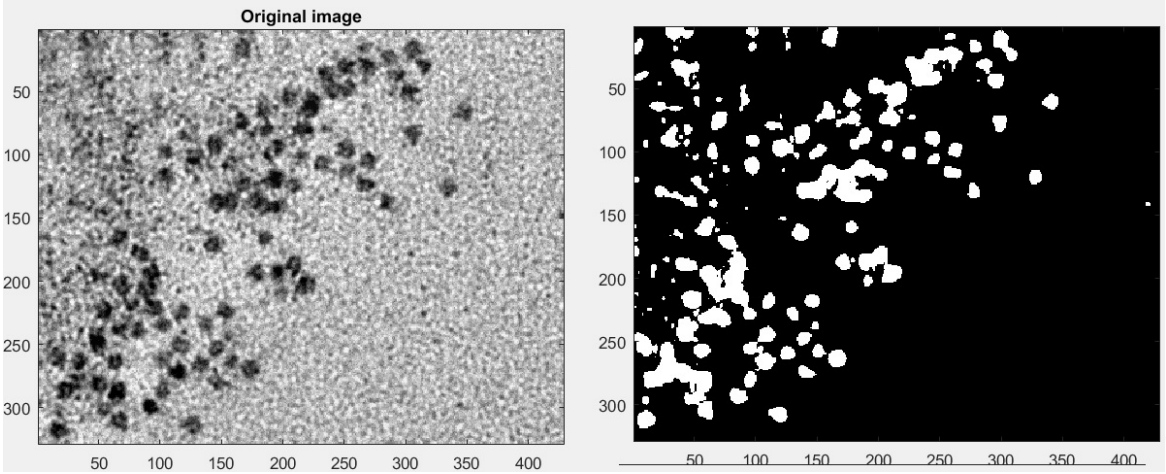


Fig. 3.17 The result for Mahalanobis distance classification on a patch that was used for validation of the model.

Once again, I obtained great performance from the sensitivity point of view but there are many pixels from tissue that result into false positives, so the specificity is lower, but this is quite expected since there is more tissue area in the test image.

As a general conclusion for this section, I would say that I obtained surprisingly good performance by using together the values of pixels from transformation with template matching and Laplacian. The Mahalanobis distance has proven to be a simple but efficient measure for implementing the calssification. However, in order to obtain a really efficient model, I had to consider several parameters for applying Template Matching and Blob Detection, annotate even more patches from several images, to try several learning techniques and then make experiments to calculate the accuracies until I found the solution that achieves the objective of the project.

3.3 Pixel classification for Ferritin detection

In this section I will present the approach and the results obtained from detection of Ferritin Molecules by binary pixel classification using the observations made in Section 3.2 that the values obtained by Laplacian of Gaussian and Template Matching can be used to make an effective classification model. This approach has the advantage that later, no matter how large is the image that is analyzed, the model can be used by applying the transformations, by making a dataset with the values obtained for each pixel, classify using the model and placing the predictions into the shape of the original image.

I started by cutting 30 patches from all images in the dataset with size 256x256 pixels that will be used for training and testing the efficiency of the classification. The patches together with the annotation that I made are shown in Appendix C. The first 20 patches contain ferritin molecules in different locations and with different levels of overlapping for obtaining a good generalization and the last 10 patches contain parts of the images without Ferritin molecules in case I need more samples for obtaining better specificity.

For these image patches, I decided to annotate only a few pixels in the center for each ferritin molecule in the images because, as Figure3.18 shows, since ferritin molecules have almost spherical shape, I can just dilate the annotation for the center of molecules until their surface is almost covered. On top of that, the values resulting from Template Matching and Blob Detection are higher in the center of the ferritin and the detection of all surface of the ferritin obtained some connected components that make molecules identification too inaccurate.

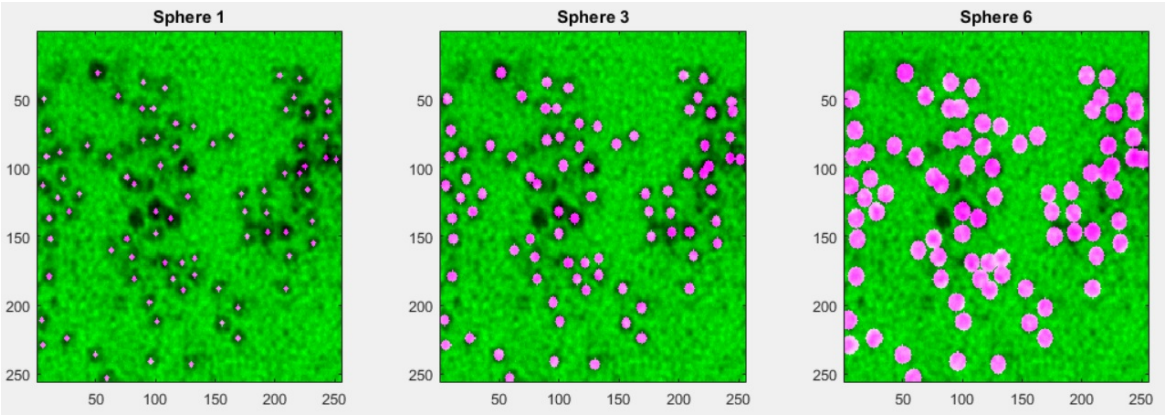


Fig. 3.18 Image fusion between patch 1 used for classification and annotation that was dilated with sphere structuring element of different sizes.

I noticed that just calculating accuracy, sensitivity and specificity for the annotated dataset is not enough to prove that the model works properly for the whole images so, for the purpose validating that the classification works, I also tested the classification on 4 different images that are shown in Appendix D and I selected zooming in areas that are significant to describe how the model will behave along the pipeline:

- A part from image 0004 that presents a big and compact cluster of ferritin molecules and this should tell if the result from my classification will enable easy identification of each particle.
- A part from image 0008 that presents smaller groups of ferritin molecules along the boundary of blood vessel and the fact that there is quite much tissue and also blood area in this zooming, it should tell how the model will behave in those areas
- A part from image 0019 that is darker and it will be useful to check if the classification works for the whole dataset
- A part from image 0021 that seems show a different texture of ferritin molecules and it will be useful to check how detection works in this case

In the next sections I will present different classifiers that were tested, with different inputs from the image transformations, I will show how they behaved with and without the different types of filters that were described in Chapter 2 .

3.3.1 Classification using Mahalanobis distance

I chose to start with building a model for Mahalanobis distance since I shown Section 3.2.4 that it is quite efficient even without many tweaks, it is fast to implement, it has short computation times and it made easier to compare different combinations of parameters and filters in order to achieve better ferritin molecules detection. The main goal is to find a distribution for ferritin and non-ferritin pixels that results in higher sensitivity and specificity when I test it with new pixel values from another image.

The parameters that I tested in different combinations are:

- Raw pixel values; it is possible that they will not have a good impact over the model but I still gave it a try.
- Values of pixels after applying Template Matching by Squared Difference with 2 different templates that are shown in Figure 3.19; one of them is a clear ferritin molecule while the other one is harder to distinguish. I considered that having two different examples of templates might increase the accuracy of detection.
- Values of pixel after convolving with Laplacian of Gaussian with scales values: 3, 5 and 7 that results in images to look like in Figure 3.2 to provide a better description for the various blob sizes during classification.

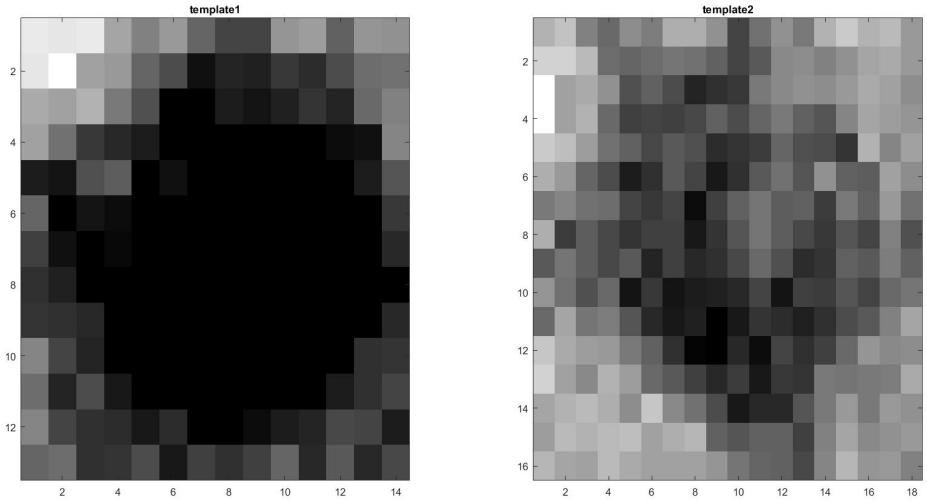


Fig. 3.19 Two different templates that I tried for Squared Difference.

Before making the classification as described in previous section, I normalized the parameters because Template Matching results in pixel values a lot higher than from Blob

Table 3.1 Performance obtained with different combinations of parameters by performing Mahalanobis Distance. TM = Template Matching, Log = Laplacian of Gaussian, Sens = Sensitivity, Spc = Specificity

Entry	Raw	Tm 1	Tm2	Log3	Log5	Log7	Sns train	Sns test	Spc train	Spc test
1		x		x	x	x	0.9200	0.9888	0.8805	0.8559
2	x	x		x	x	x	0.9200	0.9936	0.8760	0.4387
3		x	x	x	x	x	0.9082	0.5969	0.8875	0.9462
4				x	x	x	0.9248	0.9941	0.8709	0.7354
5		x		x	x		0.9047	0.9768	0.8969	0.8889
6		x			x	x	0.9098	0.9812	0.8916	0.8768

Detection and that will have a negative impact on the model. For the annotation, I applied dilatation with sphere of radius 3 because it provides quite good coverage for the ferritin molecule surface without considering too many unclear pixels.

For training I chose patches from 1 to 15, for the test I chose patches from 16 to 20 and for validation I choose the 4 larger images described previously.

In Table 3.1 I show the resulting sensitivity and specificity for using different combinations of the parameters that I specified and in Appendix E I displayed the results for using the models obtained on the images from Appendix D.

By analyzing the results, I will start mentioning that, for using the raw values for pixels, it results in having too low specificity, there are too many component wrongly classified as ferritin in the tissue and also inside the blood vessel. It also makes all the ferritin molecules to diffuse to each other and this will make too hard to identify each molecule in the image.

The other combinations of pixels have all provided good sensitivity but except for the experiment where I only considered the values from Blob Detection, there are quite many elements in blood and tissue that are wrongly considered to be positive in classification. However, the false positive components from the blood area are quite small and they can be removed by filtering the components with smaller number of pixels and I can also ignore the tissue area when I will apply statistical analysis.

On top of that, by not considering the values from Template Matching, the surface of the ferritin molecules seems to be degraded, sometimes split and that will provide inaccurate identification for the number of molecules in an area. I decided that the detection that will provide more accurate results at Spatial Statistics is the one obtained by considering the pixels values obtained after transformation with Template Matching using only the first template and Laplacian of Gaussian with scales 5 and 7 because it enabled more efficient identification for the center of each particle after removing the resulting connected components that were too small.

Table 3.2 Performance of Mahalanobis distance after filtering the images with different types of filters.

Filtering method	Sensitivity train	Sensitivity test	Specificity train	Specificity test
Median3x3	0.9156	0.9463	0.9037	0.9219
Wiener5x5	0.9151	0.9535	0.9226	0.9046
Wiene3x3	0.9197	0.9497	0.9048	0.9262
BM3D	0.8702	0.9443	0.8356	0.8533

Next, I will also present the influence of using filtering methods described in Chapter 2 on the accuracy of detection. I obtained the results shown in 3.2 and Appendix F displays the results for using the filters and Mahalanobis detector on the images from Appendix D

It seems like both: Wiener and Median filter remove some of the false positives detected without filtering which resulted in increasing the specificity of classification . The BM3D has bad effects on classification, making the specificity a lot lower. This happens because Wiener and Median filters reduce the grainy noise in the images while the BM3D has shown to degrade the texture of the particles.

I conclude this section stating that Mahalanobis Classification provides enough good results for the purpose of detection of ferritin molecules. The best combination of parameters for pixel classification is to use Template Matching with a template showing a clear ferritin molecule and Laplacian of Gaussian with scales 5 and 7. Median and Wiener filters both seem to have a good impact on the classification, reducing the specificity and removing some of the connected components that are wrongly obtained without filtering.

3.3.2 Pixel Classification using SVM and surrounding pixels

Since I noticed from Figure 3.14 that a linear separator is enough for making individual pixel classification and Mahalanobis Classifier that I described previously achieved that, I decided to try a more different approach for pixel classification inspired by **(author?)** [Dan C Ciresan]. The article describes the usage of Convolutional Neural Networks for obtaining pixel binary classification that is used at identifying the desired components in Electronic Microscopy.

The most important technique described in the article consists in using the values of each pixel together with the surrounding pixels for classification. So, for building the training set I considered the annotations from Appendix C and for each pixel marked in that annotation, I took a surrounding window with size 16x16 pixels which results in 256 parameters that were used for training. For each ferritin molecule, I annotated just around 5 pixels in the center of each particle so for each ferritin I considered about 5 samples for training. In order to also

have negative samples, I chose random samples from patches 20-30 from C based on the same procedure. I built a dataset with equal number of 16x16 patches that contain ferritin and patches that are not ferritin. Figure 3.20 shows examples of these patches.

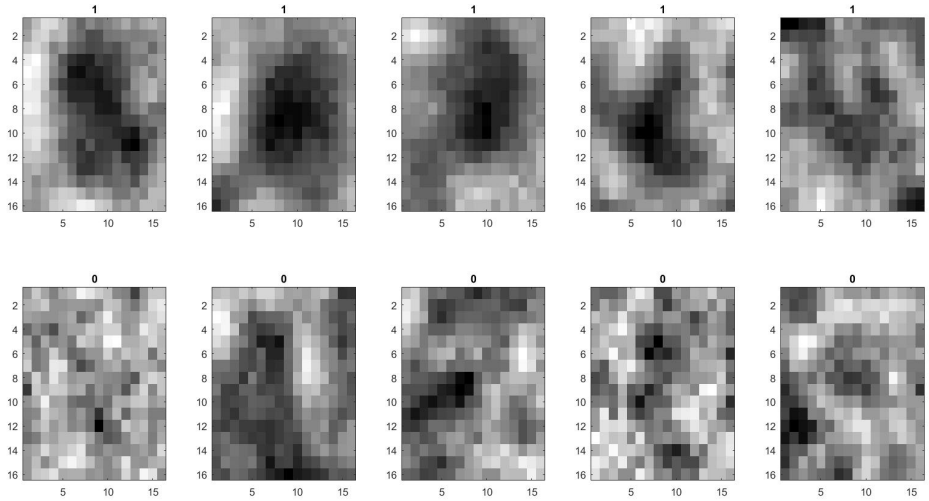


Fig. 3.20 Examples of entries for building the models: 1 means that patch contains a ferritin molecules and 0 means it doesn't

In the end, I obtained an equal number of patterns from each class of the binary classification and the data obtained was used for training SVM classifiers with different kernels . The accuracies resulting from training the models are:

- 97.2% for Linear SVM
- 98.1% for Quadratic SVM
- 98.6% for Cubic SVM
- 93.2% for Fine Gaussian
- 98.3% for Medium Gaussian
- 97.0% for Coarse Gaussian

The accuracy for this approach is very high, but as Appendix G shows how the Cubic SVM behaved on the validation set, I observed that it is very effective for detecting the ferritin molecules but there are quite many of them in the low density areas that are not identified. I think this happened because the training set was not well refined which resulted in overfitting, but the strategy described in **(author?)** [Dan C Ciresan] has proven to be very effective. I did

not have the time to make more experiments using this approach and Mahalanobis Distance Classification has obtained better results with shorter implementation and computation times.

3.4 Ferritin Detection Conclusions

I conclude this chapter stating that the ferritin molecules can be highlighted very efficiently using Blob Detection by convolving Laplacian of Gaussian and Template Matching with Sqared Difference. Each method has different drawbacks in detection of the particles, but used together, the values of pixels obtained from these transformations can be used to train very effective binary pixel classifiers for the detection.

When it comes to classifiers, Mahalanobis Distance provided very effective separation of the two classes of pixels and the resulting detection is good enough for measuring Ferritin Molecules Density. I did not manage to try many models, but I observed in Section 3.3.2 that, considering the raw values of pixels in patches with size 16x16 that are labeled if they are centered on a ferritin molecules or not, can be used to obtain effective detection of the particles by training SVM pixel classifiers. Unfortunately, the approach had the drawback of providing lower sensitivity on validation images but I think it might be more efficient than Mahalanobis distance with better selection of the training data.

However, Mahalanobis distance provided efficiency that was good enough for the purpose of the project, so I decided to use its results for measuring ferritin molecules density function.

Chapter 4

Vessel Boundary Detection

4.1 Overview

Making an automatic detection for the boundary of the blood vessels in the images from the dataset in Appendix A will be helpful for a better analysis of the ferritin molecules distribution by considering how close they are to the boundary, it will improve the molecules detection by removing the tissue area from the analysis and any success with this task will help for further research at Rigshospitalet.

However, this task was a lot more difficult than ferritin molecules detection because the contrast substance used for acquiring the images is not suited to make the boundaries easier to detect by the electrons of the microscope. The boundaries are very unclear, in some areas almost impossible to detect them with human eye, in some parts of the images the ferritin molecules diffuse into tissue making even harder to detect the boundary and in most parts they do not have a texture much more different than the tissue or the interior or the vessel.

In order to approach this problem, I started by cropping 20 small patches from the images containing various examples for the boundary and, as they are displayed in Appendix H, I annotated in two different ways: for the first annotation I marked as positive the tissue area and negative the area inside the blood vessel while in the second annotation I marked as positive just the pixels that I considered to be inside the boundary. By having two types of annotation, I had more flexibility in analyzing the boundaries and building,training and testing a model for boundary detection . For the purpose of a good generalization, the patches that were chosen contain boundaries with different shapes, angles and clarity.

In the next sections I will describe the algorithms that I used for analyzing and highlighting the boundary. In the second part I present the detection models that I tried together with the resulting accuracies and implications.

4.2 Boundary analysis and highlighting

4.2.1 Structure Tensor Experiment

In the first stage, I wanted to make a statistical analysis of the patches from Appendix H and check what useful information I could obtain. The issue is that, if I wanted to apply Principal Component Analysis, the patches had to be normalized first so the angle of the boundary in the patch is the same. Finding a method to normalize the angle of the boundaries in an image patch could also enable detection by using a template.

For obtaining the normalization, I decided to rely on the structure tensor theory and I mostly applied an algorithm inspired by Soo-Chang Pei [6] article. The article describes mainly how to use covariance matrix to obtain the values needed to translate and rotate images in order to normalize before comparing. For my task, I only needed to use the procedure for rotation normalization.

For implementing the algorithm described in Soo-Chang Pei [6], I chose the patch number 20 (Appendix H) from the dataset that I selected for this chapter, as the model since it has more straight boundary.

The purpose of the algorithm is to detect as accurately as possible the angle difference between the boundary in the model patch and the patch that has to be normalized. The first stage is to blur each image using a gaussian filter, then directional gradients on both directions (x and y) are calculated for each image. The next step is to calculate covariance and eigenvectors for the vectorized pixels in each image using the values computed by the gradients. The final step is to multiply the eigenvectors obtained from each image which results into a 2x2 matrix that, if transposed, is the rotation matrix that has to be used for normalizing an image patch. If the gradients detect the major angle in the patch correctly, the boundary should have the same angle as in the patch 20 that is the model.

I was not confident that this procedure will work since the ferritin molecules clusters that are close to boundary obtain higher gradient values, so I experimented this algorithm using the patches as they are but also the annotation tissue/blood because being a binary image, the computation of the rotation matrix should be very accurate and this could validate the algorithm described in Soo-Chang Pei [6].

So for using the boundary annotation as binary image to test the algorithm that was implemented, in Figure 4.1 appears that the patches are normalized pretty well, but it has issues if there is a change in angle of the boundary that is contained by the crop. This test reveals that for boundary detection, maybe smaller patches from the image should be analyzed at a time that do not contain curves.

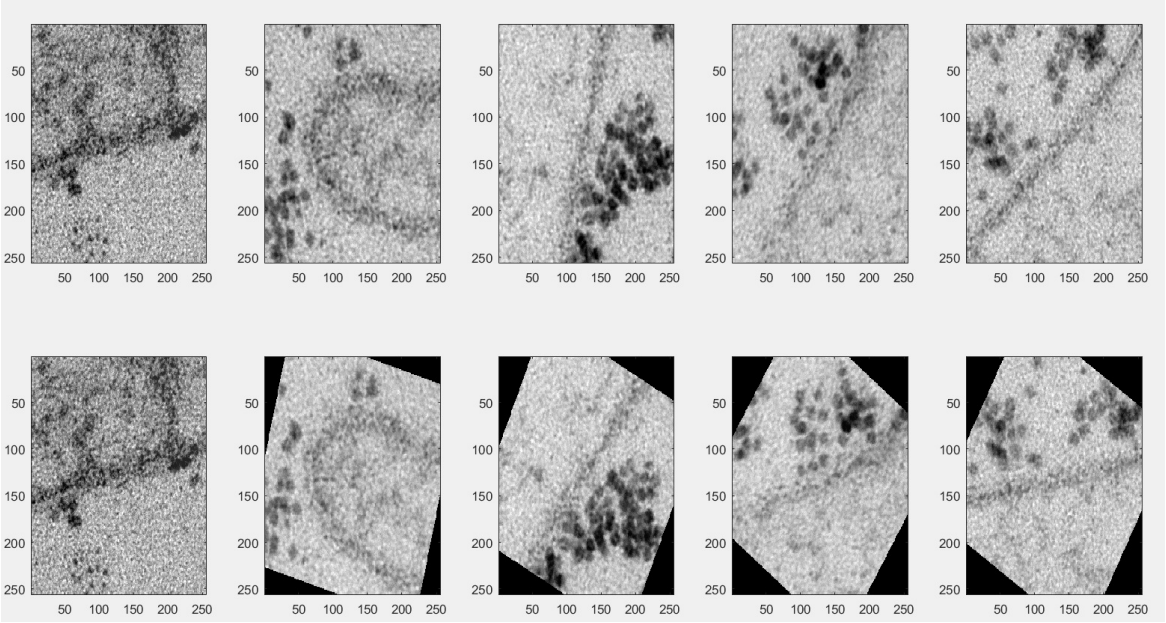


Fig. 4.1 Boundaries normalized using tensor theory applied on the blood/tissue annotation for the patches selected for this chapter(the first patch is the model). The resulting rotation angle is applied to the original patches for better visualization.

From the first test, I concluded that the algorithm described by Soo-Chang Pei [6] does not work properly if there is big difference between the shape of the objects in the images that have to be normalized, but I still tested it on the patches and I obtained the results from Figure 4.2. It is surprising to see that the 5th patch in the example is still properly normalized but for the others, but the results show that the gradient does not detect the boundary as having the higher impact on calculating the rotation angle as expected.

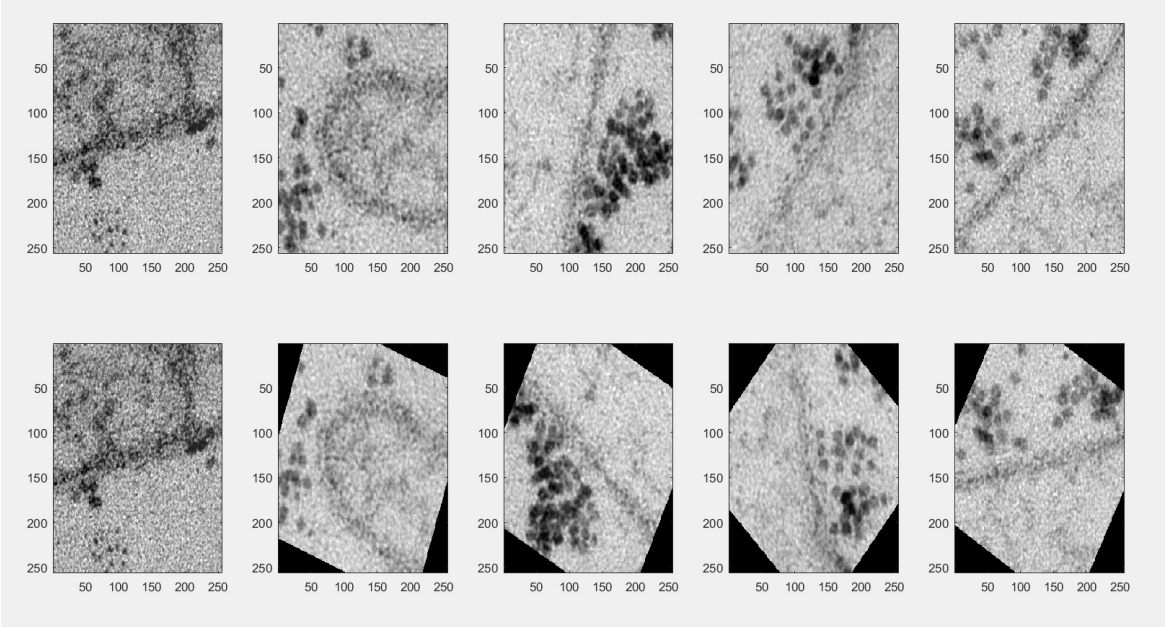


Fig. 4.2 I made experiments for several patchs and below there are examples of results obtained from normalization using the annotation(the first image is the model).

I concluded this experiment stating that normalizing the vessel boundaries' angle is hard to do automatically in the available dataset because they are very unclear and tensor theory does not manage to describe its characteristics properly. Due to these conclusions, I decided to try different approaches for analyzing the boundaries.

4.2.2 PCA for the 1d compression of boundaries

Because I considered that PCA analysis of the patches might provide useful information, I decided to select some of the patches having more straight boundary and I normalized them manually until I considered that they had the same angle of 90^o. In Figure 4.3 I displayed the patches before the normalization and Figure 4.4 shows the patches after the boundary is aligned.

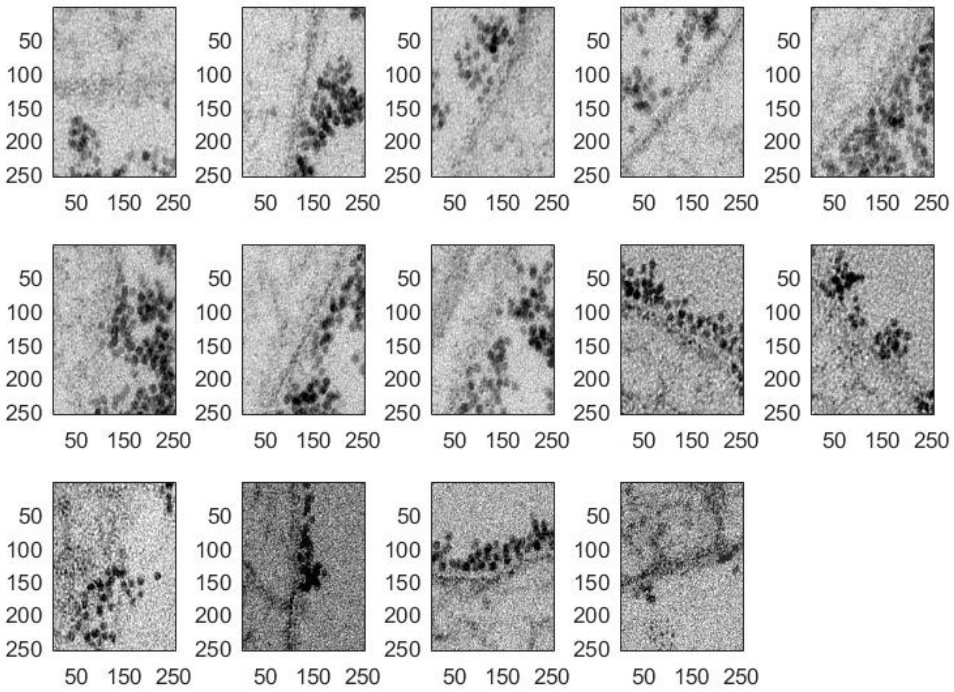


Fig. 4.3 Boundary patches before normalization.

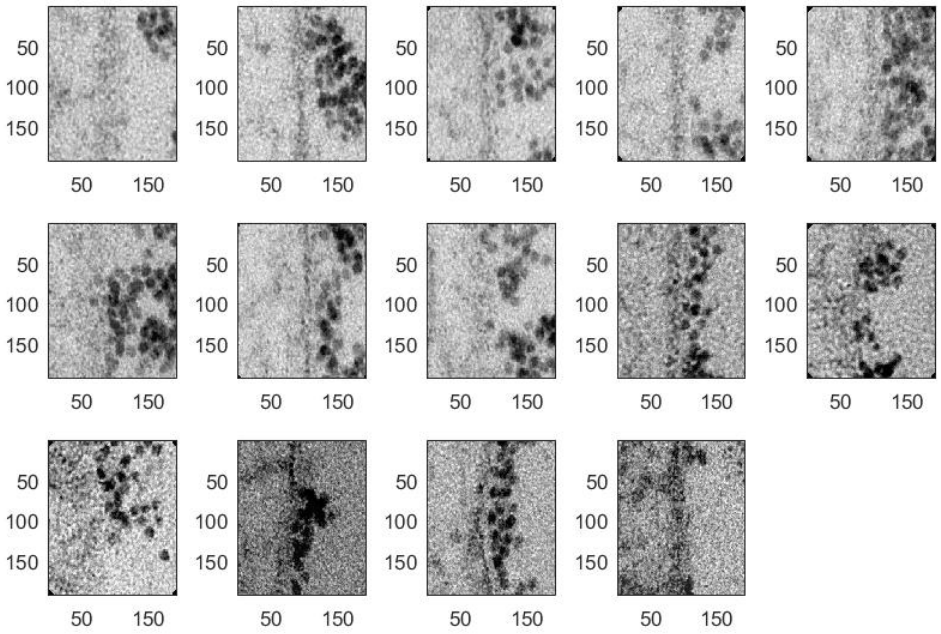


Fig. 4.4 Boundary patches after normalization.

Computing PCA for the whole 256x256 pixels patch required too much computation time so the next step I did was to sum the 2d matrices obtained over the columns so I obtained 1d compressions that are in Figure 4.5. After this, I also performed the PCA analysis for the 1d vectors that I obtained and Figure 4.6 presents the results that I obtained.

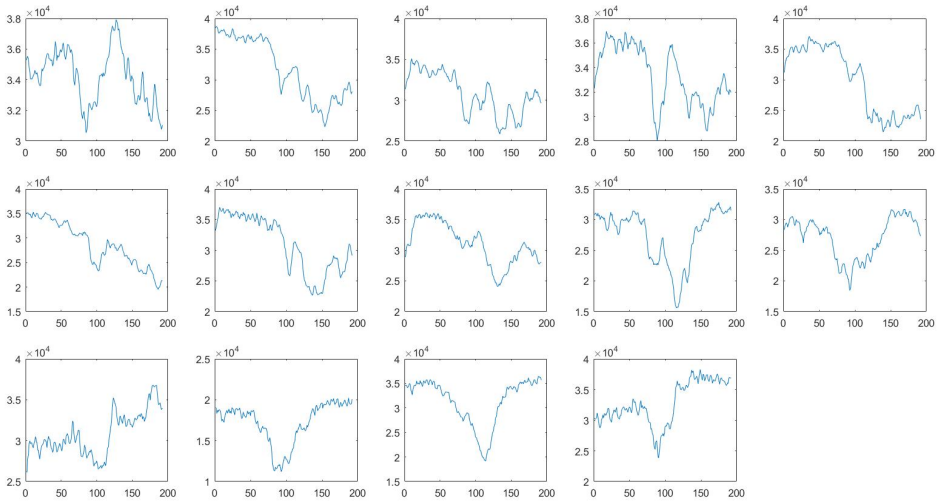


Fig. 4.5 1d compression of the patches.

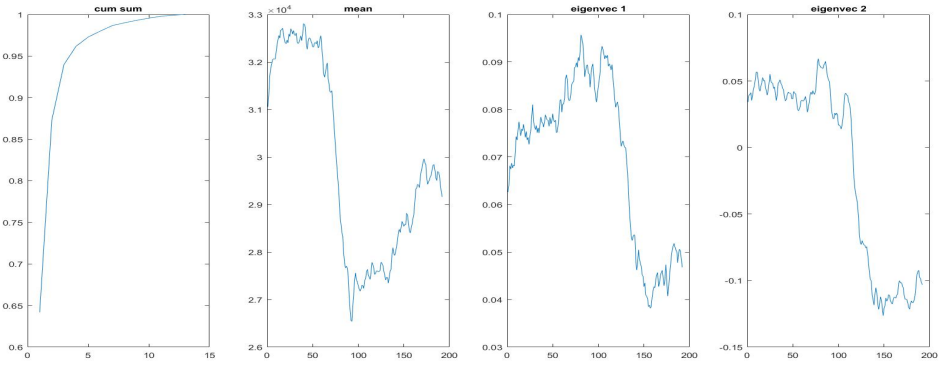


Fig. 4.6 PCA results for the 1d compression of patches: cummulative sum of the eigenvalues, the mean of the vectors and the first two eigenvectors.

From the analysis of the results. it seems like there is a tendency that there is a very low value for the sum in the middle of the vector where the boundary is located and then the values grow slowly due to the ferritin molecules that are on the right of it. The cumulative sum of the eigenvalues also show that the first 2 of them describe very well the variance within the patches.

I decided to use the observations and use the mean of the 1d extensions as a template and hoping to enable the detection of the boundary, I convolved a window with size 256x256 on the crop from Figure 4.7. At each step I computed Squared Difference of the pixels from the window with the mean of the 1d extensions rotated from 1 to 360 degrees (every 5 degrees). The sum of the Squared Differences obtained for each angle are assigned to the pixel in the middle of the windows. By testing this approach for Template Matching, I obtained the result shown in Figure 4.7.

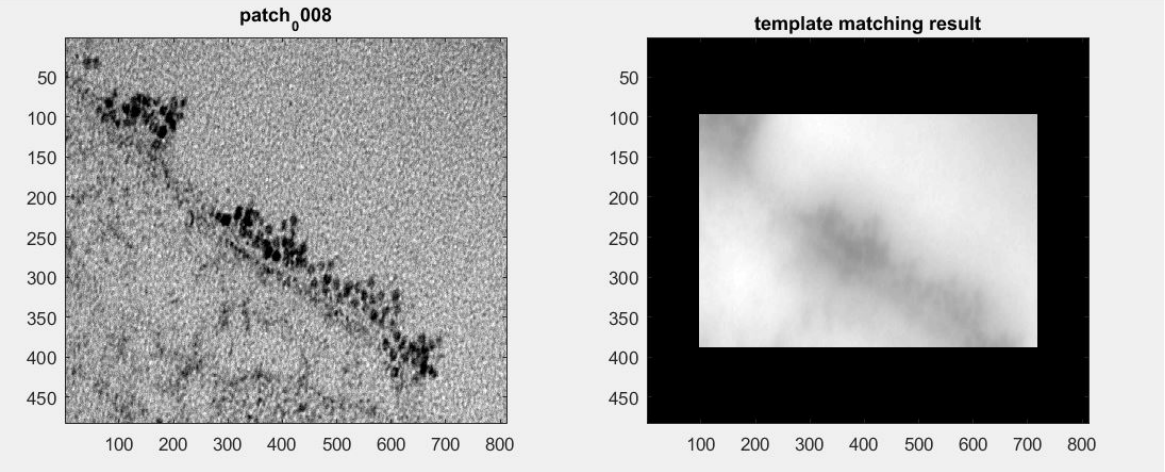


Fig. 4.7 Template matching with different rotations for 1d mean of the peatches that were analyzed. The resulting transformation has smaller size because I ignored the images that caused issues for convolution.

I did not manage to make more experiments by building a template of boundary based on the results from the analysis that was performed because of the long computation times needed to apply a template several times at different angles. I also concluded that, because of the ferritin molecules that have much stronger contrast, it is not very useful to view the characteristics of the boundary using PCA of the patches that were selected.

4.2.3 Hessian Matrix for boundary detection

In the search of methods to highlight the boundaries in images, I found that Hessian matrix (Wikipedia [8]) is commonly used for boundary detection and the transformations performed in order to obtain it might be useful as parameters for building binary classification model in the same way as I did for Ferritin detection. I will describe each step that I took during computation and I will show the results of computations.

The first step that I did was to blur the images using Gaussian filter and Figure 4.8 shows how some of the patches look after blurring. It appears that the grainy noise from the image

disappears and boundaries become highlighted in some samples. This means that values of pixels after this transformation might help me for boundary detection.

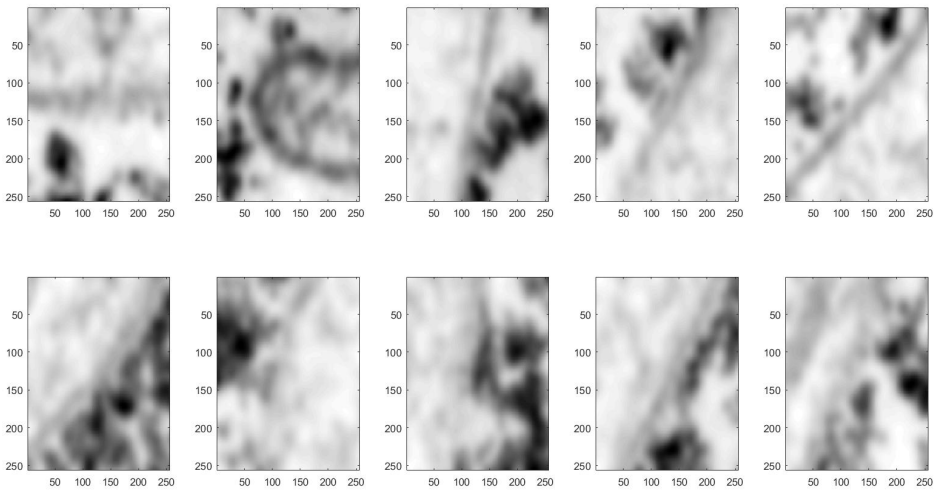


Fig. 4.8 Blurring image patches 1 to 10 from (Appendix8) with Gaussian filter of scale 8

The next step is to calculate the first and second order derivatives and this resulted in enabling to compute the Gradient Magnitude, a transformation that might be meaningful for boundary detection:

$$Gmag = (\frac{\partial I(x,y)}{\partial x})^2 + (\frac{\partial I(x,y)}{\partial y})^2 \tag{4.1}$$

For the same image patches that were blurred before, I obtained Figure 4.9 from computing the Gradient Magnitude.

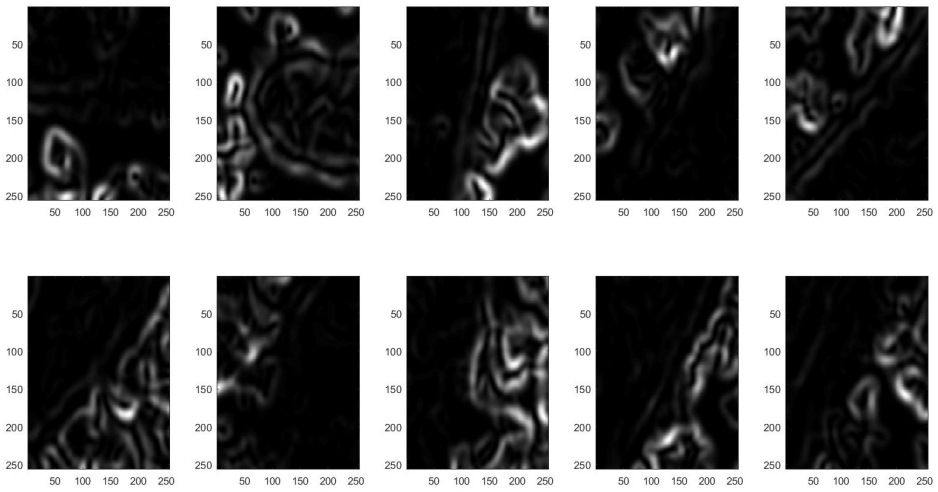


Fig. 4.9 Gradient magnitude resulting from patches 1 to 10 from Appendix H

For the patches where the boundary is clear, Gradient Magnitude shows two parallel lines and I think this transformation might also be useful for boundary detection. Unfortunately, there appears almost nothing in Gradient Magnitude for the patches where the boundary is quite unclear and the edges of the ferritin molecules clusters appear to result in higher values than the boundary of the vessel in general.

From the second order derivatives, I built the hessian matrix for each pixel:

$$H = \begin{bmatrix} \frac{\partial^2 I(x,y)}{\partial x^2} & \frac{\partial^2 I(x,y)}{\partial x \partial y} \\ \frac{\partial^2 I(x,y)}{\partial x \partial y} & \frac{\partial^2 I(x,y)}{\partial y^2} \end{bmatrix} \tag{4.2}$$

Then, for each of these Hessian matrices, I calculated the eigenvalues. Their corresponding eigenvectors should define the orientation of the objects around each pixel. So for each resulting pair of eigenvalues, I defined the larger eigenvalue as λ_1 and the smaller eigenvalue as λ_2. The resulting values for λ_1 on the patches analyzed previously are shown in 4.10.

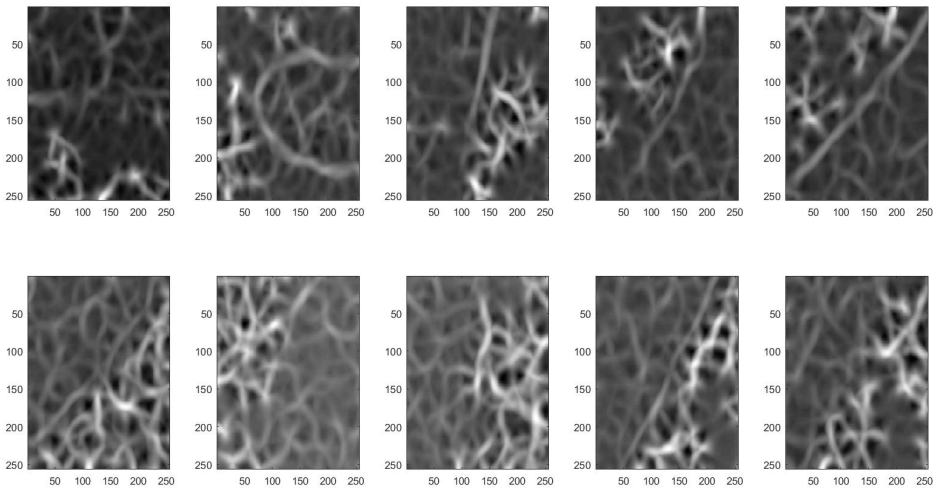


Fig. 4.10 λ_1 computed from the Hessian matrix of the patches.

As a conclusion, it appears that the eigenvalues obtained from Hessian matrix highlight the boundaries but also other elements in the images that might have a too strong impact on obtaining an accurate boundary detection. However, the graph showing together the values for lambda 1 and lambda 2 from several patches in Figure 4.11 suggests that the values in the middle might be more likely to be on the boundary but the two classes of pixels (boundary or not boundary) overlap quite much. However, used together, the values obtained for pixels on the boundary by blurring, calculating Gradient Magnitude, λ_1 and λ_2 might obtain efficient detection so in the next chapter I describe the results obtained based on these observations.

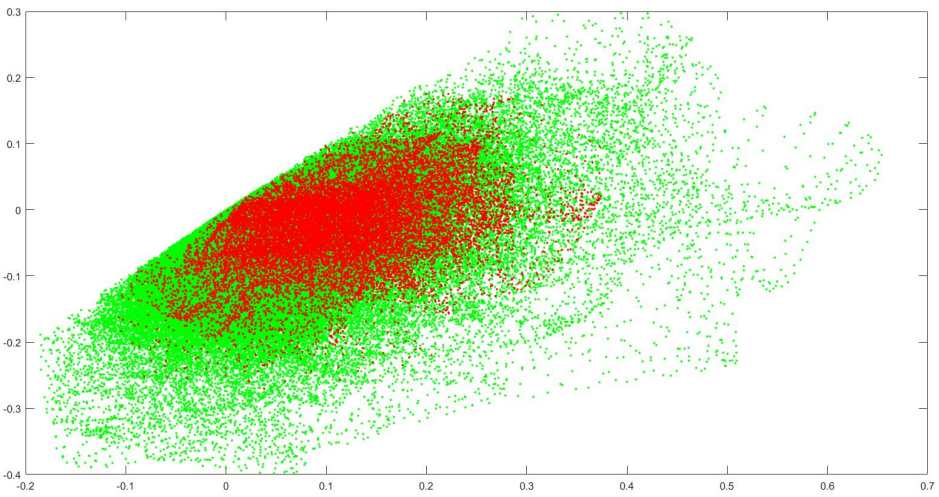


Fig. 4.11 λ_1 and λ_2 shown for each pixel in all 20 patches containing boundary. Red point means that the pixel is located on the boundary.

4.3 Boundary detection

In this section I present the results of two main strategies that I tried for boundary detection based on the observations made previously.

4.3.1 Classification using pixels values obtained from Hessian Matrix calculation

The approach on pixel classification with the purpose of boundary detection relies on using the values of pixels obtained from testing the Hessian Matrix as described in Section 4.2.3.

From the tests that I performed, I noticed that the best results are obtained from using the values obtained for each pixel in Gradient Magnitude, λ_1 and λ_2. From analyzing Figure 4.11 , I decided to use SVM for pixel binary classification into bounary or non-boundary and I obtained the following accuracies:

- 70.0% for Linear SVM
- 47.3% for Quadratic SVM
- 48.0% for Cubic SVM
- 73.7% for Fine Gaussian

- 73.6% for Medium Gaussian
- 72.7% for Coarse Gaussian

As expected, SVM with Gaussian kernel resulted in better performances but the accuracies are, unfortunately, quite low for a binary classification. Examples of using the models on larger parts of the images from the main dataset that are shown in Figure 4.12 and they prove that the approach does not work. I think it is because the boundaries in the images are too unclear and the usage of Hessian matrix does not manage to highlight them enough.

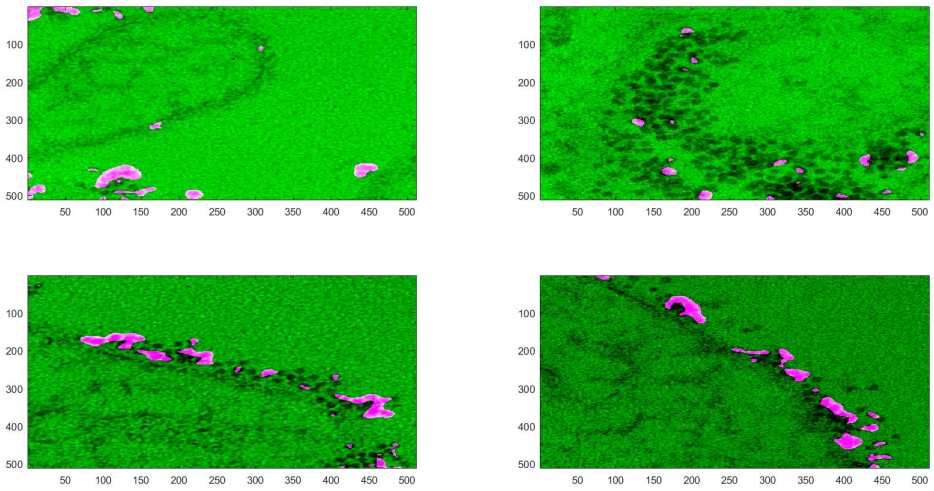


Fig. 4.12 Results for Fine Gaussian SVM model for binary pixel classification with the purpose of detecting Blood Vessels Boundaries. With pink are shown the pixels that are predicted as being on the boundary.

4.3.2 Boundary detection considering values of pixels in 32x32 windows

At Section 3.3.2 I shown that it is quite effective to make pixel classification by considering, for training, squared patches of small size centered on the elements that have to be detected, so I adapted that observation for boundary detection also.

I took a random set of 10000 squares of size 32x32 pixels from the patches in the dataset from H. Half of them are centered on boundary areas and the other half do not contain boundary area at all: some are from tissue area and the others are from the interior of the blood vessels. I decided on size 32x32 because I considered that this should contain enough

information for describing the surroundings of the boundary also. I used this data as input for training different SVM models and I obtained the following accuracies:

- 73.0% for Linear SVM
- 83.3% for Quadratic SVM
- 85.8% for Cubic SVM
- 72.9% for Fine Gaussian
- 84.4% for Medium Gaussian
- 76.1% for Coarse Gaussian

The accuracies of the models are quite low for the training set that was used, and Figure 4.13 shows that the Cubic SVM is quite effective to detecting boundary in areas where it is very clear. Unfortunately, it does not work for areas where the boundary appears to be diffused in tissue and blood.

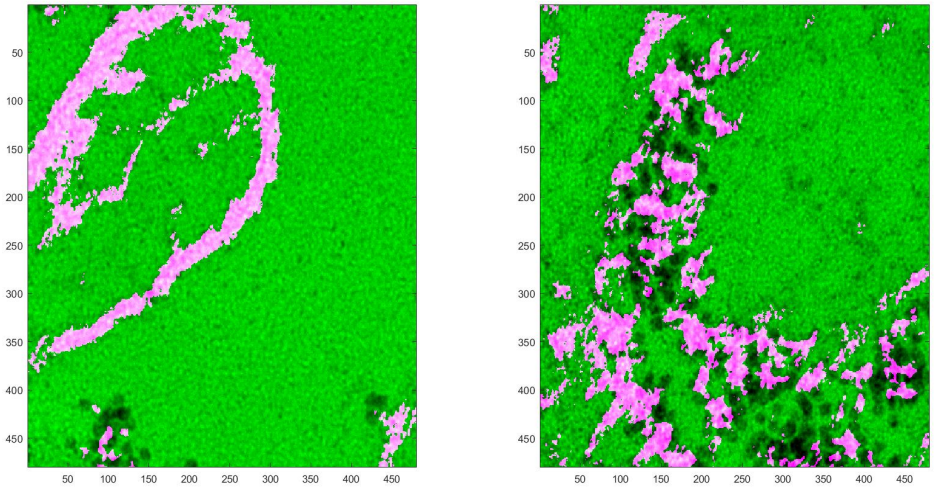


Fig. 4.13 Using Fine Gaussian SVM for binary pixel classification with the purpose of detecting Blood Vessels Boundaries. With pink are shown the pixels that are predicted as being on the boundary.

I also tried to use the same approach to classify into tissue or blood interior area relying on the second type of annotation for the patches from H but it resulted in very low accuracies (30-40%)

4.4 Conclusion

I conclude this chapter mentioning that, for boundary detection, I tried to highlight the blood vessels boundaries using a variant of Template Matching and Hessian matrix but it did not work very well because the boundaries are very unclear and the detection was negatively influenced by the ferritin molecules that are located very close to boundary.

Still, I tried to use the resulting transformations for pixel classification but unfortunately, the validation shown that I did not obtain good enough generalization based on the conclusions that I obtained from analyzing the boundary patches.

Using raw values of pixels in 32x32 window for building a classification model to detect the boundary seemed to work in areas where the boundary is very clear but it failed otherwise.

Perhaps using larger training sets and mixture of the two main approaches: transform the images and then apply pixel classification considering also the neighboring pixels might improve the results but due to limited available time, I could not make more experiments.

Chapter 5

Measuring Ferritin molecules intensity function

For the last stage of the project, I had to use the results obtained from Processing the Images in the main dataset to identify and describe the distribution of the ferritin molecules within the blood vessels with the purpose of showing if there is clustering within the molecules.

So I already proved that the best detection for the Ferritin molecules is obtained using a model for pixel classification based on calculating the Mahalanobis distance that was described at Section 3.3.1. The classification predicts which pixels from the images represent ferritin but I have to find a method to use the detection in order to identify each molecule, so the first section of this chapter will describe the method that I used.

After locating the ferritin molecules as accurately as possible, I had to apply Spatial Points Statistics methods that analyze their distribution inside the blood vessel. Since the boundary detection did not succeed with the methods that I used in Chapter 4, I had to manually annotate the parts of the image that represent the interior of blood vessels and Appendix I shows all the annotations that I performed.

The Second section of the chapter describes the functions that were computed for measuring the Ferritin molecules homogeneity and how to interpret the results. The third section concludes the chapter with discussion on the results of the statistical analysis.

5.1 Ferritin Molecules Identification

Because the ferritin molecules have almost spherical shape, I considered that it is enough to assign for each of them a pixel as close as possible to the center of the particle. Using

the coordinates of the pixel that was assigned, is enough to build a pattern describing the location of each ferritin molecule in the blood vessel.

So, for using Mahalanobis distance detection, I obtained results like in Figure 5.1. In order to process the result, I started by making extracting a list with the pixels for each connected component in the detection. By removing connected components with fewer than 50 pixels, most of the false positives were removed.

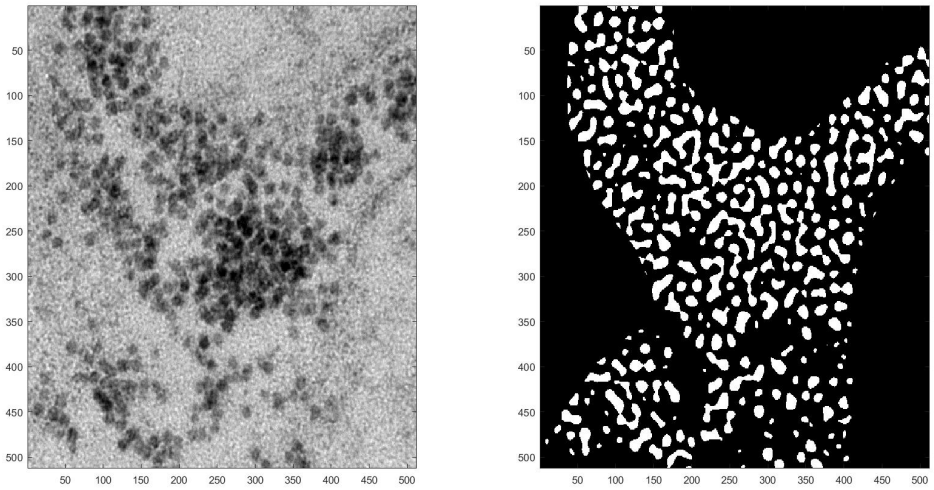


Fig. 5.1 Example of Ferritin Molecule detection using Mahalanobis distance.

In the next stage, I split each connected component in a number of groups of pixels equal with the number of pixels of the connected component divided by 180 rounded towards positive infinity. In order to get an even distribution of these pixels into groups, I used k-means clustering and considered the centers of resulting clusters to be the center of each ferritin molecule.

By applying the technique described to the detection resulting from Figure 5.1 I obtained the resulting molecules centers from Figure 5.2

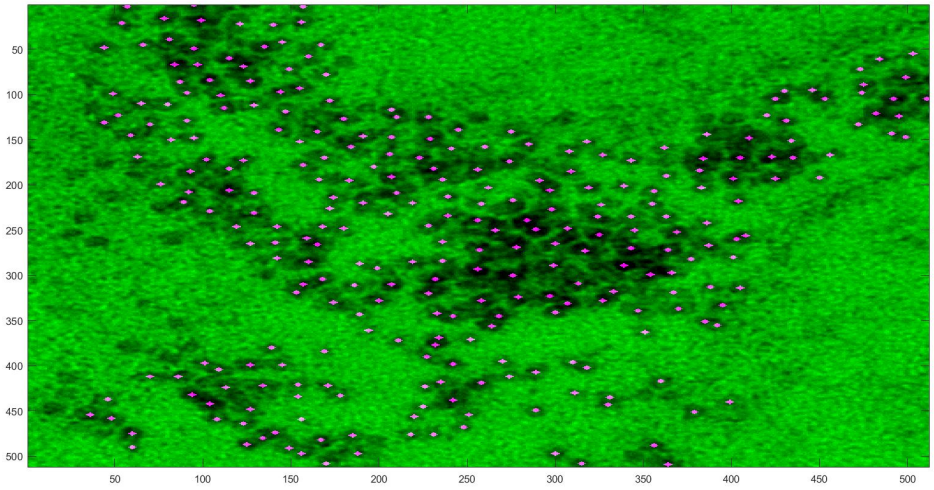


Fig. 5.2 Ferritin molecules centers as they were identified by dividing the connected components from the detection. The centers is represented with sa filled sphere for improved visibility.

I did not perform any numerical analysis on the result because, by judging the visualization of the results for this technique, I think it is very effective at identifying each ferritin molecule and it satisfies the requirements needed for the statistical analysis that I performed later. To sustain the conclusion, the results for the procedure on all the images in the dataset as they are illustrated in Appendix J, also prove that the data needed for statistical analysis is properly recognized.

Perhaps more complex algorithms and optimization of the parameters could provide more accurate results, but it was not needed for the objective of the project and it would have consumed too much time. On top of that, for human eye it is very hard to identify exactly the center for each ferritin molecule and split accurately the clusters where they overlap, so perfect validation for any algorithm applied to available dataset is almost impossible.

5.2 Spatial Point Statistics

5.2.1 Methods used for Measuring Ferritin Molecules intensity

Now that I have the detections for the centers of Ferritin molecules in Appendix J and the annotated boundaries of the blood vessels in Appendix I for all images in the dataset, I can apply Spatial Point Statistics in order to measure Ferritin molecules intensity. In order to do this, I used the R package *spatstat* for analysing spatial point patterns and the theory

presented in Baddeley [1]. The R code that I used was written by Jon Sporring on the work he did before I started working on the dataset.

The purpose for this analysis is to determine if the ferritin molecules form cluster patterns or if they are just having a regular distribution.A homogeneous set of points in the plane is a set that is distributed such that approximately the same number of points occurs in any circular region of a given area. A set of points that lacks homogeneity is spatially clustered.

Ripley's K function represents descriptive statistics for detecting deviations in spatial homogeneity . The K function is defined as:

$$\widehat{K}(t) = \lambda^{-1} \sum_{i \neq j} I(d_{ij} < t)/n \tag{5.1}$$

Where d_{ij} is the Euclidean distance between the i^{th} and the j^{th} points in the set containing n points, t is the search radus and λ is the average density of points. I is the indicator function (1 if its operand is true, 0 otherwise) Wikipedia [10].

For the available dataset, I chose to use an extension of this function, as described by Baddeley [1], named inhomogeneous K-function that relies on fitting a linear-exponential model for each of the images in the dataset. For coomputing this function, it is needed to calculate for each image the true intensity function $\lambda(u)$ that accounts for each point the dependency on the distance to the border of the blood vessels. The reason for this is that, as noticed in the images from the dataset, the ferritin molecules are in general grouped close to the vessel's walls, so I want to check if it is not just because of the distance to blood vessels that there are not many molecules in some parts of the vessels. Figure 5.3 shows an example for computing the true intensity function for one image in the dataset.

One important thing to notice is that the point patterns for each image is the detection for the Ferritin Molecules that are only inside the blood vessels, as annotated in I, but they appear rotated in the resulting graphics because there is difference between how the coordinates are considered in Matlab and R but this does not have any impact on the results of the statistical analysis.

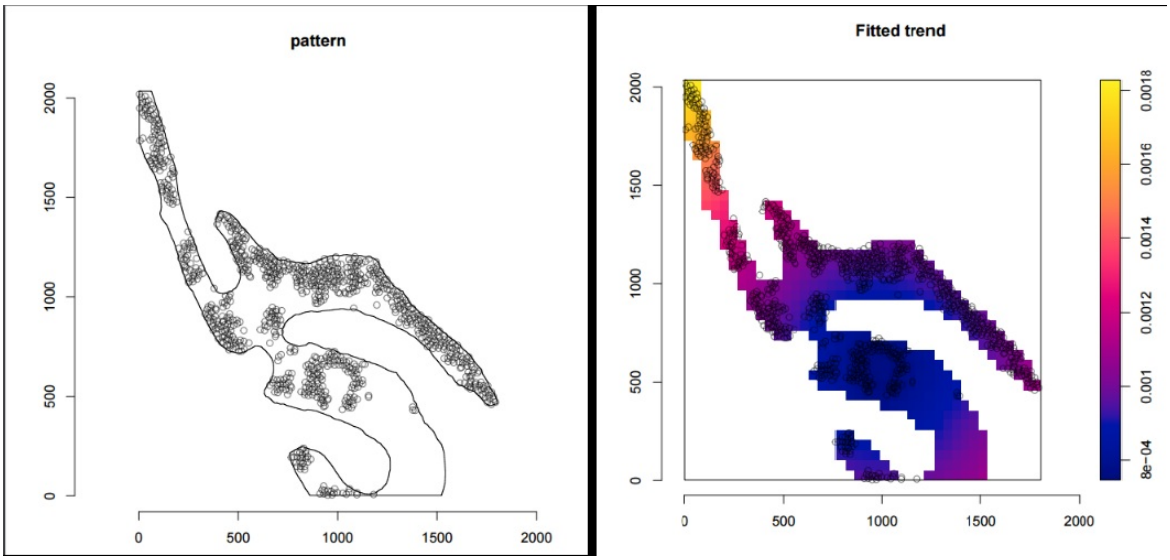


Fig. 5.3 To the right is the point pattern associated to image 0002 that is considered for statistics and to the right is the true intensity function that is used for K-function.

The inhomogeneous K-function is defined as:

$$K_{inhom}(r) = E[\sum_{x_j \varepsilon X} \frac{1}{\lambda(x_j)} \mathbf{1}\{0 < ||u - x_j|| \leq r\} \,|u \varepsilon \mathbf{X}] \tag{5.2}$$

where r is the radius that is considered around each point, u is the current point and x_j is any other point from the pattern towards which the distance is calculated.

As example, in Figure 5.4 is the result for calculating the inhomogenous K-function for image 0002. The plot is shown together with inhomogenous Poisson process with intensity function $\lambda(u)$, whose inhomgeneous K function is:

$$K_{inhom,pois}(r) = \pi r^2 \tag{5.3}$$

and it is represented by the red curved line in the graph. If the inhomgeneous K function for the points in the pattern that is analyzed results to be higher than πr^2, then it suggest that we have clustering. Otherwise, it is just a regular pattern.

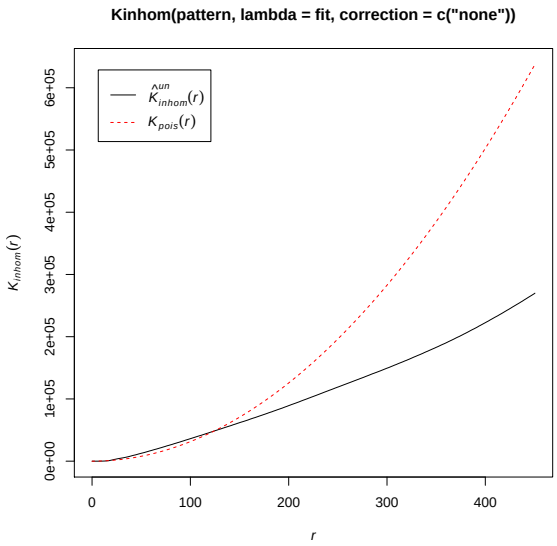


Fig. 5.4 The result for using the inhomogeneous K-function of the points pattern from image 0002 drawn with black line. The red line is the K function for inhomogeneous Poisson process for comparison.

L-function is a commonly used transformation of K and is defined as:

$$L(r) = \sqrt{\frac{K(r))}{\pi}} \tag{5.4}$$

which transforms the Poisson K function into a straight line, $L_{pois}(r) = r$ and as noticed in Figure 5.5 it makes the visual assessment of the graph much easier. Just like before, higher values than the L function for Poisson process suggests clustering, otherwise there is just regular points patter.

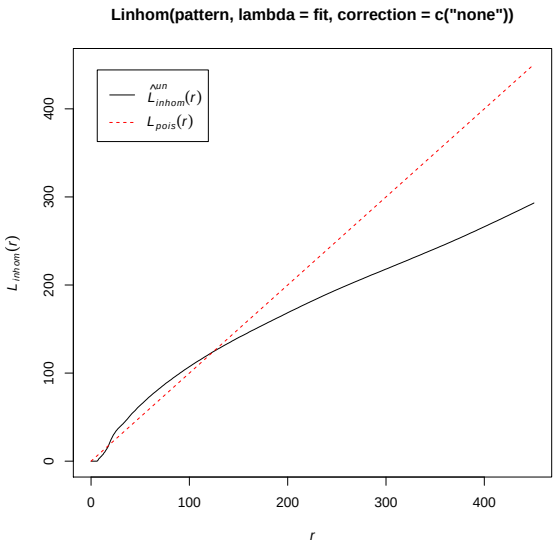


Fig. 5.5 The black line respresents L function for the points patter corresponding to image 0002. The red line is the L function for Poisson process.

Another summary function is the pair correlation function:

$$g_{inhom}(r) = \frac{K'_{inhom}(r)}{2\pi r} \tag{5.5}$$

where $K'_{inhom}(r)$ is the derivative of K_{inhom}.The function describes the probability of observing a pair of points at certain locations separated by a distance r, divided by the corresponding probability for a Poisson processes of the same intensity. The value $g(r) = 1$ corresponds to complete randomness, $g(r) > 1$ suggests clustering or attraction at distance r, while values $g(r) < 1$ suggest inhibition or regularity. Figure 5.6 is an example for result of using pair correlation function on a pattern of points from the dataset.

So I used these three summary functions to describe the Ferritin molecules intensity function and in Appendix K I displayed, for each image in the dataset, the corresponding pattern that is analyzed, the true intensity function, the inhomogenous K-function, the L function and the pair correlation function g.

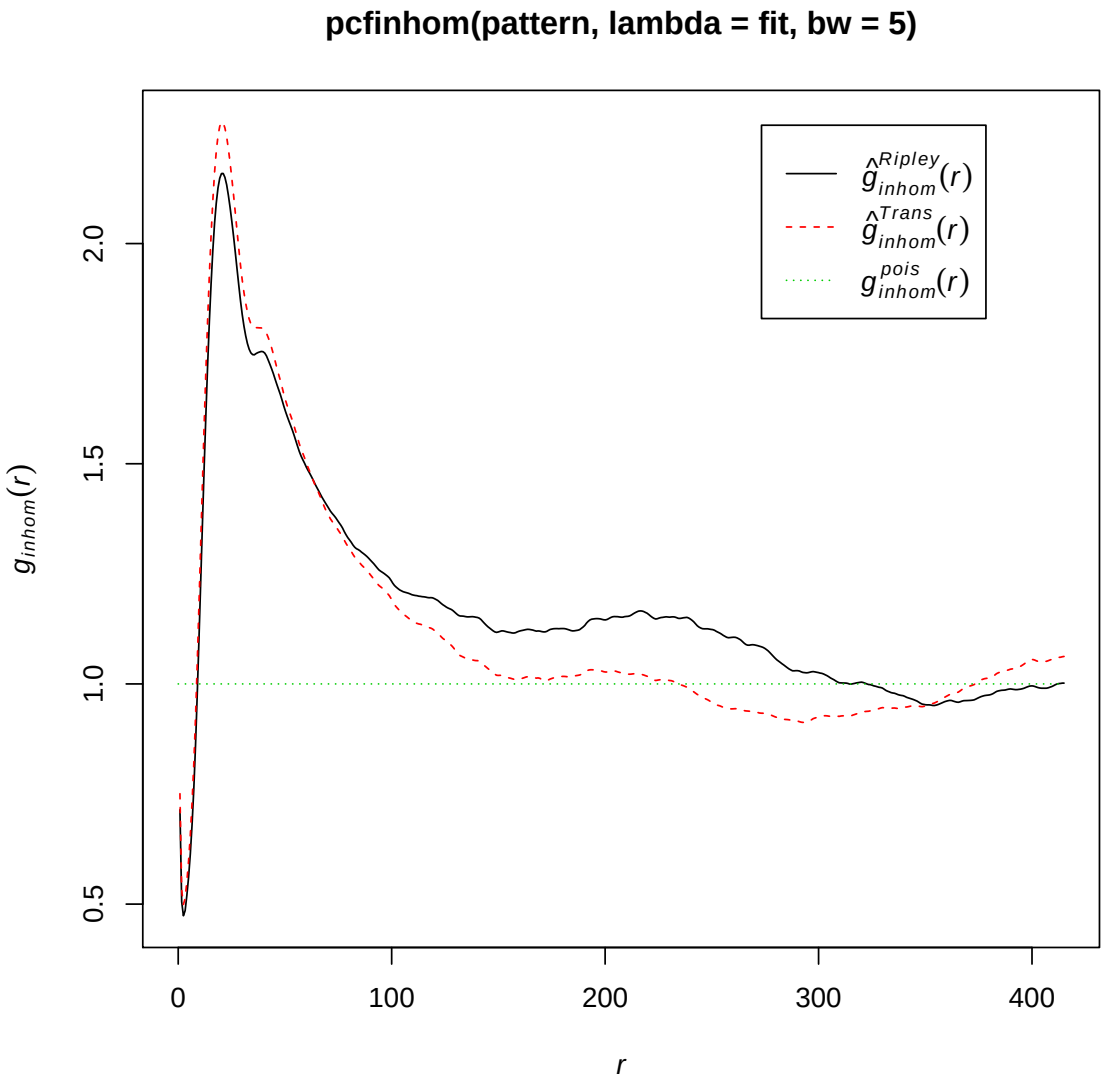


Fig. 5.6 The black line respresents g function for the points patter corresponding to image 0002. The red line is the translation corrected estimate for g and the green line is the g function for Possion process.

5.2.2 Analysis of the results

As I already mentioned, in Appendix K there are all results for analyzing statistically the point patterns corresponding to the Ferritin molecules location in the available dataset. The patterns appear rotated in this appendix than they are in the original image because I used R packages for the analysis which interpreted the coordinates differently than Matlab does. But this does not have any impact on the molecules intensity function.

I noticed from the images in the main dataset (Appendix1) are in two categories: the first five of them show thin blood vessels where the Ferritin is more uniformly distributed in large number on all the blood area. But the last 5 images show parts of thicker vessels that have most of their ferritin molecules located very close to boundary and this aspect is also revealed by the intensity function shown for each image. The first five images have more even distribution of the true intensity while the last five images show high molecules density close to the boundary of the vessel and very low value otherwise.

The plots of the inhomogenous K-function, the L function and the pair correlation function g show that there is clearly clustering in most of the images from the dataset. But in the images that show thin blood vessels with very large number of ferritin molecules evenly spread in the vessel, like in image 0005, it results in no clustering. However, in the images representing section from thicker vessels, most of the values of functions K, L and g are above the Poisson distribution's plot. So even if I computed the intensity function depending on the distance to the vessel's boundary, the clustering is still proved by the analysis that I performed.

I would conclude the analysis saying that the clustering is clearly proved by the Spatial Point Statistics that I computed for the last 5 images in the dataset, but there is also some clustering in the images showing thin blood vessels because they have some regions that have very high density of ferritin molecules but they also present parts with low density of ferritin, and the molecules in those regions are still clustered. I think that areas with very high density of ferritin molecules are perhaps located just in some specific areas on the blood vessels and maybe having a scan of a large system of veins can show, by using the intensity function, that there are just some larger clusters located in some parts on the vessels.

Chapter 6

Conclusions

So, with the work I did on my project I managed to find efficient algorithms for automatically detecting the Ferritin molecules in the images that were provided and I applied Spatial Point Statistics methods to prove graphically that there is clustering within the Ferritin molecules in the blood vessels. I also managed to find methods to improve the detection accuracy by filtering the noise but the detection of the Blood Vessels boundary was not successful despite trying several techniques for it.

Isolating and analyzing the noise signal produced by the TEM imaging did not achieve any result but Bias Correction followed by Histogram Equalization and Median or Wiener filters improved slightly the visibility of the particles and had a positive influence on the efficiency of the detection algorithms.

For the Ferritin Molecules' detection, template Matching with Squared Difference and Laplacian of Gaussian have proven to be effective transformations to highlight the particles. The results of these transformations allowed very effective detection using a simple pixel classification algorithm based on Mahalanobis Distance. Mostly for experimental reasons, I tried to make pixel classification that uses as input data squares with size 16x16 and predicts if there is a ferritin molecule in the center of the square or not. SVM with nonlinear kernels provided very good accuracy but the technique requires more tuning for the structure of the models and selection of training data since to avoid overfitting. As a result, I decided to use the Mahalanobis distance model whose results were good enough for the purpose of the project.

Unfortunately, the Blood Vessels Boundary analysis and detection did not achieve the results that I was hoping for. The low quality for their texture did not allow to obtain much information from statistical analysis of their shape in order to construct a template for highlighting the boundaries. I computed Gradient Magnitude, Hessian Matrix, I considered the pixels neighborhood but I did not manage to find a set of parameters for building an

effective detection model. The main reason is that each of the transformations that I applied obtained stronger focus on other particles, especially ferritin molecules clusters that are located very close to the boundary. Perhaps a better choice of training data and tuning of the classification methods could provide better results. However, the results on this objective meant that I had to manually annotate the boundaries in each of the images in the dataset before applying statistical analysis.

Finally, the Spatial Point Statistics analysis that I performed on the distribution of the detected ferritin in the blood vessels has proven graphically that there is strong clustering for the thicker blood vessels, but also clustering in lower density areas for the images showing thinner blood vessels. Perhaps I could get a better view of the molecules intensity function if I could have scan of a larger structure of blood vessels, because it seemed to be wide and dense clusters of ferritin molecules in some parts of thinner blood vessels.

Personally, for me, the work on project was very useful to improve the skills on applying the knowledge on Image Processing and Machine Learning that I acquired during the studies. The most challenging part was that I had to choose on my own examples for the training datasets used for the modeling different algorithms that I wanted to analyze. And the interesting observation is that just calculating the accuracy, sensitivity and specificity of the classification was not enough to prove its performance until using it on a large image and visualizing the results.

The study that I performed managed to conclude that there is clustering within the ferritin molecules in the blood vessels located in brain, answering the issue that was stated by Casper Hempel. This information will certainly be useful for the research project that he conducts, but I am also sure that the other observations that I described in the thesis can be useful for future work on the same type of topics.

References

[1] Baddeley, A. (2010). *"Analysing spatial point patterns in R"*. "CSIRO and University of Western Australia".

[Dan C Ciresan] Dan C Ciresan, Alessandro Giusti, L. M. G. J. S. Deep neural networks segment neuronal membranes in electron microscopy images.

[3] Himmat S. Kushwaha, Sanju Tanwar, K. S. R. (2012). De-noising filters for tem (transmission electron microscopy) image of nanomaterials. *2012 Second International Conference on Advanced Computing & Communication Technologies*, pages 276–281.

[4] Kostadin Dabov, Alessandro Foi, V. K. and Karen Egiazarian, Senior Member, I. (2007). Image denoising by sparse 3d transform-domain collaborative filtering. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 16(8):1–16.

[5] Naresh Marturi, Sounkalo Dembele, N. P. (2014). Scanning electron microscope image signal-to-noise ratio monitoring for micro-nanomanipulation. *Automatic Control and Micro Mechatronic Systems (AS2M), Institute FEMTO-ST, Universite´ de Franche-Comte´/ CNRS/ENSMM/UTBM, Besanc¸on, France*, 36:419–429.

[6] Soo-Chang Pei, C.-N. L. (1995). Image normalization for pattern recognition. *Image and Vision Computing*, 13:711–723.

[7] Wikipedia. Blob detection.

[8] Wikipedia. essian matrix. https://en.wikipedia.org/wiki/Hessian_matrix.

[9] Wikipedia. Mahalanobis distance. https://en.wikipedia.org/wiki/Mahalanobis_distance.

[10] Wikipedia. Spatial descriptive statistics. https://en.wikipedia.org/wiki/Spatial_descriptive_statistics.

[11] Wikipedia. Template matching. https://en.wikipedia.org/wiki/Template_matching.

Appendix A

Main dataset

This appendix contains all images that were used for the research in the project, as they were provided by Casper Hempel from Rigshospitalet.

There are 10 images in total showing blood vessels from brain tissue.

Sometimes, in the main documentation, I used the last 4 digits of the file name to refer to a specific image.

Each image has size 2048x2048 pixels and they came in single channel color.

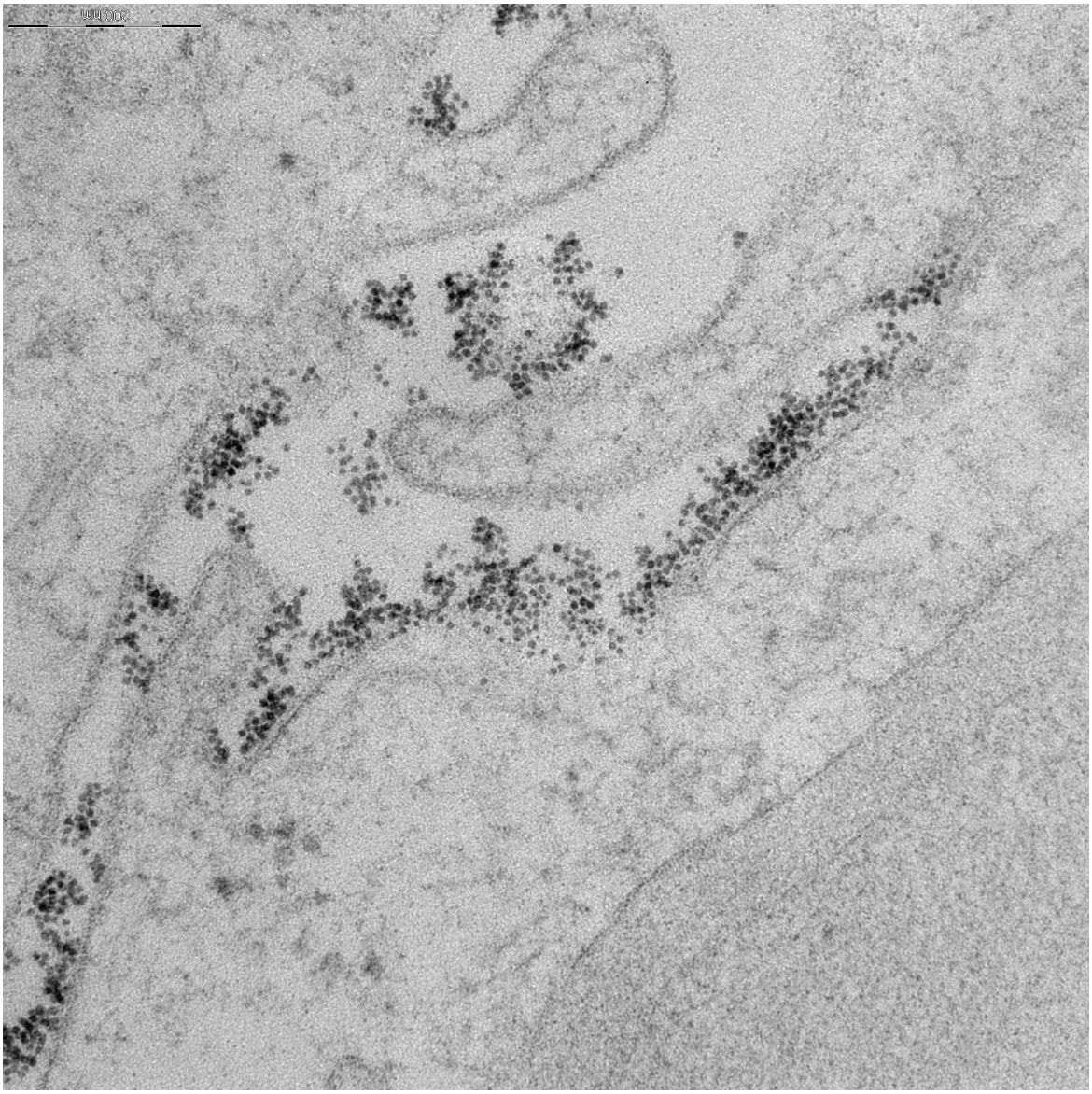


Fig. A.1 File name *Brain ferritin HPF 0002*

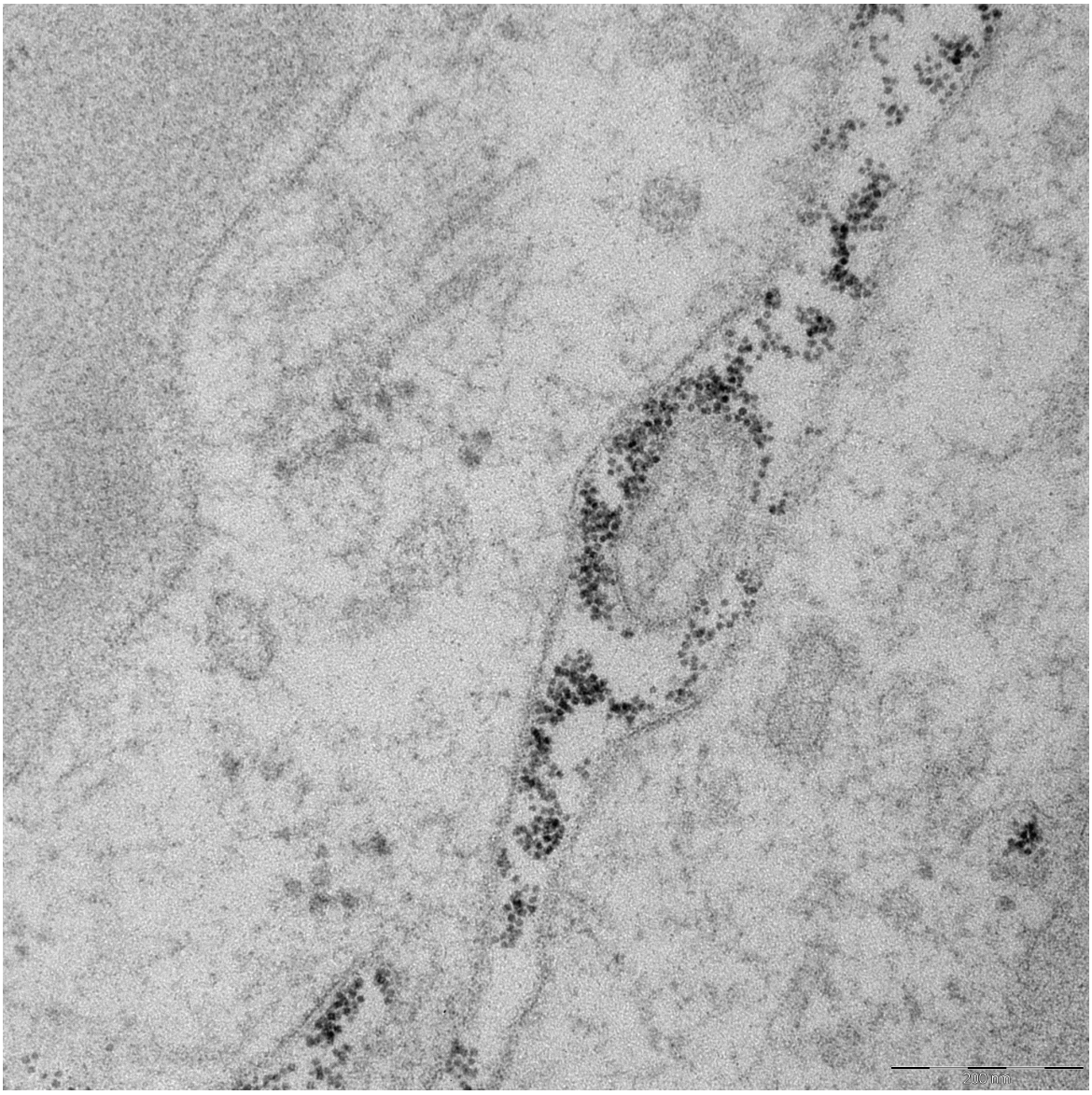


Fig. A.2 File name *Brain ferritin HPF 0003*

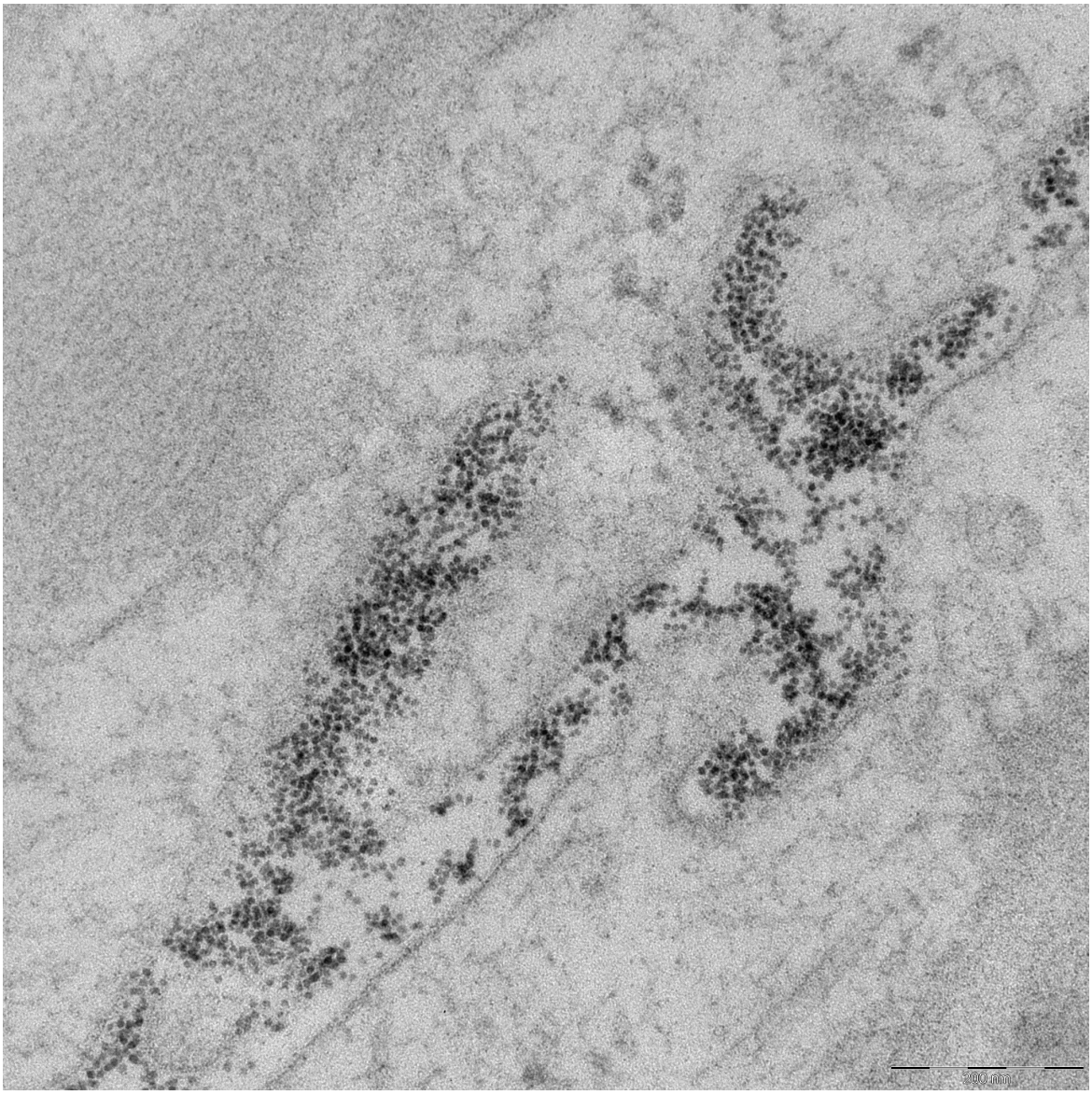


Fig. A.3 File name *Brain ferritin HPF 0004*

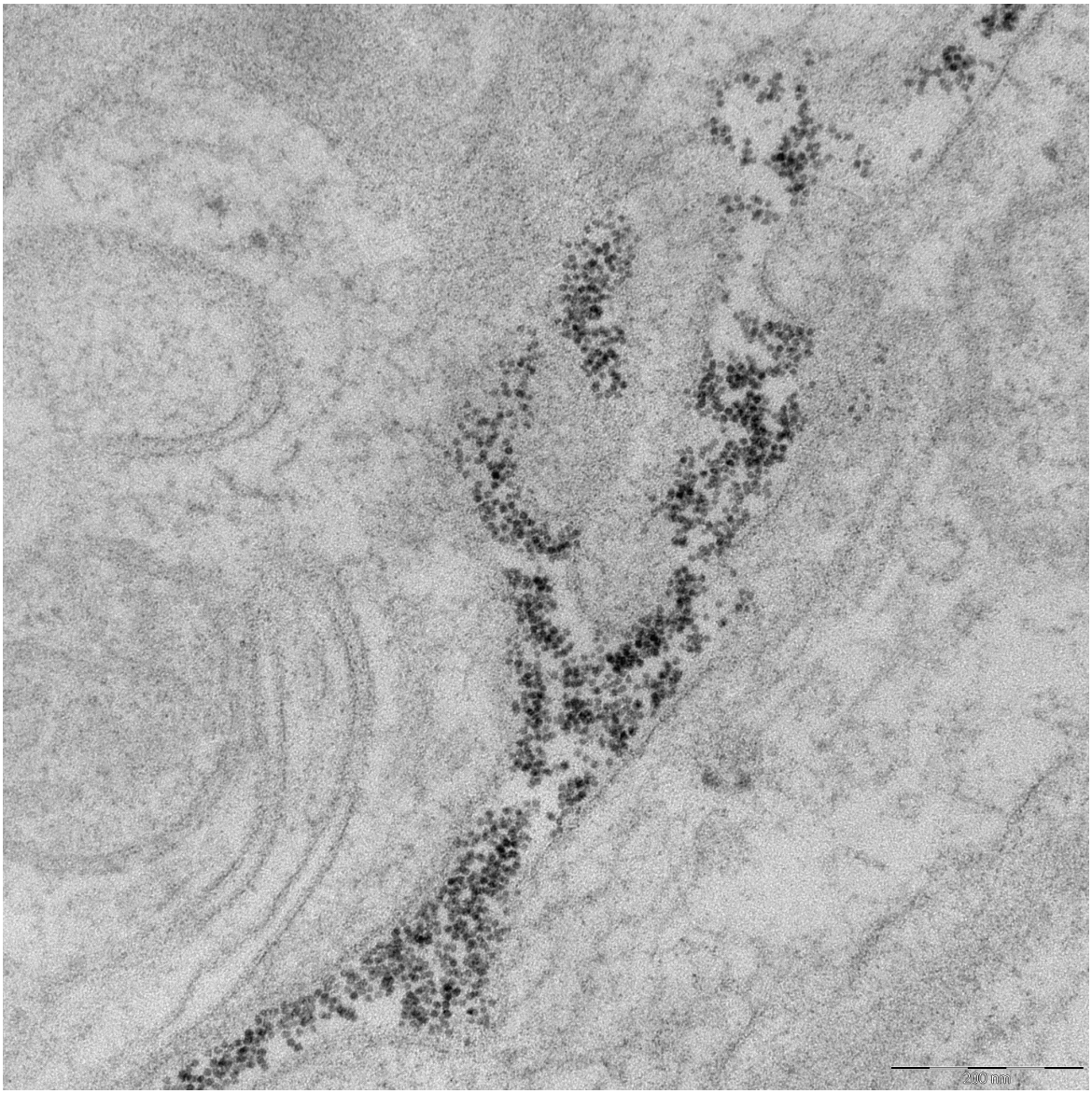


Fig. A.4 File name *Brain ferritin HPF 0005*

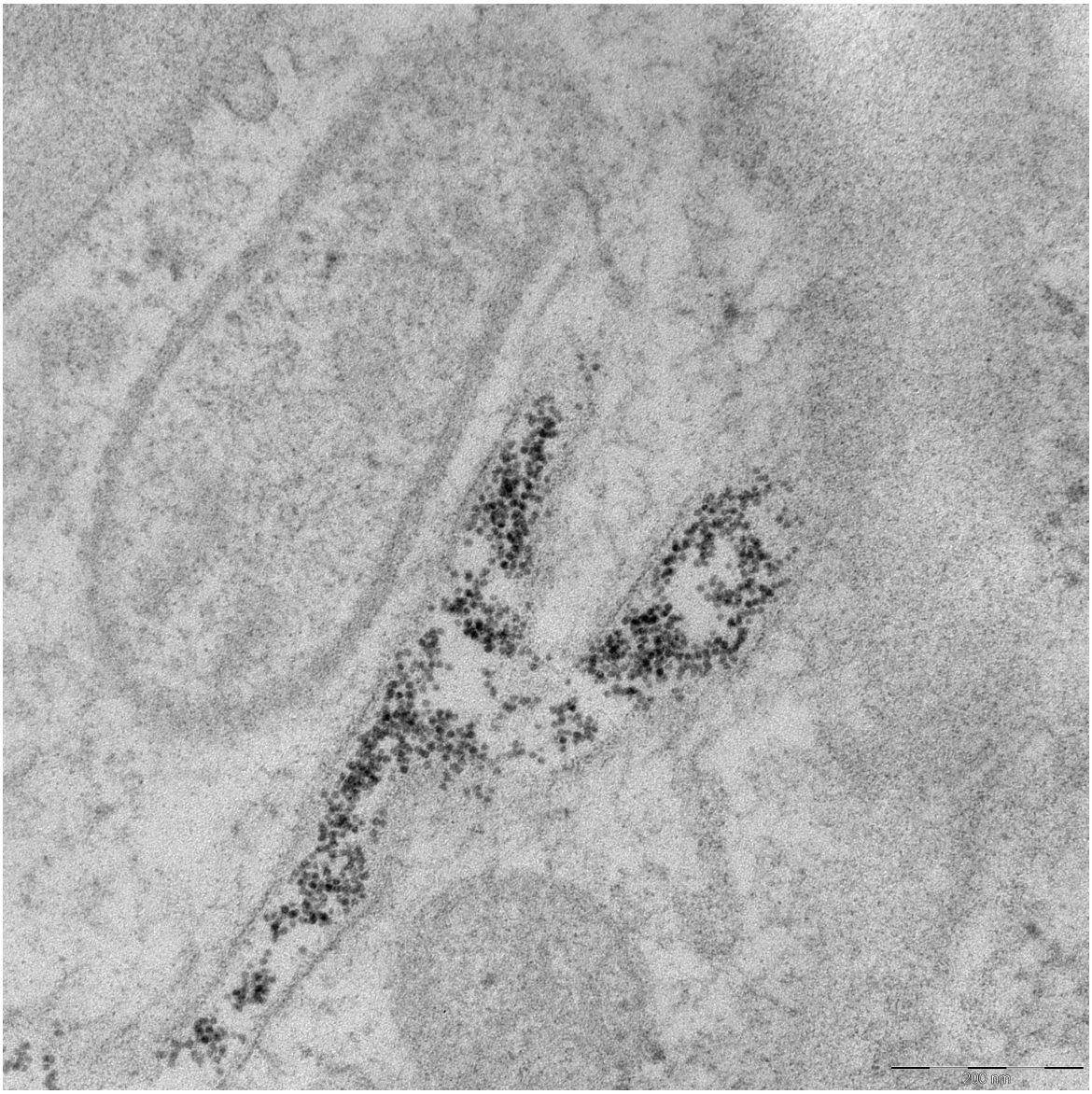


Fig. A.5 File name *Brain ferritin HPF 0006*

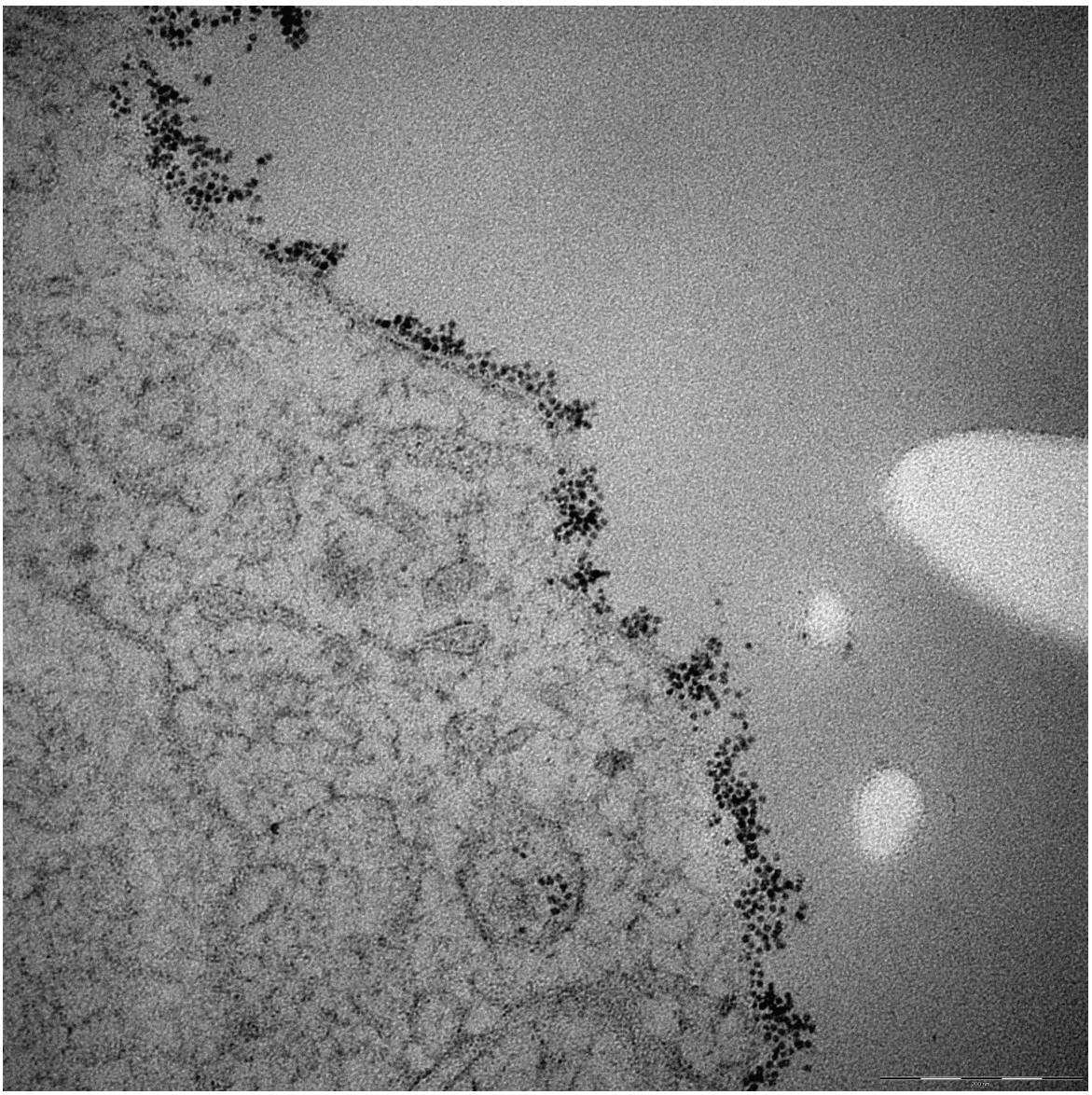


Fig. A.6 File name *Brain ferritin HPF 0008*

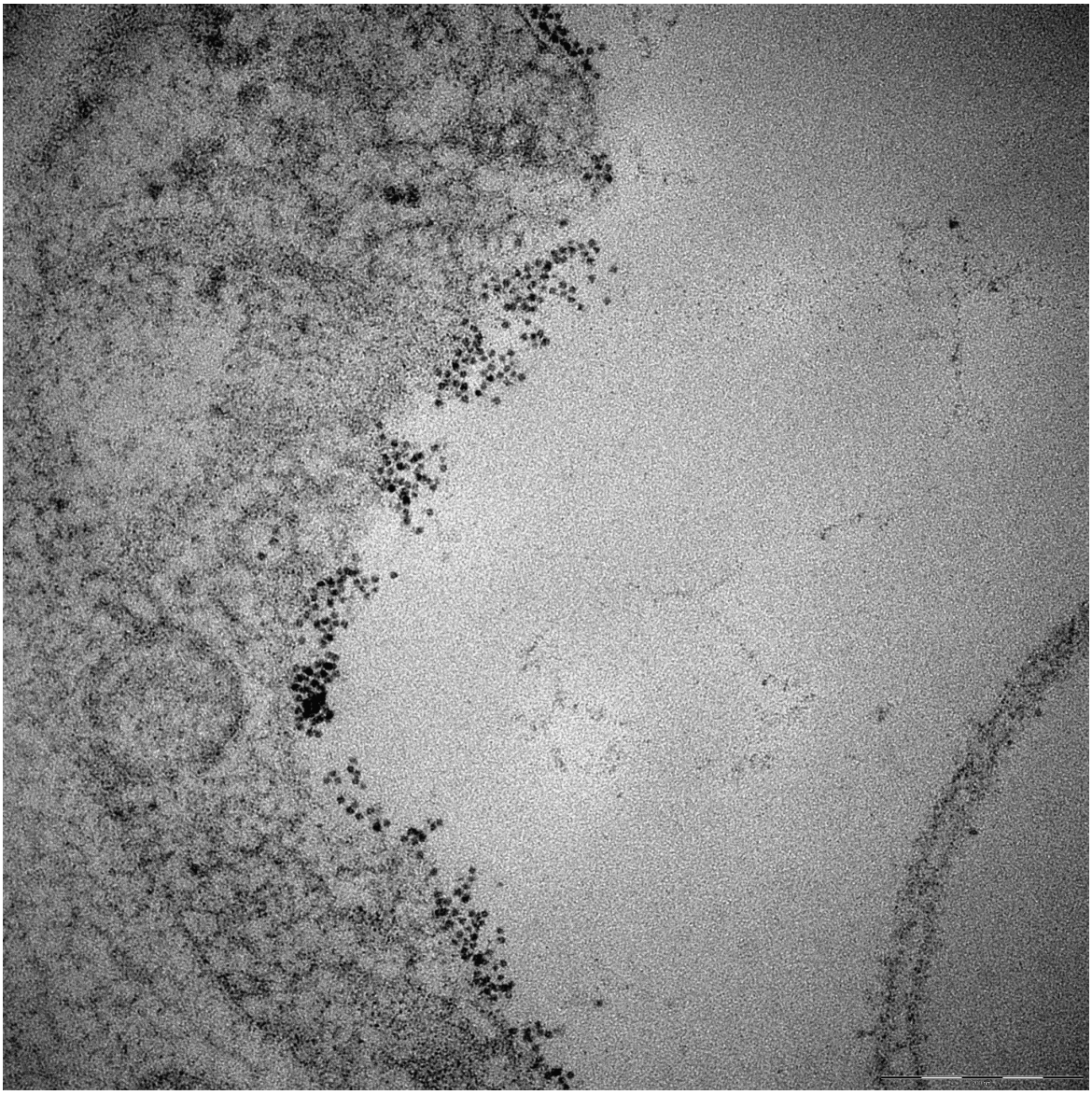


Fig. A.7 File name *Brain ferritin HPF 0016*

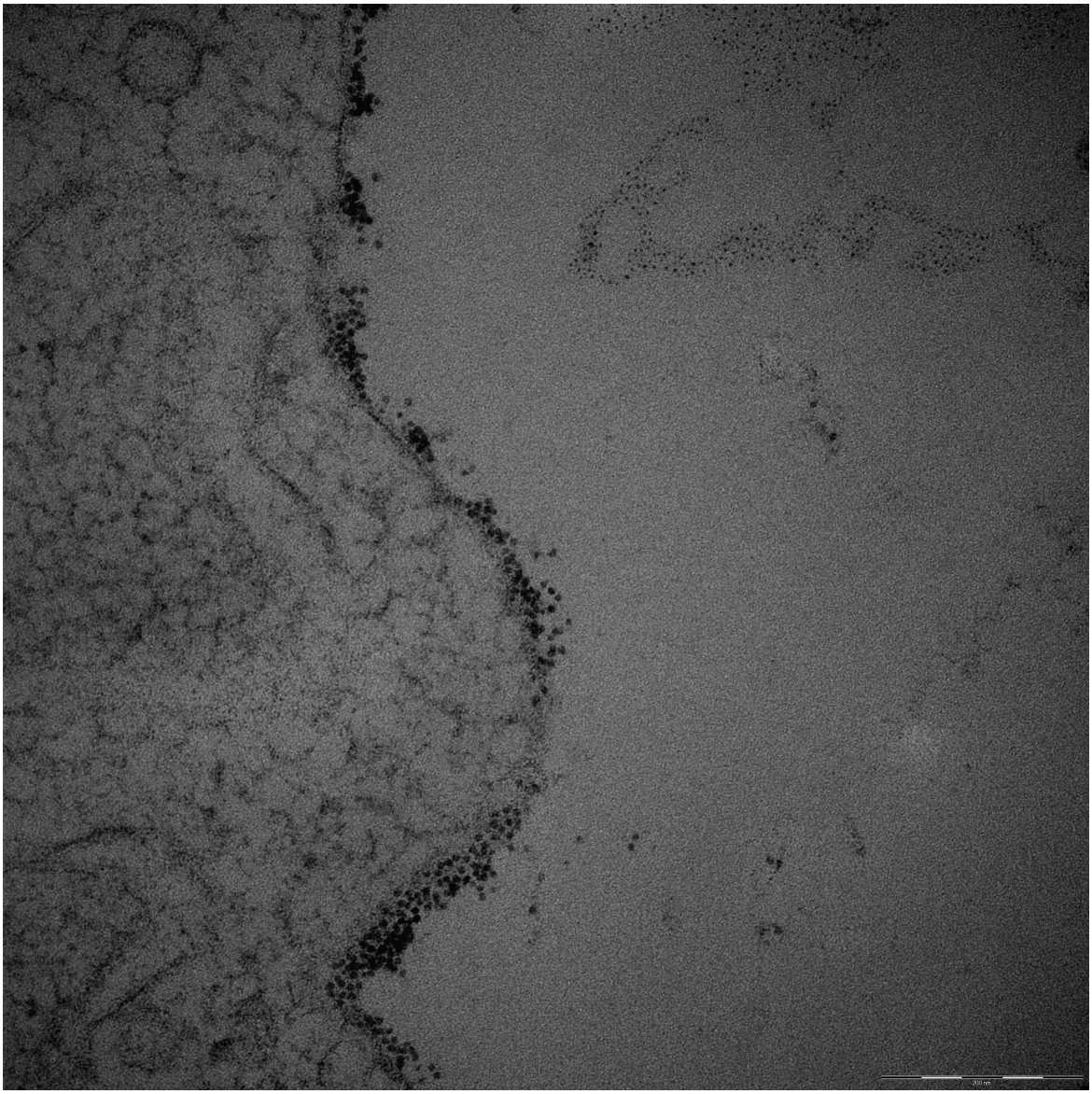


Fig. A.8 File name *Brain ferritin HPF 0019*

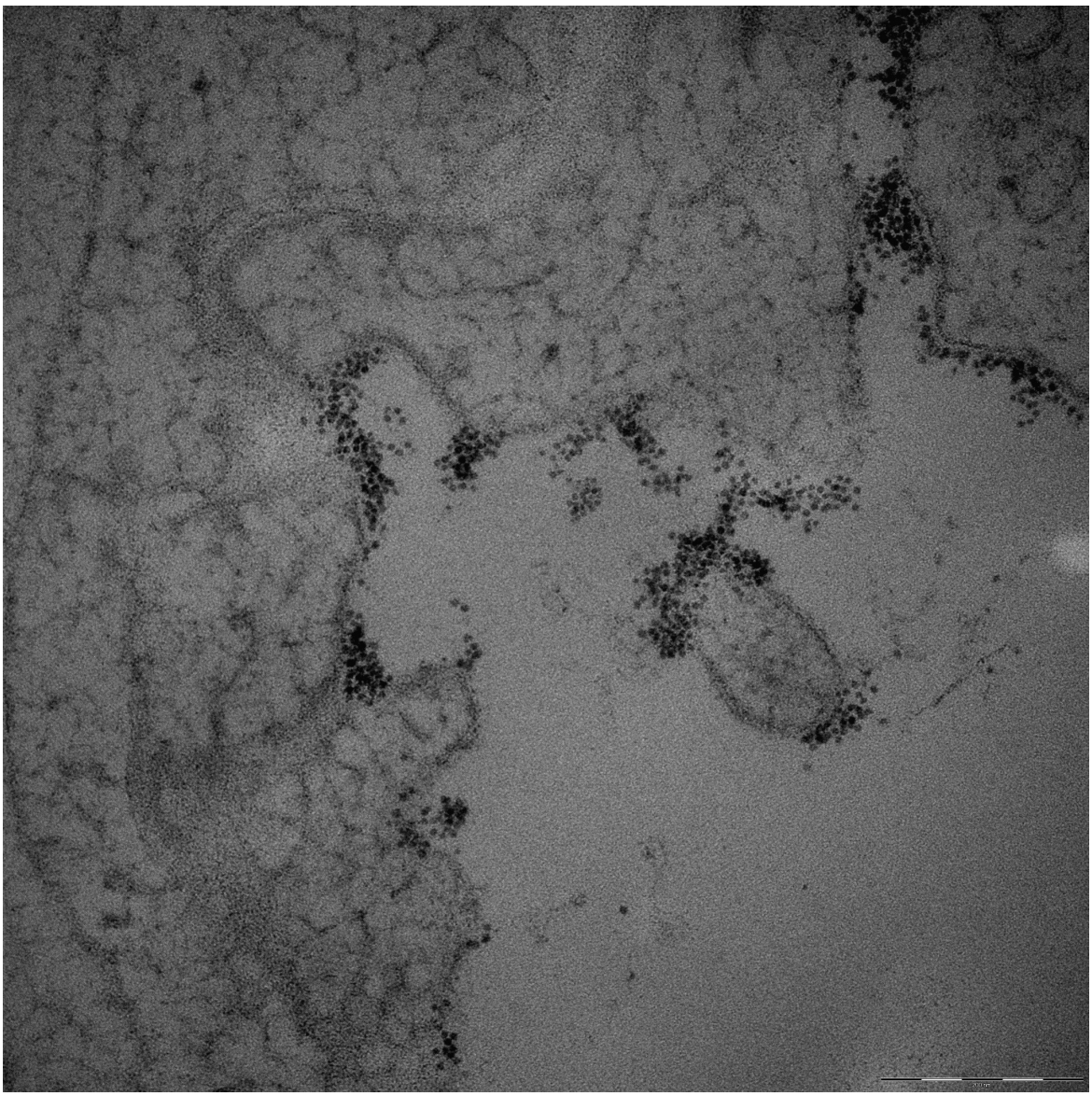


Fig. A.9 File name *Brain ferritin HPF 0020*

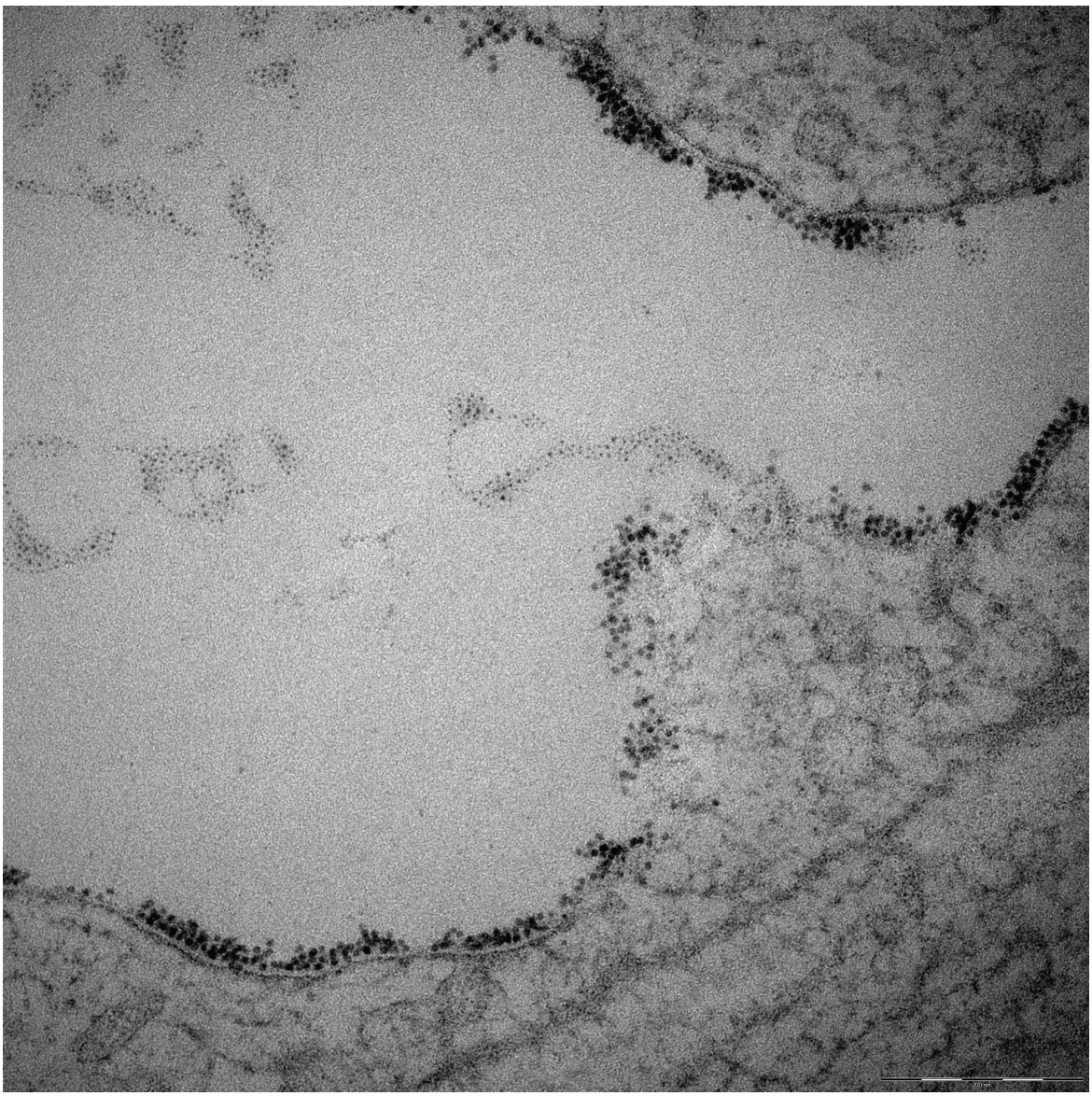


Fig. A.10 File name *Brain ferritin HPF 0021*

Appendix B

Noise Samples

Display of the noise patch selections together with the name of the image they come from and the coordinates of the noise patches. Each patch that was used for analyzing the noise has size 256x256 pixels.

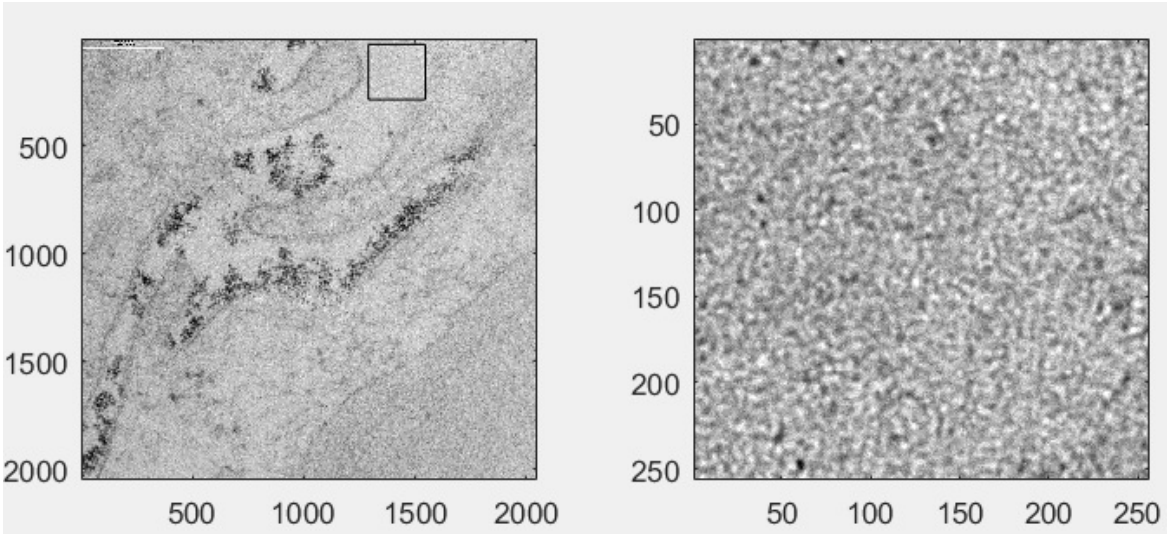


Fig. B.1 From image *Brain ferritin HPF 0002* I took sample from ($X = 1290$, $Y = 28$)

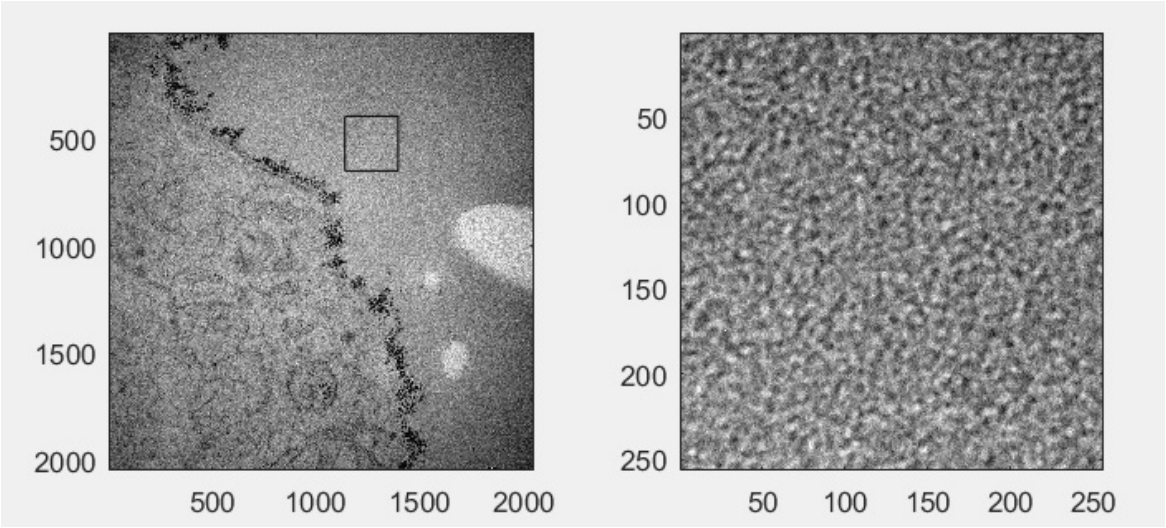


Fig. B.2 From image *Brain ferritin HPF 0008* I took sample from ($X = 1138$, $Y = 391$)

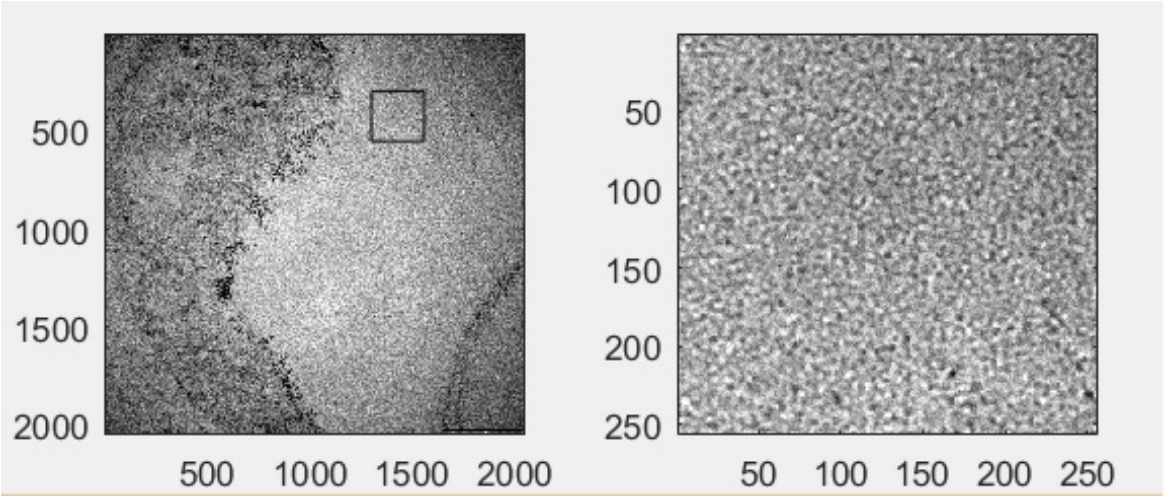


Fig. B.3 From image *Brain ferritin HPF 0016* I took sample from ($X = 1300$, $Y = 294$)

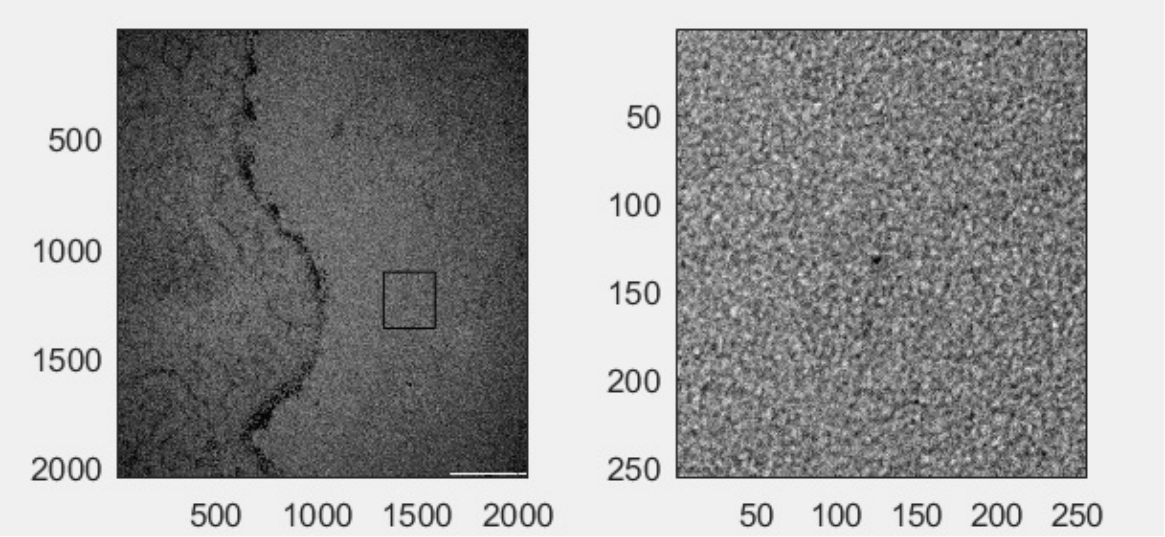


Fig. B.4 From image *Brain ferritin HPF 0019* I took sample from ($X = 1328$, $Y = 1108$)

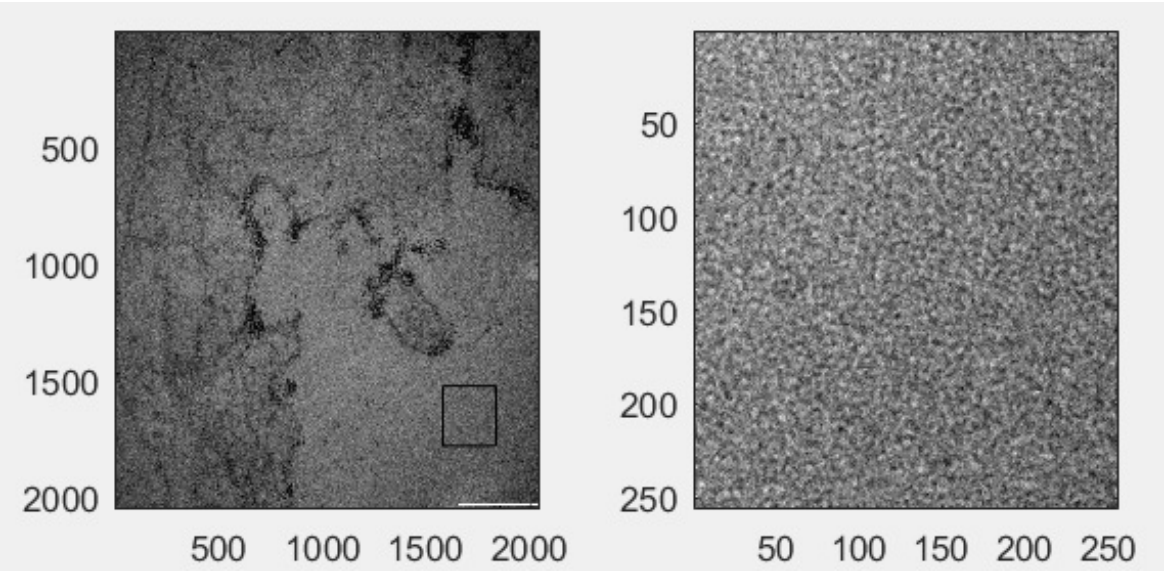


Fig. B.5 From image *Brain ferritin HPF 0020* I took sample from ($X = 1581$, $Y = 1520$)

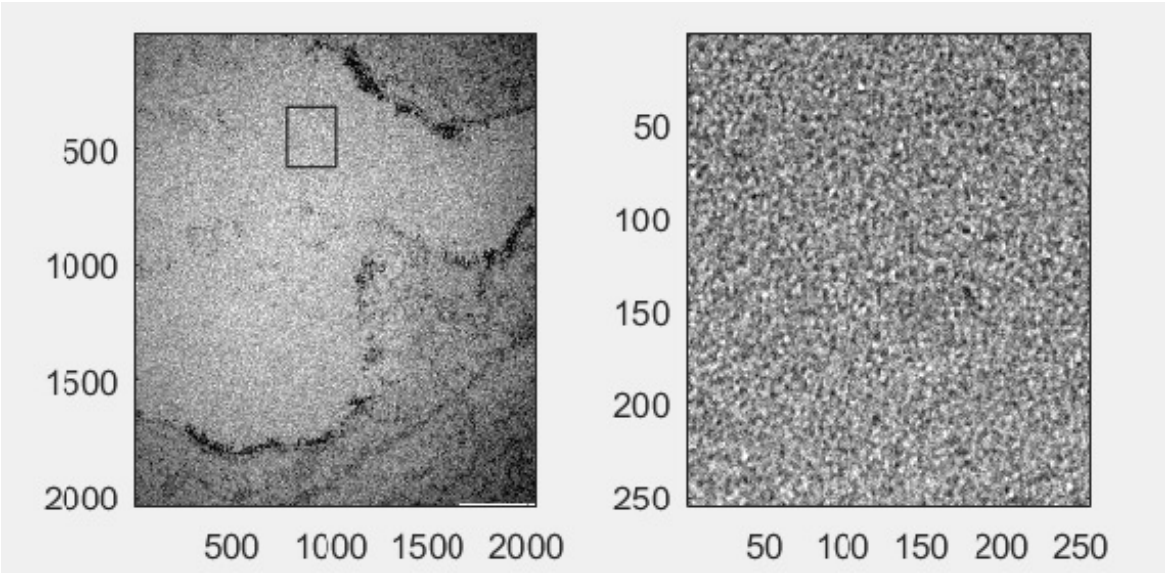
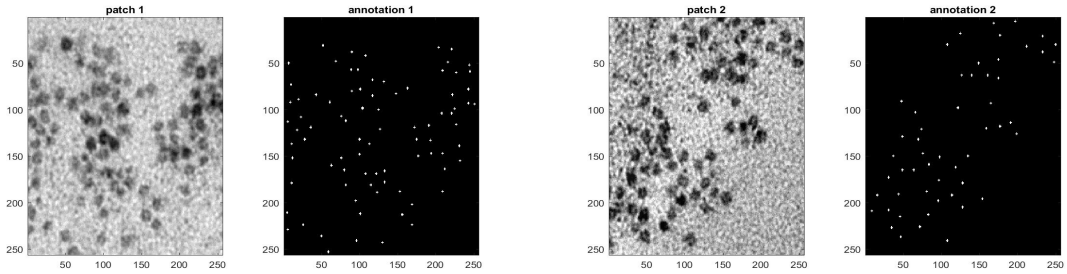


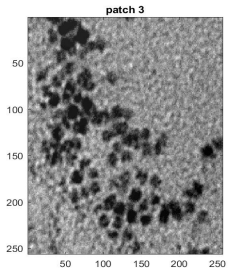
Fig. B.6 From image *Brain ferritin HPF 0021* I took sample from ($X = 778$, $Y = 324$)

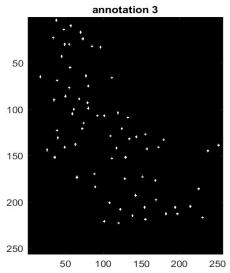
Appendix C

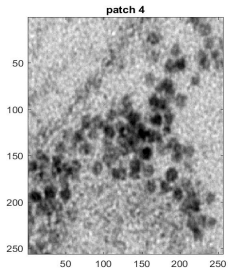
Patches with Ferritin Molecules used for pixel classification

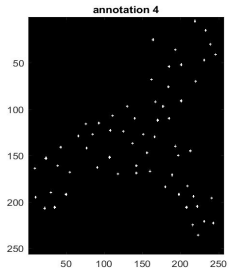
In this appendix I show the patches from the images that were used for training and testing the Ferriting Detection models. The patches are displayed together with the annotation for the center of each molecule.

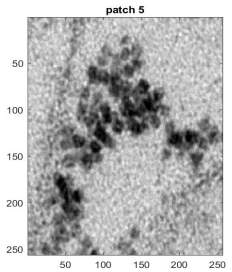


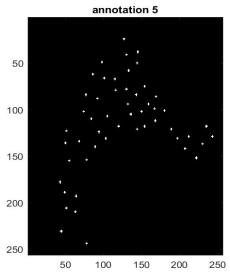

patch 3

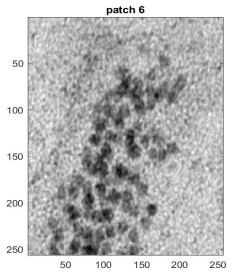

annotation 3

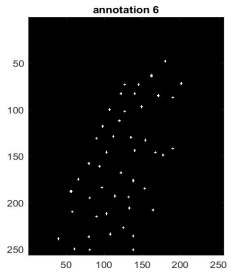

patch 4

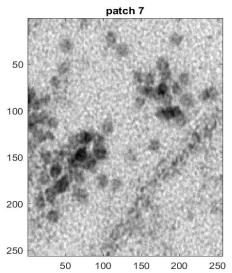

annotation 4

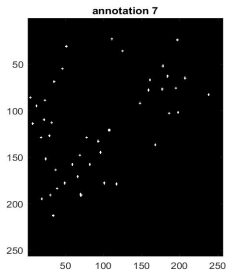

patch 5

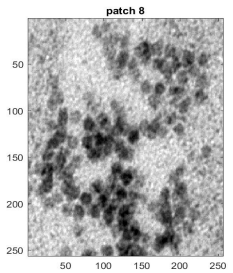

annotation 5

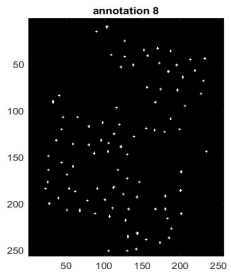

patch 6

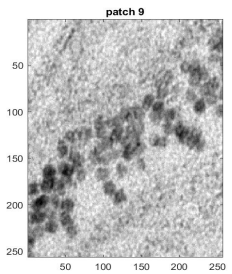

annotation 6

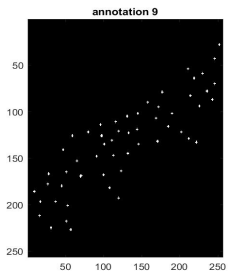

patch 7

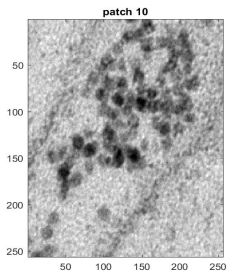

annotation 7

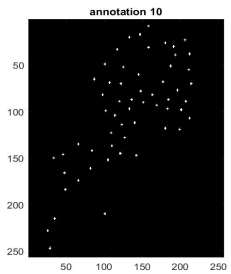

patch 8

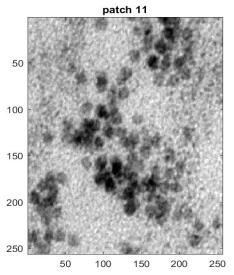

annotation 8

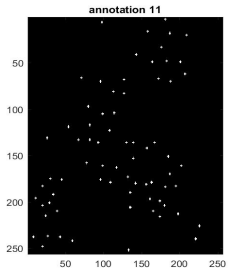

patch 9

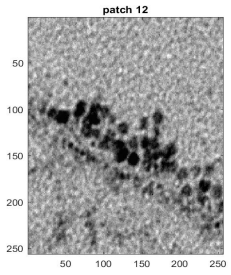

annotation 9

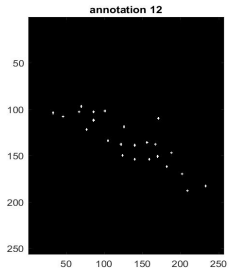

patch 10

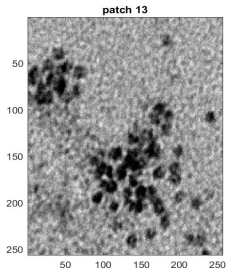

annotation 10

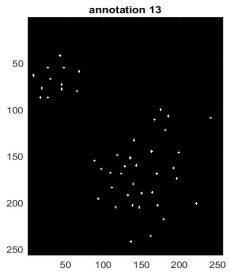

patch 11
50
100
150
200
250
50 100 150 200 250

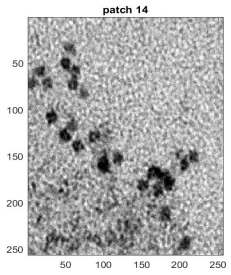

annotation 11
50
100
150
200
250
50 100 150 200 250

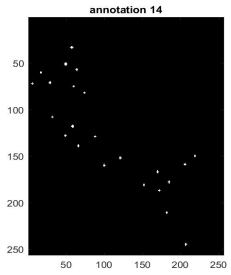

patch 12
50
100
150
200
250
50 100 150 200 250

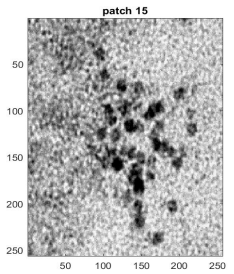

annotation 12
50
100
150
200
250
50 100 150 200 250

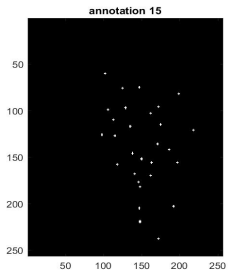

patch 13
50
100
150
200
250
50 100 150 200 250

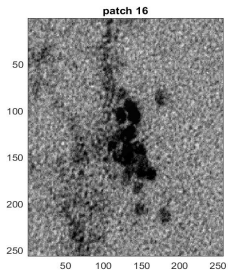

annotation 13
50
100
150
200
250
50 100 150 200 250

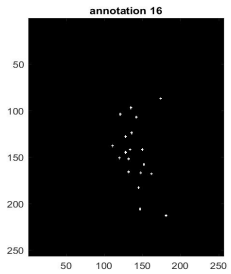

patch 14
50
100
150
200
250
50 100 150 200 250

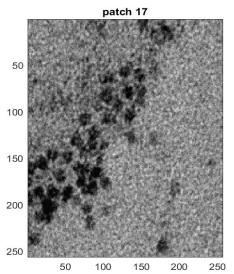

annotation 14
50
100
150
200
250
50 100 150 200 250

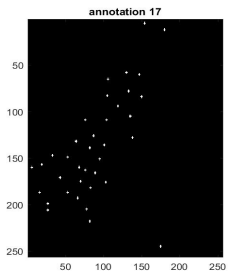

patch 15
50
100
150
200
250
50 100 150 200 250

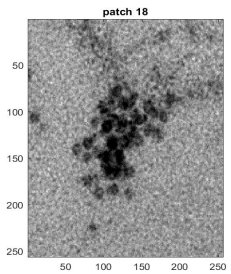

annotation 15
50
100
150
200
250
50 100 150 200 250

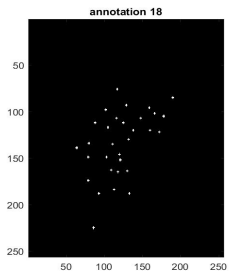

patch 16
50
100
150
200
250
50 100 150 200 250

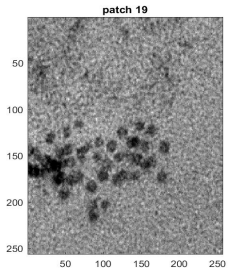

annotation 16
50
100
150
200
250
50 100 150 200 250

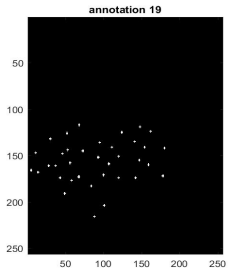

patch 17
50
100
150
200
250
50 100 150 200 250

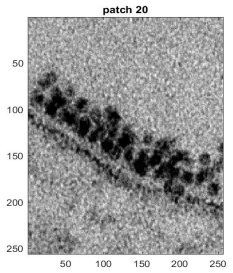

annotation 17
50
100
150
200
250
50 100 150 200 250

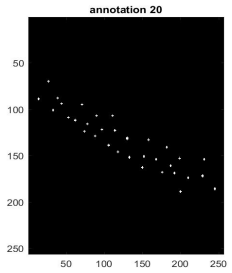

patch 18
50
100
150
200
250
50 100 150 200 250

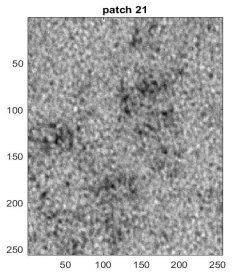

annotation 18
50
100
150
200
250
50 100 150 200 250

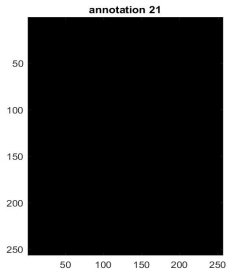

patch 19
50
100
150
200
250
50 100 150 200 250

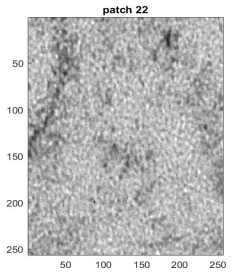

annotation 19
50
100
150
200
250
50 100 150 200 250

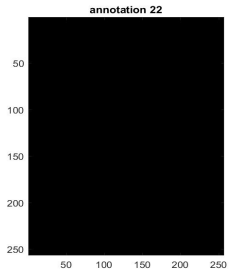

patch 20
50
100
150
200
250
50 100 150 200 250

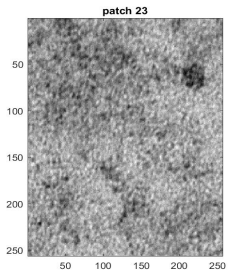

annotation 20
50
100
150
200
250
50 100 150 200 250

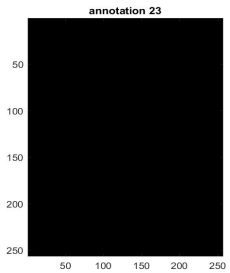

patch 21
50
100
150
200
250
50 100 150 200 250

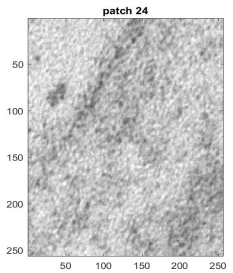

annotation 21
50
100
150
200
250
50 100 150 200 250

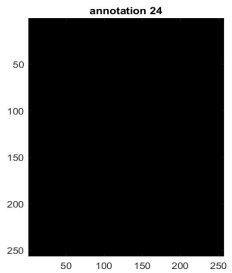

patch 22
50
100
150
200
250
50 100 150 200 250

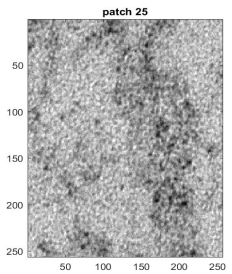

annotation 22
50
100
150
200
250
50 100 150 200 250

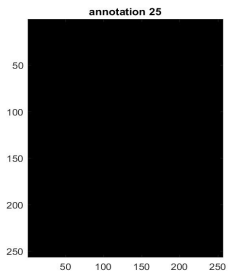

patch 23
50
100
150
200
250
50 100 150 200 250

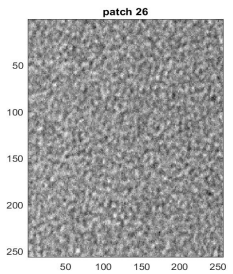

annotation 23
50
100
150
200
250
50 100 150 200 250

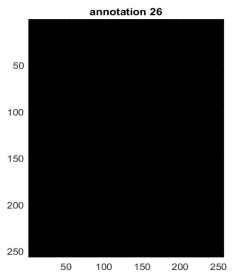

patch 24
50
100
150
200
250
50 100 150 200 250

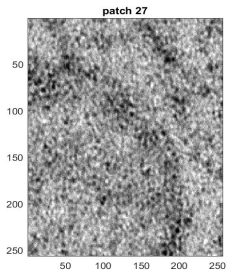

annotation 24
50
100
150
200
250
50 100 150 200 250

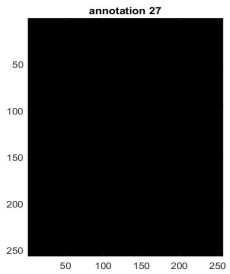

patch 25
50
100
150
200
250
50 100 150 200 250

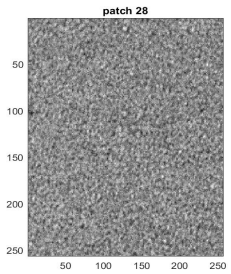

annotation 25
50
100
150
200
250
50 100 150 200 250

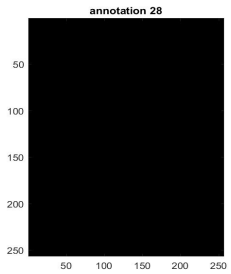

patch 26
50
100
150
200
250
50 100 150 200 250

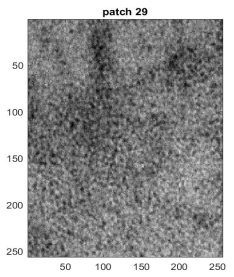

annotation 26
50
100
150
200
250
50 100 150 200 250

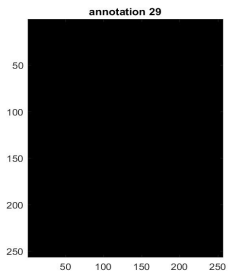

patch 27
50
100
150
200
250
50 100 150 200 250

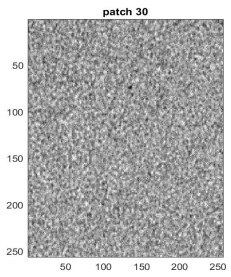

annotation 27
50
100
150
200
250
50 100 150 200 250

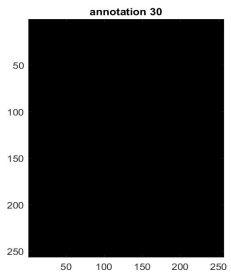

patch 28
50
100
150
200
250
50 100 150 200 250


annotation 28
50
100
150
200
250
50 100 150 200 250


patch 29
50
100
150
200
250
50 100 150 200 250

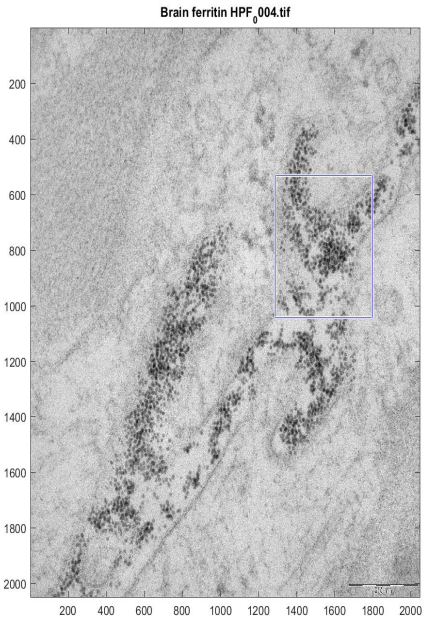

annotation 29
50
100
150
200
250
50 100 150 200 250

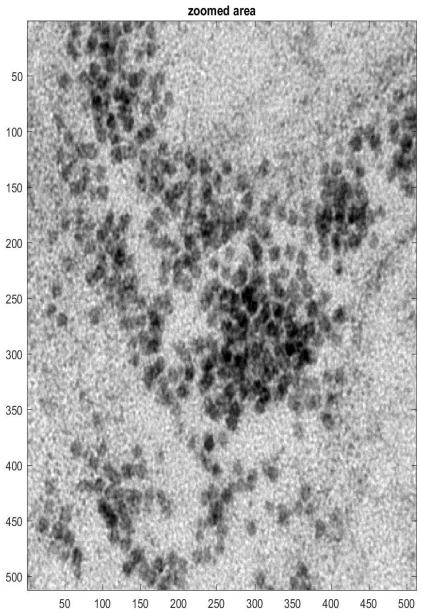

patch 30
50
100
150
200
250
50 100 150 200 250

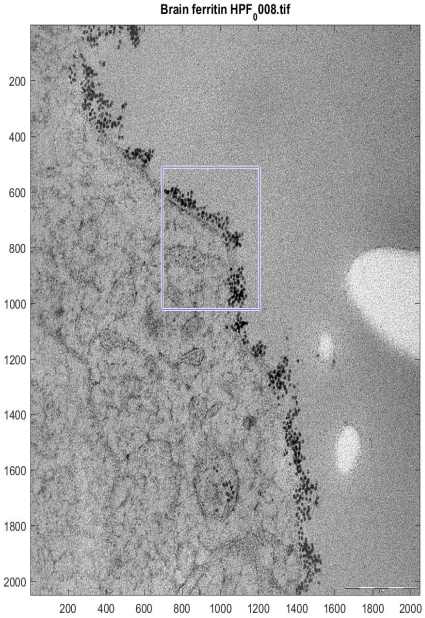

annotation 30
50
100
150
200
250
50 100 150 200 250

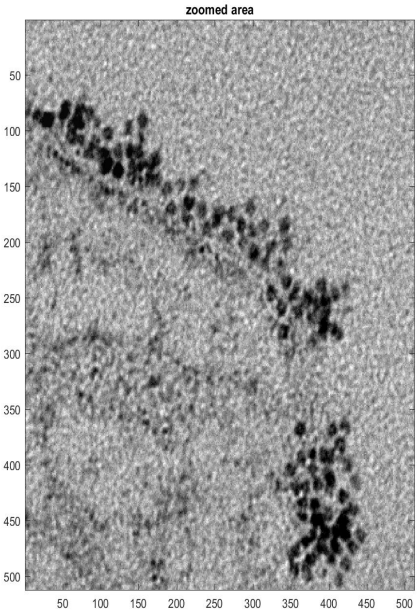
Appendix D

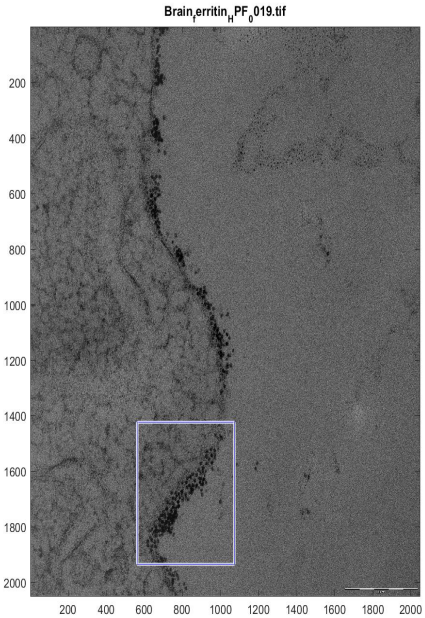
Images used for validating pixel classification algorithms

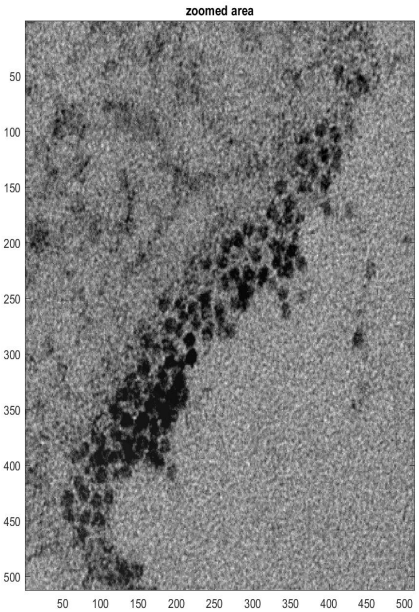


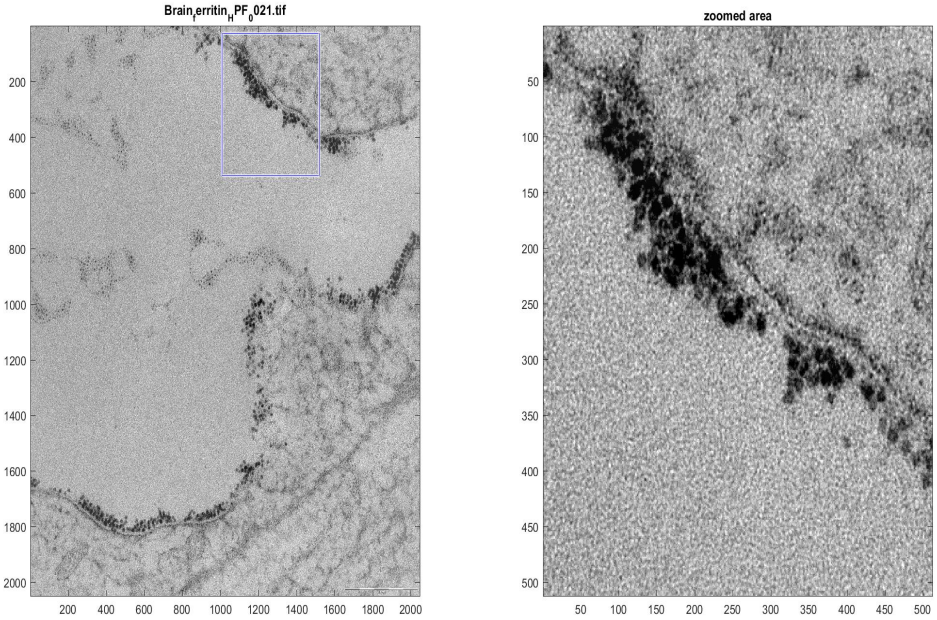



Brain ferritin HPF0008.tif
200
400
600
800
1000
1200
1400
1600
1800
2000
200 400 600 800 1000 1200 1400 1600 1800 2000


zoomed area
50
100
150
200
250
300
350
400
450
500
50 100 150 200 250 300 350 400 450 500


BrainFerritinHPF0019.tif
200
400
600
800
1000
1200
1400
1600
1800
2000
200 400 600 800 1000 1200 1400 1600 1800 2000


zoomed area
50
100
150
200
250
300
350
400
450
500
50 100 150 200 250 300 350 400 450 500


Brain_ferritin_HPF_021.tif
200
400
600
800
1000
1200
1400
1600
1800
2000
zoomed area
50
100
150
200
250
300
350
400
450
500

Appendix E

Results for Mahlanobis pixel classification with different parameters

Visualizations correspond to entries from Table 3.1 in Section 3.3.1.

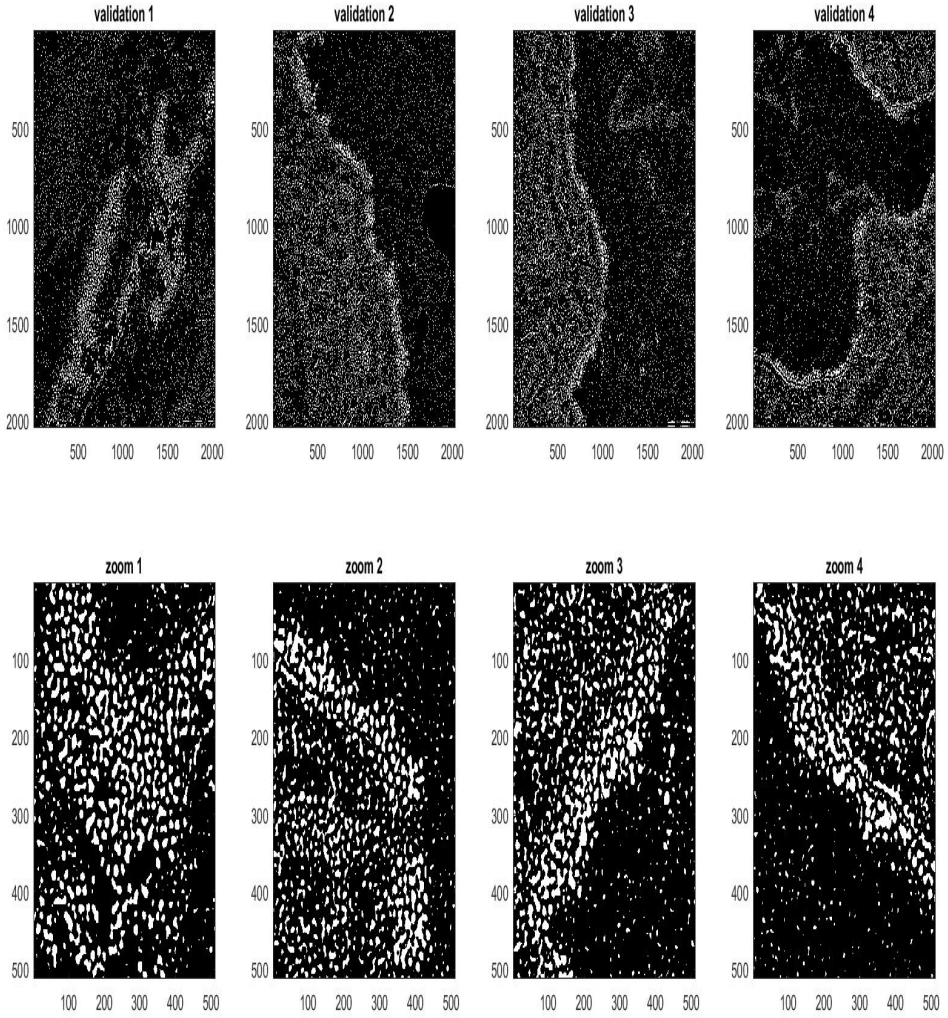


Fig. E.1 Tm1,Log3,Log5,Log7

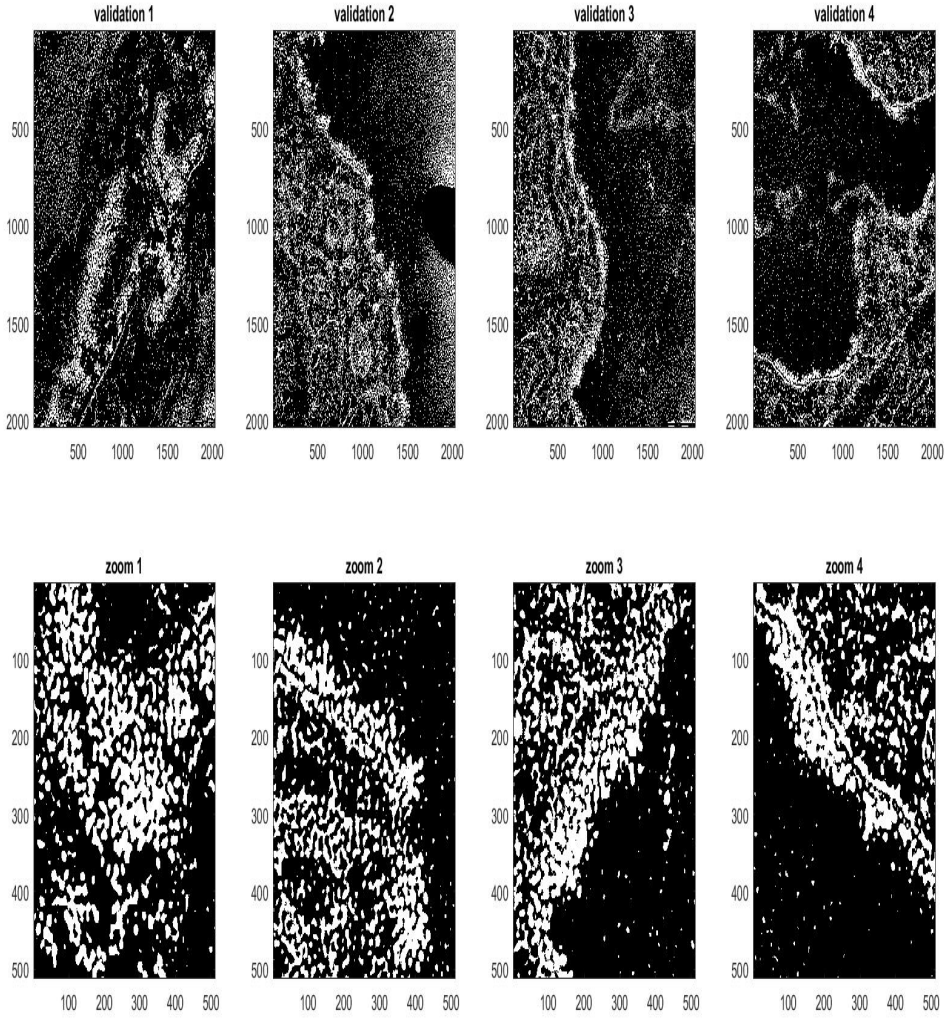


Fig. E.2 Raw,Tm1,Log3,Log5,Log7

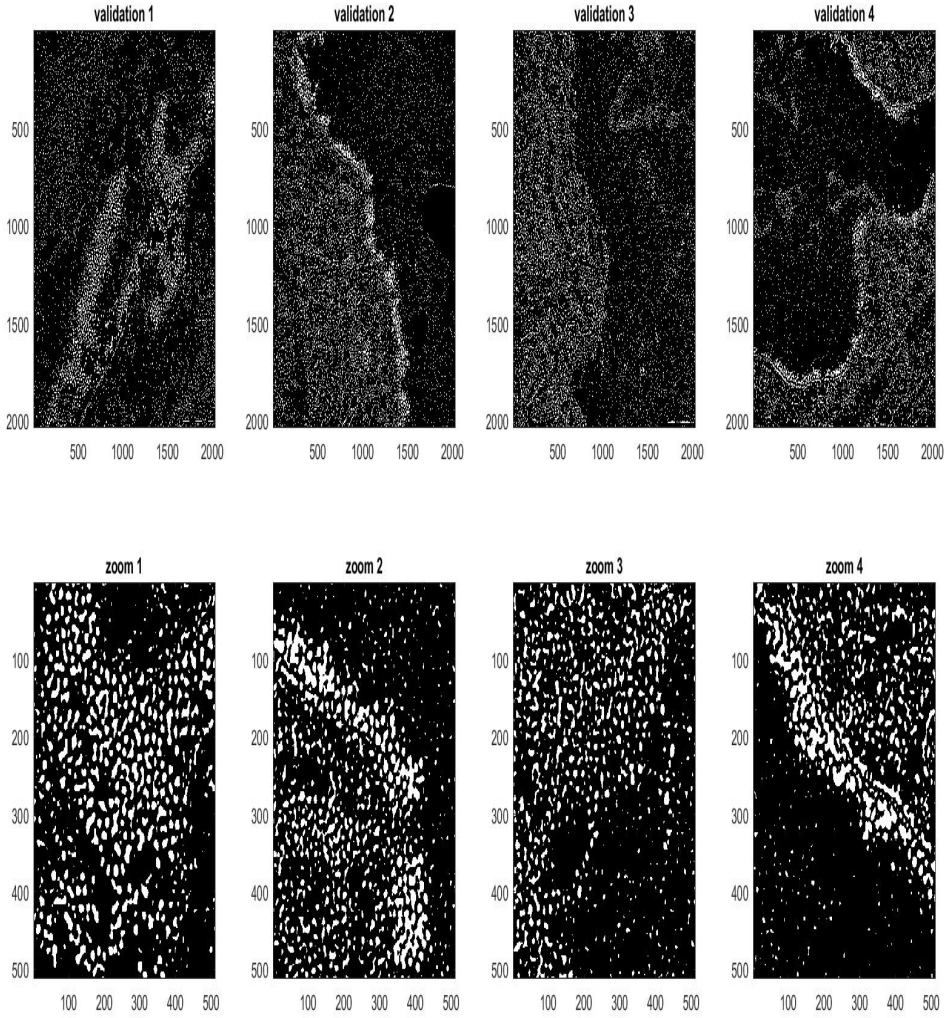


Fig. E.3 Tm1,Tm2,Log3,Log5,Log7

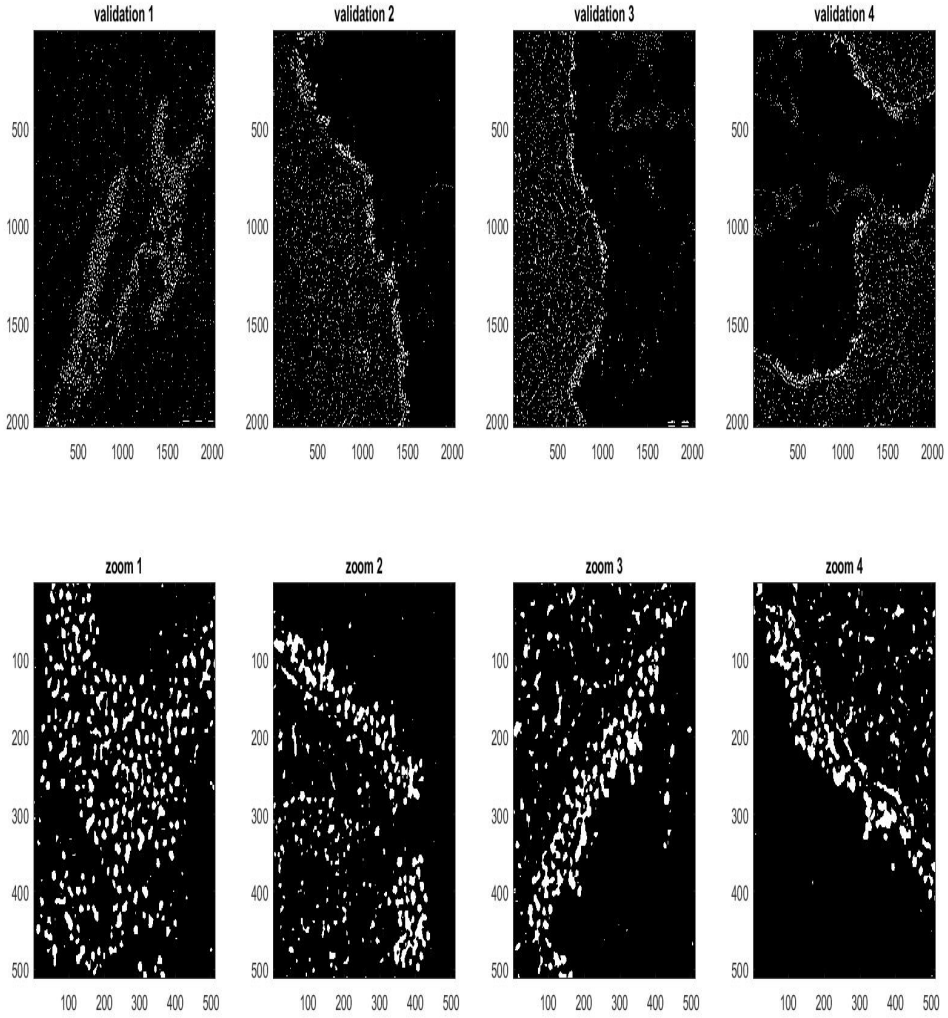


Fig. E.4 Log3,Log5,Log7

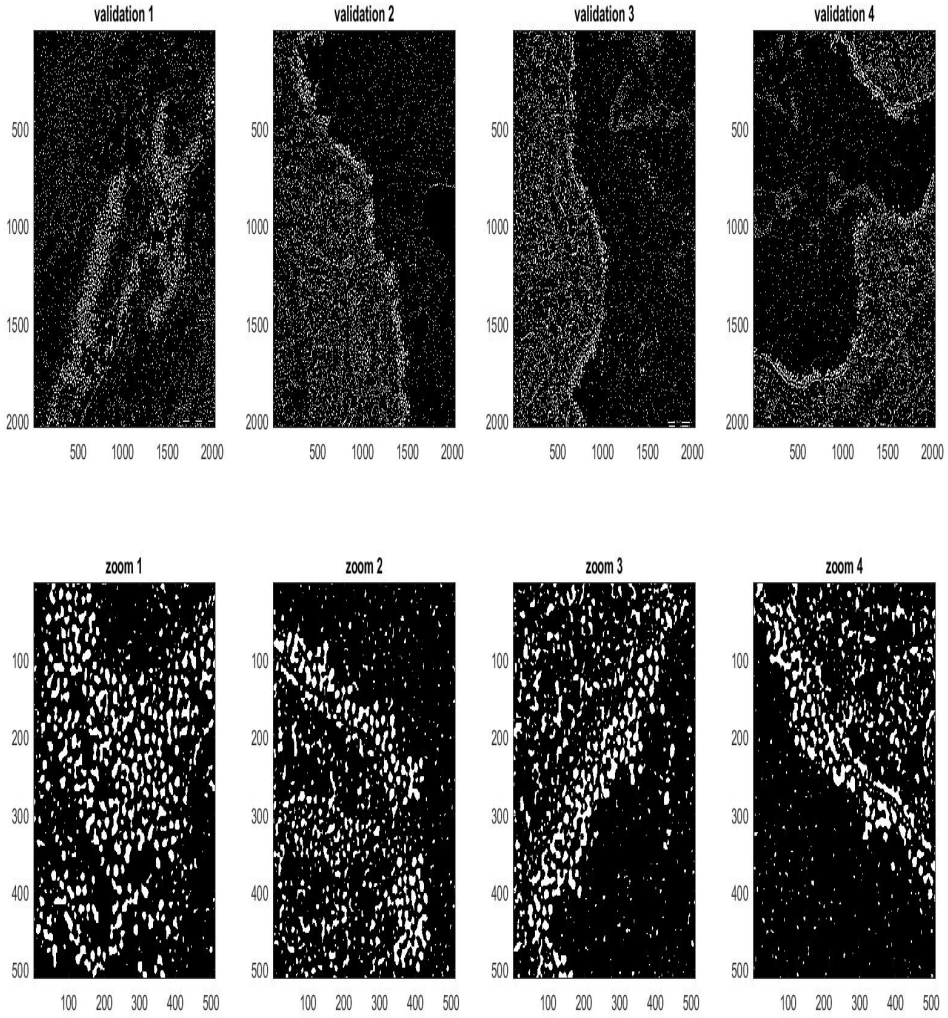


Fig. E.5 Tm1,Log3,Log5

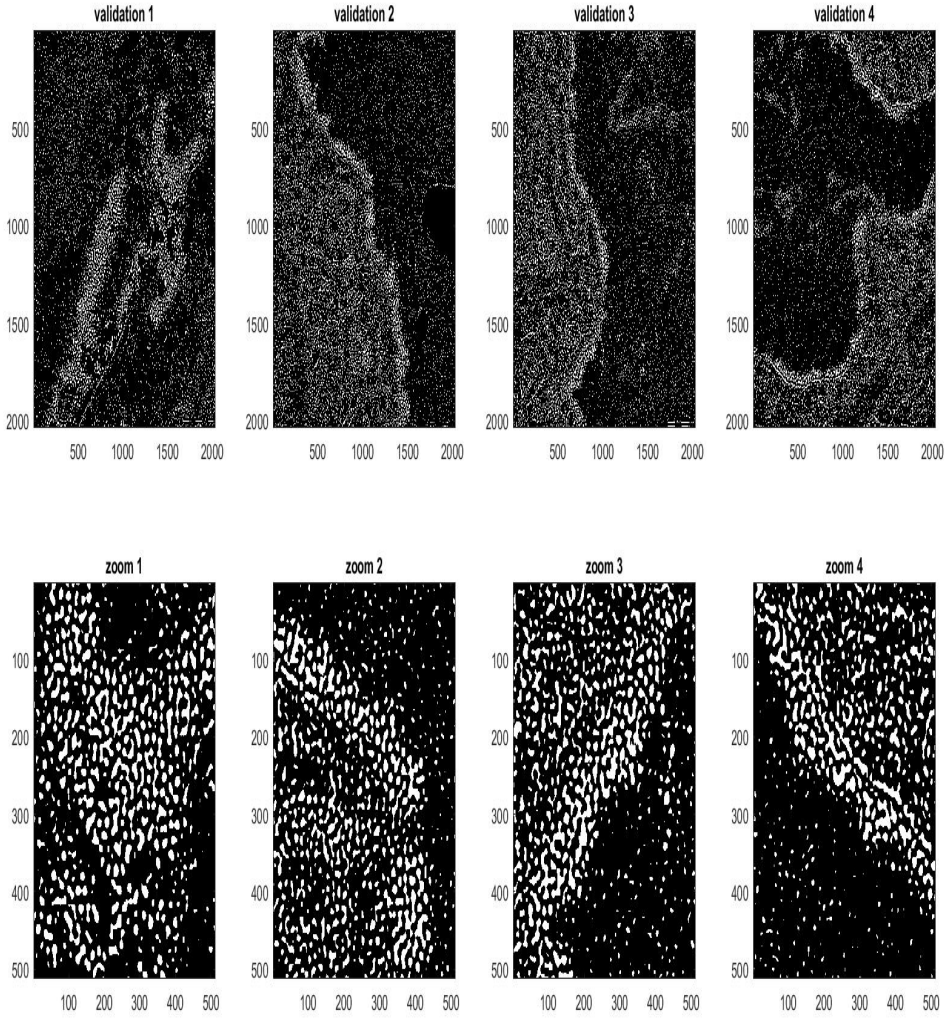


Fig. E.6 Tm1,Log5,Log7

Appendix F

Results for Mahlanobis pixel classification with different filters

Visualizations correspond to entries from Table 3.2 in Section 3.3.1.

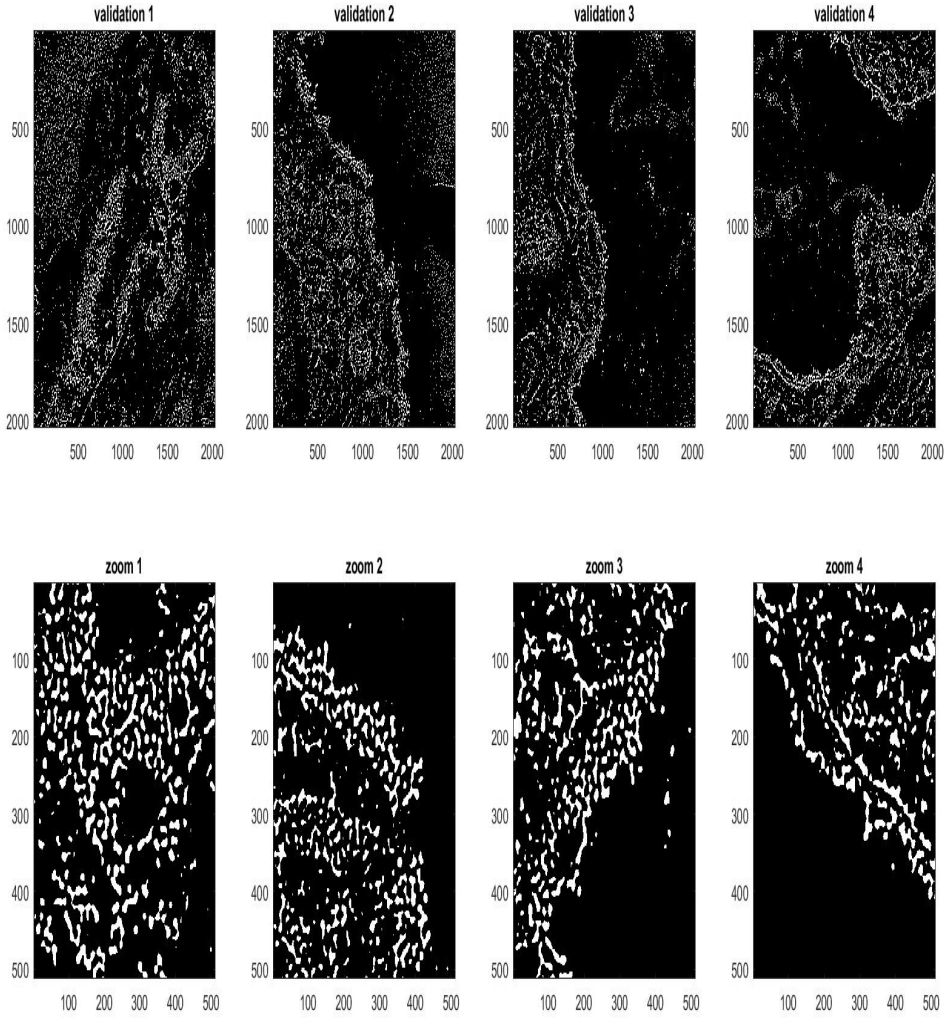


Fig. F.1 Histogram Equalization and Median filter 3x3

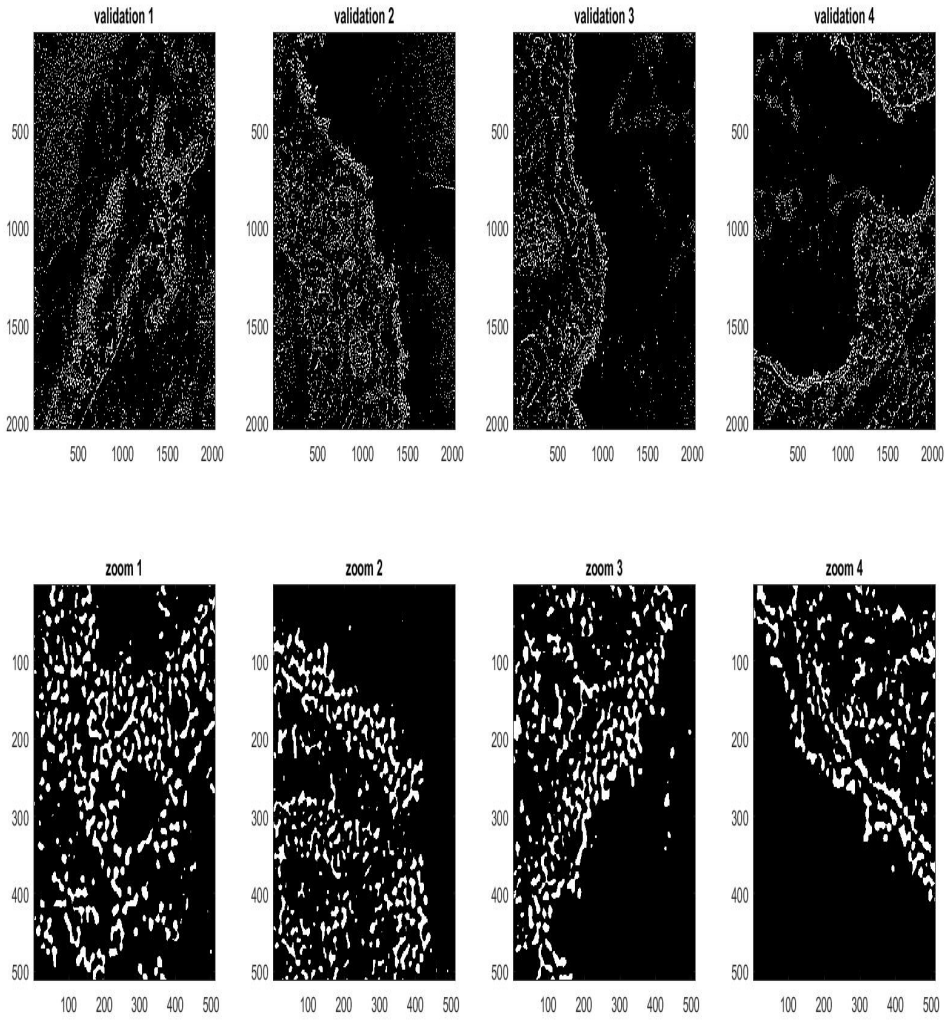


Fig. F.2 Histogram Equalization and Wiener filter 5x5

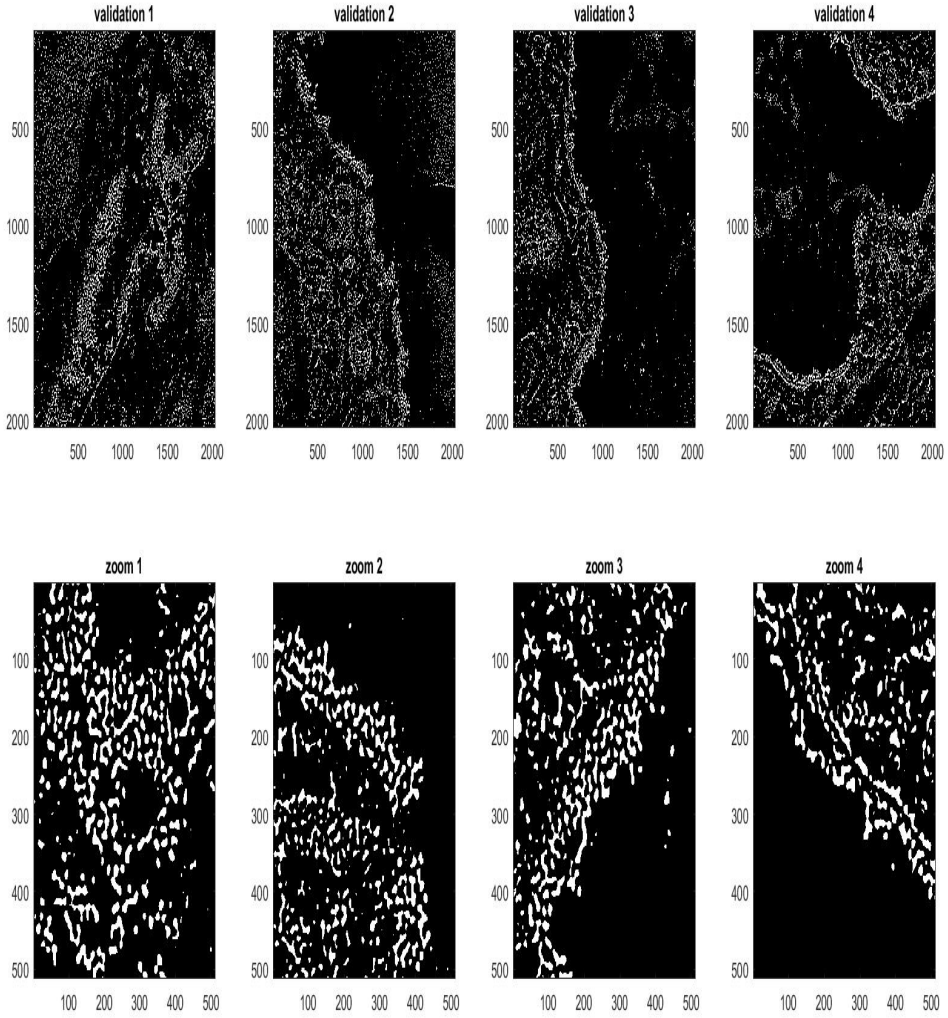


Fig. F.3 Histogram Equalization and Median filter 3x3

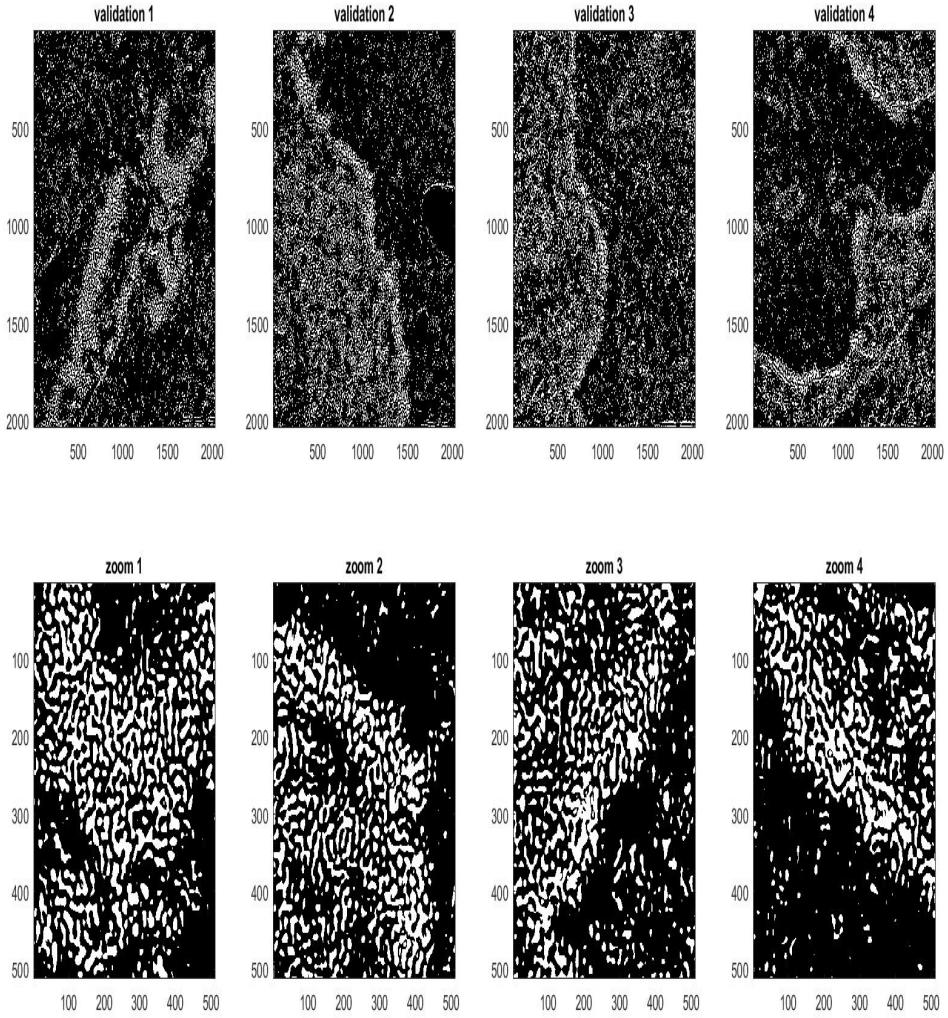
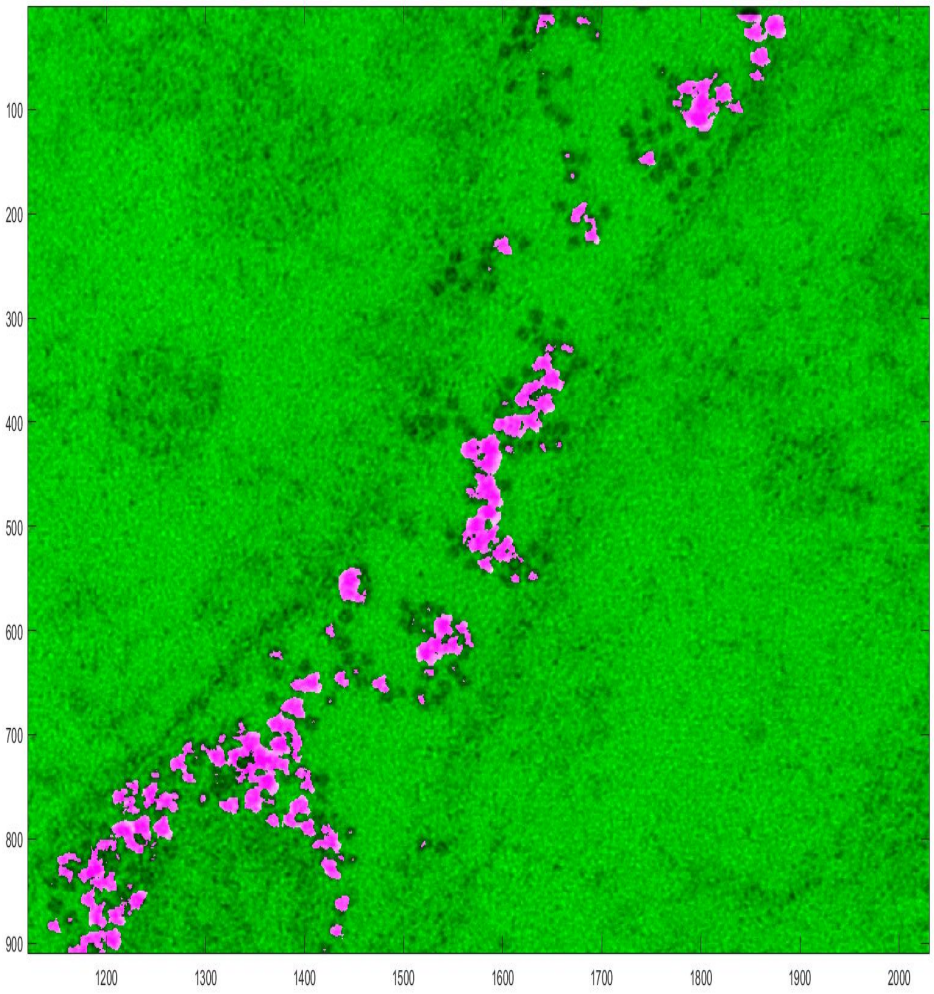

validation 1
validation 2
validation 3
validation 4
zoom 1
zoom 2
zoom 3
zoom 4

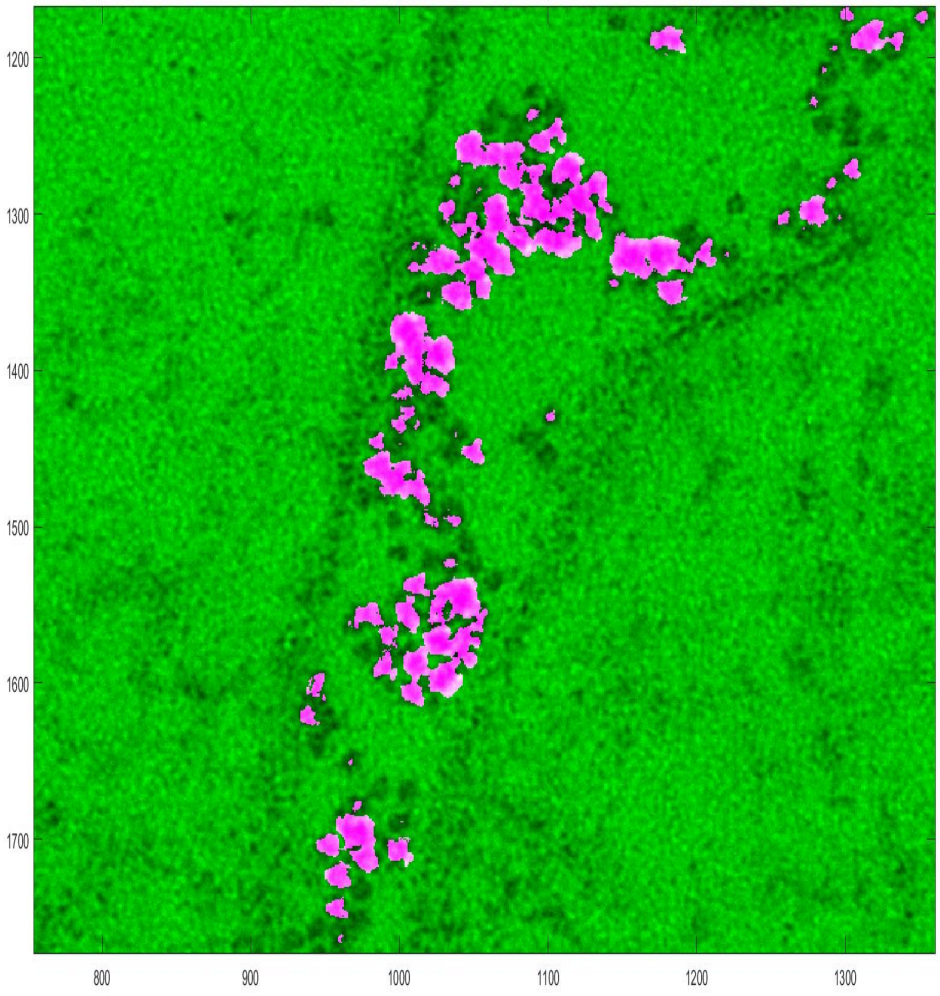
Fig. F.4 BM3D

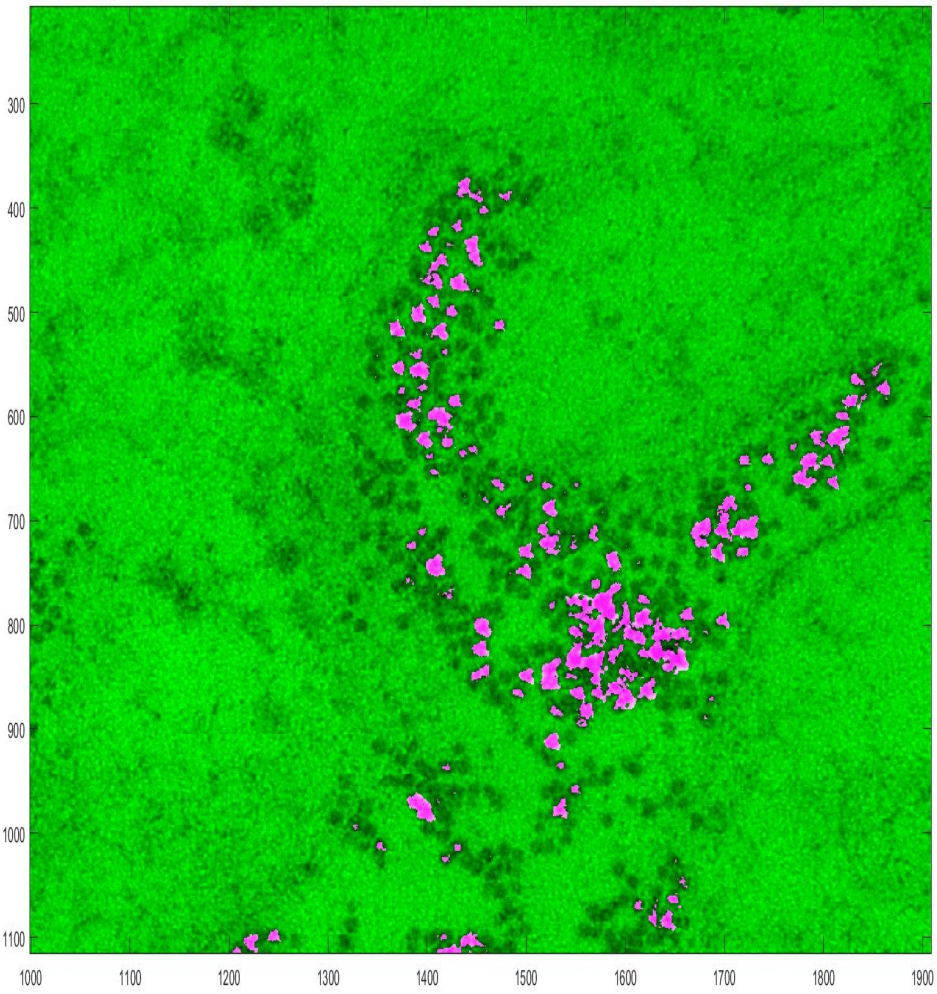
Appendix G

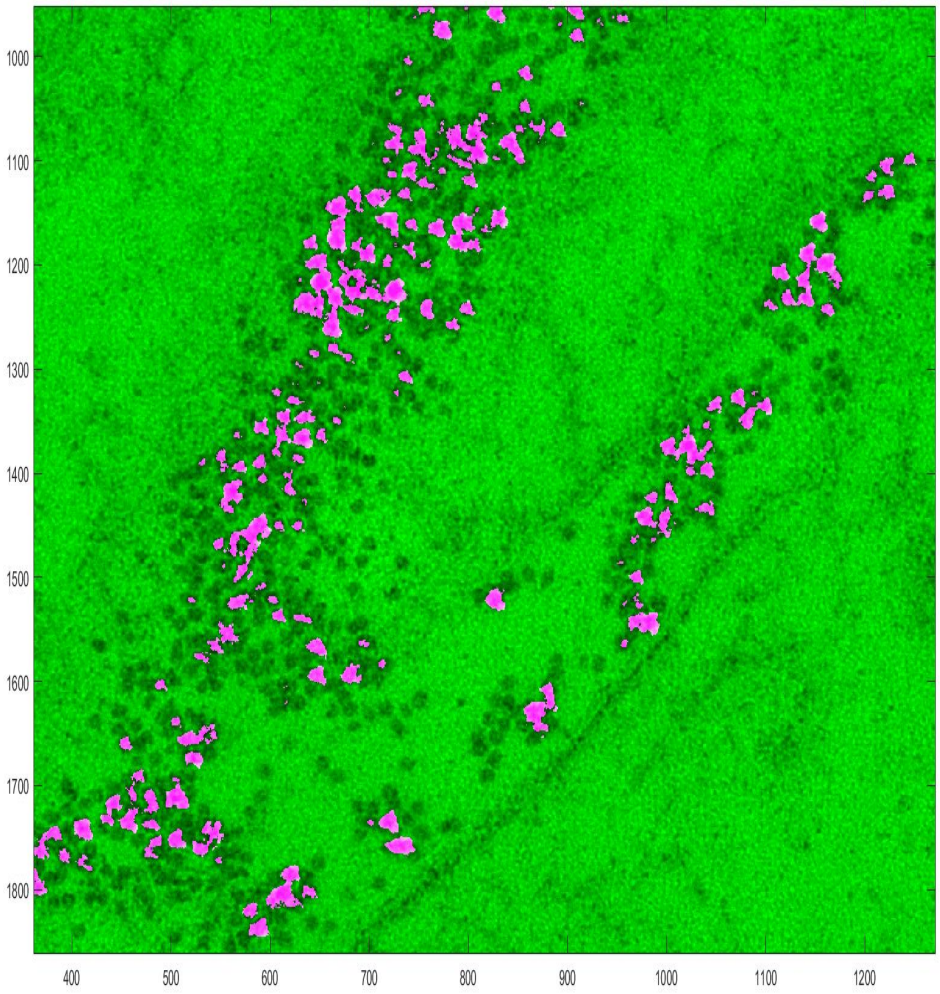
Examples for SVM results for Ferritin detection

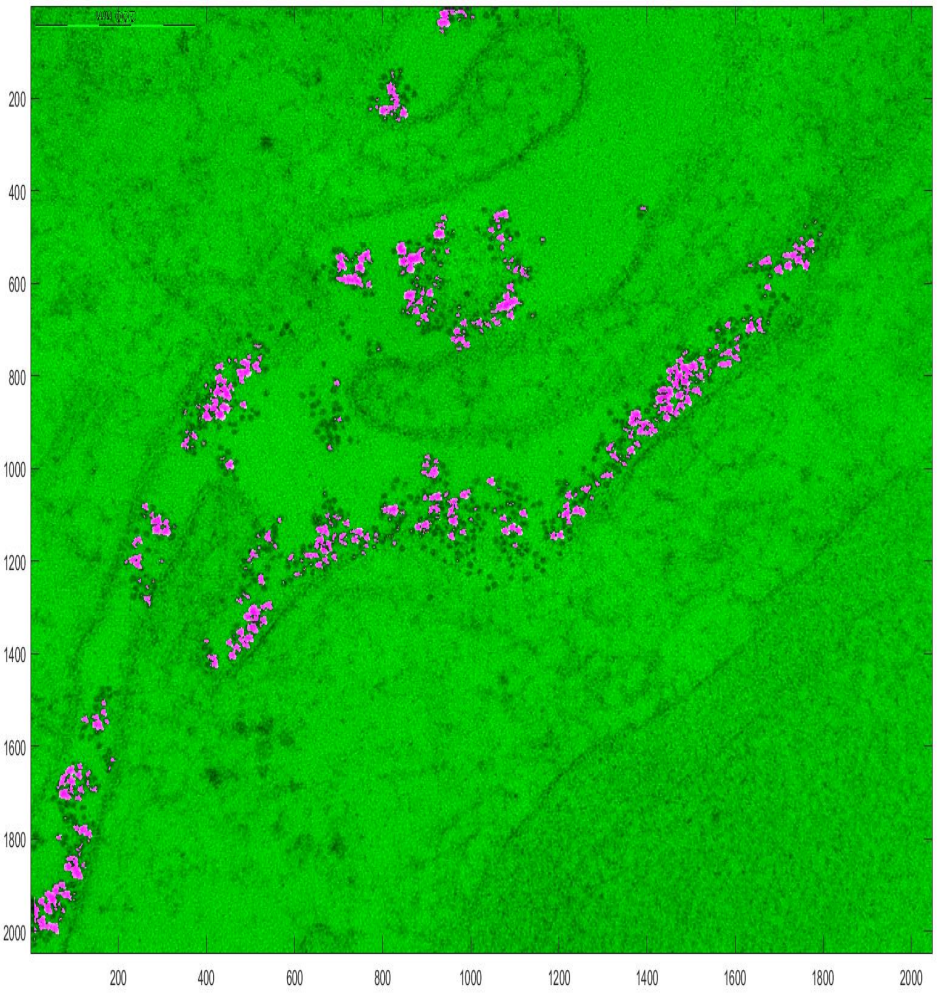
The areas marked with pink represent the pixels predicted to be in ferritin molecule by the classifier.

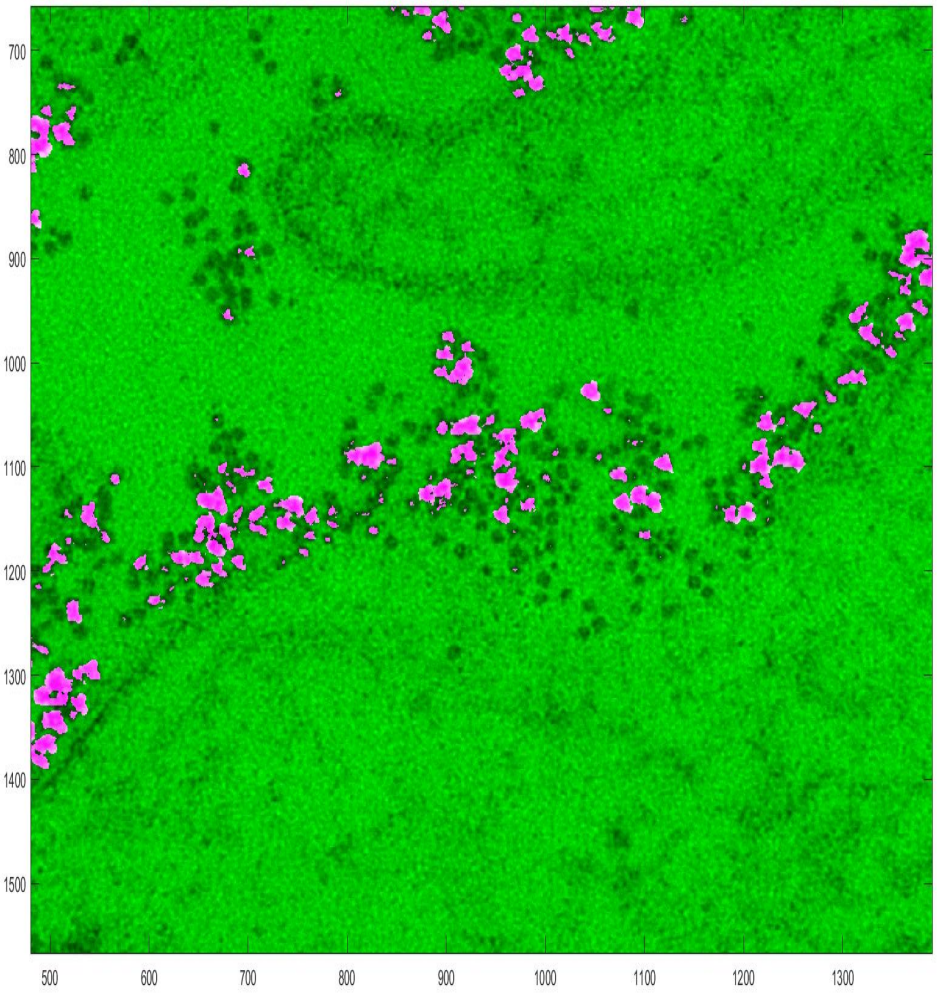

100
200
300
400
500
600
700
800
900
1200
1300
1400
1500
1600
1700
1800
1900
2000

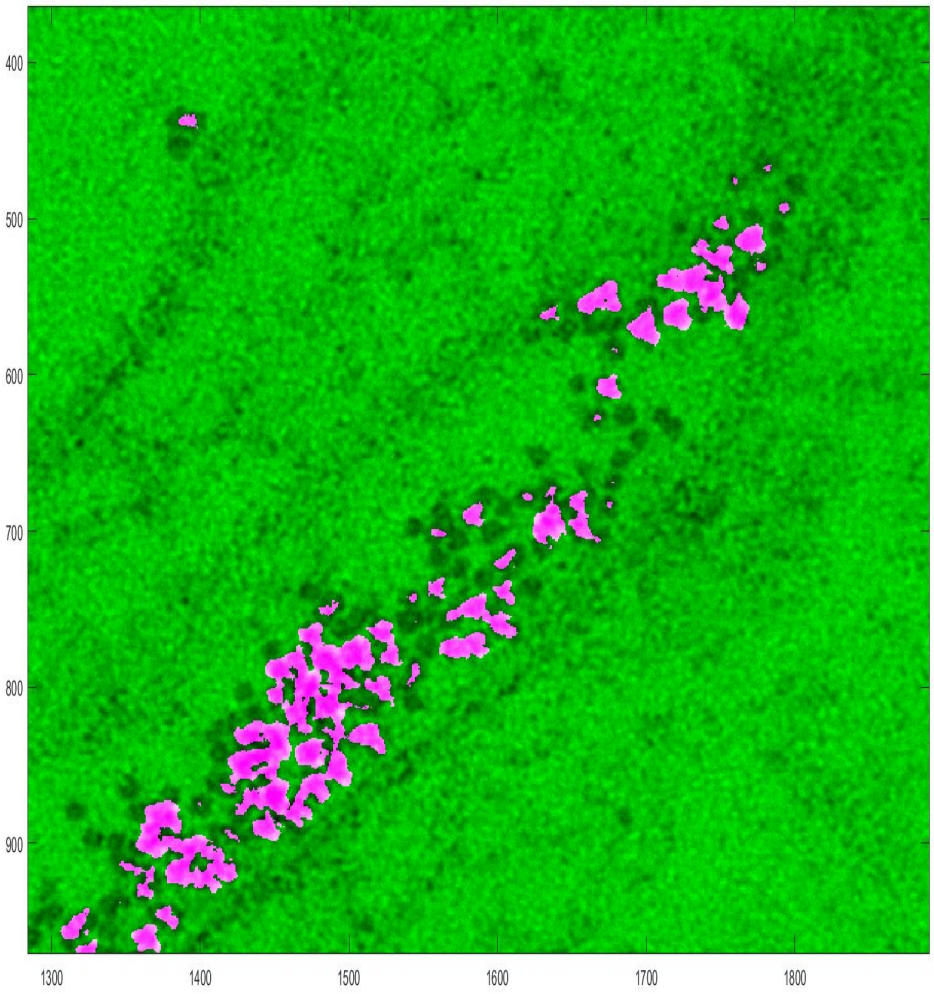

1200
1300
1400
1500
1600
1700
800
900
1000
1100
1200
1300


300
400
500
600
700
800
900
1000
1100
1000
1100
1200
1300
1400
1500
1600
1700
1800
1900


1000
1100
1200
1300
1400
1500
1600
1700
1800
400
500
600
700
800
900
1000
1100
1200

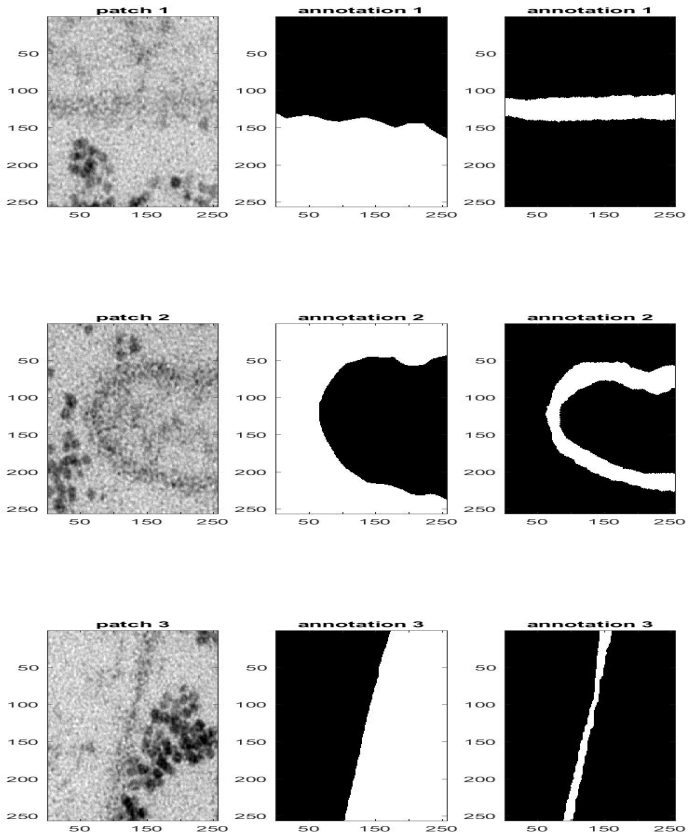

200
400
600
800
1000
1200
1400
1600
1800
2000
200
400
600
800
1000
1200
1400
1600
1800
2000

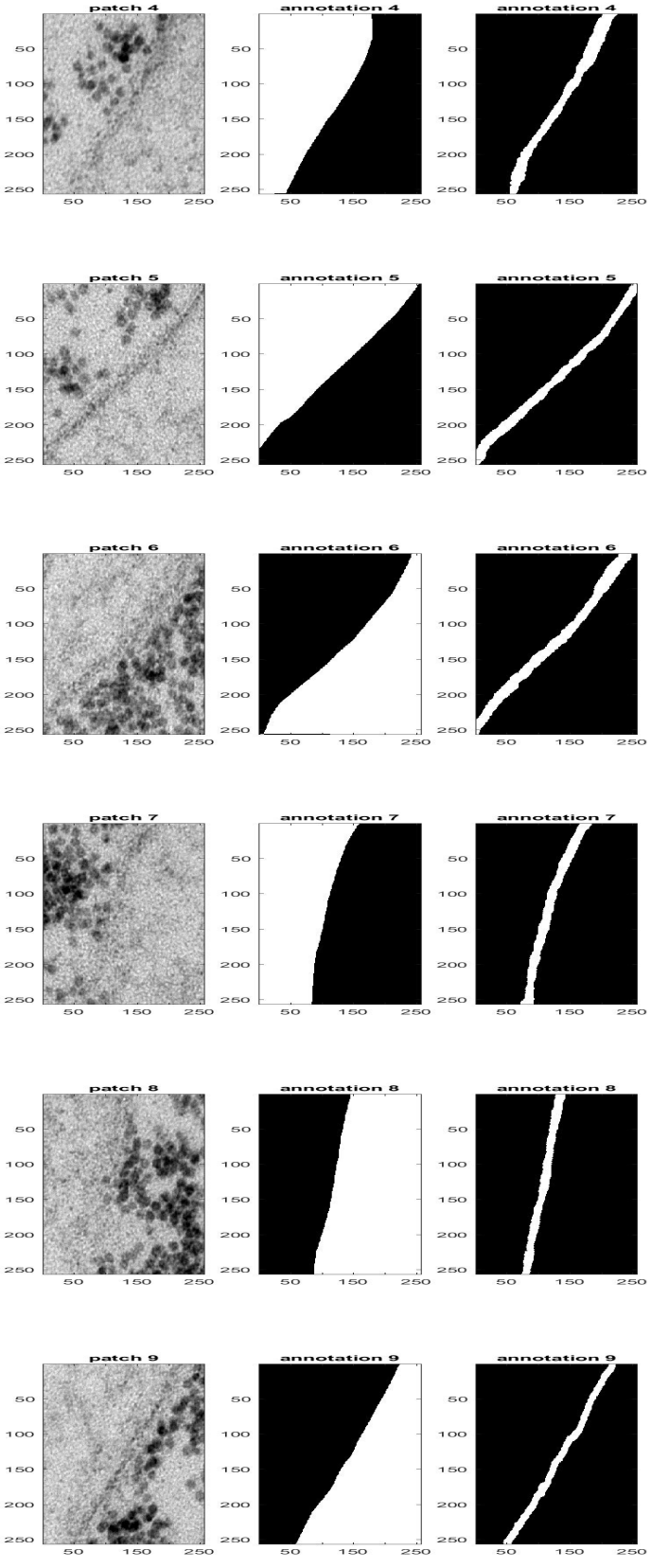

700
800
900
1000
1100
1200
1300
1400
1500
500
600
700
800
900
1000
1100
1200
1300

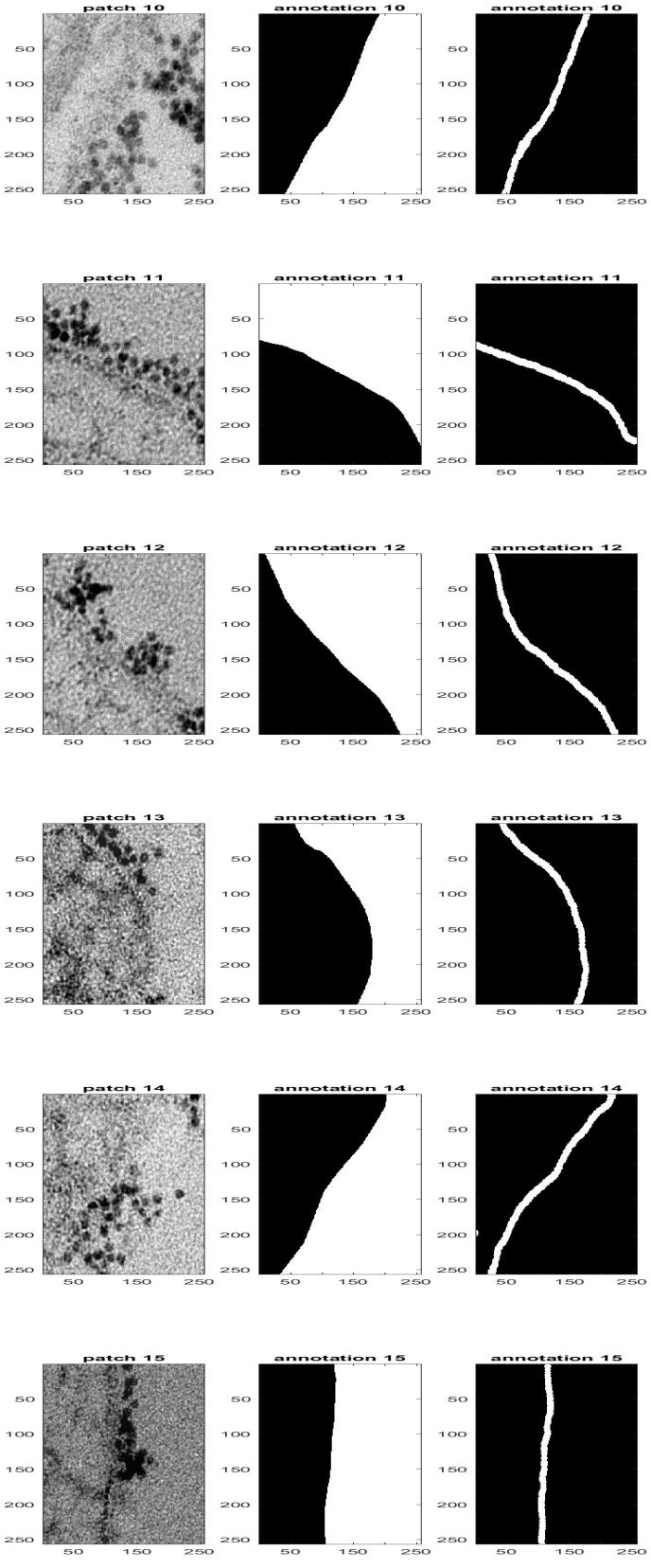

400
500
600
700
800
900
1300
1400
1500
1600
1700
1800

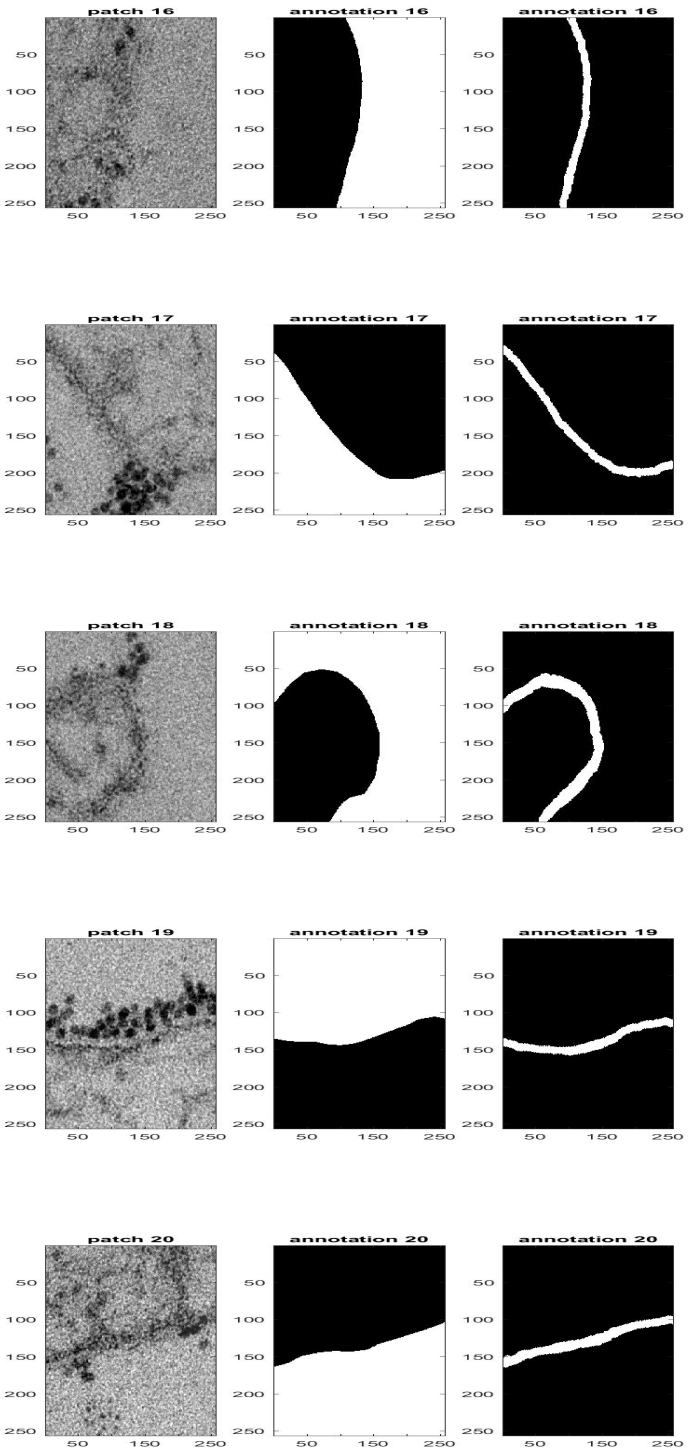
Appendix H

Patches used for boundary detection together two types of annotations.




patch 4
annotation 4
annotation 4
patch 5
annotation 5
annotation 5
patch 6
annotation 6
annotation 6
patch 7
annotation 7
annotation 7
patch 8
annotation 8
annotation 8
patch 9
annotation 9
annotation 9
50
100
150
200
250
50
150
250

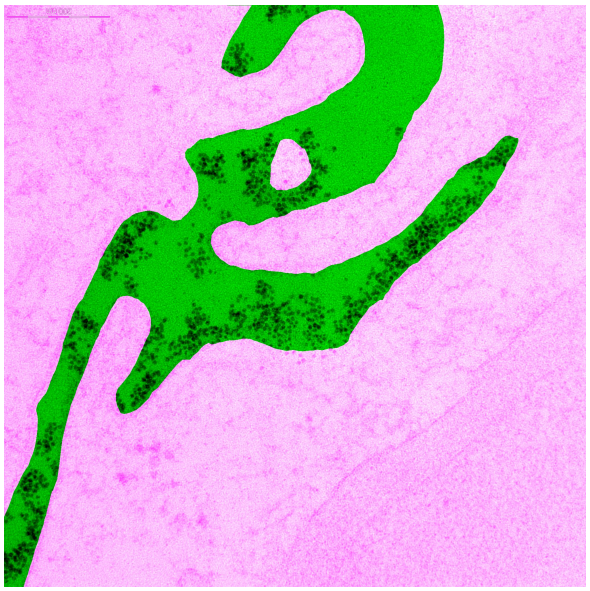

patch 10
annotation 10
annotation 10
patch 11
annotation 11
annotation 11
patch 12
annotation 12
annotation 12
patch 13
annotation 13
annotation 13
patch 14
annotation 14
annotation 14
patch 15
annotation 15
annotation 15
50
100
150
200
250
50
150
250

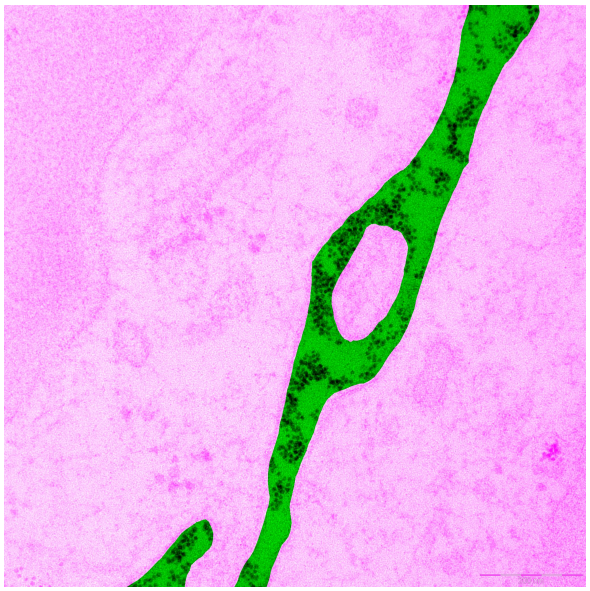

patch 16
annotation 16
annotation 16
patch 17
annotation 17
annotation 17
patch 18
annotation 18
annotation 18
patch 19
annotation 19
annotation 19
patch 20
annotation 20
annotation 20
50
100
150
200
250
50
150
250

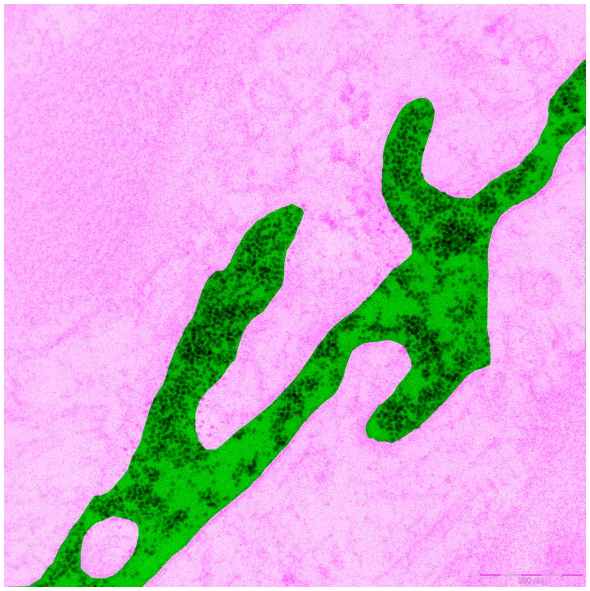
Appendix I

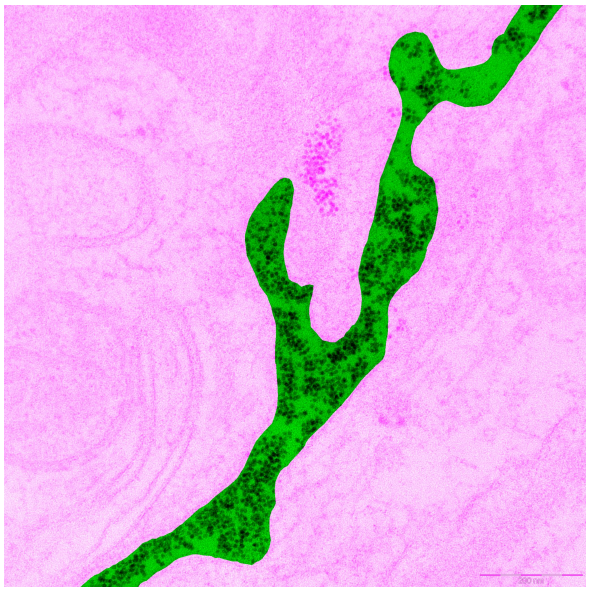
Tissue area annotation for the images in dataset

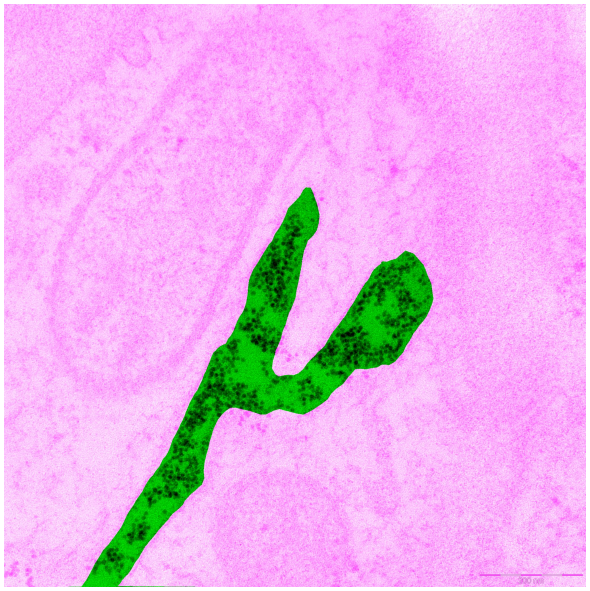
Ferritin molescules detection and Spatial Point Statistics at Chapter 5 is only applied in the areas marked as green on the annotations shown in this appendix.

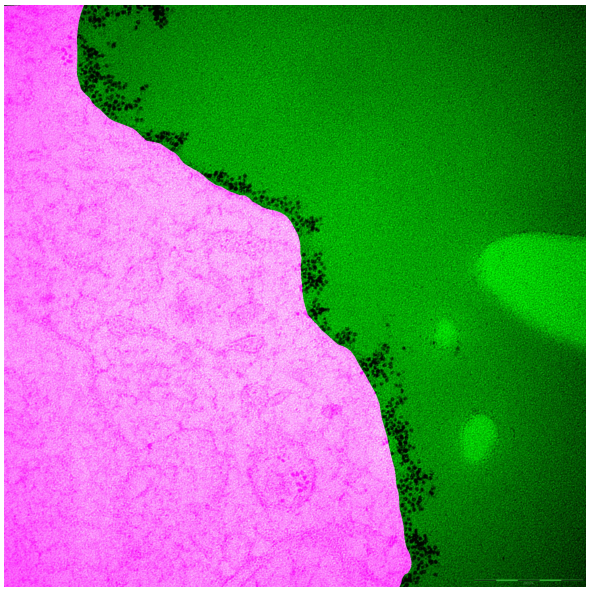


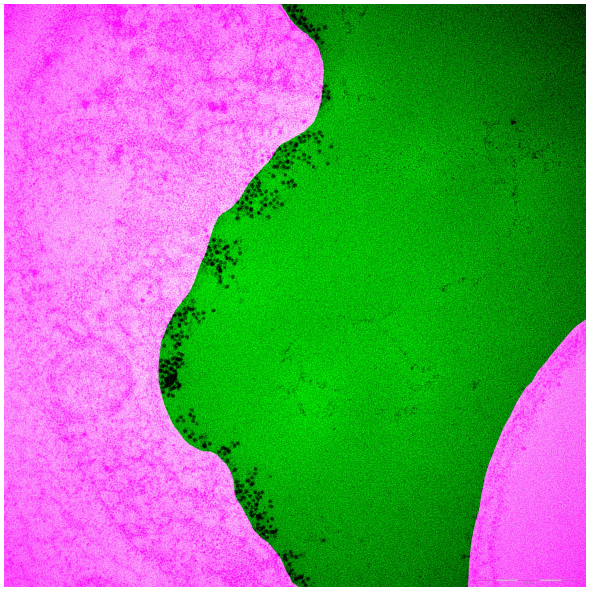


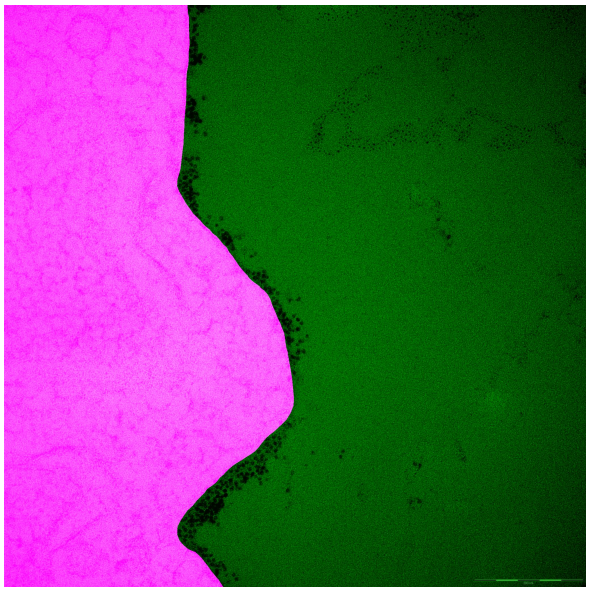


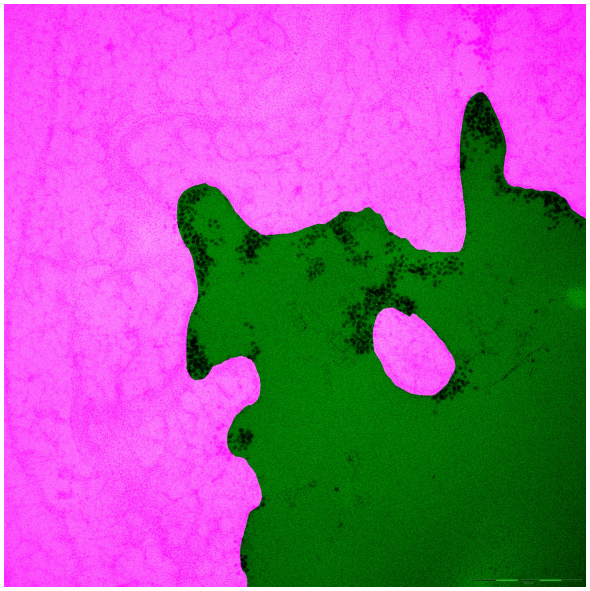


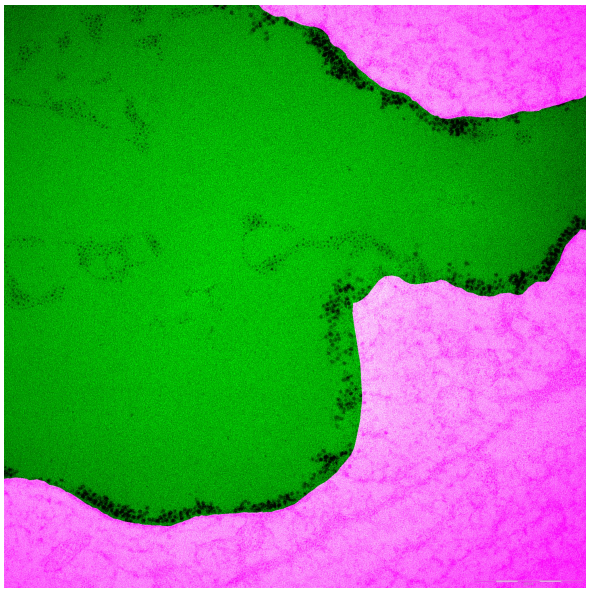








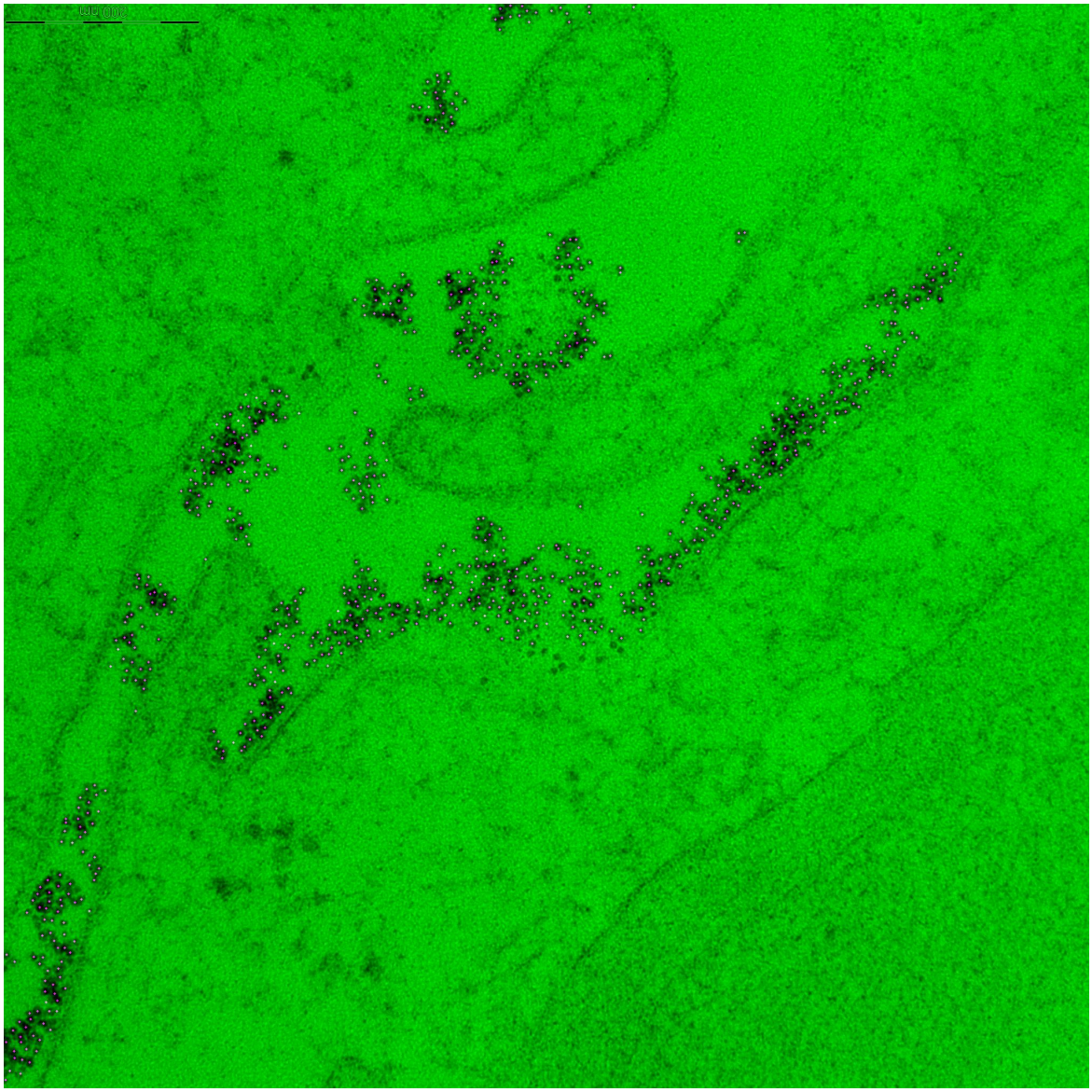


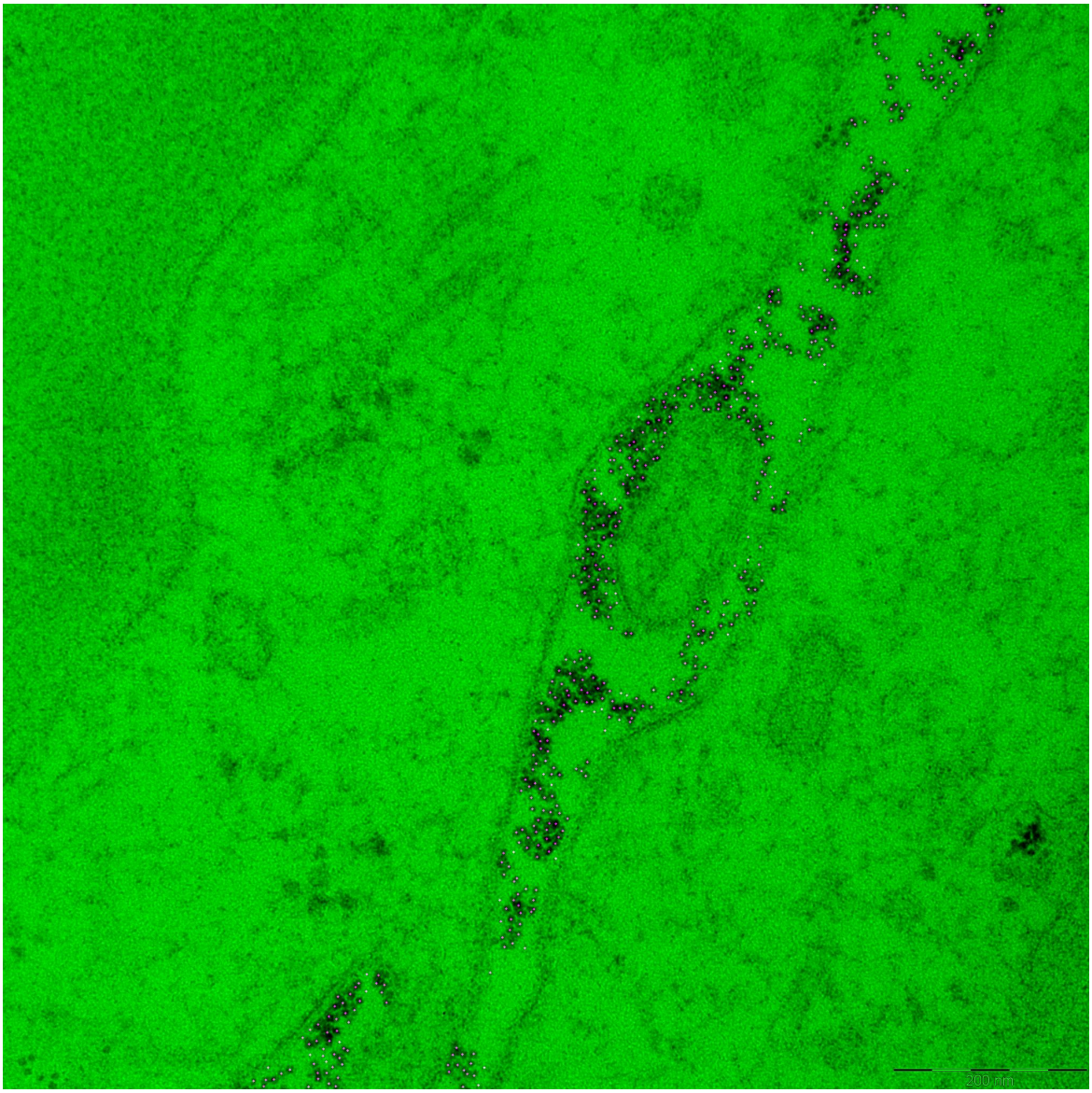


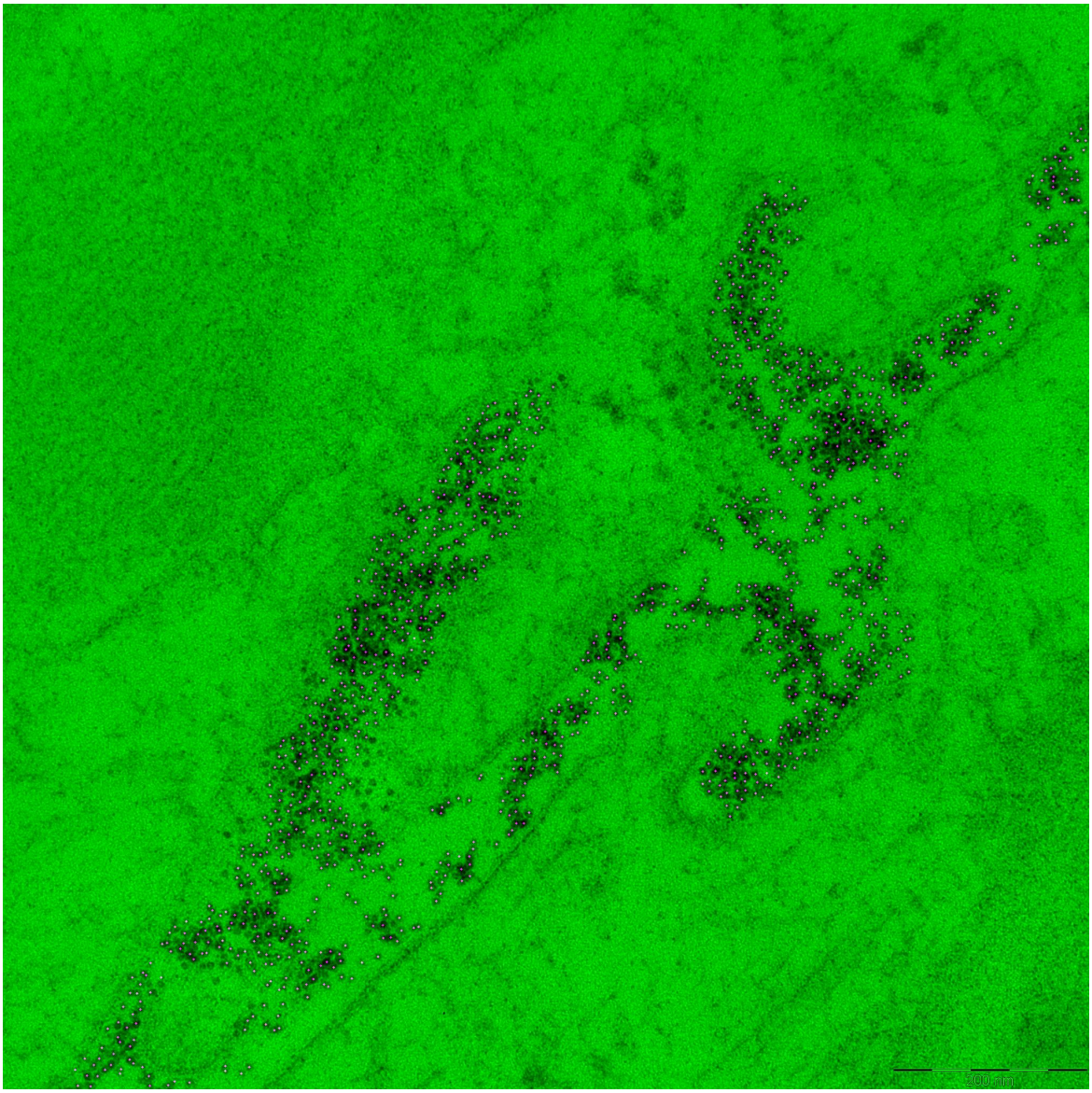
Appendix J

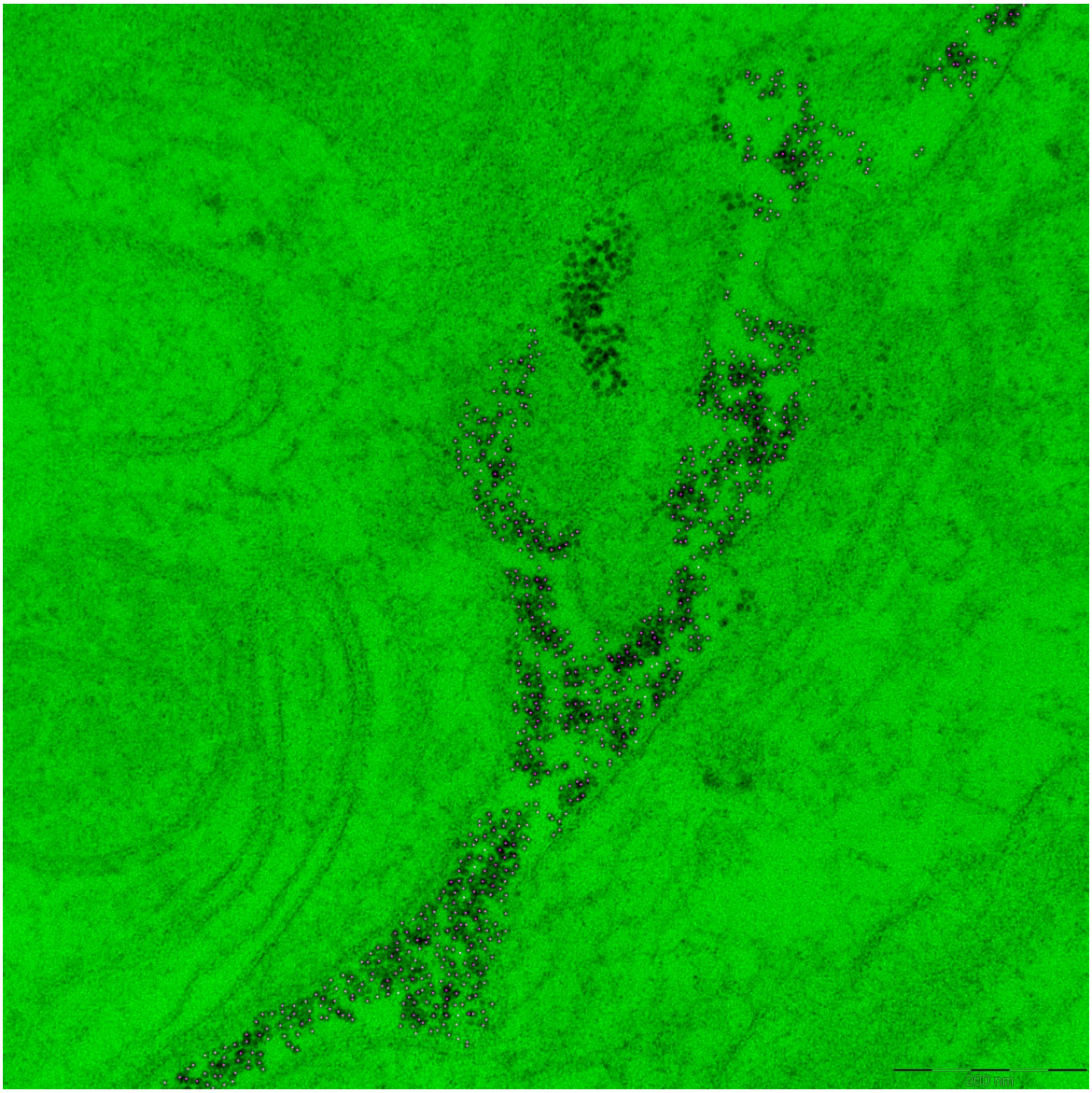
Ferritin molecules centers detection

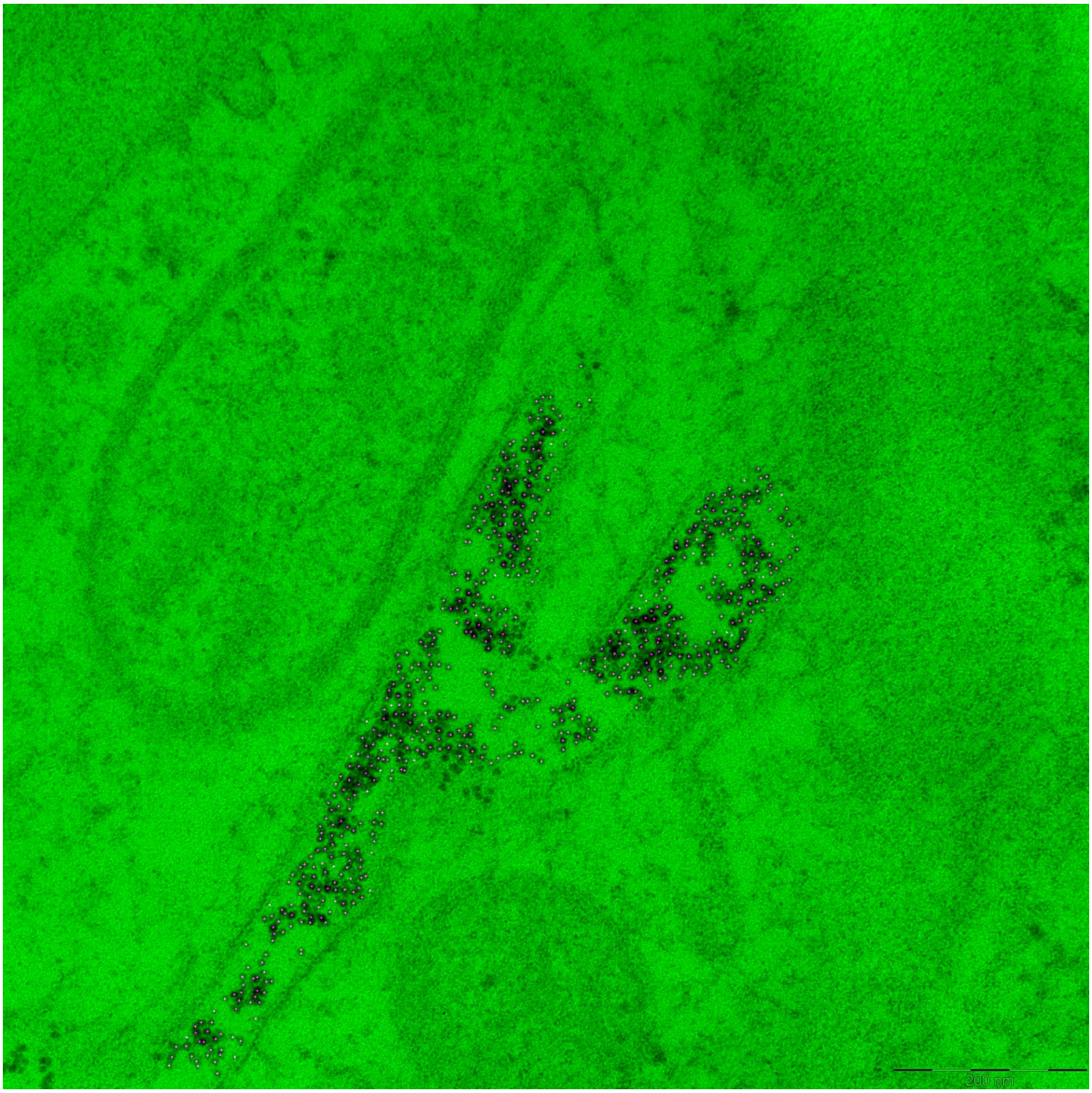
The results for using Mahalanobis Classification and Ferritin Molecules centers identification for each image in the dataset. Only the interior of the blood vessels is considered for the detection, as marked in (Appendix 9). The pixel representing the center for each molecule is dilated so it will be more visible.

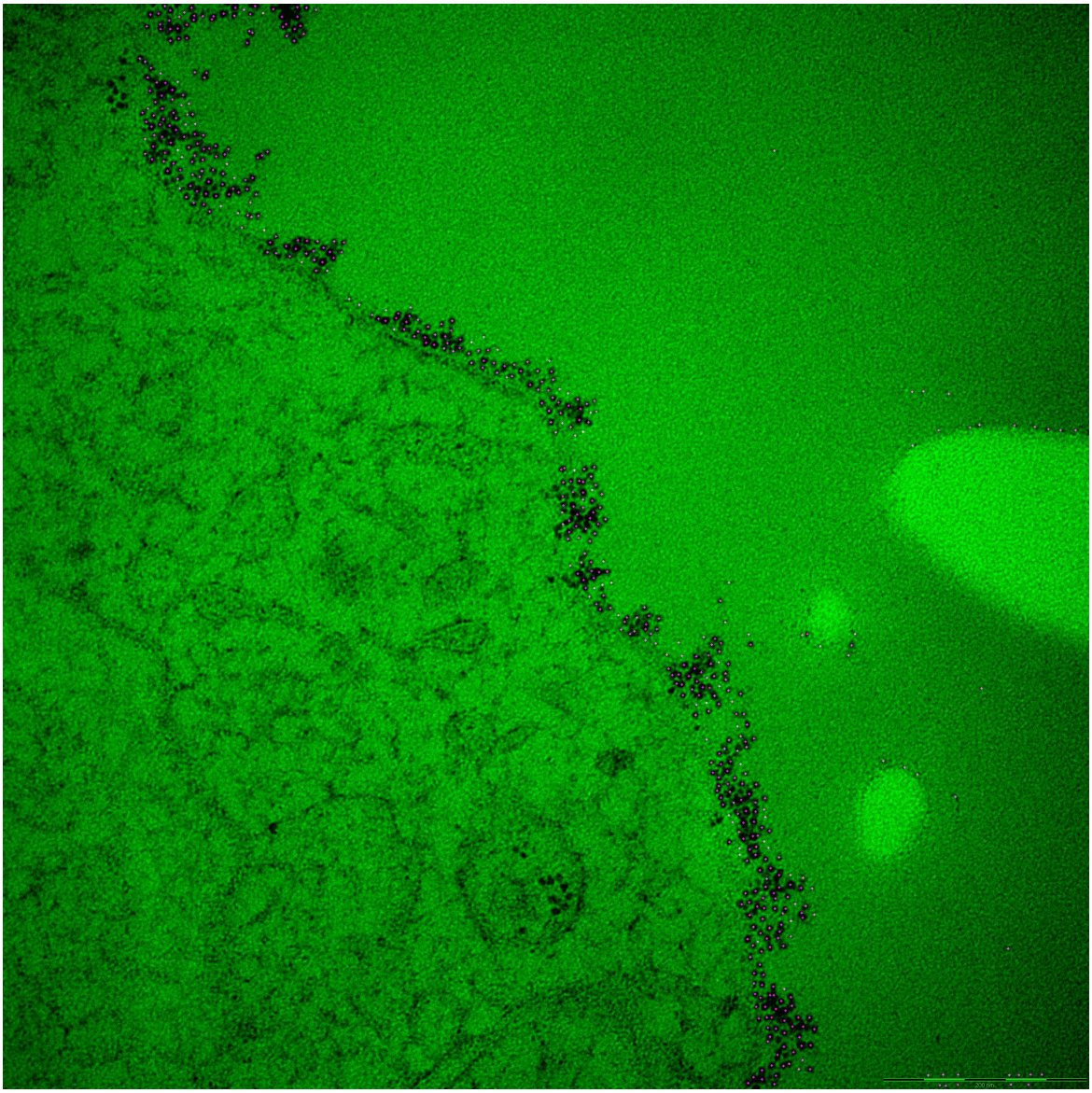


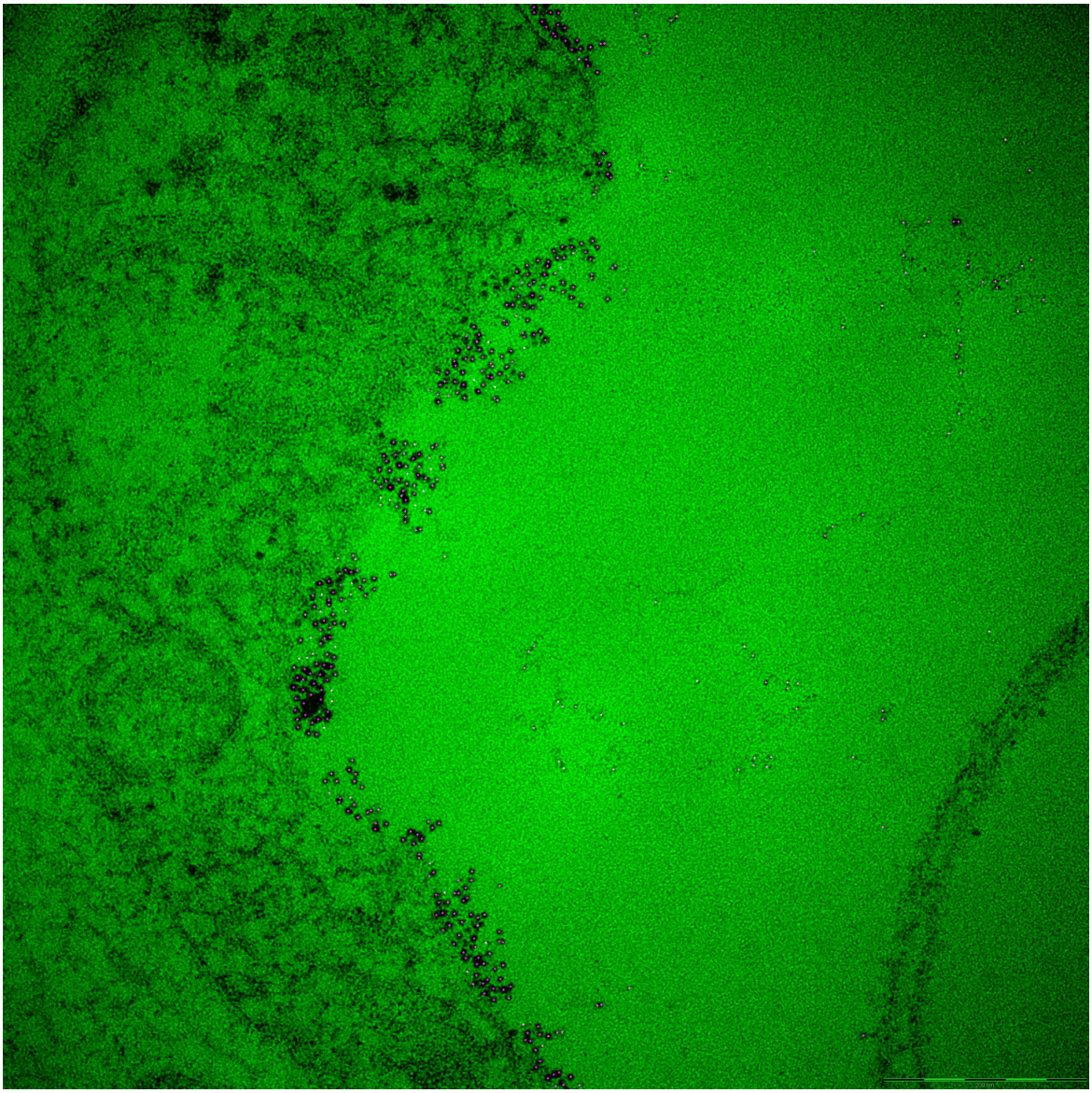

200 nm

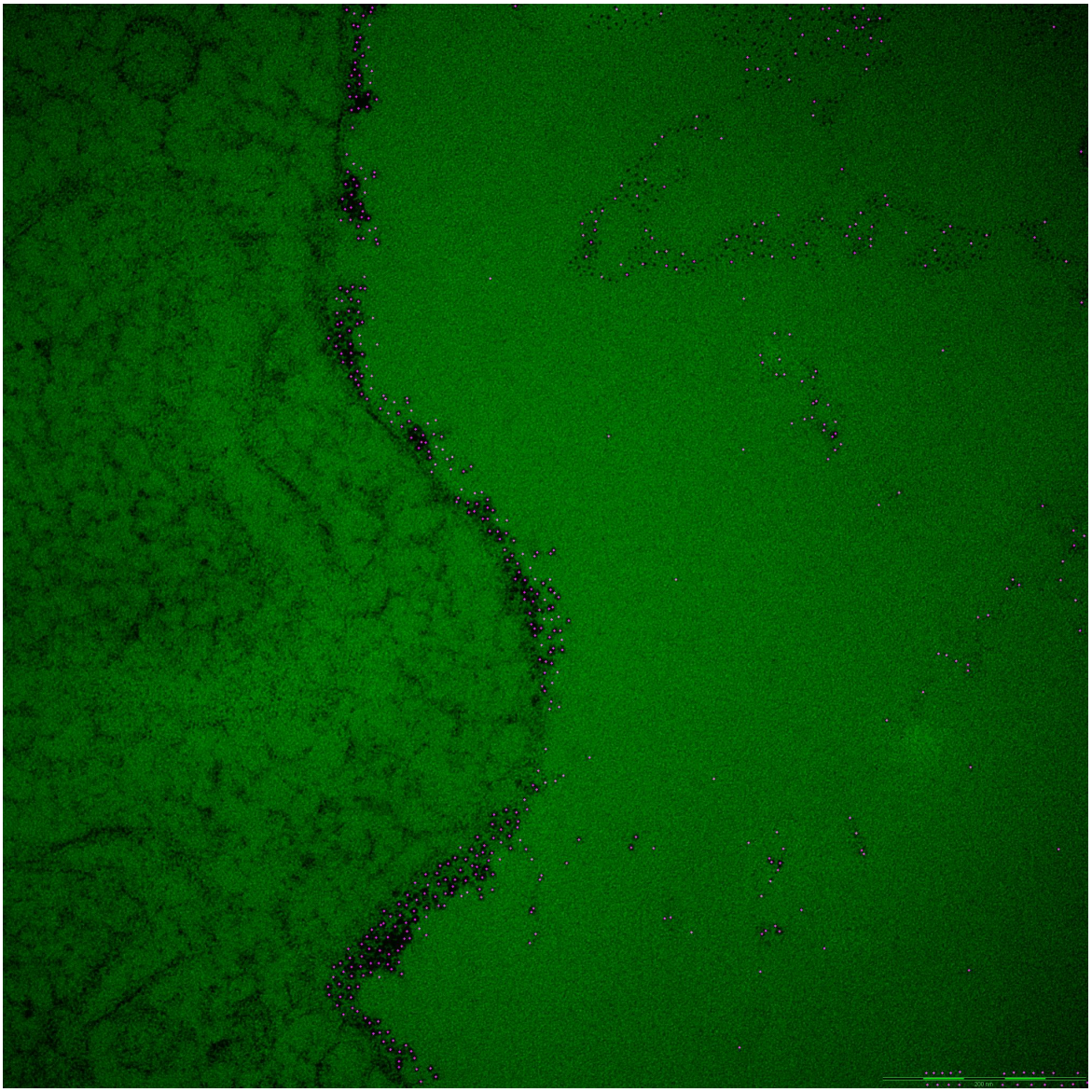

200 nm

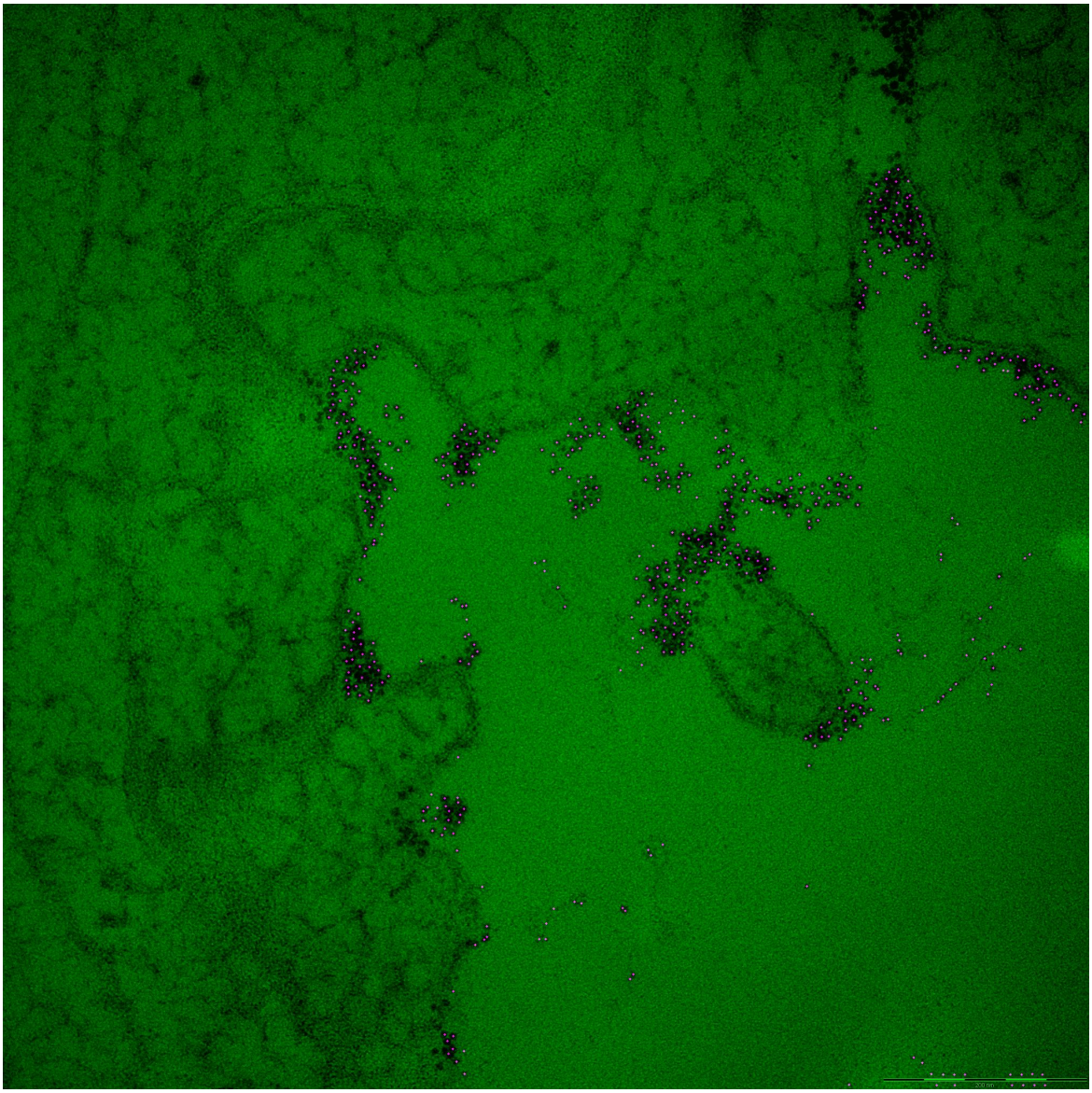

200 nm

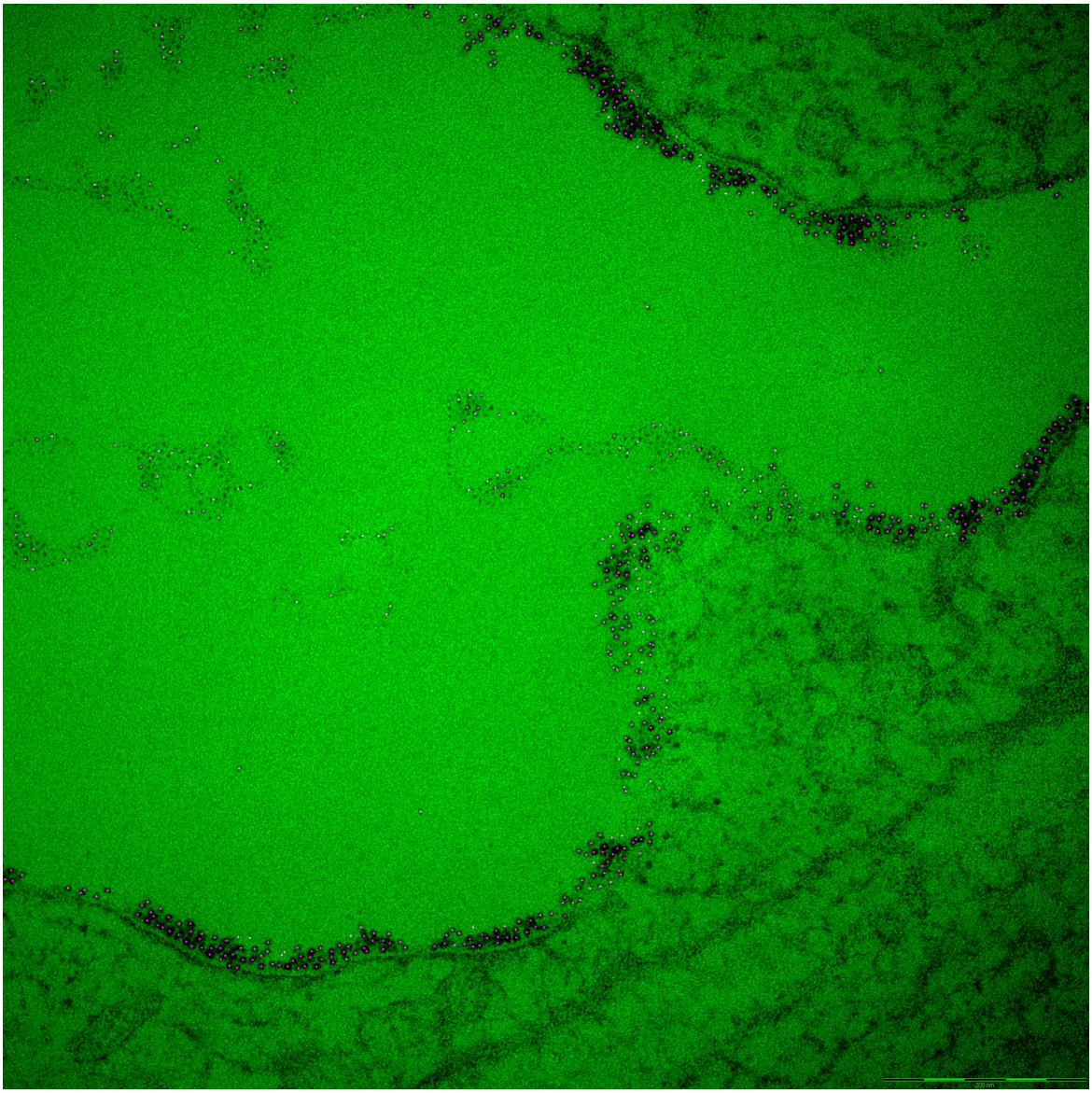

200 nm









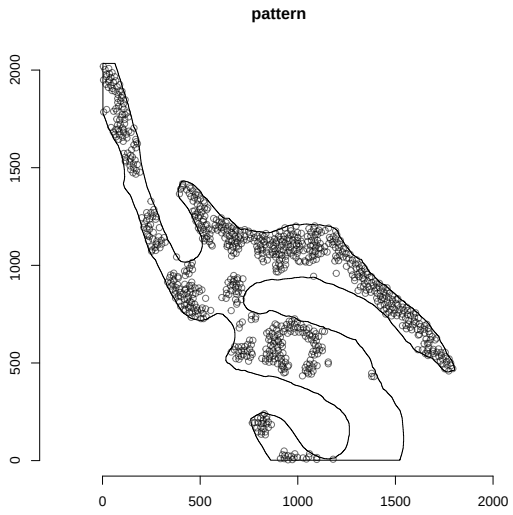


Appendix K

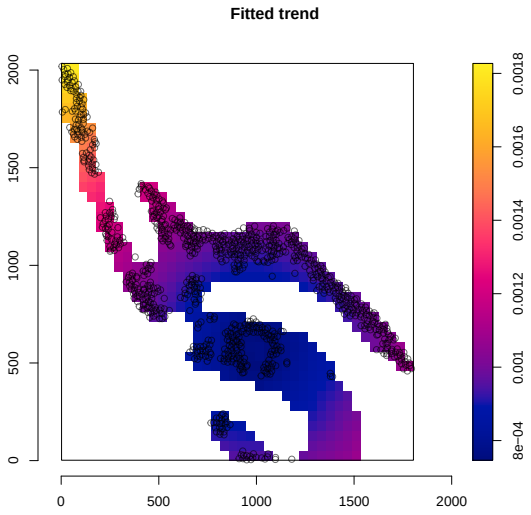
Ferritin molecules intensity description

This Appendix displays the results obtained from Chapter 5 on analyzing the spatial distribution of Ferritin Molecules in the blood vessels in each image from Appendix K.

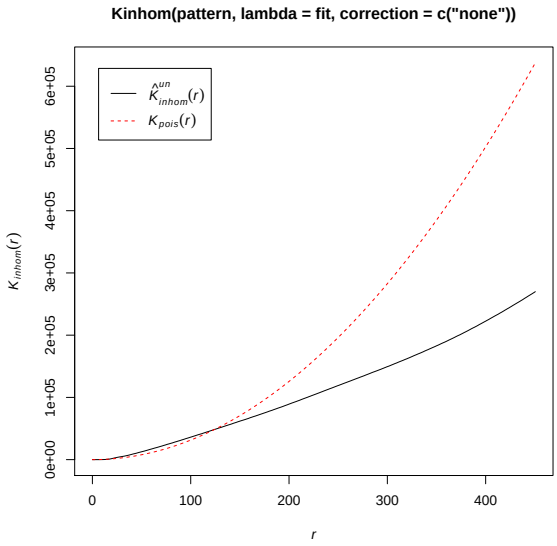
K.1 Image 0002



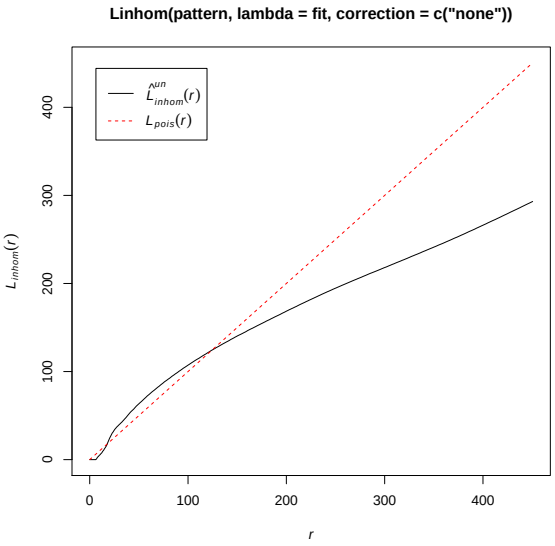
(a) Pattern for image 0002.



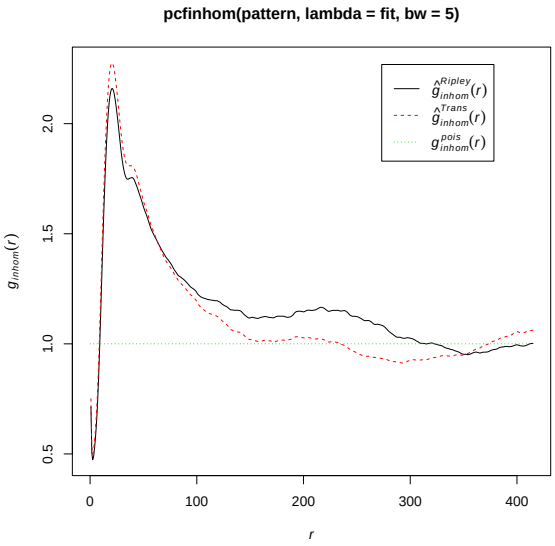
(b) Intensity function for image 0002.



(a) Inhomogeneous K function.

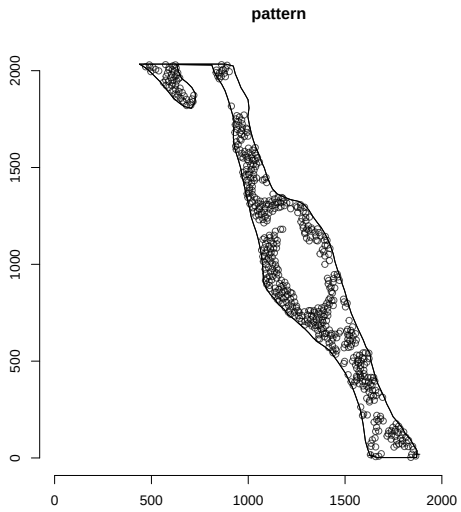


(b) Inhomogeneous L function.

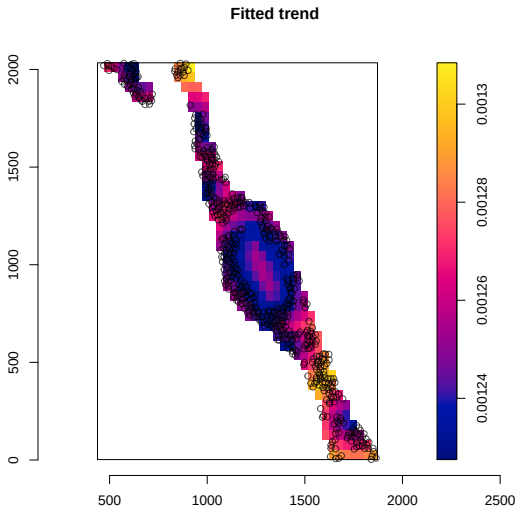


(c) Inhomogeneous pair correlation function.

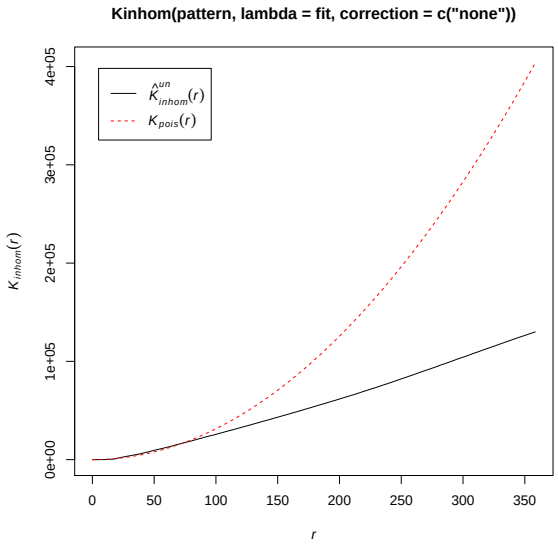
K.2 Image 0003



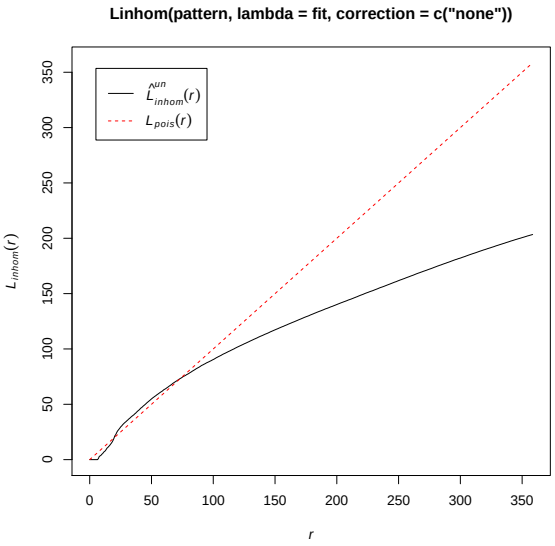
(a) Pattern for image 0003.



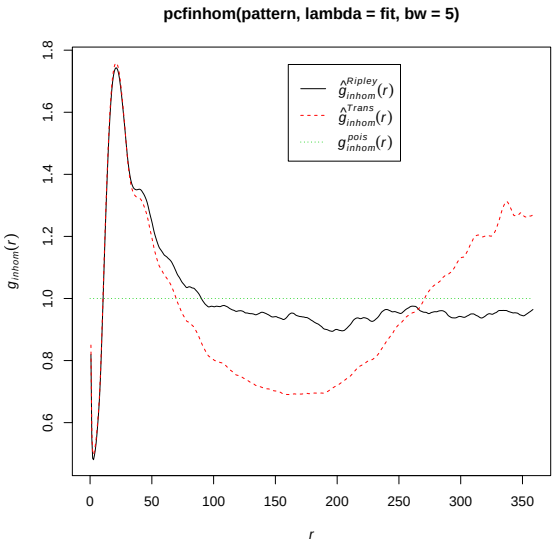
(b) Intensity function for image 0003.



(a) Inhomogeneous K function.

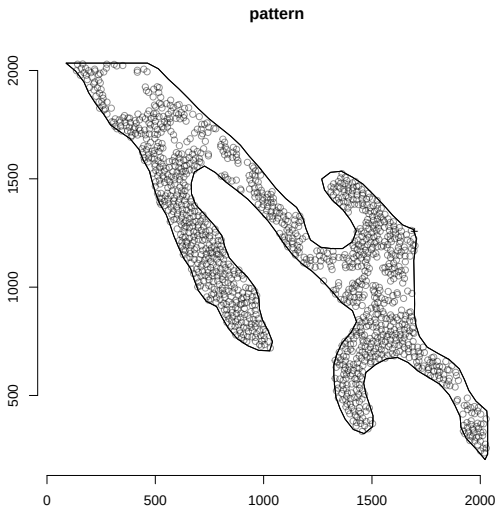


(b) Inhomogeneous L function.

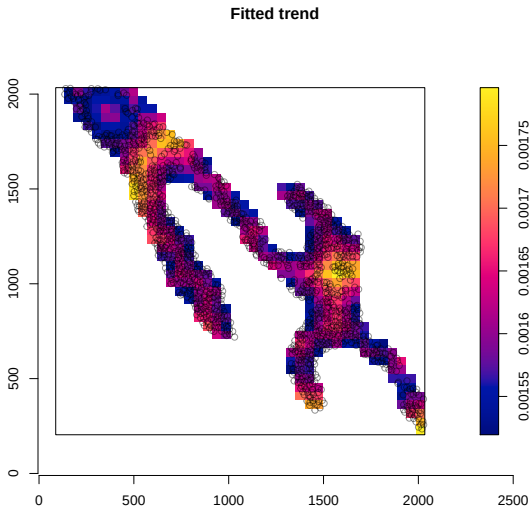


(c) Inhomogeneous pair correlation function.

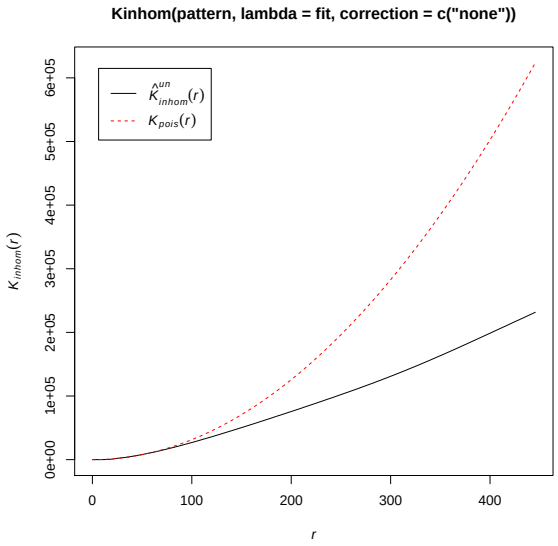
K.3 Image 0004



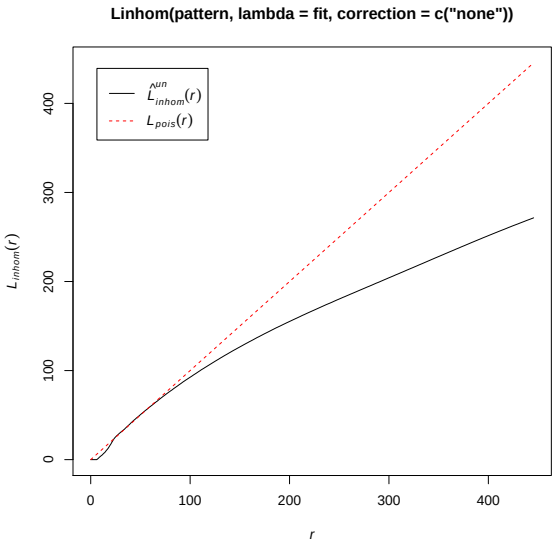
(a) Pattern for image 0004.



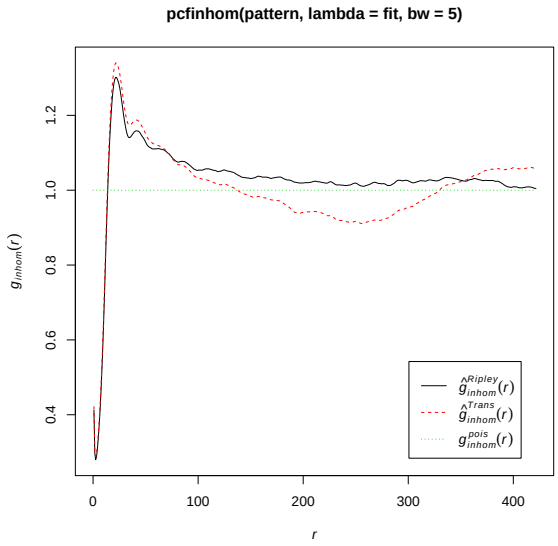
(b) Intensity function for image 0004.



(a) Inhomogeneous K function.

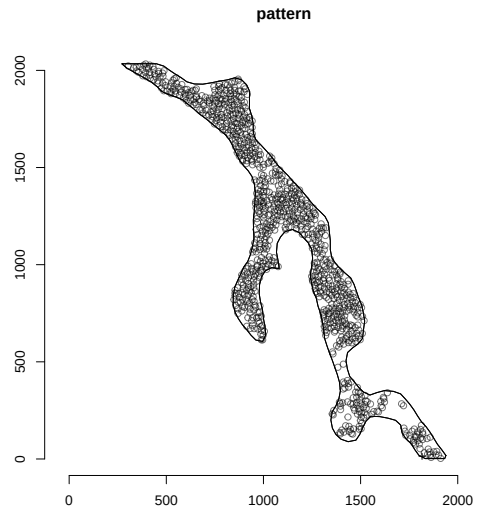


(b) Inhomogeneous L function.

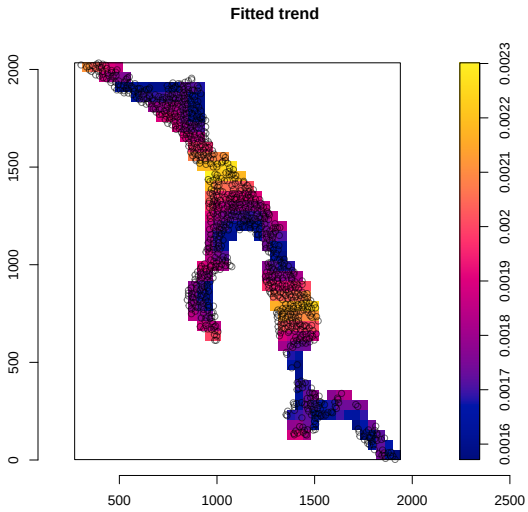


(c) Inhomogeneous pair correlation function.

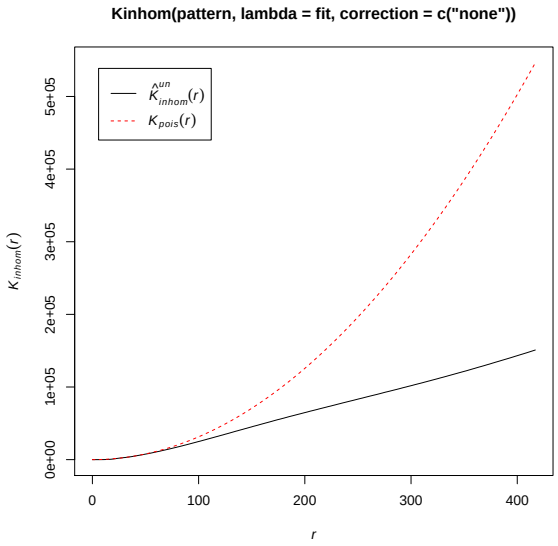
K.4 Image 0005



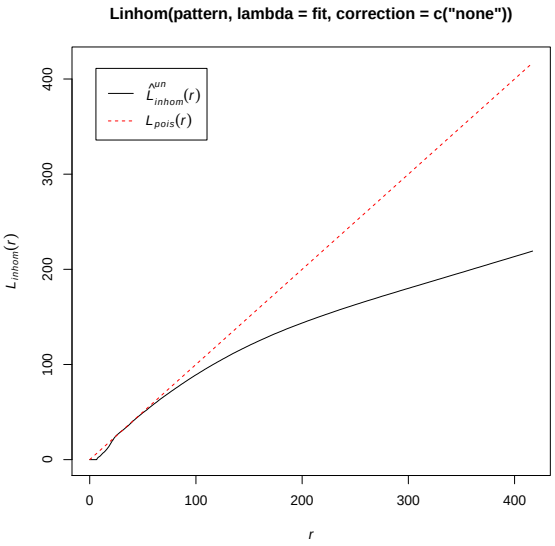
(a) Pattern for image 0005.



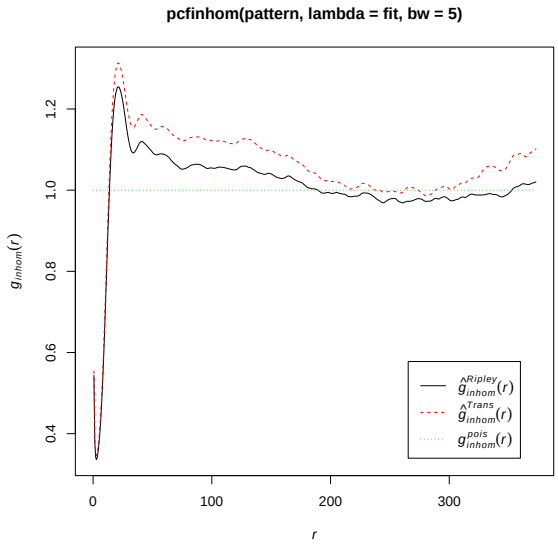
(b) Intensity function for image 0005.



(a) Inhomogeneous K function.

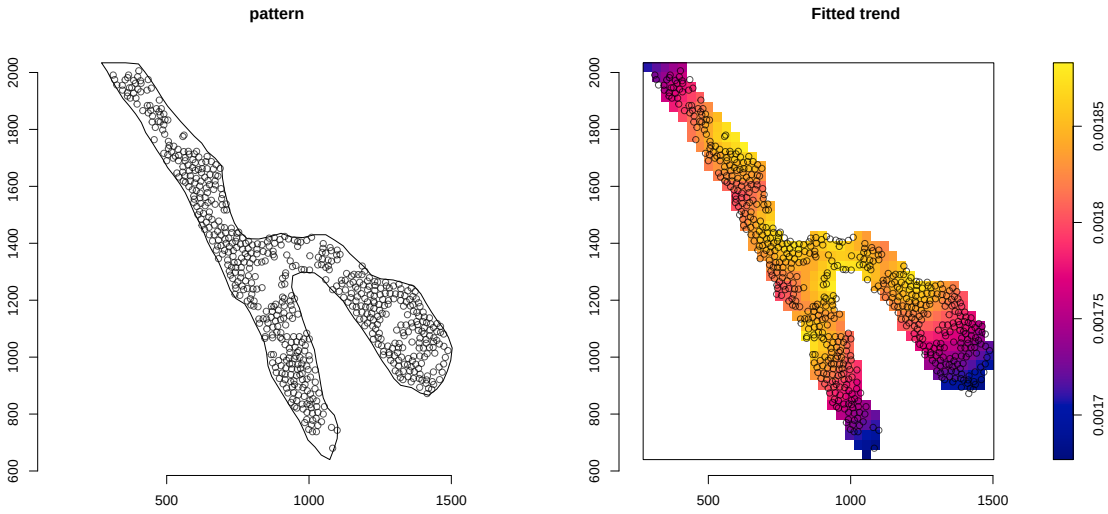


(b) Inhomogeneous L function.



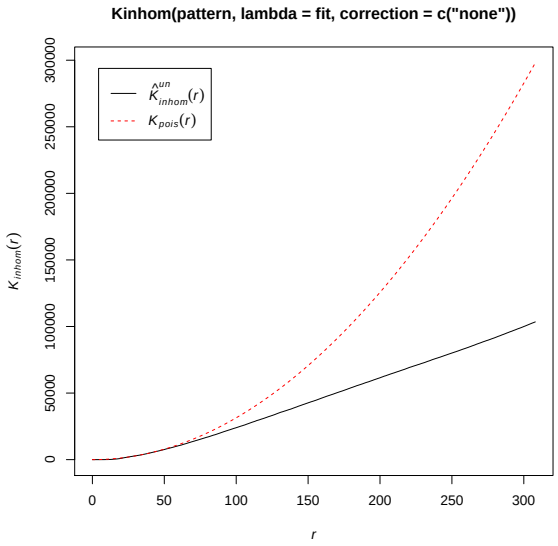
(c) Inhomogeneous pair correlation function.

K.5 Image 0006

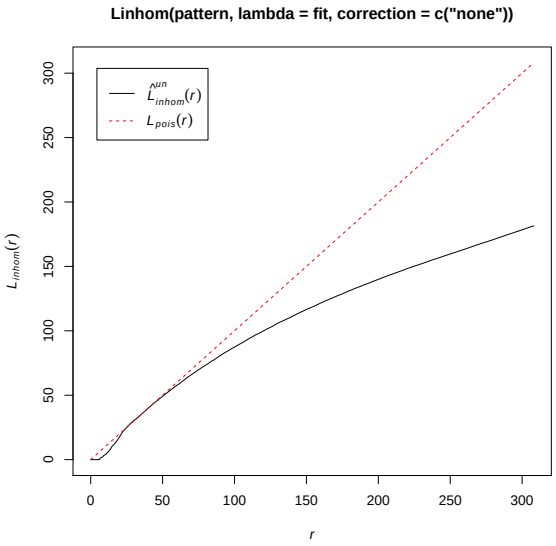


(a) Pattern for image 0006.

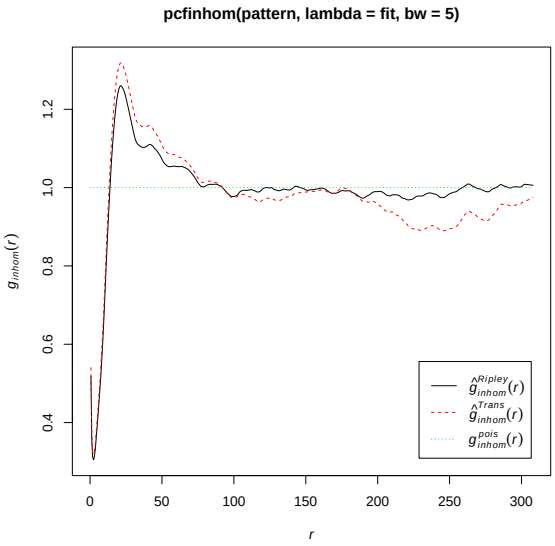
(b) Intensity function for image 00056.



(a) Inhomogeneous K function.

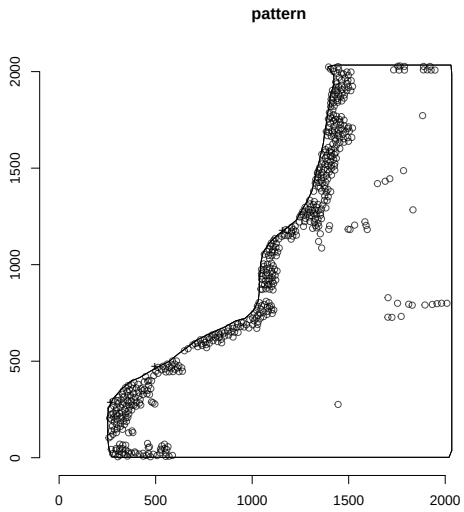


(b) Inhomogeneous L function.

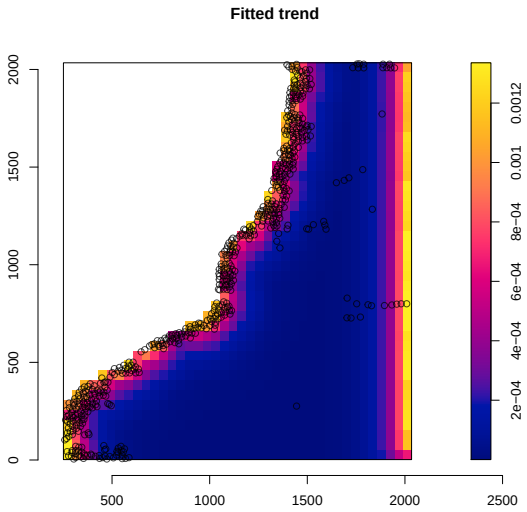


(c) Inhomogeneous pair correlation function.

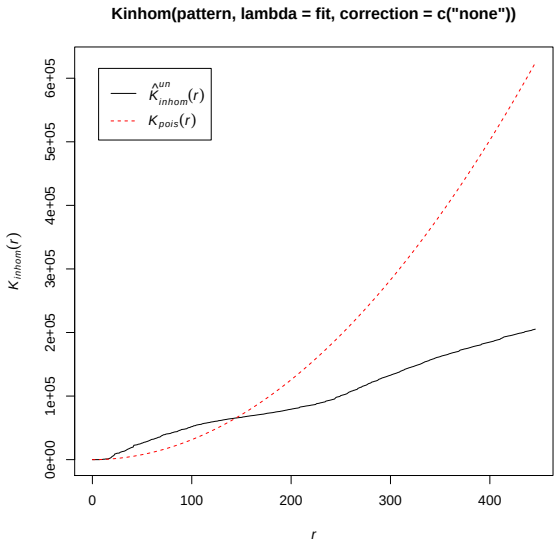
K.6 Image 0008



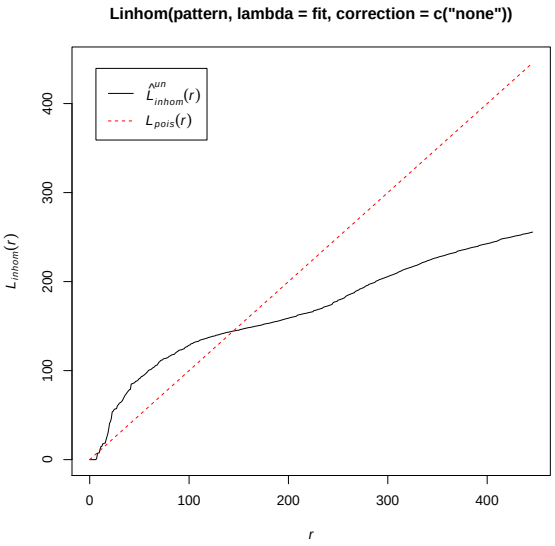
(a) Pattern for image 0008.



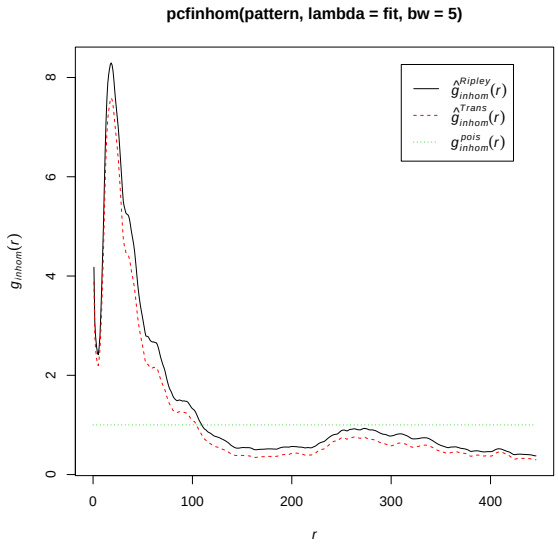
(b) Intensity function for image 0008.



(a) Inhomogeneous K function.

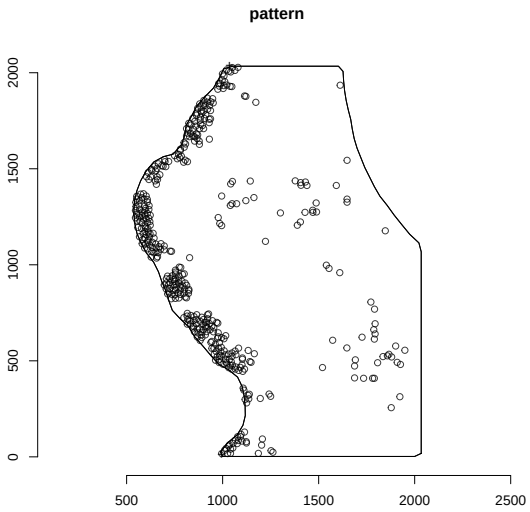


(b) Inhomogeneous L function.

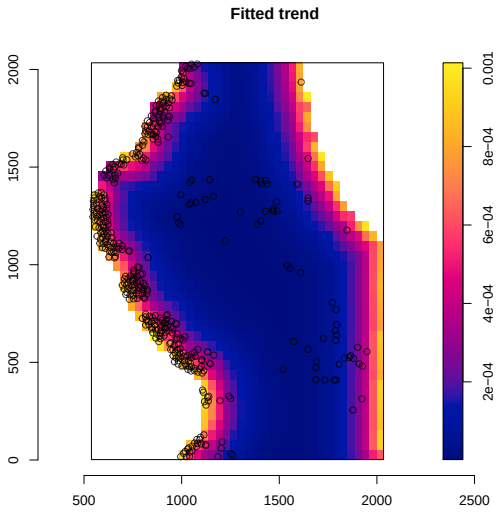


(c) Inhomogeneous pair correlation function.

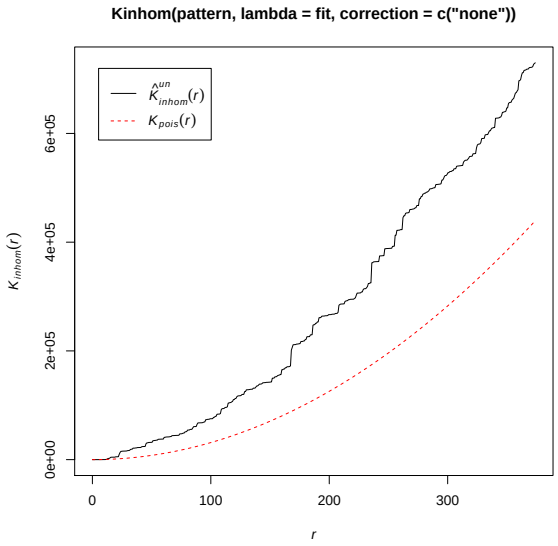
K.7 Image 0016



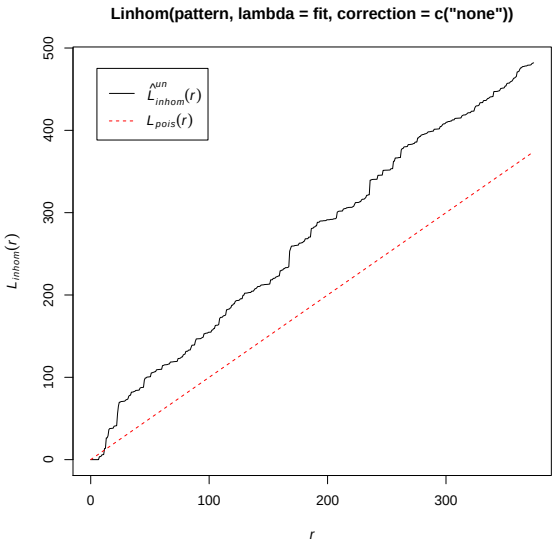
(a) Pattern for image 0016.



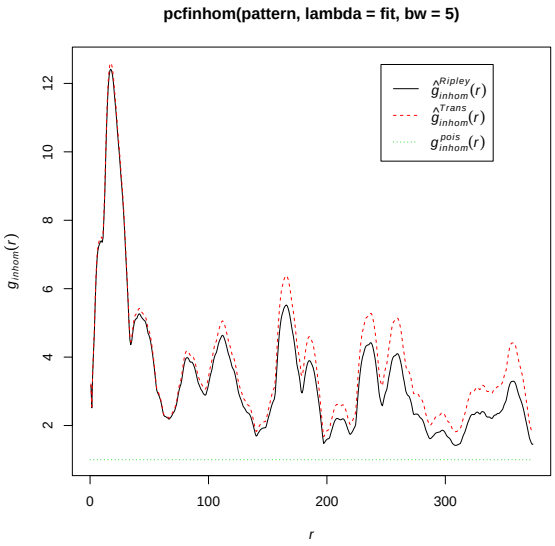
(b) Intensity function for image 0016.



(a) Inhomogeneous K function.

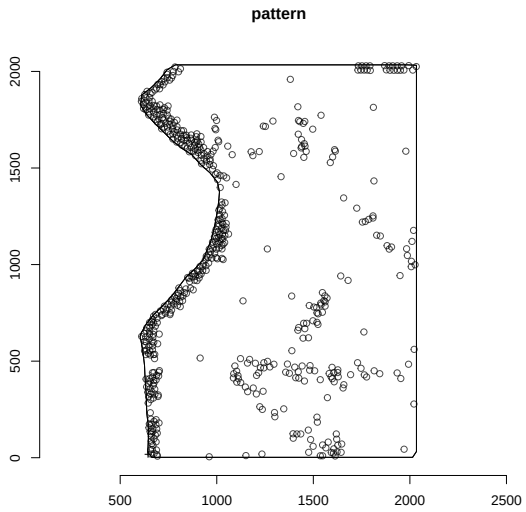


(b) Inhomogeneous L function.

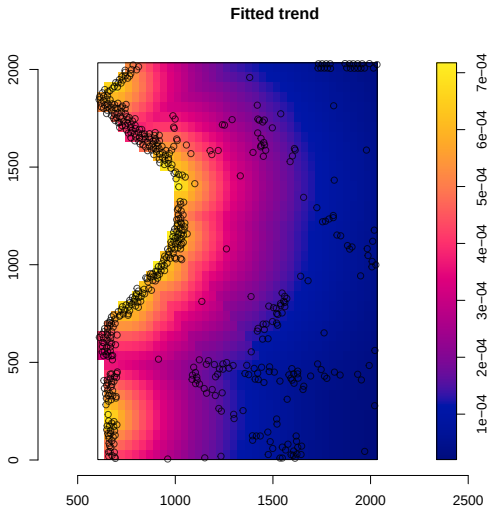


(c) Inhomogeneous pair correlation function.

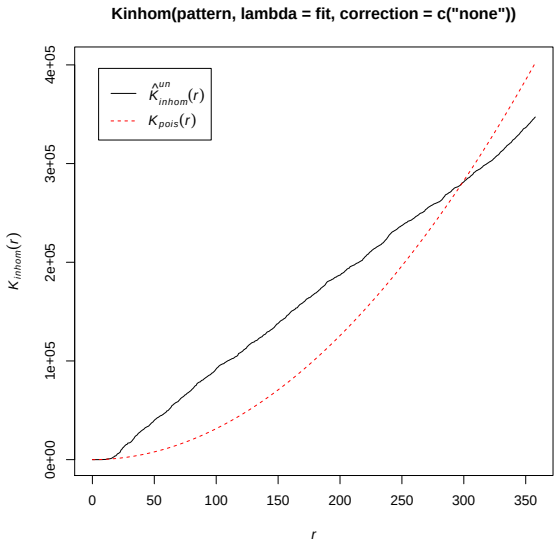
K.8 Image 0019



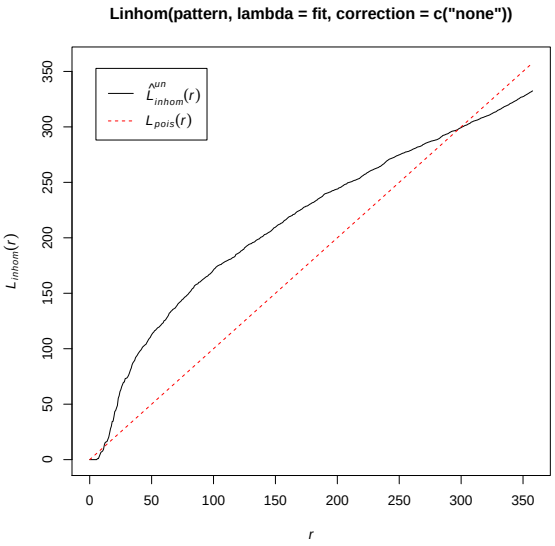
(a) Pattern for image 0019.



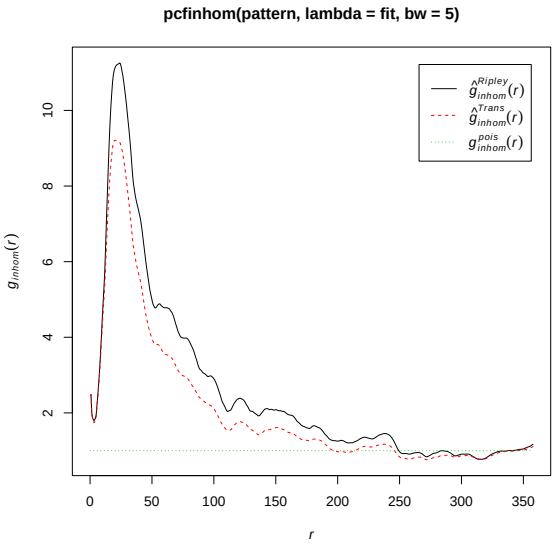
(b) Intensity function for image 0019.



(a) Inhomogeneous K function.

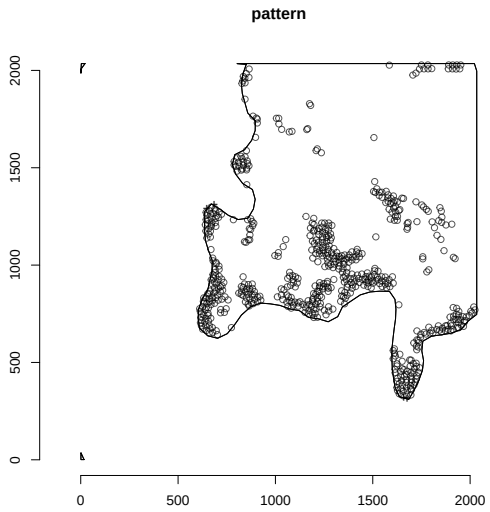


(b) Inhomogeneous L function.

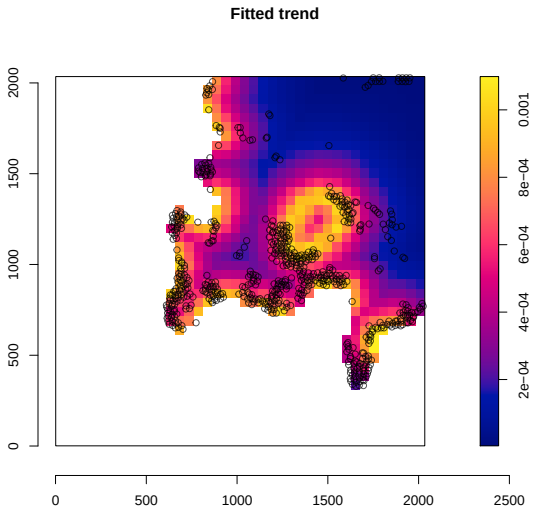


(c) Inhomogeneous pair correlation function.

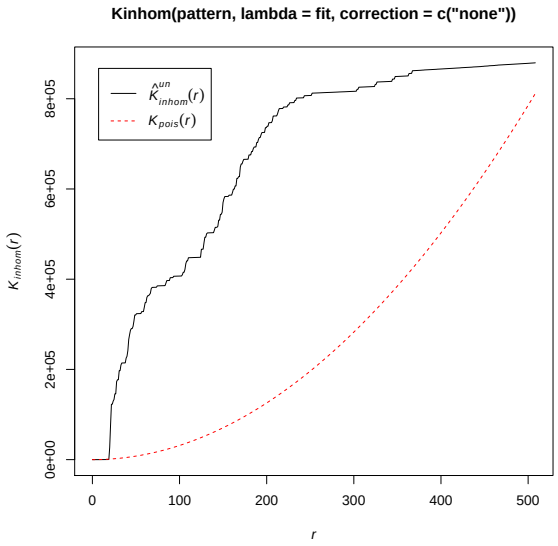
K.9 Image 0020



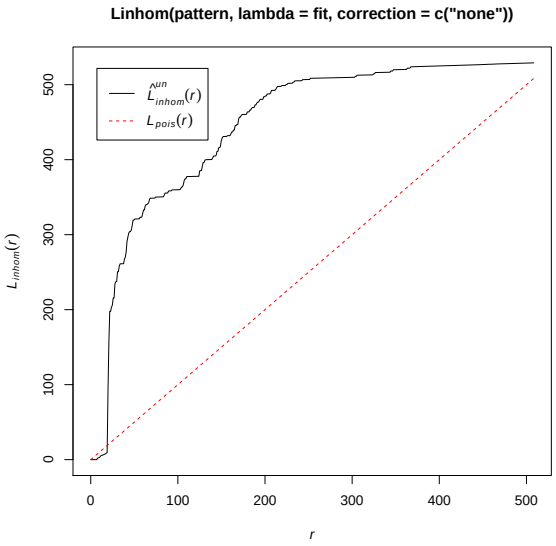
(a) Pattern for image 0020.



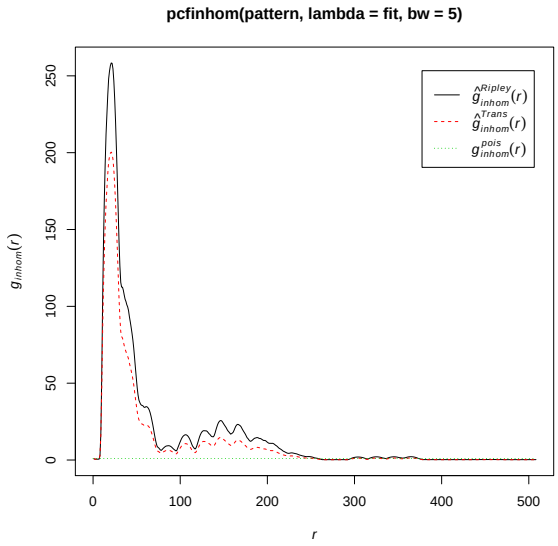
(b) Intensity function for image 0020.



(a) Inhomogeneous K function.

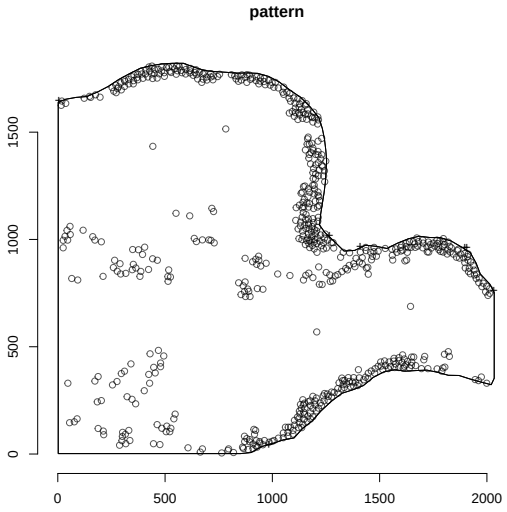


(b) Inhomogeneous L function.

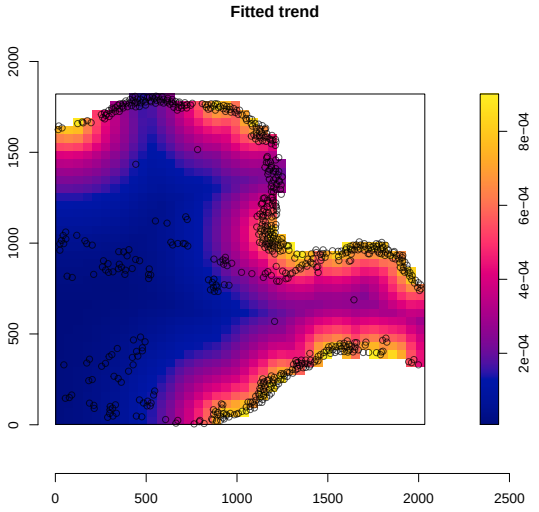


(c) Inhomogeneous pair correlation function.

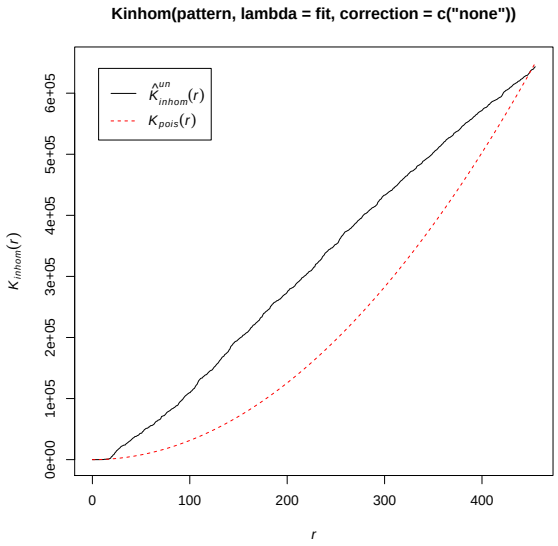
K.10 Image 0021



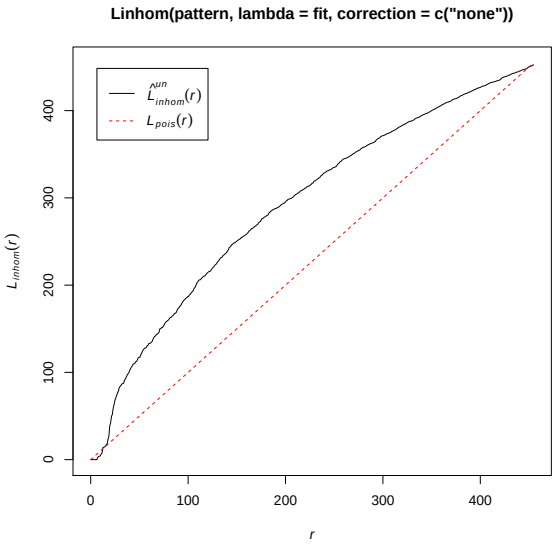
(a) Pattern for image 0021.



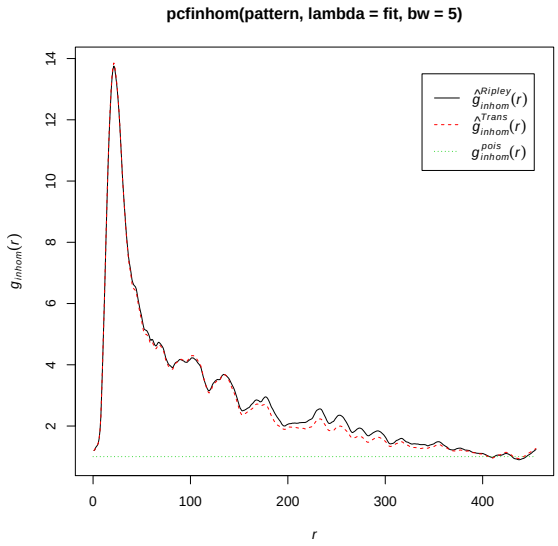
(b) Intensity function for image 0021.



(a) Inhomogeneous K function.



(b) Inhomogeneous L function.



(c) Inhomogeneous pair correlation function.