Image Analysis for the Mouse Brain Architecture Project

A Dissertation presented

by

Angeliki Pollatou

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Department of Physics and Astronomy

Stony Brook University

(include this copyright page only if you are selecting copyright through ProQuest, which is optional)

Copyright by
Your Name
2019

Stony Brook University

The Graduate School

Your Name

We, the dissertation committe for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation

Name - Dissertation Advisor
Include title and department

Name - Chairperson of Defense
Include title and department

Type the remaining committe members using the above format. Type each member's name, title and department. Always make sure the signature page is kept to one page in length.

Type the outside member's name last. Include discipline and affiliation.

This dissertation is accepted by the Graduate School

Dean of the Graduate School

Abstract of the Dissertation

Title of Dissertation

by

Your Name

Doctor of Philosophy

in

Full Name of Degree Program

(Concentration - optional)

Stony Brook University

2019

Begin typing abstract here.

Dedication Page

This page is optional.

Frontispiece

The frontispiece is generally an illustration, and is an optional page.

Contents

1 Introduction **1**
- 1.1 Biological Background . . . 1
- 1.2 The Mouse Brain Architecture Project . . . 2

2 Image Processing **8**
- 2.1 Background . . . 8
- 2.2 Smoothing Spatial Filters . . . 12
  - 2.2.1 Averaging filter . . . 12
  - 2.2.2 Gaussian Filter . . . 15
  - 2.2.3 Nonlinear Filters . . . 16
- 2.3 Sharpening and Edge Detection Spatial Filters . . . 16
  - 2.3.1 Roberts, Prewitt and Sobel filter . . . 17
  - 2.3.2 Canny operator . . . 19
- 2.4 Morphological operators . . . 19
  - 2.4.1 Erosion and dilation . . . 19
  - 2.4.2 Opening and closing . . . 21

3 Edge detection **29**
- 3.1 Median filter . . . 30
- 3.2 Sobel filter . . . 34
- 3.3 Mathematical morphology operations . . . 34

4 Automatic detection of out-of-focus images **54**
- 4.1 Introduction . . . 54
- 4.2 Out-of-focus image metric . . . 56
- 4.3 Results . . . 58
- 4.4 Visual inspections and algorithm comparison . . . 61
- 4.5 Applications of the method in other fields . . . 63
- 4.6 Discussion about the method . . . 64
- 4.7 Conclusion . . . 65

5 Striping removal **102**
- 5.1 Introduction . . . 102
- 5.2 Previous methods of removing striping effects . . . 102
- 5.3 Data description . . . 104
- 5.4 Proposed Destriping Method . . . 106

5.4.1 Background and Sample tissue image separation 107
5.4.2 Spatial location of stripes 107
5.4.3 Smoothing the stripes 108
5.4.4 Balancing the uneven illumination 109
5.4.5 Correction of panel intensity variations 110
5.5 Results 111
5.5.1 Visual inspection 111
5.5.2 Mean and standard deviation 112
5.5.3 Mean Relative Deviation 113
5.5.4 Structural Similarity Index 114
5.5.5 Image Focus 115
5.5.6 Other metrics 115
5.6 Discussion 116
5.7 Conclusion 118

6 Supporting information **136**

List of Figures

1 Human brain depiction. Credit: Medical Gallery of Blausen Medical 2014. 1
2 Neuron structure (from [19]). 2
3 Structure of a synaptic connection(from [19]). 3
4 Microscopic image of a neuron in the brain of a living mouse. Fluorescent probes are use to brighten up the synapses, the red dots denote inhibitory synapse while the green dots mark excitatory synapses. Image adapted from [22]. 4
5 Example of an F image from the MBA project. The image depicts section 32 from stack 2941. 6
6 Example of an N image from the MBA project. The image depicts section 161 from stack 2958. 7
7 Example of an IHC image from the MBA project. The image depicts section 149 from stack 2958. 7
8 Example of a RGB Color Image. View of the Earth from the Earth-observing satellite Suomi NPP. Image credit: NASA. . 9
9 Grayscale Earth image 10
10 Binary image of Earth after imposing a threshold in a grayscale image. 11
11 Spatial filtering of an image. By combining a mask and an operator through all the points of the image, we create a filtered image. 12
12 Average filter of varying sizes, showing the effect on different shapes and sizes. 14
13 Galaxy extraction from the image depicting Stephan's Quintet. The original image is filtered by using an average filter of size 401×401 pixels. In order to extract the five large galaxies we use a threshold on the filtered image. Original image credit: NASA/STScI. 22
14 The original image with added noise is shown on top and the bottom image shows the same image after using a median filter of size 5×5 pixel. Original color image license: Creative Commons CCO. 23
15 Effect of Sobel Filter in an image depicting an Arctic fox in Iceland. Original color image license: Creative Commons CCO 1.0 24

16 Monarch butterfly image edge detection comparison. Original image adapted from Wikimedia Commons. 25
17 Example of the dilation effect of a 3×3 square structuring element on data points 26
18 Example of the dilation effect of a disk structuring element on input point 26
19 Example of the erosion effect of a square 3×3 pixel structuring element on input point 27
20 Application of erosion in determining an object perimeter. Ophiura ophiura (common brittlestar) image by © Hans Hillewaert / CC BY-SA 4.0 28
21 Comparison of a whole F image size versus the sample size. The image depicts section 119 from stack 2941. 30
22 Examples of different kinds of noise and artifacts in the original F images. Such artifacts do not affect the tissue but in order to analyze the images we have to crop the images so the artifacts do not affect the subsequent analysis. 31
23 Examples of different kinds of noise and artifacts in the original N images. Such artifacts do not affect the tissue but in order to analyze the images we have to crop the images so the artifacts do not affect the subsequent analysis. 32
24 Examples of different kinds of noise and artifacts in the original IHC images. Such artifacts do not affect the tissue but in order to analyze the images we have to crop the images so the artifacts do not affect the subsequent analysis. 37
25 Fluorescent image with noise similar to the salt and pepper noise we discussed in the previous chapter. 38
26 Nissl stained image with a lot of noise similar to the salt and pepper noise we discussed in the previous chapter. 39
27 IHC image with noise. 40
28 Original image that we will use to show the edge detection process. This is image 149 in stack 2941. 41
29 Grayscale original image. 42
30 Grayscale image after using the median filter 43
31 Grayscale mage after the Sobel filter is applied. 44
32 Binary image after the threshold is applied to the Sobel filtered image. 45
33 Grayscale image after dilation with a rounded shape is applied. 46

34 The gaps inside the perimeter have been filled. 47
35 Erode the filled image. 48
36 Estimated sample edge overlayed to the original color image. 49
37 Example of edge detection on an F noisy and out-of-focus image 50
38 Examples of edge detection of an F noisy image 51
39 Examples of edge detection of an N image with several bubbles 52
40 Examples of edge detection of an IHC image with a few artifacts and letters from the slides 53
41 Examples of F images that due to damage, noise, artifacts will be removed from the sample with visual inspection. 66
42 Examples of noisy and damaged Nissl stained images 67
43 Examples of IHC images that due to damage,noise, artifacts would be removed from the sample through visual inspection . 68
44 Examples of different fluorescent out of focus images 69
45 Examples of different Nissl stained out-of-focus images 70
46 Examples of different IHC out-of-focus images 71
47 Example of fluorescent (F) image that is out-of-focus when magnified. When the image is in full view it does not look OOF but once the image is magnified the image is clearly OOF. 72
48 Image of section 136 from stack 2681 does not look like OOF when inspecting the whole image but when we magnify in several regions of the image, it is clear that the image is OOF. 73
49 Image of section 85 from stack 2797 does not look like OOF when inspecting the whole image but when we magnify in several regions of the image, it is clear that the image is OOF. 74
50 Example of fluorescent (F) image that is partially out-of-focus. The top part of the figure shows a magnified region of interest (region A) while the bottom part of the figure shows another magnified region of interest (region B) of the same image. It is shown that, even though both regions are from the same image, region A is out-of-focus while region B is in focus. . . 75
51 Ratio of FV of sharp vs OOF images for different kernels from a sample of different brain image datasets. The numbers displayed in the legend indicate the brain dataset. 76
52 Images from stack 2888 77
53 Images from stack 2888 78
54 Images from stack 2888 79

55 OOF detection for stack 2888 80
56 OOF images indentified by the algorithm and visual inspection for series 2888 81
57 Section 53 identified as OOF. The image does not look like OOF when looking at the whole image but when we zoom in some regions it is clear that it is OOF. 82
58 OOF detection of N images in stack 3079 83
59 N section 22 from stack 3079 identified as OOF visually and from the program 84
60 N sections 121(top) and 122(bottom). The top image was identified as OOF by the algorithm while the bottom was not. Looking at both whole images we can not tell the difference. . 85
61 Zoom in areas from N sections 121(top) and 122(bottom). The top image was identified as OOF by the algorithm while the bottom was not, at this resolution it is evident the top is OOF 86
62 N sections 141(top) and 142(bottom). The top image was not identified as OOF by the algorithm while the bottom was identified as OOF. Looking at both whole images we can not tell the difference. 87
63 Zoom in areas from N sections 141(top) and 142(bottom). The top image was not identified as OOF by the algorithm while the bottom was identified as OOF, at this resolution it is evident the bottom is OOF. 88
64 Focus value (FV) versus image ID for different IHC brain image dataset 2695, where the black $\circ$ symbol represents all the data, the straight blue line is the moving median and the OOF candidates are shown as the red $*$ symbol. 89
65 IHC sections 40(top) and 41(bottom) from stack 2695. The top image was identified as OOF by the algorithm while the bottom was not. Looking at both whole images at this magnification, we can not tell the difference. 90
66 Zoom in areas from IHC sections 40(top) and 41(bottom). The top image was identified as OOF by the algorithm while the bottom was identified as OOF, at this resolution it is evident the top is OOF so the algorithm was successful in identifying this OOF image. 91

67 IHC sections 114(top) and 115(bottom) from stack 2695. The top image was identified as not an OOF by the algorithm while the bottom was identified as an OOF. Looking at both whole images at this magnification, we can not tell the difference. . . 92
68 IHC sections 116(top) and 117(bottom) from stack 2695. Both images were identified as an OOF by the algorithm. 93
69 Magnified 2000×2000 pixel areas from IHC sections 114, 115, 116 and 117. Images 115, 116 and 117 were identified as OOF by the algorithm while section 114 was identified as not an OOF, at this resolution it is evident the algorithm was successful in identifying the OOF images 94
70 IHC sections 187 (top) and 188 (bottom) from stack 2695. Both images were identified as OOF but it is not clear at this magnification. 95
71 IHC sections 189 (top) and 190 (bottom) from stack 2695. The top image was identified as an OOF by the algorithm while the bottom was identified as not an OOF. Looking at both whole images at this magnification, we can not tell the difference. . . 96
72 Magnified 2000×2000 pixel areas from IHC sections 187, 188, 189 and 190. Images 187, 188 and 189 were identified as OOF by the algorithm while section 190 was identified as not an OOF, at this resolution it is evident the algorithm was successful in identifying the OOF images 97
73 Frames 100-190 from traffic video 98
74 Frames 191-281 from traffic video 99
75 Focus value (FV) versus frame ID for the traffic video. The black $\circ$ symbol represents all the data, the blue solid line is the moving window data points and the OOF candidates are shown as the red $*$ symbol. 100
76 Blurred traffic images identified as OOF by the algorithm. . . 101
77 **Original fluorescent color image section 147 from stack 2941 with stripes.** This is an original color image with stripes before any processing, the size of the image is 19726 (rows) $\times$27156 (columns). There are at least ten visible vertical stripes on the sample, there a few more in the background but not visible due to its color. 105

78 **Cropped original fluorescent color image of section 147 from stack 2941.** Three random areas were picked so that we inspect the structure of the stripes, the whole cropped image is 14231 (rows) ×20759 (columns) pixel size 106
79 **Details of the fluorescent original color image of section 147 from stack 2941 with stripes**.This is a magnification of three random small sections of the original image that include two clearly visible vertical stripes before any processing.119
80 **Color Sample image and Background image.** The top image depicts a full color Sample image and the bottom image shows the Background image as defined in Section 5.3.1. . . . 120
81 **Sobel filtered Background image.** This is the gray scale Background image after it has been processed with a Sobel filter. The ten stripes of the cropped image are clearly visible. We can see how they split the image into eleven panels. 120
82 **Mean cross track profile of the Background.** The mean cross track profile of the Sobel Background gray scale image reveals the local minima which are marked in red. This plot allows us to identify the size and exact location of the stripes. 121
83 **Mean cross track profiles for the Sample color image.** The mean cross track profile of the Sample image in all three channels (R, G, B). The black arrows show the minima positions that were identified from the Sobel filter of the Background in the previous section. The additional peaks in the center of the red channel are due to the staining with a marker in that area, which can clearly be seen on the bottom of the image in Fig. 80. 122
84 **Profile of the second and fourth stripe of the Sample image.** We are presenting two different stripes from the red channel of the Sample image. The middle red point in both images is the location of the minima while the other two points show the beginning and end of the stripe. 123
85 The second stripe before and after the polynomial fit. After the correction we need to calculate the scaling factor between points A and B in subfigure (b) so that the discontinuity is removed and the stripe is completely smoothed. 123

86 **Mean cross track profiles for R, G, B channels of the image after smoothing and scaling of the stripes.** The rising illumination is the result of an uneven illumination of each panel. 124
87 **Cross track profiles for R, G, B Background after smoothing and scaling of the stripes.** These are the mean cross track profiles for the Background image after the stripes were smoothed. Similar to the Sample image, we see the rising illumination towards the right side of the image. 125
88 **Mean cross track profile for Red Background with an exponential fit.** The mean cross profile for the Background image in the Red channel, after we have smoothed and scaled the stripes, is shown in blue. The red line is the exponential fit directly from the mean cross profile and the green line is the fit resulting from using our scaling versus panel data points. Based on the g raph, there is an agreement between the two fits. 126
89 **Mean cross track profiles for the Sample R, G, B channels after balancing the illumination.** After the correction the Background does not have the ascending trend anymore but we see a the individual structure in every panel. . . 127
90 **Mean cross track profiles comparison between original Sample R, G, B and corrected Sample images.** After we complete the smoothing of the stripes and balance the illumination of the image, the mean cross profile of the resulting Sample image is shown in blue and has been overlayed by the original Sample image profile (in red) to highlight the slight rotation between the two sets of data. 128
91 **Mean cross track profiles for Sample R, G, B before and after smoothing the lines.** In order to showcase the slight rotation between the cross track profile of the original image (in red color) and the corrected image (in blue color), we concentrate on the fourth panel of the Sample image. . . . 129
92 **Modeling the intensity panel variations.** The difference between the image without stripes and the original image has a linear form and is shown in red. The blue color shows the Fourier model we used to model panel four so that the mean cross profile resembles the initial profile. 130

93 **Color Sample image without stripes.**. This is the resulting color image after all the steps have been completed. The stripes have been removed and the illumination is balanced. . 131
94 **Comparison of different areas of the image before and after destriping.** We selected five random regions to show the effectiveness of the method in removing the stripes while the quality and structure of the image is preserved. 132
95 **Mean cross profile of destriped image in R, G, B channels.** The sudden drops in intensity and the uneven scaling of the panels has been removed, while the illumination is balanced throughout the image. The fluctuations we see in the mean cross profile now reflect the structures in the Sample. . 133
96 **Original image with selected ROIs for comparison**. Original image with selected 1000 × 1000 pixel ROIs away from the stripes. The ROIs are used to compare the values before and after the destriping of the image in order to assess the quality of the method. 134
97 **SSIM index map.** This map shows the local values of the SSIM index that is derived after comparing the initial image with the destriped image. We notice that the largest changes are in the areas of the stripes (darker areas), while further away the changes are smaller (lighter areas). 135

List of Tables

1 Roberts cross operators 17
2 Prewitt Operators 18
3 Horizontal and vertical Sobel operators 18
4 F and N brain image datasets results from testing phase. Results from a random sample of 31 fluorescent brain image datasets (8527 total images) and 25 Nissl stained brain image datasets (6980 total images) showing false positive and false negatives for each separate dataset. 61
5 F, N, IHC brain image datasets results from production. Results from a random sample of 25 fluorescent, 94 Nissl stained and 89 IHC brain image datasets during production runs, showing false positive and false negatives for each separate dataset. 62
6 Confusion matrix from 6812 fluorescent images during production runs. 62
7 Confusion matrix from 25849 N images during production runs. 62
8 Confusion matrix, precision and recall from results from 24165 IHC images during production runs. 62
9 Precision and recall from results from 6812 F, 25849 N and 24165 IHC images during production runs. 63
10 **Mean and standard deviation comparison**. Mean and standard deviations of the different regions of the gray scale image that are away from the original stripes for the original and the destriped image. The last two columns show the percentage change. 113
11 MRD and SSIM values of the selected ROIs of the image . . . 114
12 Focus values of the different ROIs of the image for the original and destriped image. The last column shows the percentage change in the FV value. 116

List of Abbreviations

Include this list if applicable.

Preface

This page is optional.

Acknowledgements

This page is optional.

Vita, Publications and/or Fields of Study

This page is optional for doctoral students only.

1 Introduction

1.1 Biological Background

The human brain is the most important and complex organ in the human body. Our brain acts as a control center for all our motor functions but it also controls our thoughts, emotions and behavior. Our memory and learning and intellectual abilities are also controlled by the brain. A simple map of the brain is shown in Figure 2 when we can see different areas of the brain.

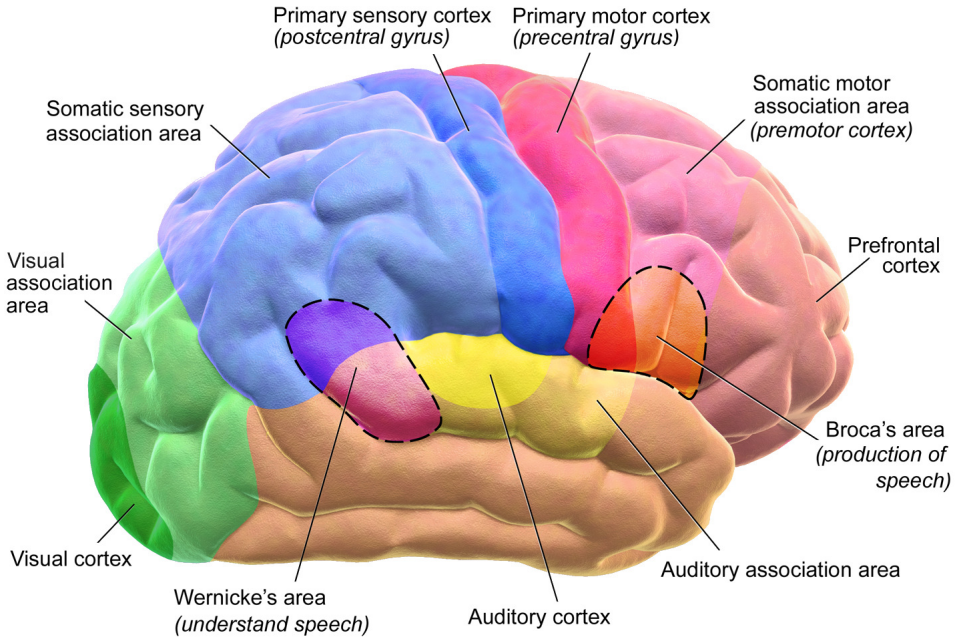


Figure 1: Human brain depiction. Credit: Medical Gallery of Blausen Medical 2014.

The *neurons* are the messengers of the brain, they are the cells that pass signals within the brain. Neurons have specialized functions and they can have varied shapes and sizes. One of the difference of neurons compared to other cells is that they can transmit information over long distances. A typical neuron consists of a *cell body*, *dendrites* and *axon* (Figure 2).

In order for information to flow from one neuron to the other it has to

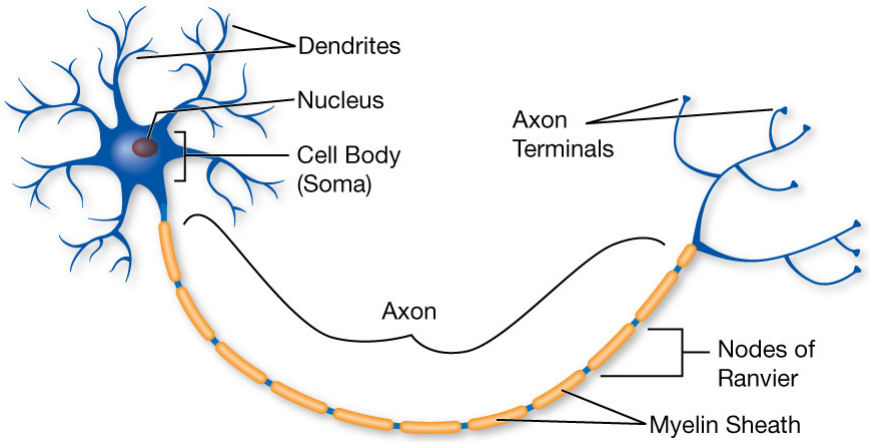


Figure 2: Neuron structure (from [19]).

pass through a small gap known as *synapse* (Figure 3). A neuron can send and receive information from several neurons. An average neuron can form and receive $1,000$ to $10,000$ synaptic connections and it is estimated that an average human brain contains close to 100 billion neurons [23] (Figure 4).

The scale of these numbers is astronomical, that is why the human brain is considered one of the most complicated structures in the know universe. The complexity of the structure of the brain at that level is enormous and understanding these connections will give an insight into the function of the brain, which is a critical scientific challenge.

1.2 The Mouse Brain Architecture Project

At the microscopic level the brain has a great degree of variability due to the astronomical number of neurons while at the macroscopic level the patterns are relatively stable to each species. The mesoscopic scale is the transitional point between the two and it allows us to study the circuit architecture of entire brains. Animal brains like mice can be used to create a mesoscopic level map of the neural brain circuit architecture. Although mice brains are smaller and less complicated than human brains, much of the structure and neural connections that exist in human brains can also exist in mice, so they

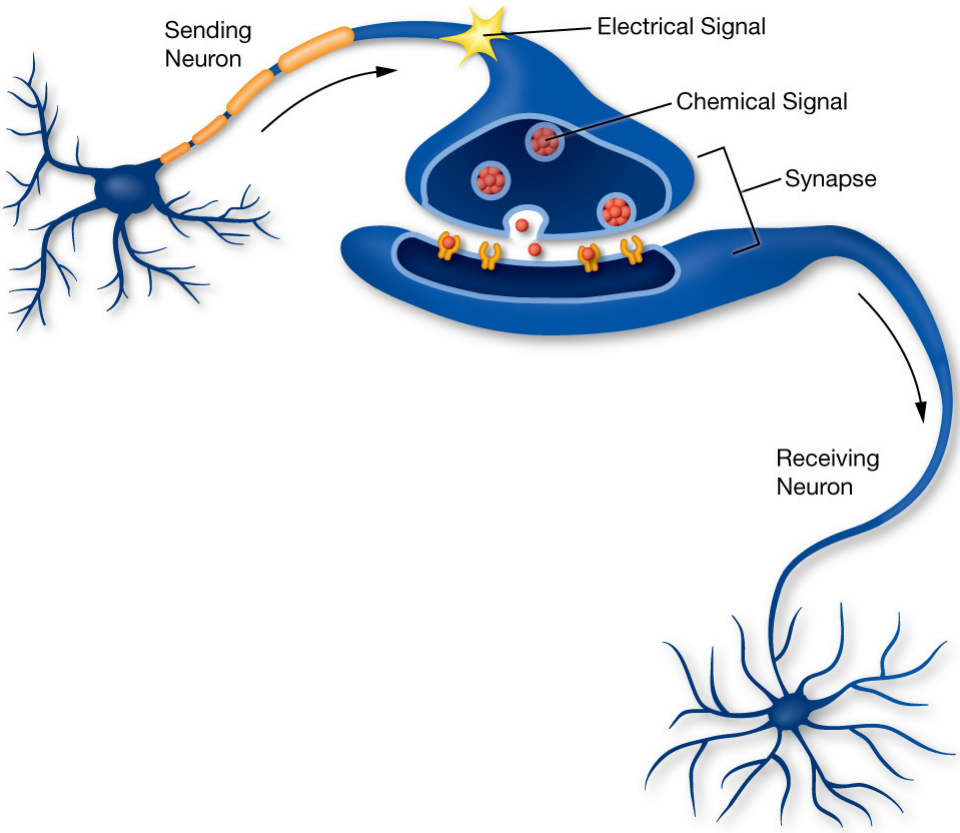


Figure 3: Structure of a synaptic connection(from [19]).

can be used as a model for human brains.

The Mouse Brain Architecture (MBA) project aims to create a map of the mesoscale cicruits of the whole brain images [30, 3]. The goal of the project is to map the mouse brain at a mesoscopic scale so that we can explore the neural connections throughout the whole brain. In order to create a 3D map of a mouse brain it has to be first sectioned into very thin layers so that they can be scanned([37]) individually.

In order for the neurons to be visible we pick about 250 brain sites for injections of different kinds of *tracers*. Tracers are chemical probes that are used in order to artificially stain the tissue sample so that we can differenti-

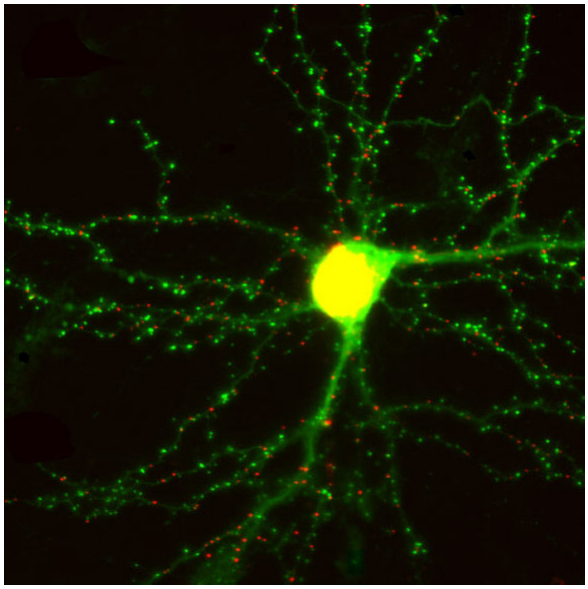


Figure 4: Microscopic image of a neuron in the brain of a living mouse. Fluorescent probes are use to brighten up the synapses, the red dots denote inhibitory synapse while the green dots mark excitatory synapses. Image adapted from [22].

ate the different structures and they are characterized by the direction they travel. *Anterograde* tracers show the path from the source to the termination so we can study which cells are receiving information while *retrograde* tracers reveal the cells that are sending information to the population neurons in the injection site [6].

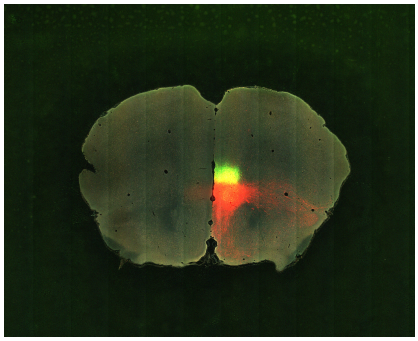
Each site is injected with four tracers in four different mice. Two of the tracers (one viral retrograde and one viral anterograde) were tagged with fluorescent dyes (F). Additionally, two other tracers (one retrograde and one anterograde) will be labeled immunohistochemically (IHC) and visualized using *brightfield* imaging. Each brain consists of approximately 600 sections and alternate sections are Nissl stained (N). The Nissl stain is useful because it allows us to study the cytoarchitecture of neurons in the different parts of the brain ([1]). Although the Nissl stain does allow us to see the different patterns of a brain area it does not allow us to see the detailed morphology of

a neuron. *Brightfield microscopy* is a common form of optical microscopy but unless the sample is artificially stained we can not differentiate the different structures. *Fluorescence microscopy* is an ideal for observing large samples at the micrometer scale because it allows us to access the individual structures with a great deal of detail. Scanning the samples creates 2D images that allows us to visualize the individual neurons and their processes. Stacking those images will result in a 3D representation of each brain that was scanned.

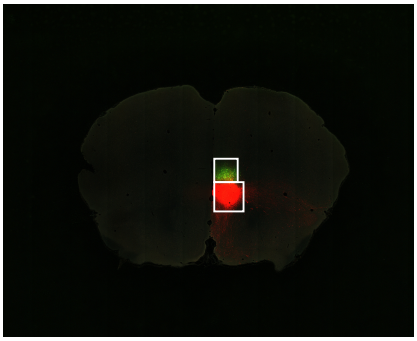
An example of each kind of image is shown to visually demonstrate their differences and their strength in identifying different structures. Figure 5a shows an F image where the injection sites are much brighter while Figure 5b shows the same image less illuminated so that the injection areas are emphasized more. The two regions enclosed in white squares are shown in Figures 5c, 5d. These images exhibit higher contrasts compare to the N and IHC images so the individual structures are clearly visible. Figure 42a showns an Nissl stained section from a mouse brain and a close up in Figure 42b. As shown in the image the background is white and the cell bodies are blue. Figure 43a shows an IHC image from a section of a mouse brain where a close up of the black square is shown in Figure 43b where we can clearly see the labeled fibers.

Because of variability in brain shape and size between different mice, we have to repeat the imaging numerous times using various samples in order to create a high quality brain atlas. This process creates images with a staggering amount of detail and complexity. Each brain is sectioned approximately 20 μm thick which results in approximately 600 images, which are referred to as 'stacks'. The digital imaging of the different sections is performed with a 20X objective (0.46 μm per pixel) which produces images with close to 1 billion pixels. The images are stored as a 2D array of pixels and each pixel can be characterized by its (x, y) coordinates and its value, so each image can reach a gigapixel size. Considering the number of samples and the high definition of our data, our entire dataset is larger than 1.5PB. Because of the massive volume and complexity of our data, any interactive work on the dataset would take years to be completed, so it has to be replaced by robust automatic processing methods in order to be completed in a reasonable amount of time.

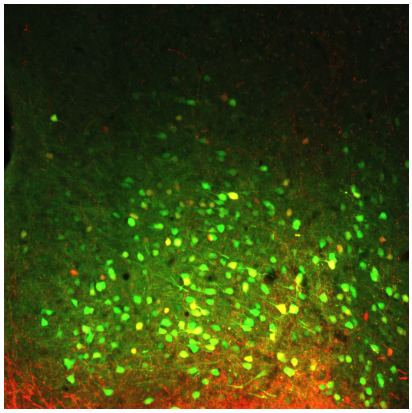
The main goal of the project is to eventually comprehend the architecture of the human brain. Creating a high quality mouse brain atlas is a first step towards that goal. The data collected will be available to all researchers through a data repository in order to study and analyze specific areas and



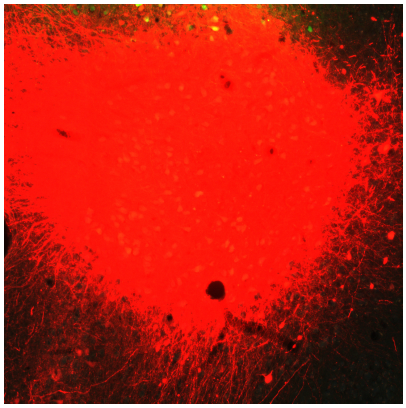
(a) F image



(b) Dim Image



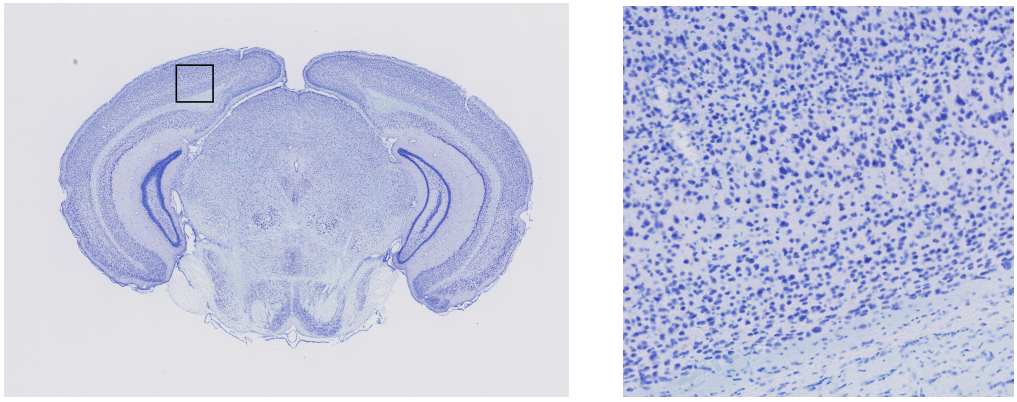
(c) Close up of top region



(d) Close up of bottom region

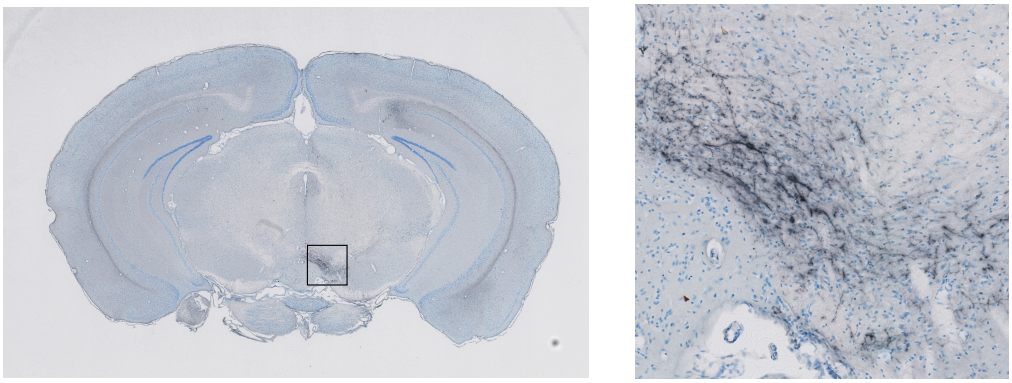
Figure 5: Example of an F image from the MBA project. The image depicts section 32 from stack 2941.

circuits of the brain. But the study of the brain connectivity at this level can open the gates to the understanding of many brain diseases and disorders. Abnormal brain connectivity has been linked to the autism spectrum disorders(ASD) ([54]), schizophrenia([47], dyslexia ([15]) among others. Any progress towards understanding any of these diseases could potentially lead to more accurate diagnosis and potential drug development.



(a) N image of a mouse brain section (b) Magnification of the black square area

Figure 6: Example of an N image from the MBA project. The image depicts section 161 from stack 2958.



(a) IHC image of a mouse brain section (b) Injection site magnification

Figure 7: Example of an IHC image from the MBA project. The image depicts section 149 from stack 2958.

2 Image Processing

2.1 Background

A digital image is a representation of a real image that can be stored as two-dimensional array (i.e. matrix). In order to translate an image into numbers we can divide it into small areas called *pixels* so a single pixel in the displayed image corresponds to an element of the matrix. The size of the matrix does not correspond to the size of the image in the real world. A factor called *resolution* determined the spatial scale (smallest discernable detail) of the image pixels and is necessary if we want to traslate the pixels to real world representation (size). For example an image with 3600×3600 pixels that has a resolution of 300 pixels per inch in the real world occupies a size of 12×12 in. *Intensity* is the numeric value of a pixel, the higher the number the brighter the pixel is.

True color images require a 3D array, where the first plane in the third dimension represents the red pixel intensities, the second plane represents the green pixel intensities and the third plane represents the blue pixel intensites. Those will be called R, G, B colors or channels throughout this thesis. An example of a color image is shown in Figure 8 depicting Earth. The dimensions of the image are $8000 \times 8000 \times 3$.

If we take a particular weighted combination of the three color channels of an images the result is an *grayscale* image that has a continuous range of gray values. The grayscale image carries only intensity information with black to be the weakest intensity while white to the strongest intensity. The weighted sum of the R, G, B components is:

$$0.2989 * R + 0.5870 * G + 0.1140 * B \tag{1}$$

By using the above equation we derive the grayscale image of Earth that is shown in Figure 9. The colors now have a continuous range of gray values and the size of this image is 8000×8000.

A *binary* image is a digital image that has only two colors: black or white. While a grayscale image can have any value between 0 and 1, a binary image can only have a value of 0 or 1. We can create a binary image from the grayscale image by imposing an intensity threshold. The pixels that have values lower than the threshold have their values replaced with zero, while the pixels with values higher than the threshold have their values replaced with one. Using as a threshold 75% of the maximum value of the grayscale



Figure 8: Example of a RGB Color Image. View of the Earth from the Earth-observing satellite Suomi NPP. Image credit: NASA.

image, the outcome of such an image is shown in Figure 10. This image has only two values, zero (background) and one (foreground).

Image processing algorithms in the spatial domain are based on direct manipulation of images pixels. Alternatively, an image can be processed in a transform domain which will require the image to be transformed in the transform domain first, complete any processing steps and then revert back

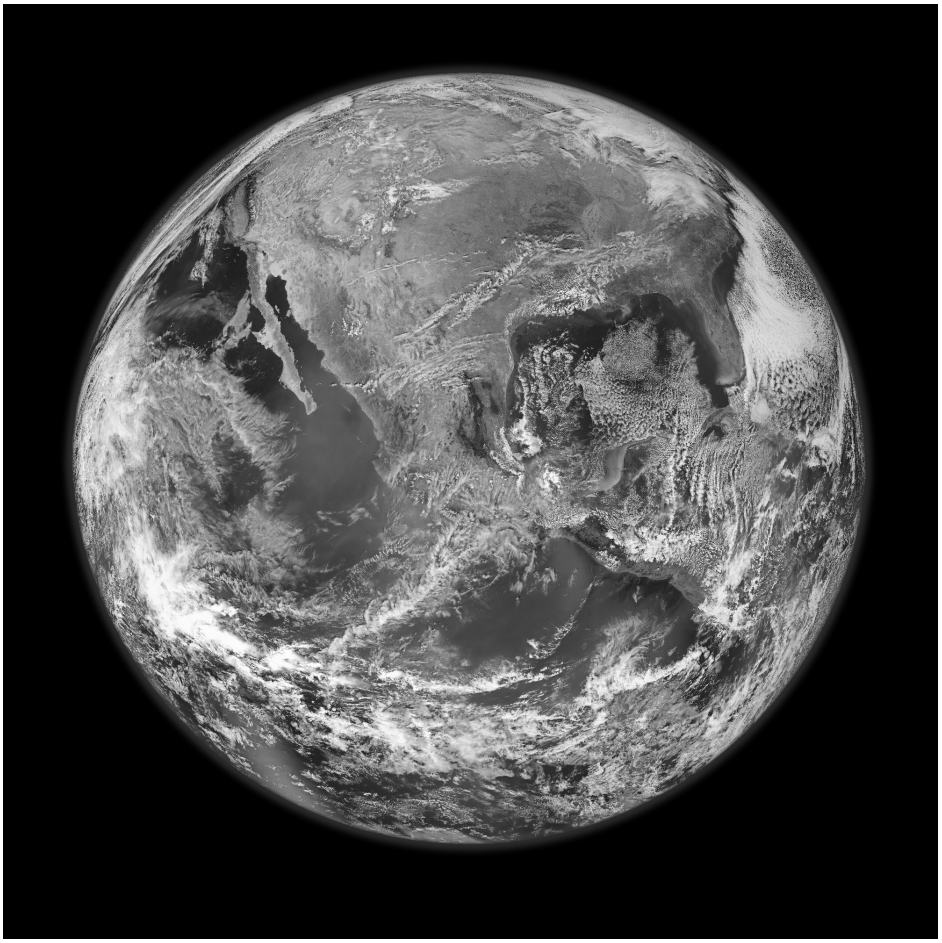


Figure 9: Grayscale Earth image

to the spatial domain. One of the most common transform domains is the frequency domain, where we use the Fourier transform of an image to make any changes required. The choice of which domain is better suited depends on the processing tasks and the kind of image we have to manipulate. In this work we will mostly make use of spatial processing. An example of an image *filter* is shown in Figure 11

An arbitrary point A shown in position (x,y), along with a small neigh-

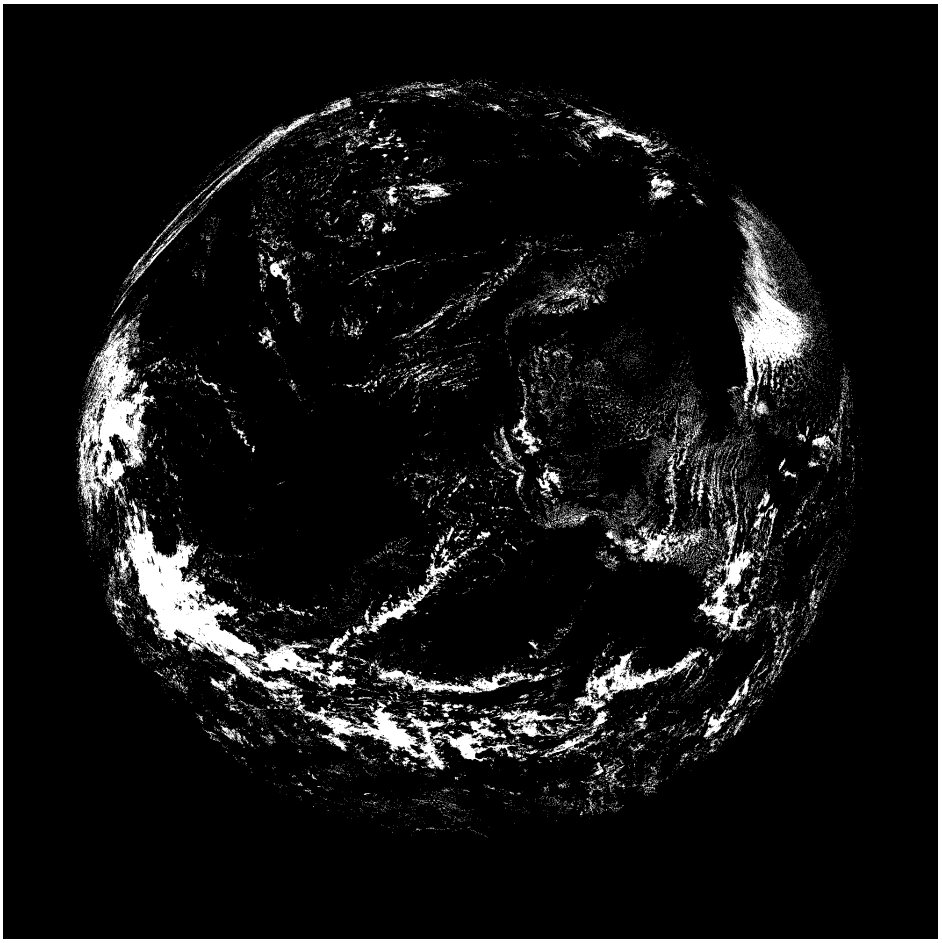


Figure 10: Binary image of Earth after imposing a threshold in a grayscale image.

borhing region. If we move the origin of the neighborhood from region to region and apply an operator in that area the outcome would be a *filtered* image. The new filtered image has new values for each point A which are the result of the filtering operation. A *mask* as shown in the image is a rectangle (it could be any other shape) with the data point we want to change in the middle. The operator we impose could be a function or a process. The

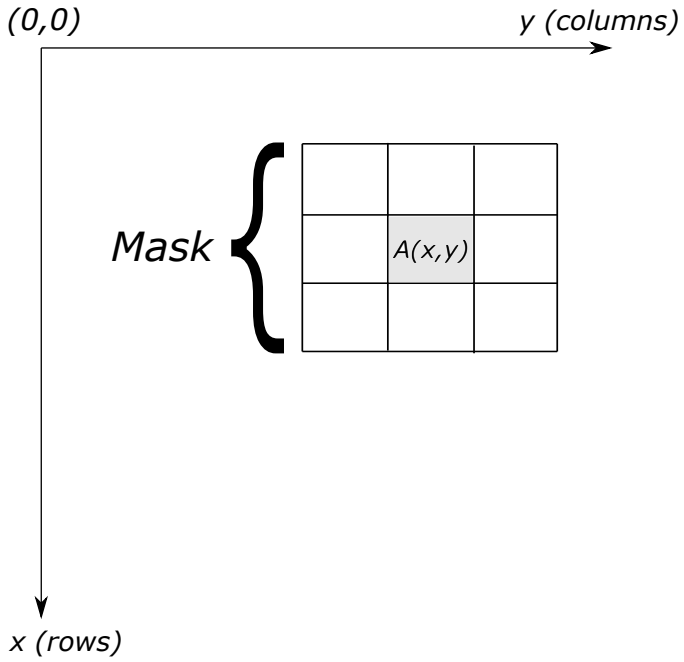


Figure 11: Spatial filtering of an image. By combining a mask and an operator through all the points of the image, we create a filtered image.

procedure of combining a mask and an operator in each point of the image is called *spacial filtering*, while the operator along with the neighborhood is called a *spatial filter* (or *spatial mask*, *kernel*, *window*).

There are numerous filters that do different tasks like remove an unwanted component or feature of an image, enhance or unveil a structure, etc. In this chapter we will discuss only filters that are necessary in order to understand this work.

2.2 Smoothing Spatial Filters

2.2.1 Averaging filter

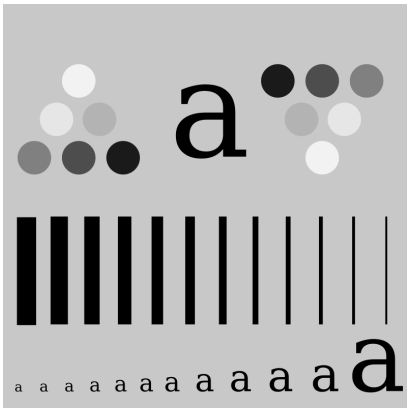
An image that has gone through an average filter is an image where all the pixel values are replaced by the average values of its neighbors. For example a $3x3$ average filtered image has all values substituted with:

$$P(x,y) = \frac{1}{9}\sum_{i=1}^{9} N_i(x,y) \tag{2}$$

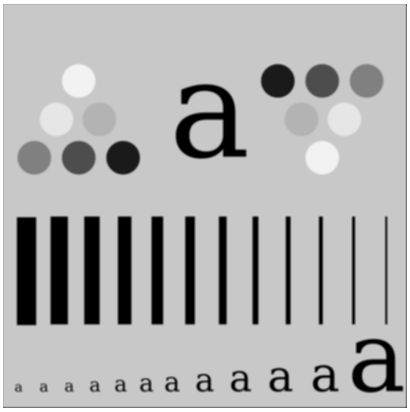
where P is the new data point in position (x, y) and N_i is the sum of all values of the 8 neighboring points and the point itself. The size of the mask is decided on depending on the kind of data we have and what is the goal of the project. The average filter is also called a *lowpass filter* because it passes lowpass frequencies but it filters out higher frequencies. To demonstrate the utility we present an image with different size elements and apply different filters.

Visually an image that has been processed with an average filter will look blurred and the amount of blurring will depend on the size of the mask. Blurring an image can be useful in a situation where we want to extract objects of a specific size in an image where we depict objects of different size. As we increase the filter size smaller objects will blend with the background and larger objects will look like 'blobs'. The object of interest that we want to extract will dictate the size of the filter that we will use.

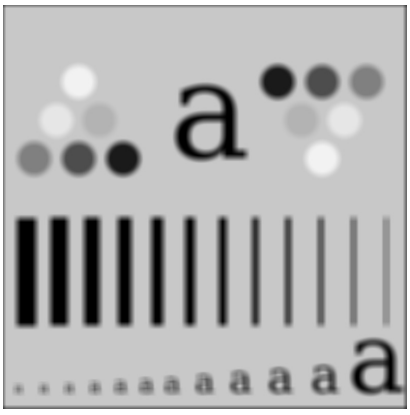
In Figure 12 we present how the image smoothing is affected by the different size of average filters. Subfigure 12a shows the original image while the images Subfigures 12b - 12f show the results after smoothing with square average 5×5, 15×15, 25×25, 51×51, 75×75 pixel filters respectively.



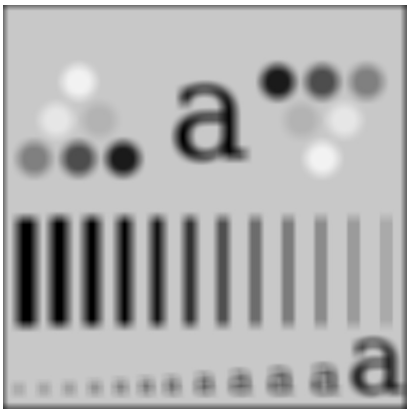
(a) Original Image



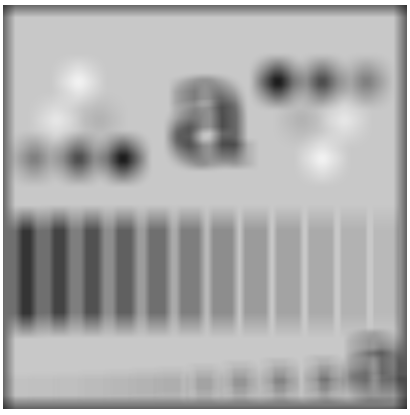
(b) 5x5 Filter



(c) 15x15 Filter



(d) 25x25 Filter



(e) 51x51 Filter



(f) 75x75 Filter

Figure 12: Average filter of varying sizes, showing the effect on different shapes and sizes.

We notice that the image becomes progressively smoother and the smaller structures start to blend in the background. For example the small 'a' on the left side of the image is slightly blurred in Subfigure 12b, while at Subfigure 12d it starts to blend in with the background. In Subfigure 12e the six smallest 'a's are completely blended with the background since the filter we are using is larger than the first six letters. An important practical application of the average filter is that it allows us to retrieve objects of interest from an image that includes objects of different sizes and shapes. We will use an example from the field of Astronomy using an image taken with the Hubble Space Telescope depicting a galaxy group, located in the constellation of Pegasus, called Stephan's Quintet. The image in Subfigure 13a shows the original image in color which depict the galaxies with other background galaxies and stars. The size of the original image is 5840×4894 pixels.

We can use the average filter to extract the five main galaxies depicted. Because of the size of the overall image and the size of the galaxies a very large mask is required. In order for the background galaxies and stars to be remove we will use a mask that has a size of 401×401 pixels, shown in Subfigure 13c. It is common practice to follow this operation by using a threshold so that objects that are fainter can be eliminated. In this example we use a threshold equal to the median value and one standard deviation of the image. The result of the threshold is shown in Subfigure 13d where we see a representation of the brightest objects of the image, in our case the galaxies shown in the foreground.

2.2.2 Gaussian Filter

The Gaussian filter is similar to the mean filter but it uses a different kernel that resembles a Gaussian bell shape. It is also called a Gaussian blur and it is a low pass signal so it diminishes the high frequency signals. Contrary to the mean filter the value we use to replace each pixel is a weighted average of the surrounding pixels, a process which is generally called *convolution*. In a Gaussian blur, the closest pixels are more influential and the weights come from the Gaussian probability distribution. In order to create the Gaussian mask we start by defining the 2D Gaussian kernel as:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{3}$$

where x, y is the pixel position and σ determines the width of the kernel.

A Gaussian filter is a *linear filter* and it can be used to blur an image or reduce noise. The Gaussian filter can be a little slow due to the fact that it is using convolution but it is very powerful in the frequency domain.

2.2.3 Nonlinear Filters

A nonlinear filter is a filter where the output of the image after it has been filtered is not a linear function of the input image. The best known filter in this category in the *median filter*. A median filter will replace every value of the image with the median value within the neighborhood that we have defined. Median filters are used to remove noise from images with much less blurring than a linear filter. They are very powerful in removing very sharp peaks of data and replacing them with the median value of their neighborhood. Noise can be introduce in various ways; for example when we scan a photograph, the film grain can be a source of noise or the scanner can introduce some noise while in digital images the mechanism that gathers the data can introduce noise. An example of noise removal is shown in Figure 14 using a snow landscape example that has a size of 2248×4000 pixels. Figure 14a shows an image after we have added a 'salt and pepper noise' to about 20% of the pixels. Salt and pepper noise represents itself as a percentage of corrupted pixels that visually present themselves as black and white pixels in images. We can remove these sharp disturbances by using a 5×5 pixel size median filter. The result is shown in Figure 14b and it shows good noise removal without visible blurring.

2.3 Sharpening and Edge Detection Spatial Filters

The main goal of sharpening an image is to reveal transitions in intensity. The uses of such filters are varied and necessary in many fields from medical imaging to satellite imagery to visual tracking (surveillance?). Sharpening an image can be achieved by spatial differentiation as opposed to the image smoothing that is achieved by averaging which is analogous to integration [21]. So any areas that have a sharp changes in intensity (like an edge or a discontinuity) will be enhanced using image differentiation. The size of the operator depends on the how large the intensity discontinuity of the image is, mirroring how the smoothing filter size depends on the size of the noise or object we want to remove.

A digital image can not be defined as a continuous function of the spatial

variables but rather a discrete function of the integer spatial coordinates. Based on the above definition, in areas of constant intensity our first derivative will be zero while it will have a non-zero value in the other areas. The second derivative will also be zero in areas of constant intensity but will also be zero in areas where the intensity change has the same slope. That makes the second derivatives more sensitive to edge detection for more fine details.

In the following sections we will discuss filters that are based on first and second derivatives. There are several filters of this kind but we will only focus on the ones that are mentioned in this work.

2.3.1 Roberts, Prewitt and Sobel filter

The Sobel filter uses the image gradient to determine edges in an image. The operator takes advantage of the fact that the adges in an image have the maximum value. The gradient of an image is estimated by the partial derivatives $\partial f/\partial x$ and $\partial f/\partial y$ for every (x, y) pixel location in the image. Since we are dealing with a digital function the approximation of the first-order partial derivatives are estimated as differences between pixels:

$$\frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y) \qquad \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y) \quad (4)$$

The above equation is equal to filtering f(x,y) with a 1D mask. One of the earliest attempts to use an 2D mask when we are interested in the diagonal edge direction was by Roberts ([41]). The operator is named the *Roberts cross operator* and the kernels are shown below:

1	0
0	-1

0	1
-1	0

Table 1: Roberts cross operators

Such 2×2 masks are not useful for computing edge direction (they are very sensitive to noise) and they are also awkward to use since they do not have a center of symmetry. We need masks that are symmetric around the center point so the smallest mask can be 3×3 pixels size. In order to achieve that symmetry (so that the derivative is centered at the pixel we want to manipulate) we add a zero in the middle that way the data from opposite

sides are taken into account and we gain information about the direction of the edge. Another way of approaching the approximation is to fit a quadratic surface over the 3×3 neighborhood and then compute the gradient for the fitted surface. So the simplest approximations to partial derivatives are given using the 3×3 pixel masks which are called Prewitt operators [39]:

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Table 2: Prewitt Operators

The Sobel operator is a slight variation of the Prewitt operator where a little more weight is given in the center row (or columns). This allows us to remove any noise by taking into consideration the previous and next rows (or columns) but not having them influence the result so much. The 3×3 Sobel operator is both directions([49]) is:

-1	0	1
-2	0	2
-1	0	1

+1	+2	+1
0	0	0
-1	-2	-1

Table 3: Horizontal and vertical Sobel operators

The Sobel operator in both directions can also be decomposed as the products of an averaging and differentiation kernel which has a smoothing effect to the image so it makes it less sensitive to noise present in images. We will present an example to show how powerful the Sobel filter is for edge detection. An image of an Arctic fox is shown in the original color image in Figure 15a and the grayscale image in Figure 15b. The image depicts a white fox on an white background which, even visually, could be difficult to detect. If we want to find the edges of the fox using an automatic detection we can use the Sobel filter, the results of the image after we apply the Sobel filter are shown in Figure 15c. We notice that the edges of the fox are emphasized and if we want to them more clear we can create a binary image by using a threshold (Figure 15d). In order to trace the boundary of the fox at this point we will have to use morphological operators which will be discussed in the next section, but the Sobel filter has emphasized the regions of high spatial frequency that correspond to the edges of the image.

2.3.2 Canny operator

The Canny operator [5] is another edge detection method that is more computationally intensive but it could be superior for certain kinds of images. The Canny edge detection algorithm consists of several steps. Initially the image has to be smoothed with a Gaussian filter and then the gradient magnitude and direction are calculated. The next step includes the suppression of pixels that are not local maxima and two thresholds are determined, weak and strong. The pixels that have values higher than the strong threshold are considered pixels that belong in the edge and pixels lower than the weak threshold do not belong in an edge. The pixels that are between the two thresholds will be considered true edge pixels if they are connected to pixels that their value is higher than the strong threshold.

We are presenting an image of a Monarch butterfly (Figure 16) using the Sobel and Canny operator in order to showcase their differences. Although both filters manage to find the overall edge of the butterfly, the Canny operator (Figure 16c) seems superior for fainter edges for example the shadows around the thorax, the stripes on the abdomen and the lighter spots on the wings. So, if we wanted to detect the finer details a Canny operator would be ideal but if we wanted find the overall edge of the butterfly the Sobel filter would give us a good result much faster.

The choice of which filter to be used for edge detection depends on the kind of data we analyse but also the structures we want to unveil. Most of the times they have to be combined with morphological operators (which will be discussed in the next section) in order to get the desired result. Since there are no specific rules about the use of these filters it is very common to test them empirically until one that suits the needs of the particular project is found.

2.4 Morphological operators

2.4.1 Erosion and dilation

Morphological operators are not technically filters but they can act like them since they change the morphology of the image. Two of the most basic operations are *erosion* and *dilation*. The operations need three elements : the data points we want to operate on (input image) and the shape and size of a *structuring element*. The dilation operator will expand the shape of the data points in the input image and the way the expansion is done depends

on the structuring element. Most of the implementations of this operator expect the image to be binary. Figure 17 shows an example of a dilation operation that has a square 3×3 structural element and it will be an array of this form:

1	1	1
1	1	1
1	1	1

In Figure 17a, black is the background and white are the data points that we want to apply the structuring element. The result is shown in Figure 17b where we notice that the data points have grown thicker. Because of that property, a common application of the dilation is to brige gaps in images.

The structuring element can have a variety of elements, a common shape is a disk. A disk shaped structuring element with a radius of 2 pixels is has this form:

0	0	1	0	0
0	1	1	1	0
1	1	1	1	1
0	1	1	1	0
0	0	1	0	0

Figure 18 shows the effect a disk shape structuring element will have on an input image. The choice of the shape and the size of the structuring element depends on the shapes that we want to keep, for example if we want to keep only lines in an image then we pick a linear structuring element.

Erosion has the opposite effect to dilation, it removes a layer of pixels around the boundaries. Figure 19 shows the effect erosion has when using a square 3×3 structuring element.

From this example we can see that the effect of erosion is thinning the boundaries of foreground pixels but it can also disconnect areas when using the right structuring element. Erosion can be viewed as a filter that removes image details smaller than the structuring elements.

One of the practical applications of dilation is region filling. By applying conditional dilation we can close regions in images, a very important step in edge detection. This is an iterative process which combines a dilation operator and a logical operator. Erosion can be used to acquire the perimeter of binary objects. Figures 20a, 20b depict the color and grayscale image

respectively of a brittle star. When applying erosion with a small structuring element the perimeter of the object reduces slightly compared to the original image. The eroded image is shown in Figure 20c and subtracting that image from the original image reveals the perimeter of the star shown in Figure 20d.

2.4.2 Opening and closing

Two other important morphological operators are the *opening* and *closing*. If A is a set and B is a structuring element, the opening ($\circ$) and closing ($\bullet$) operators are defined as:

$$\mathrm{A} \circ \mathrm{B} = (\mathrm{A} \ominus \mathrm{B}) \oplus \mathrm{B} \tag{5}$$

$$\mathrm{A} \bullet \mathrm{B} = (\mathrm{A} \oplus \mathrm{B}) \ominus \mathrm{B} \tag{6}$$

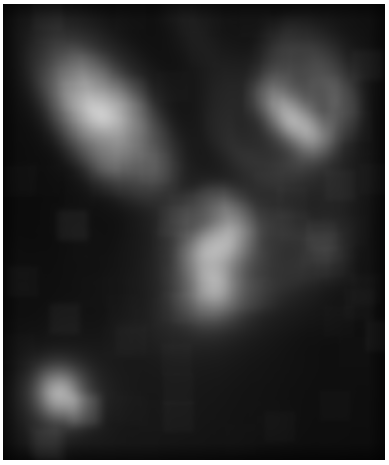
where $\ominus$ and $\oplus$ denote the erosion and dilation respectively ([38]). These two operations can be very powerful in image processing. For example, opening can remove protrusions, smooth the contour of an object or disconnect narrow isthmuses. On the opposite side, closing can connect data points that are close to each other and fills regions and gaps. Two more morphological filters called *top-hats* are built upon the opening and closing operators ([50]). A *white top-hat* of an image is the difference between the original and its opening, while a *black top-hat* is the difference between the closing of the original image and the original image. Such operators have numerous applications such as feature extraction, contrast enhancement and removal of certain elements from images.



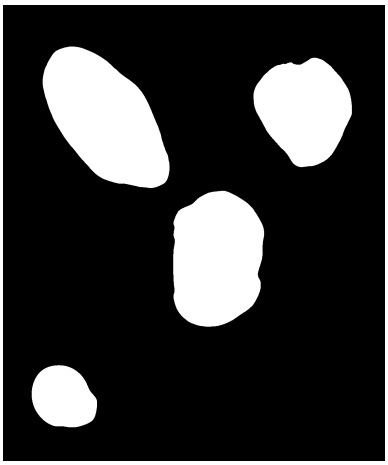
(a) Color Image



(b) Grayscale Original Image

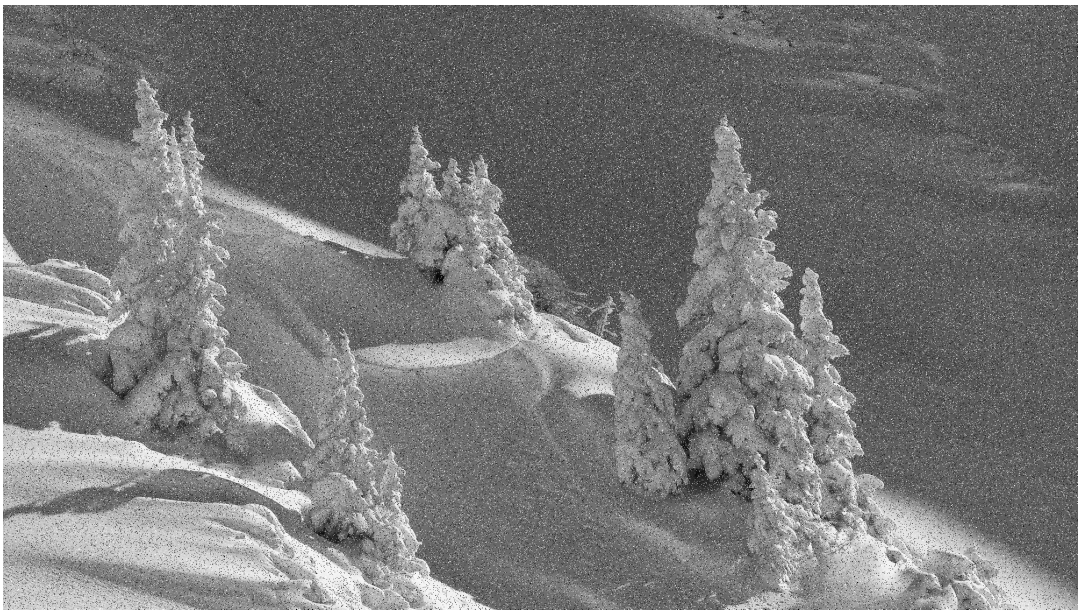


(c) Filtered Image



(d) Thresholding result

Figure 13: Galaxy extraction from the image depicting Stephan's Quintet. The original image is filtered by using an average filter of size 401×401 pixels. In order to extract the five large galaxies we use a threshold on the filtered image. Original image credit: NASA/STScI.



(a) Original grayscale image with added noise



(b) Image after the noise is removed

Figure 14: The original image with added noise is shown on top and the bottom image shows the same image after using a median filter of size 5×5 pixel. Original color image license: Creative Commons CCO.



(a) Original Image



(b) Grayscale image

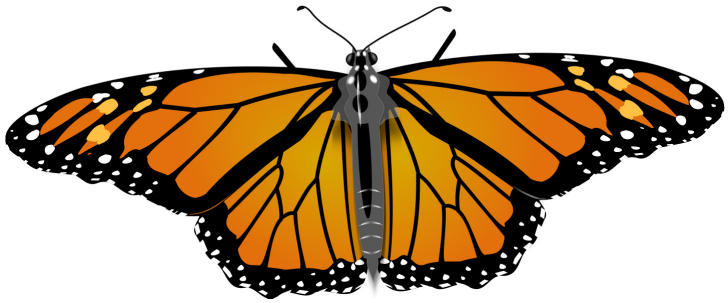


(c) Sobel filter

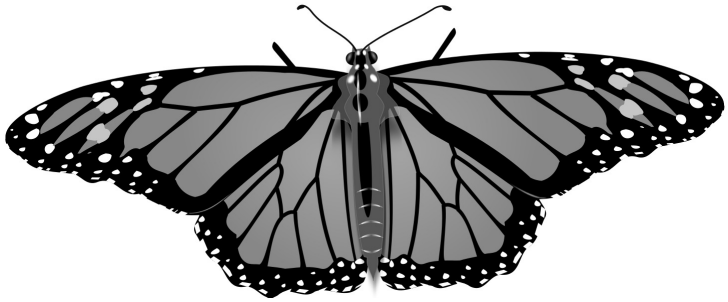


(d) Binary image after using threshold

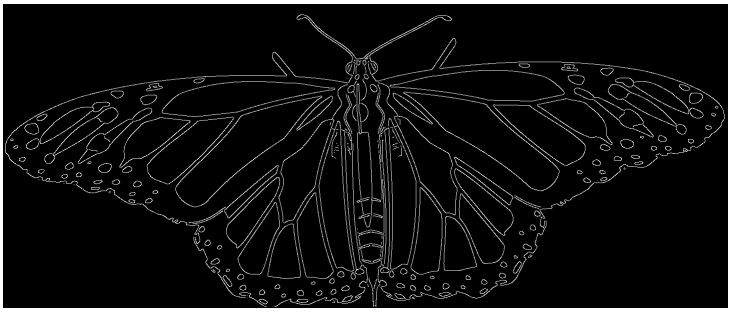
Figure 15: Effect of Sobel Filter in an image depicting an Arctic fox in Iceland. Original color image license: Creative Commons CCO 1.0



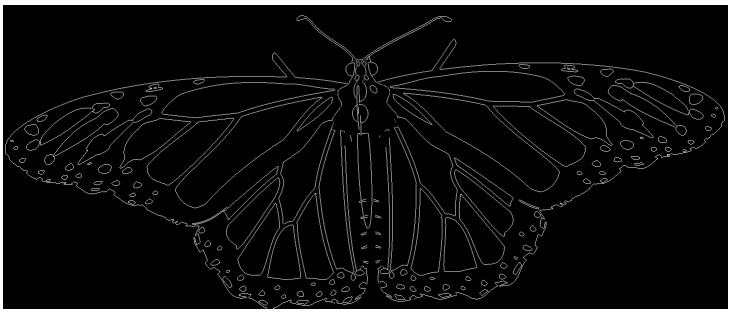
(a) Color Image of a Monarch butterfly



(b) Grayscale image

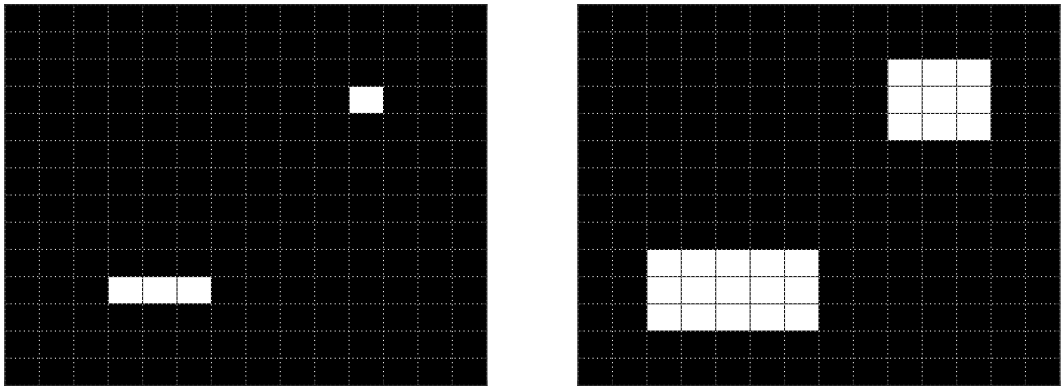


(c) Binary image after using Canny operator



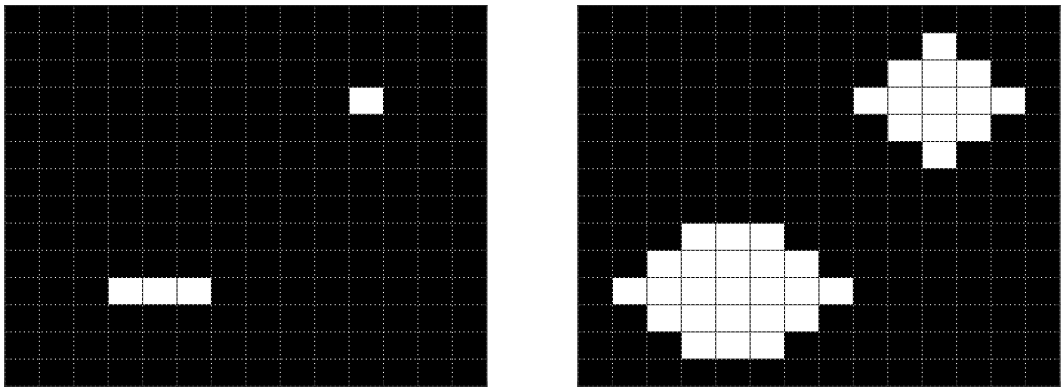
(d) Binary image after using Sobel filter

Figure 16: Monarch butterfly image edge detection comparison. Original image adapted from Wikimedia Commons.



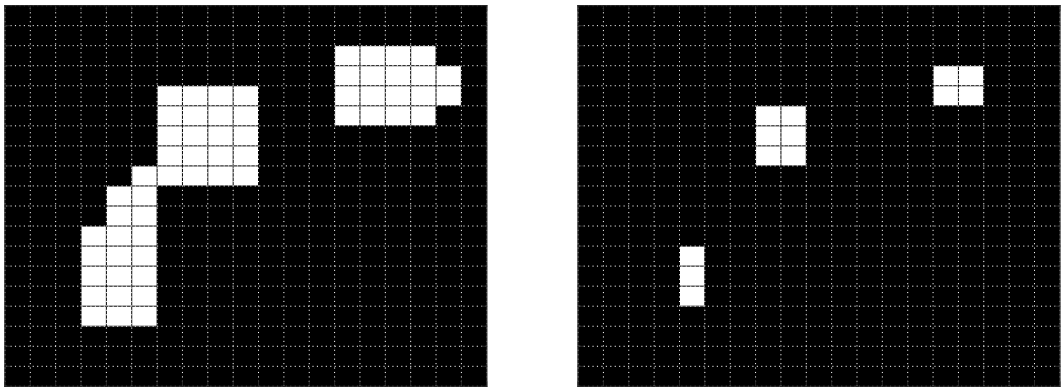
(a) Original Image (b) Image after dilation

Figure 17: Example of the dilation effect of a 3×3 square structuring element on data points



(a) Original Image (b) Image after dilation

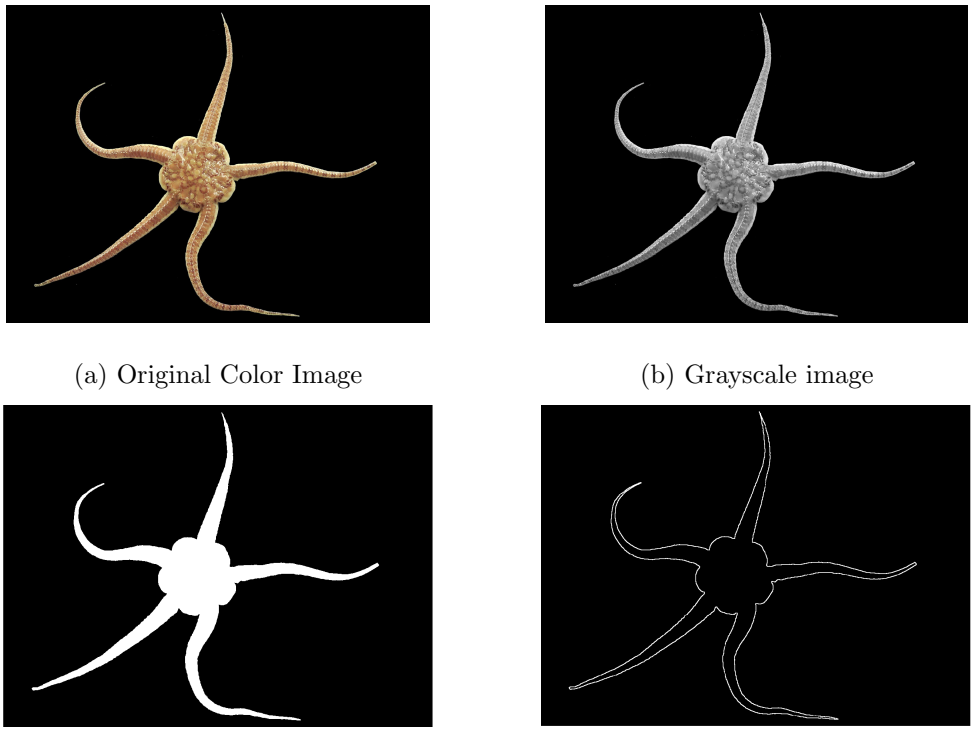
Figure 18: Example of the dilation effect of a disk structuring element on input point



(a) Original Image

(b) Image after erosion

Figure 19: Example of the erosion effect of a square 3×3 pixel structuring element on input point



(a) Original Color Image

(b) Grayscale image

(c) Binary image after erosion

(d) Perimeter of object

Figure 20: Application of erosion in determining an object perimeter. Ophiura ophiura (common brittlestar) image by © Hans Hillewaert / CC BY-SA 4.0 .

3 Edge detection

In the previous chapter we discussed the individual filters and their varied uses. Most of the times in order to achieve specific tasks we have to apply a combination of techniques in order for our result to be successful. Edge detection of our samples is a very important step for sucessful results. There are several reasons for that:

- **Decrease the image size**. As we have discussed before, our images are very large which make the analysis be computationally expensive. Any decrease in the size of the images can help drastically reduce the amount of time it takes for any analyses. An example of a typical image is shown in Figure 21. The original image dimensions is 19615×26558 pixels and it is about 1 GB in size. After we perform the edge detection the image that we will work on is the image inside the white rectangle which has dimensions 13400×20500, so the cropped image will occupy about 53% of the initial image. Such a reduction in size is typical in our images and it speeds up our further calculations so this is a significant step for our data analysis.

- **Noise removal**. A large number of images are not pristine their background could include noise, bubbles, tape or foreign objects on it. Removing the background makes any algorithm run more efficiently and eliminates any errors that could come from misidentifying noise and artifacts for sample. Figure 22 shows several examples of the different noise and artifacts that we can encounter during the analyses in our F images. Figure 22a, 22d, 22e show different obstructions, notice a part of letters in Figure 22e. Figures 22b, 22c, 22f show air bubbles trapped inside the slides. As a side note, notice also that our images are not all the same size and that the samples are not always in the center of the images. We face the same challenges in our N and IHC images as well, a selection of them with similar artifacts is shown in Figures 23, 24.

- **Model background**. In one of our tasks we have to model the background of the image so removing the sample gives us a clean image of the background. The reason why this is an essential step will be discussed further in the destriping chapter.

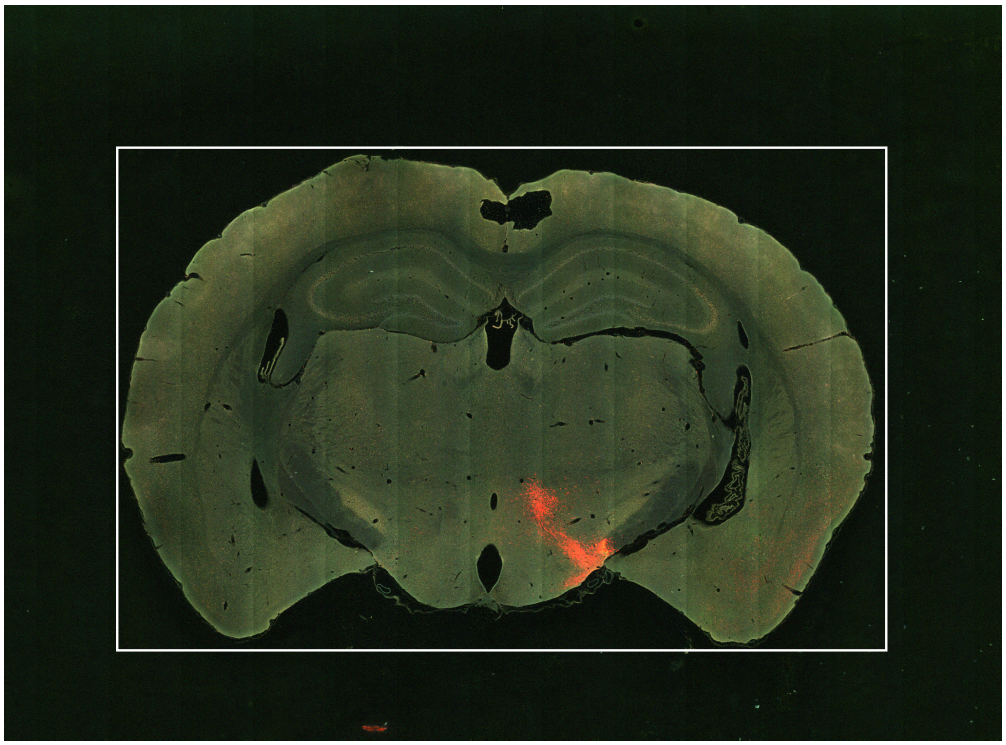


Figure 21: Comparison of a whole F image size versus the sample size. The image depicts section 119 from stack 2941.

We should note here that although not all steps are needed for all images because we are creating algorithms that can run automatically without intervention we have to think of all the different scenarios that can appear in the images. For example, removing the noise from an image might not seem necessary for a clean image but because we will not know which images need it or not we have to include it.

3.1 Median filter

The first step is to apply a median filter to the image. As we have discussed Chapter 2, a median filter is suitable for noise removal and generally sharp peaks of data. This step is necessary because we want to have an image that

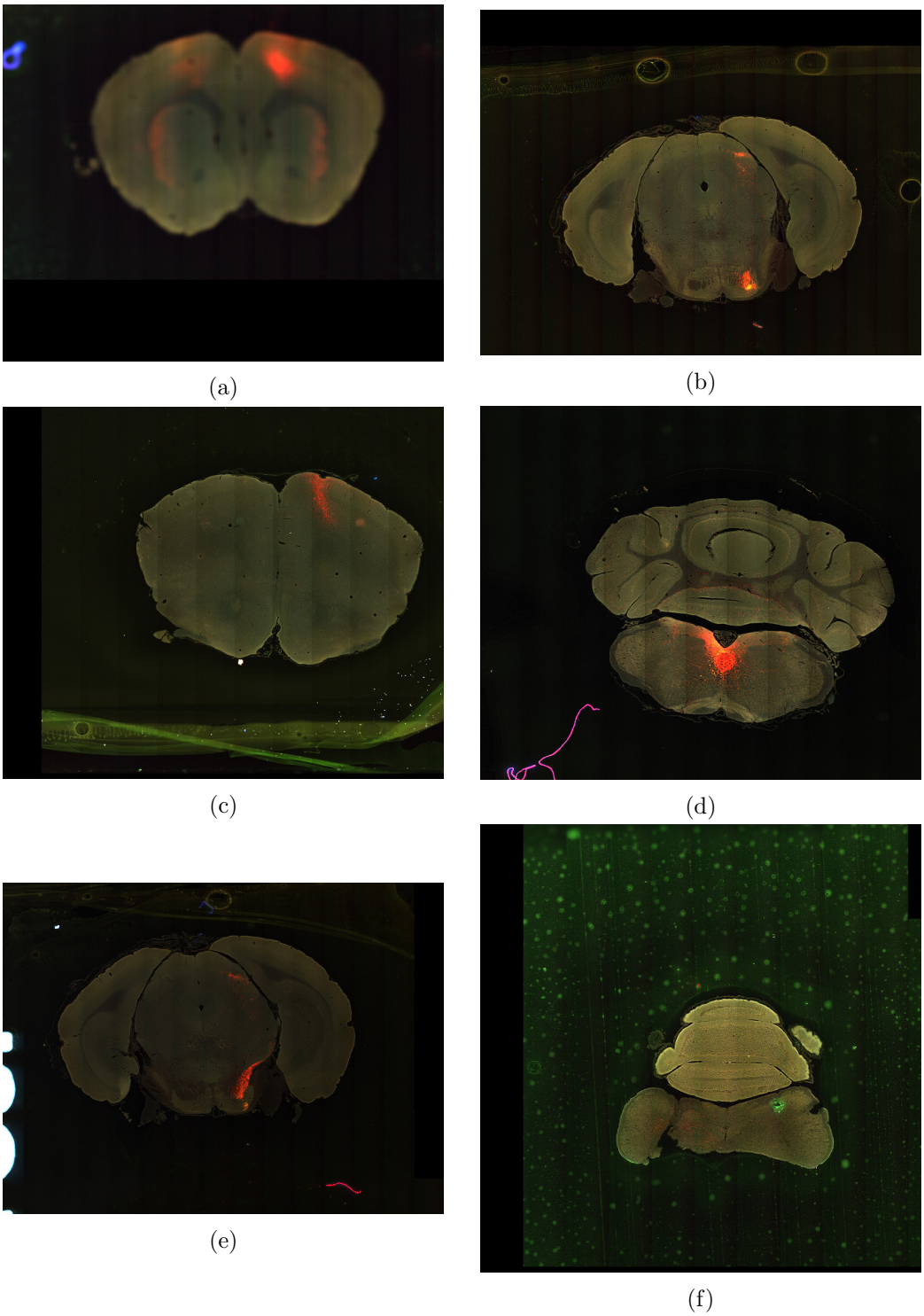
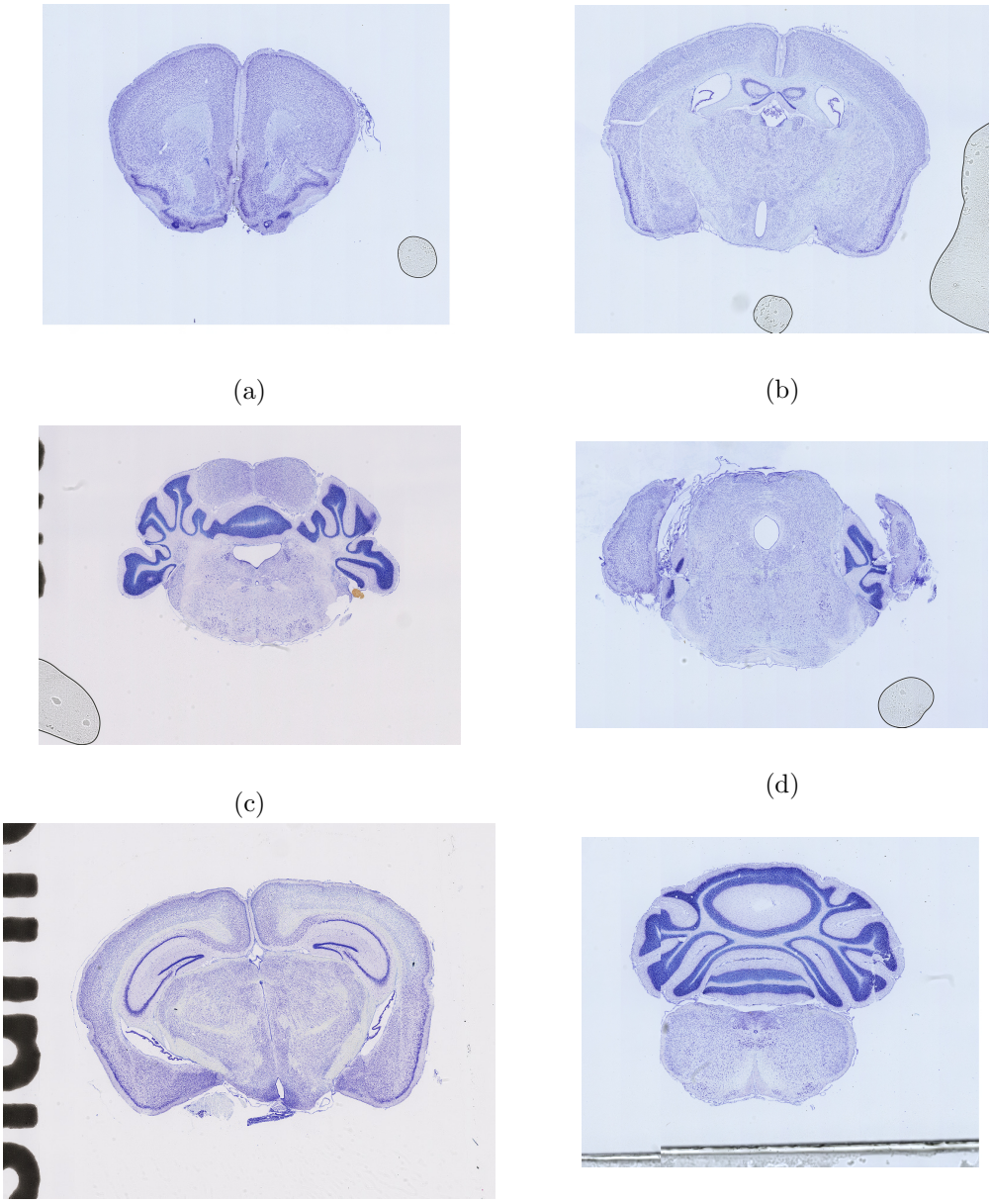


Figure 22: Examples of different kinds of noise and artifacts in the original F images. Such artifacts do not affect the tissue but in order to analyze the images we have to crop the images so the artifacts do not affect the subsequent analysis.



(a) (b) (c) (d) (e) (f)

Figure 23: Examples of different kinds of noise and artifacts in the original N images. Such artifacts do not affect the tissue but in order to analyze the images we have to crop the images so the artifacts do not affect the subsequent analysis.

is as smooth as possible and any peaks to be mostly from the edges not from noise. We also want to use the filter to remove peaks from the injection site so the edge detection later on does not mistake those pixels for edges. One important point is that our dataset also includes images that are out of focus as well which creates problems with the edge detection. That means that we have to strike a balance between having a filter kernel that is big enough to remove any noise or sharp peaks of data in images that are in focus but at the same time to not over-blur the out of focus images so much that the edges of the brain are not visible anymore. So we have to pick the smallest size that removes all noise and peaks.

In Figure 25, we show an F image with a lot of noise that needs to be removed in order to find the edges of the brain sample. In the image we also notice an obstruction artifact on the bottom of the image, which has likely caused the sample to be out-of-focus. This image is an example of how carefully the filter size has to be selected, if we pick a very small kernel the noise and the artifact will not be removed sufficiently. On the other side, if we pick a larger kernel the sample edges will be smoothed to the point that the edges will not be visible anymore (since the image is already out-of-focus). The noise is not only limited to F images, as we see in Figures 26, 27 the N and IHC images present the same difficulties as well so the noise has to removed from those images as well before we proceed further with the edge detection.

We will use the image shown in Figure 28 as an example to show the different stages of the edge detection process. Because of the large size of the images we can expedite the image edge detection by using a grayscale image which is shown in Figure 29. Using the grayscale image is not going to change the quality of the code, since the grayscale includes all the colors.

The median filter that works the best for our F data has a small 7×7 pixel size. The choice of the kernel depends on the size of the background noise that we want to remove and is confirmed with visual inspections.

After we use the median filter we notice that the very few specks in the background disappear and the sample, although slightly blurred, has kept its original shape. Figure 30 shows the resulting image after we use the median filter.

3.2 Sobel filter

After we have removed the background noise and artifacts we will use an edge detection filter to enhance the edges of our sample. The Sobel filter is only a 3×3 pixel size so if it is not appropriate for the data we can not change the mask size, so a different filter has to be used. The image after it is filtered is shown in Figure 31. A threshold has to be applied to that image and the outcome binary image is the one we see in Figure 32. The mask works very well for our data and easiliy finds the the edges of the sample. If all our images were ideal (no noise, damage or artifacts) then we would not need more steps. But our images are not all as ideal as this one so the following steps, although they might not seem that are needed for this image, would be necessary for other images.

We additionally tested other methods for edge detection for our dataset such as the Canny filter [5], Prewitt operator [39] and Roberts cross operator [41]. Although all the different methods yielded satisfactory results for most images, all methods except the Sobel filter failed to detect the tissue edges of many images that were out-of-focus. Since the goal of the project is to detect the out-of-focus images, it is important to achieve proper edge detection in any particular group of images. Therefore for the purpose of this work the Sobel filter will be used for edge detection.

3.3 Mathematical morphology operations

At this point we have to use an operation that in mathematical morphology is called *dilation*. Applying this operation to the image will expand the data points from Figure 32 ([21]), so every single pixel will turn into a new shape as is dictated by a *structuring element*. Visually this will result in an image where the objects are much thicker and any gaps between lines in the contour of the sample will close. The choice of the shape of the structuring element depends on the size and shape of the objects in our image. In our case the outline of the brain has a rounded/smooth shape so if for example we use a square or diamond shape we might use some of the roundness of the shape, that is why we will use a circular structuring element. When we apply the structuring element the image will have a much thicker outline and some lines that were far away from each other (like on the bottom right) have connected (Figure 33).

The next step is called region filling and it will allow us to fill in the area

inside the perimeter of the sample. The result is shown in Figure 34

Now that we have an enclosed area we will "thin" out the image by using the opposite of dilation, called *erosion*. The reason for that is because when we dilated the image we made some areas artificially thicker (which was needed because we want to connect some areas that look disconnected) and now that we have the filled in area we need to correct for that. This step is not absolutely necessary for the specific calculation but in the case that we want a very exact outline it will be necessary. Since we are doing this calculation to get the the edges of the sample in order to remove most of the background, a few pixels of extra padding in the final edge sample are not going to cause any problems with our calculations. Nevertheless, for the sake of precision we will add this step. The result of the erosion is shown in Figure 35, notice that the outline is a little thinner now.

If we locate the data points of the outline we can trace the edges of the image. Figure 36 shows the estimated perimeter of the sample overlayed to the original image. We notice that there is more than one areas, the assumption is that the sample is always the largest region.

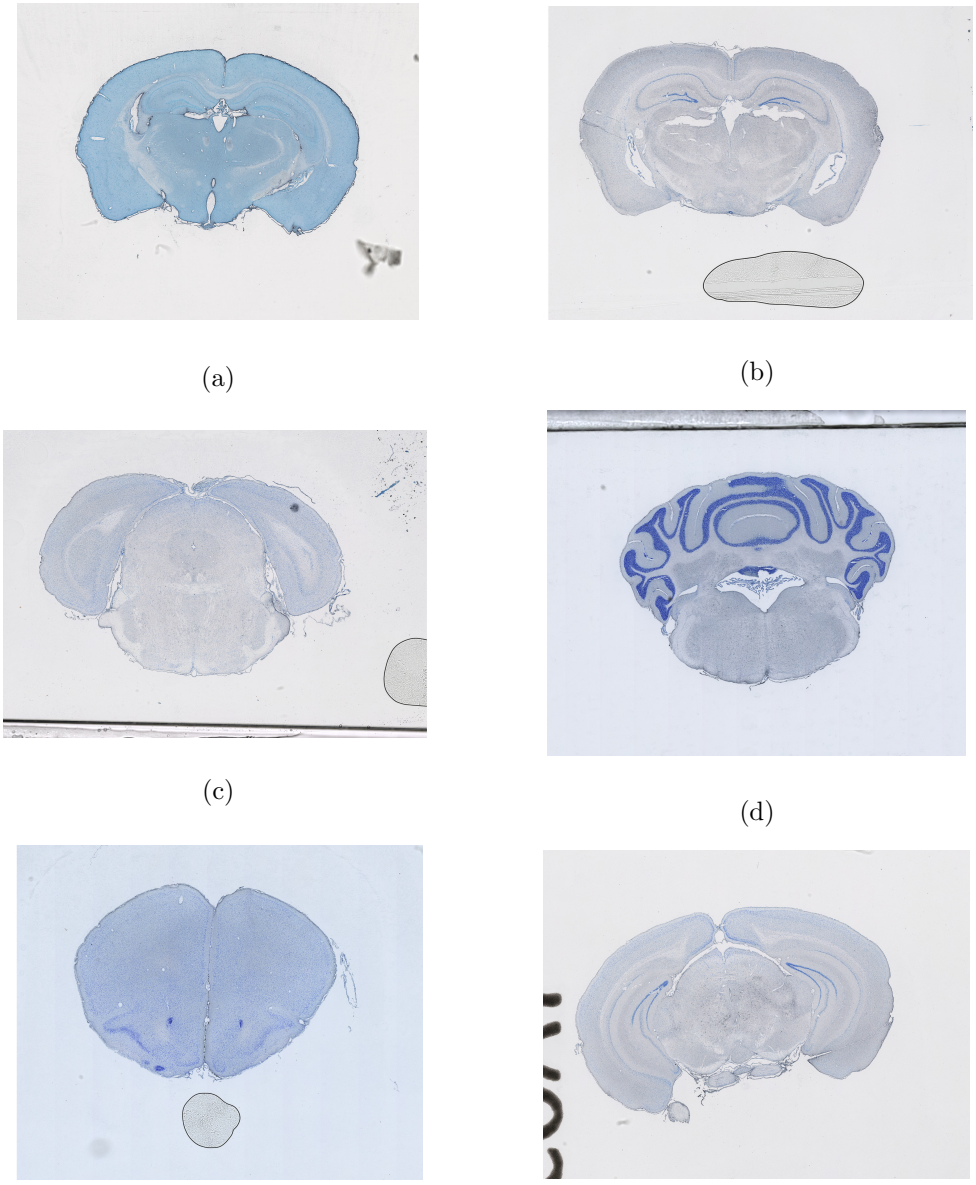
The edge detection method we described has several steps that might not seem necessary for an image like the one we used as an example since such an images is free from severe noise, air bubbles, letters or other artifacts. We will show a few examples of images that the edge detection deemed all the steps necessary for a successful outcome.

Figure 37 shows that the initial blurring is necessary to remove some of the background noise that can influnce the edge detection but since the original image is also out-of-focus we can not blur the image too much. Notice that after we apply the Sobel filter, one of the image stripes is traced as well but since it is inside the image that problem is resolved after the fill the region. Figure 38 shows another example where we display the need for the expansion of the points after imposing the Sobel filter. As we see in on the bottom of Figure 38d there are gaps so if we tried to fill the image without dilating the white data points then the two rounded areas on the bottom will not be included in the outline of the image.

An example of an N image edge detection is shown in Figure 39. Those images have a smaller contrast than the F images so the median filter that will be used will be in a smaller neighborhood than the F images, otherwise the sample edges will not be strong enough to be detected by the Sobel filter (especially for OOF images). Despite that difference, the N images also require all the steps that have been enumerated in this section for edge

detection. Notice that after applying the Sobel filter (Figure 39d there are many more edges in the image, even inside the tissue sample, unlike the F images, Because of the low contrast between the sample and the background, the Sobel filter is more sensitive to other background artifacts as well. The original image (Figure 39a) has a white straight line on the left side that is part of the image (the black border was added for emphasis) that the program identifies as another region as well. Additionally it has several air bubbles that the program identifies as additional "blobs". The program identifies as the tissue area the largest blob so these smaller areas will not be misidentified as tissue.

An example of an IHC image edge detection is shown in Figure 40. This image has a few artifacts and a patch of white area on the bottom left but the biggest issue are the letters from the slide on the left side as shown in Figure 40a. After applying the Sobel filter the letters are also identified as a region as a whole (Figure 40d but since their area is smaller than the tissue region and connected to an edge the image will not identify them as the tissue but output the large "blob" as such. The filter also picks up the the white block on the bottom as another region but due to the shape and location of the area it will not be misidentified as a tissue.



(a) (b) (c) (d) (e) (f)

Figure 24: Examples of different kinds of noise and artifacts in the original IHC images. Such artifacts do not affect the tissue but in order to analyze the images we have to crop the images so the artifacts do not affect the subsequent analysis.

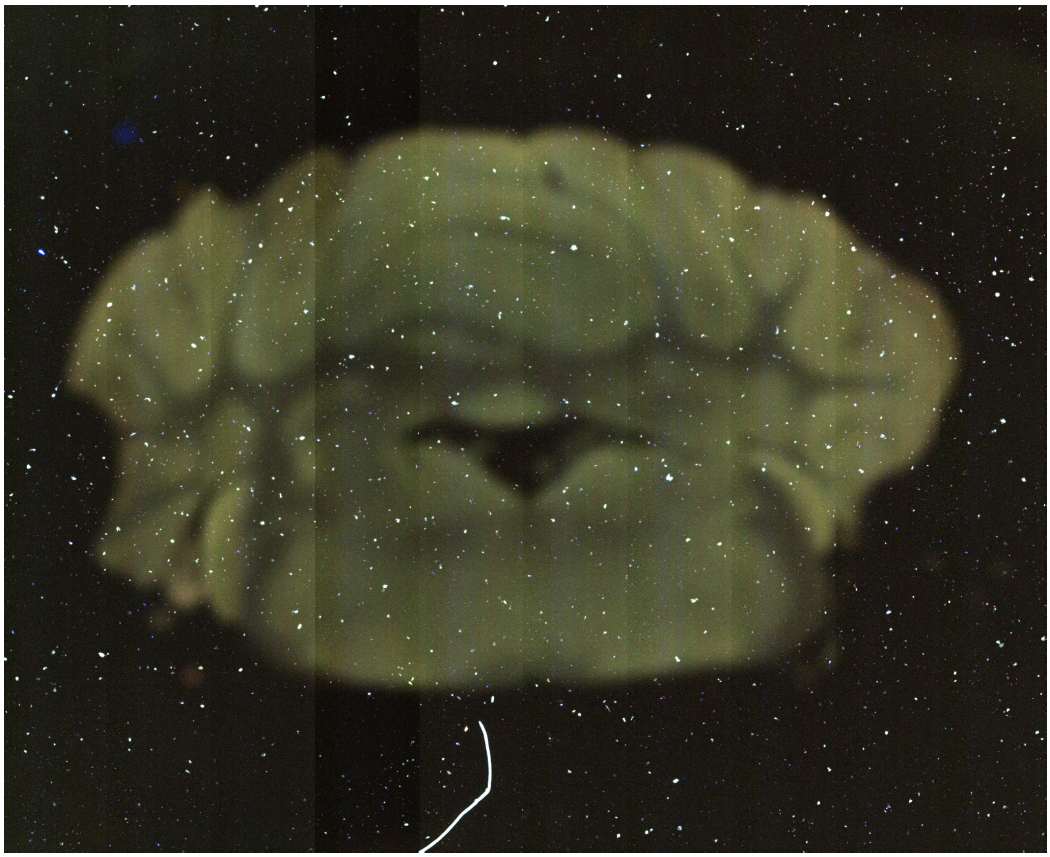


Figure 25: Fluorescent image with noise similar to the salt and pepper noise we discussed in the previous chapter.

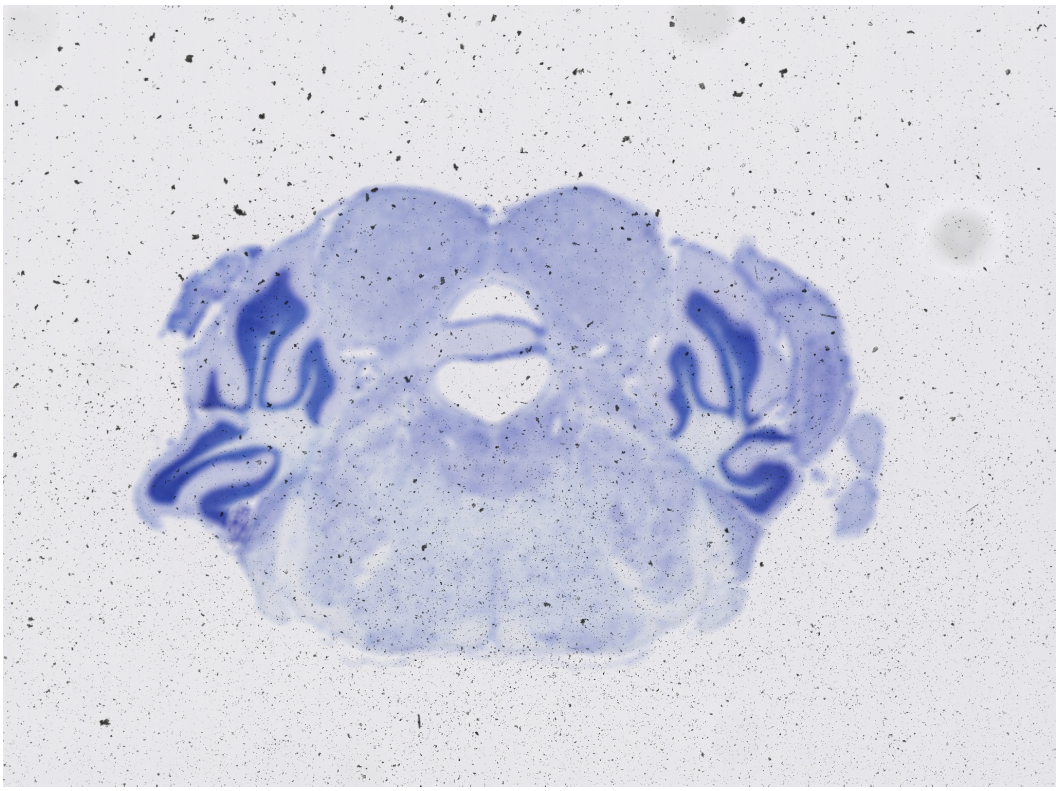


Figure 26: Nissl stained image with a lot of noise similar to the salt and pepper noise we discussed in the previous chapter.



Figure 27: IHC image with noise.

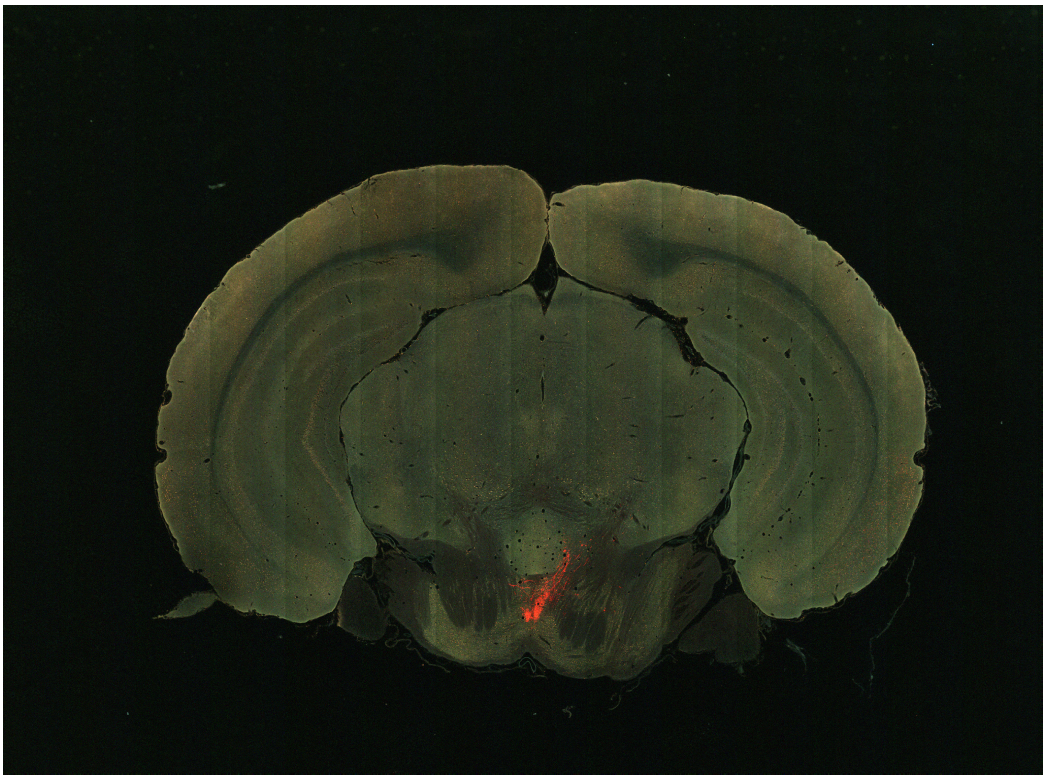


Figure 28: Original image that we will use to show the edge detection process. This is image 149 in stack 2941.

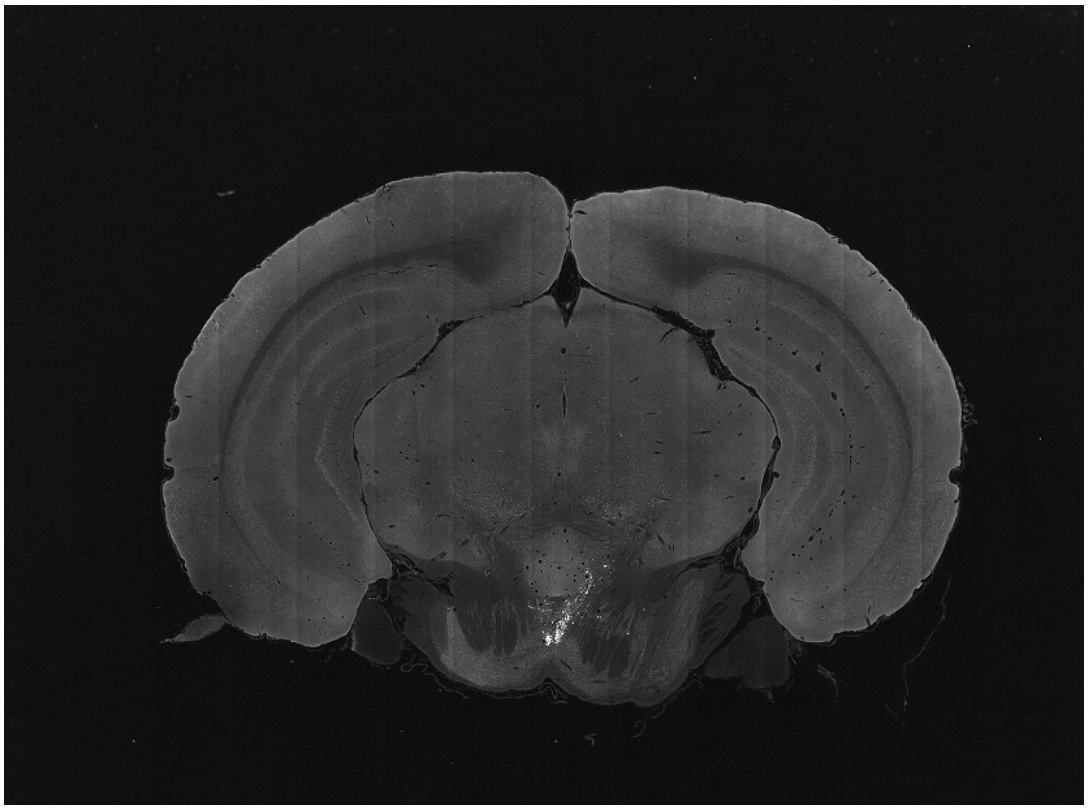


Figure 29: Grayscale original image.

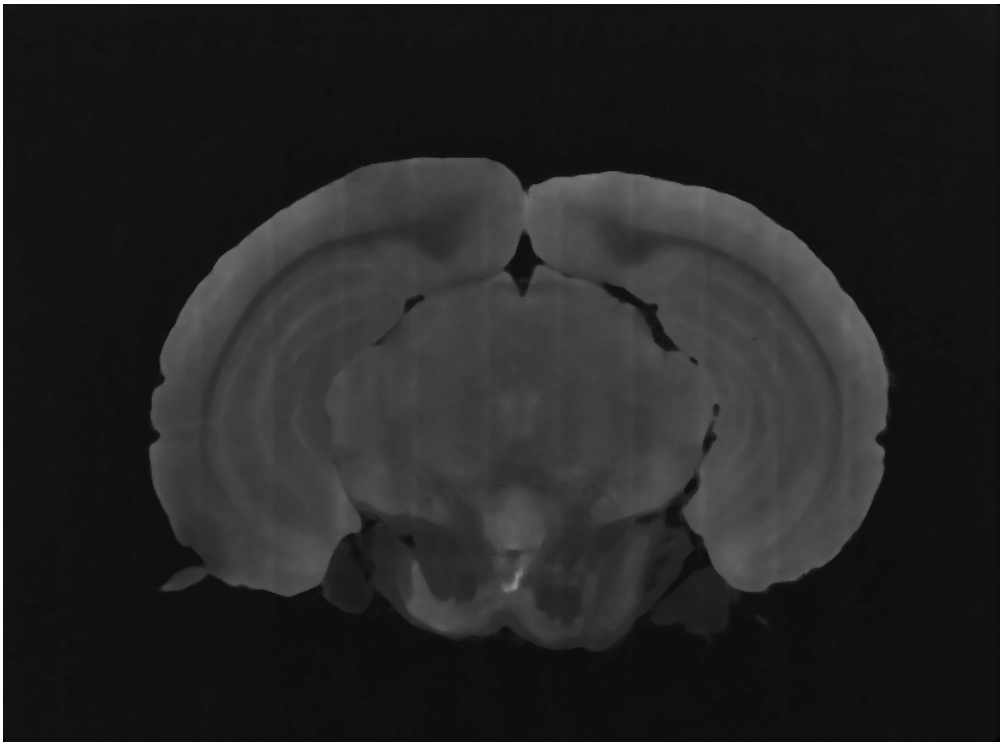


Figure 30: Grayscale image after using the median filter .

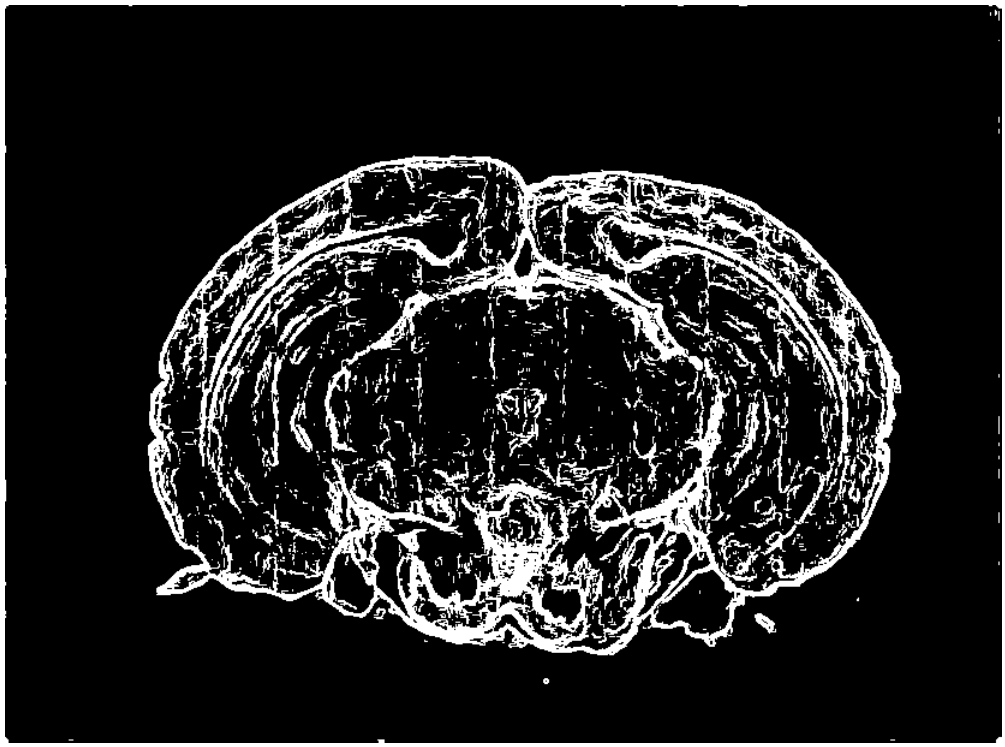


Figure 31: Grayscale mage after the Sobel filter is applied.

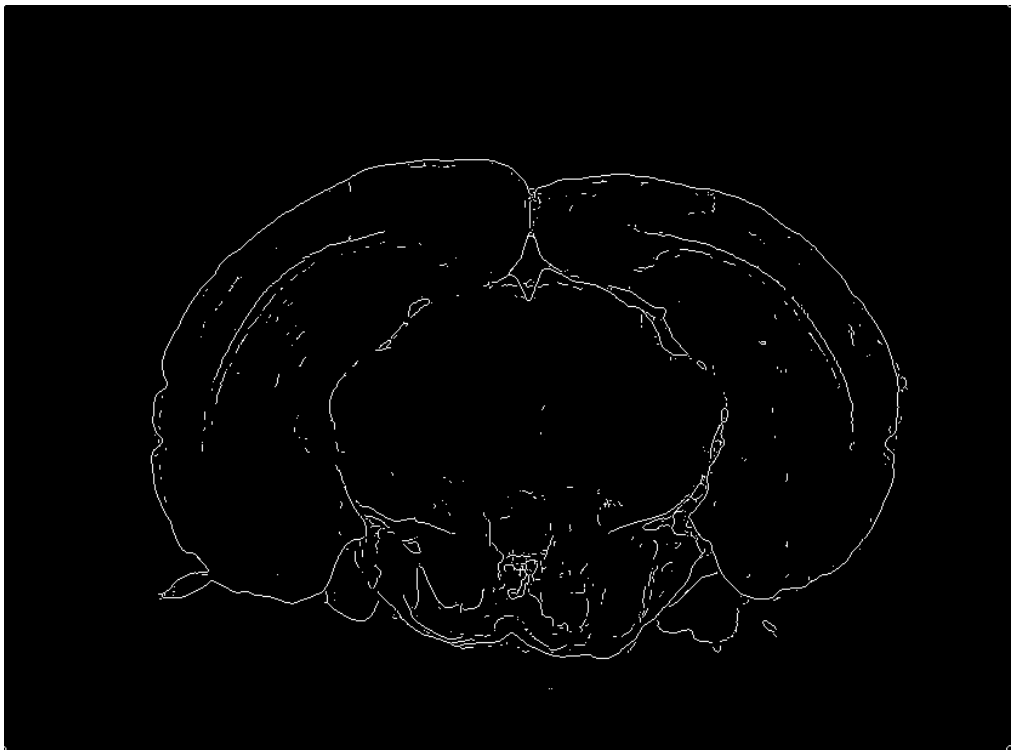


Figure 32: Binary image after the threshold is applied to the Sobel filtered image.

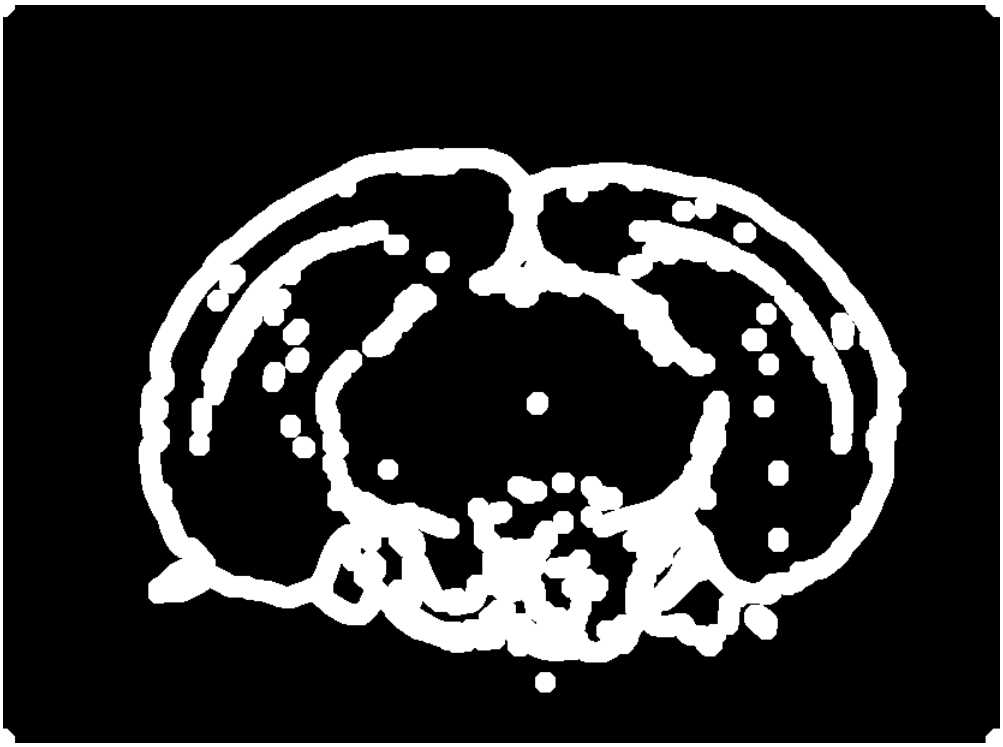


Figure 33: Grayscale image after dilation with a rounded shape is applied.

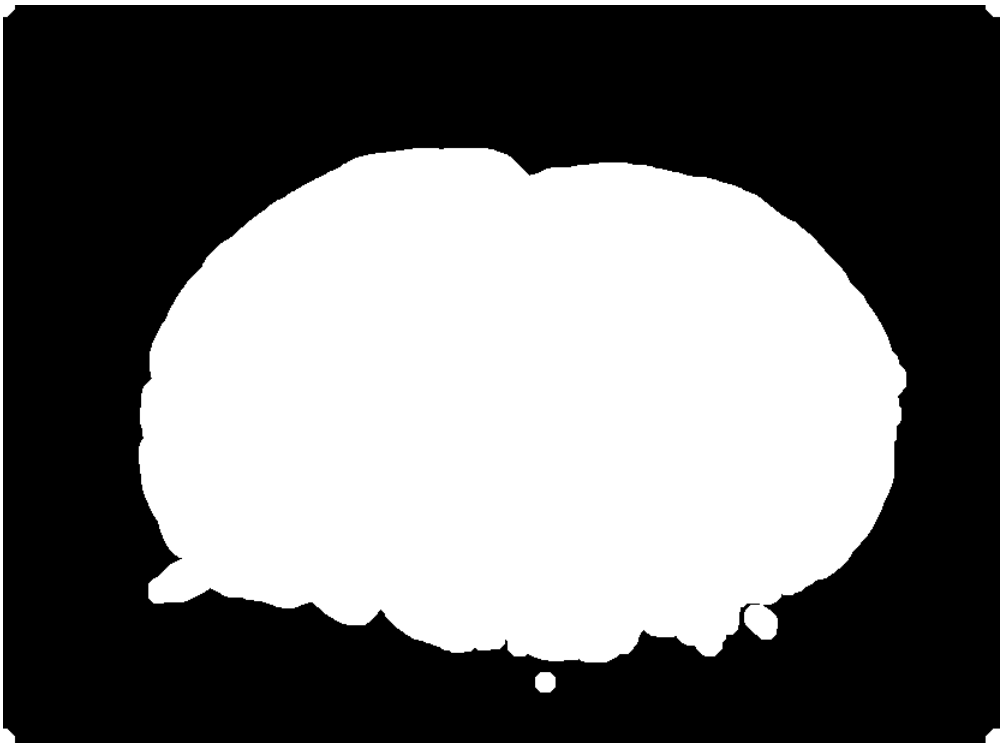


Figure 34: The gaps inside the perimeter have been filled.

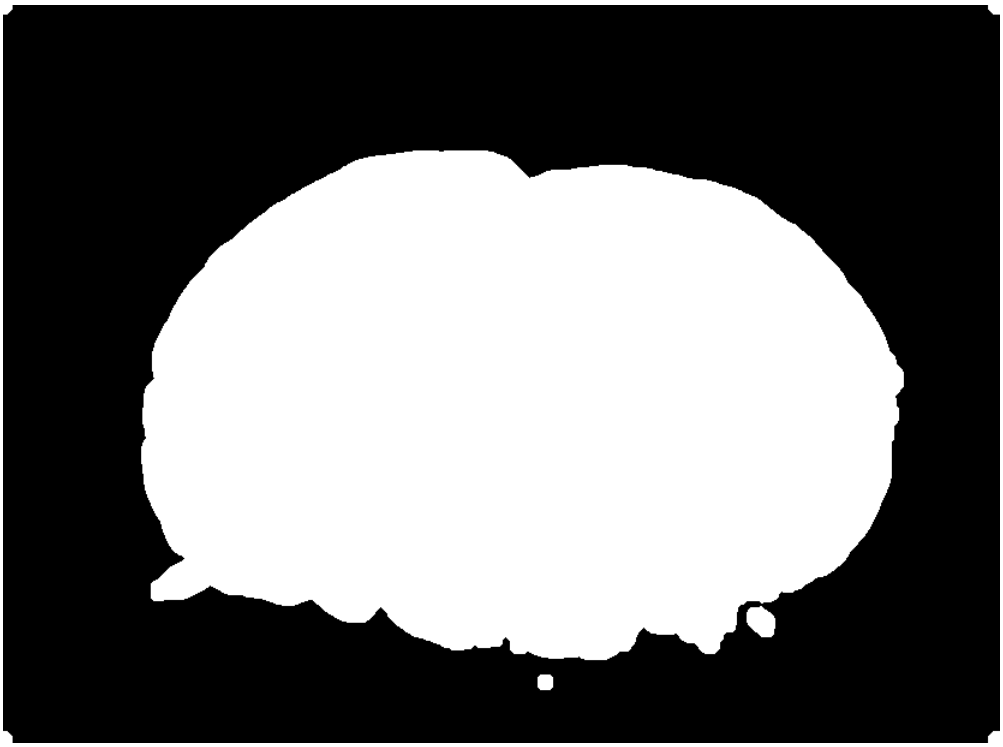


Figure 35: Erode the filled image.

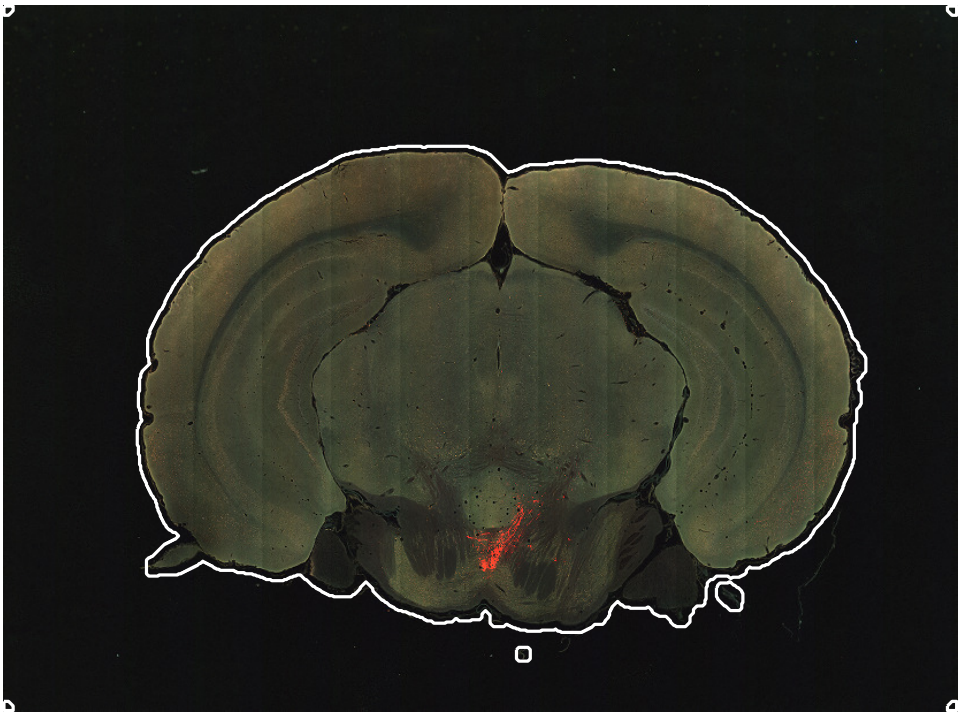
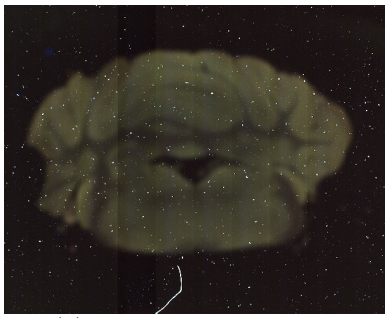
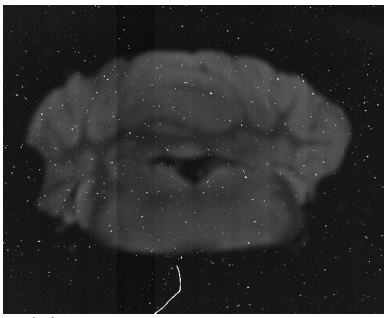


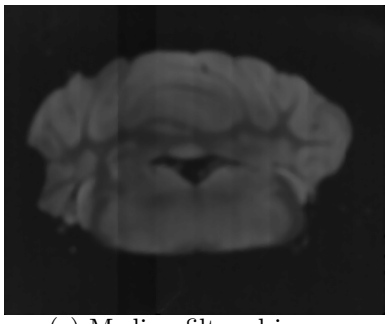
Figure 36: Estimated sample edge overlayed to the original color image.



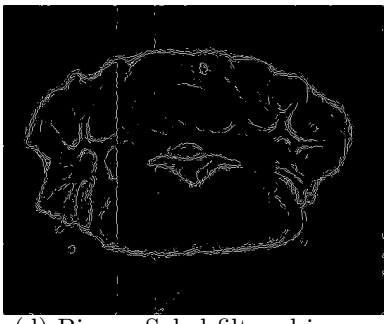
(a) Original Color image



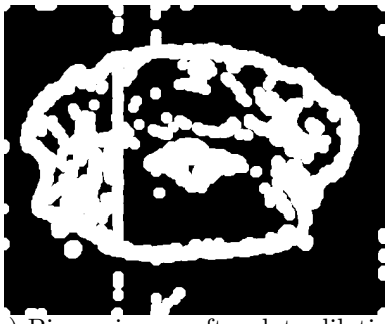
(b) Original Greyscale image



(c) Median filtered image



(d) Binary Sobel filtered image



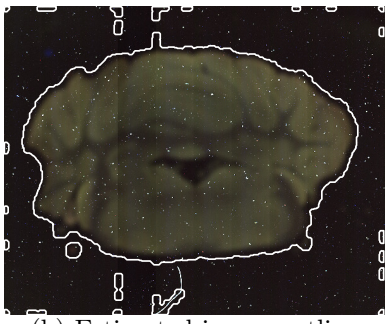
(e) Binary image after data dilation



(f) Binary image after region fill

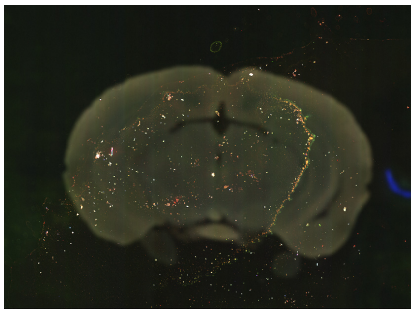


(g) Binary image after data erosion

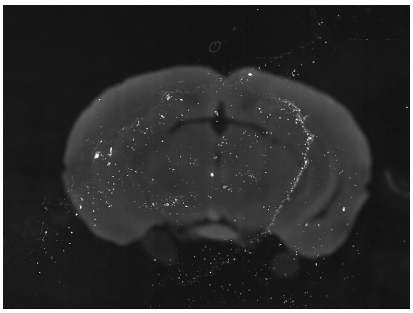


(h) Estimated image outline

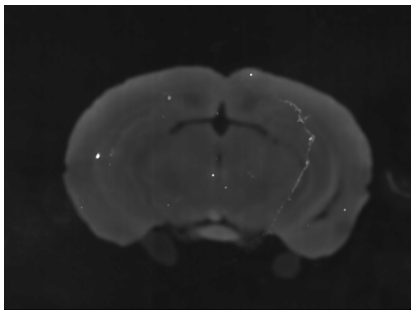
Figure 37: Example of edge detection on an F noisy and out-of-focus image



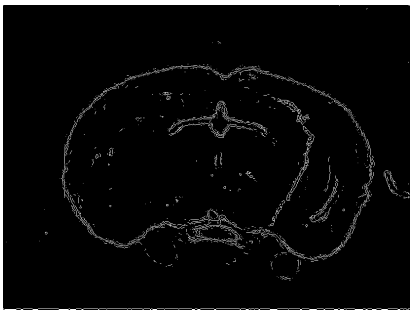
(a) Original Color image



(b) Original Greyscale image



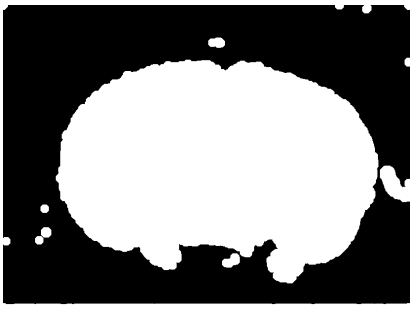
(c) Median filtered image



(d) Binary Sobel filtered image



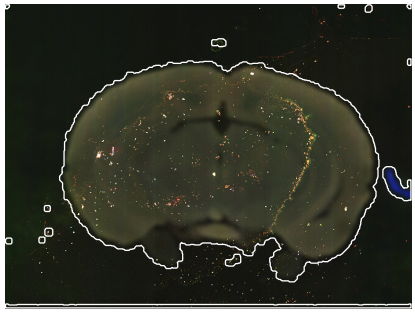
(e) Binary image after data dilation



(f) Binary image after region fill

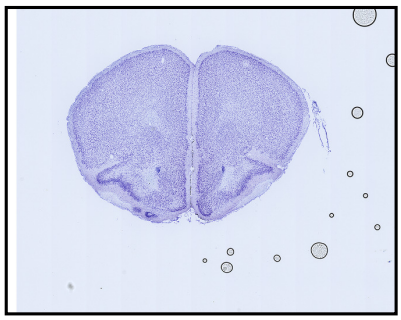


(g) Binary image after data erosion

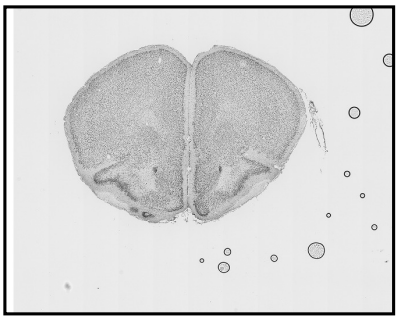


(h) Overlayed image outline

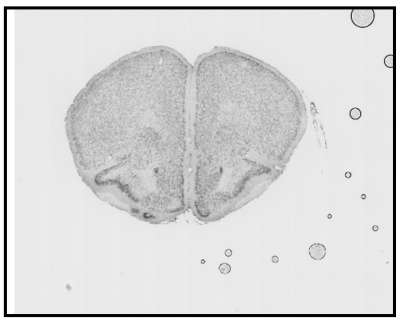
Figure 38: Examples of edge detection of an F noisy image



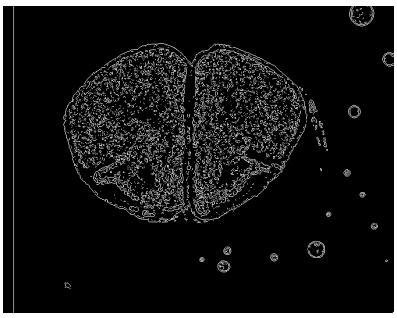
(a) Original Color image



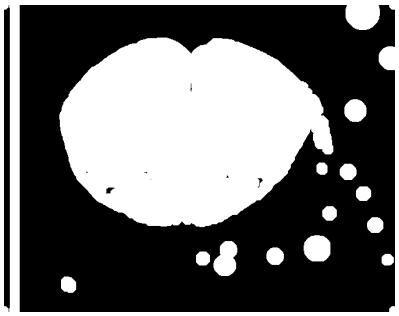
(b) Original Greyscale image



(c) Median filtered image



(d) Binary Sobel filtered image



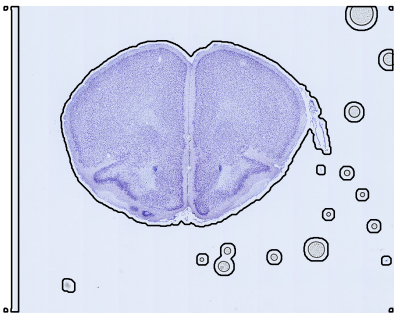
(e) Binary image after data dilation



(f) Binary image after region fill

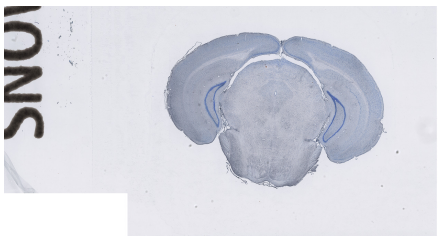


(g) Binary image after data erosion

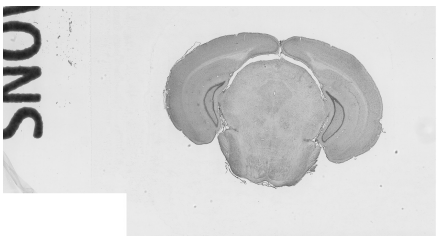


(h) Overlayed image outline

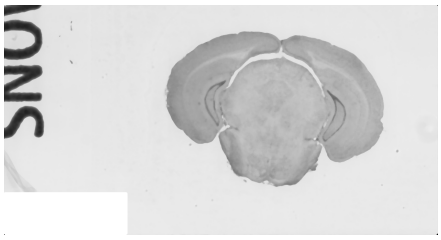
Figure 39: Examples of edge detection of an N image with several bubbles



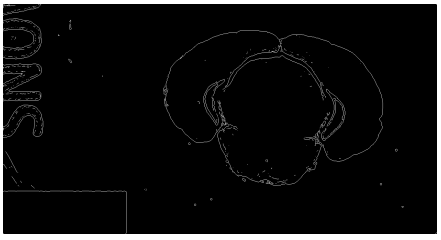
(a) Original Color image



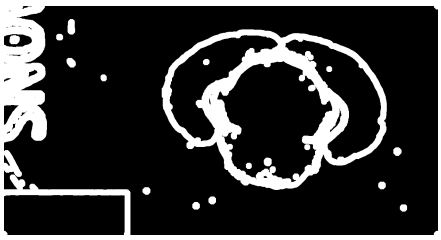
(b) Original Greyscale image



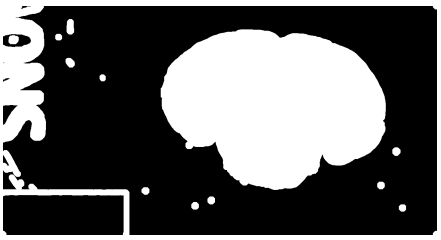
(c) Median filtered image



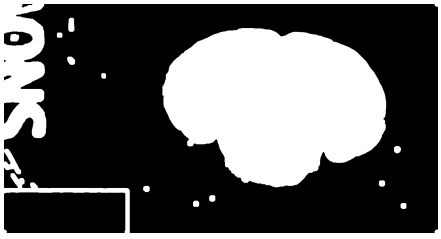
(d) Binary Sobel filtered image



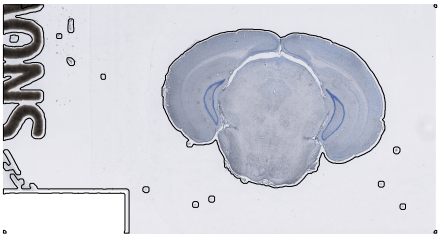
(e) Binary image after data dilation



(f) Binary image after region fill



(g) Binary image after data erosion



(h) Overlayed image outline

Figure 40: Examples of edge detection of an IHC image with a few artifacts and letters from the slides

4 Automatic detection of out-of-focus images

4.1 Introduction

A large part of the image processing work flow in brain imaging is the quality control which is typically done visually. The samples can present several problems while they are prepared, for example there can be damages from the slicing of the sample and sections can be torn, cracking, missing or folded. Any of these issues can be easily spotted and removed from the sample. Each sample section adheres to a tape that is then placed onto a glass slide. This manual procedure can create additional issues like folded tape and sample, air bubbles between the tape and the slide. Additionally the slide can also introduce artifacts and noise due to contamination from foreign materials like dust. These issues can also become apparent by any user and easily removed and replaced if needed. Some examples of different kinds of problematic images are shown in Figures 41, 42, 43. Such images can be easily identified by even an inexperienced observer and removed from the sample.

The next step is digitizing the slides using a digital scanner which may result in images that are out-of-focus (OOF). Although the scanner is equipped with autofocus it can fail due to a selection of focus points that are not in the same tissue height or because of using noisy focus points. Classifying an image as out-of-focus is an important step of the quality control and it is crucial for producing an unbiased dataset. Such task is not only very time consuming but its sucess depends on the skills of the observer. There are many images that are clearly out-of-focus (Fig. 44, 45, 46), which can be characterized as such with high confidence by any observer. However, some images can be either slightly out-of-focus or partially out-of-focus and those are challenging to detect through visual detection. In the former case, in order for the observer to find whether the image is out-of-focus, a magnified view of the image has to be examined in order to determine its sharpness (Fig. 47) which can be time consuming.

Such OOF images are also part of the N images dataset. Because these particular images have lower contrast than the F images many times it is difficult to access whether the image is OOF without increasing the magnification and inspecting several tiles to determine whether the image is OOF. An example of such an image is shown in Figure 48. When visualizing the image as a whole (Figure 48a there is no indication that the image is OOF. But magnifying different regions (48b,48c) of the image reveal that the im-

age is OOF. The IHC images also include samples that when inspected in regular magnification look in focus but when magnifying specific regions, the fact that they are OOF is revealed. An example of such an image is shown in Figure 49. An inspection of that level for all images of just a particular stack will take a considerable amount of time and if we take into account the massive number of all stacks then the amount of time it takes to complete such an inspection for all images will be impractical.

In the partially out-of-focus case, in order for the observer to find the part of the image that is out-of-focus, several small tiles of the image have to be closely examined (Fig. 50). In the partially out-of-focus case there is a high chance that an observer will miss the out-of-focus region because usually a visual inspection will not cover all the pixels of an image but a random sample of several areas. This case is not only time consuming but it also needs a skilled observer in order to evaluate enough image parts so that they do not miss the area of the image that is out-of-focus. Both cases can be lengthy and laborious and the skill of the observer will determine whether the identification of OOF images is complete and accurate. Utilizing an automated way to identify the OOF images will improve the speed of the analysis and avoids mistakes that can be introduced due to human error or lack of skills and experience in identifying these issues.

It is clear based on these representative examples that a visual inspection of determining OOF images is not advisable. The main reasons are:

1. Visual inspection for OOF images can be subjective, unreliable and tedious so we do not want to insert in the work flow such a task, because it can possibly allow OOF images to contaminate an otherwise clean dataset. Creating an algorithm with a metric that classifies OOF images according to specific criteria will resolve this issue.

2. Visual inspection for determining OOF images is impractical. Due to the massive amount of images that we have such a task could be very prolonged. Creating an algorithm that can be implemented automatically for different stacks without user intervention and in parallel with each other will allow us to complete this task in a reasonable amount of time.

In order to classify whether images are out-of-focus we have to devise an automated method that can be done fast without user intervention.

4.2 Out-of-focus image metric

The images contain the brain tissue but also a background that does not include any information that is relevant to the focus calculation. Additionally, often there are artifacts in the background (like moisture, bubbles, unattached tape on the slide as shown in Fig. 41, 42, 43) and by removing these artifacts the program differentiates the brain tissue from the background more accurately. Therefore, the background of the images has to be removed. Furthermore, by removing the background the image analysis is faster because the image to be analyzed is smaller. This is a great advantage because, as it will be discussed later, the analysis is time intensive because of the large size of the images so any opportunity to limit the volume of the data should be taken. In the previous chapter we discussed in detail the edge detection steps that have to be taken so we will not repeat it here.

Several algorithms [36] were tested for the purpose of identifying out-of-focus (OOF) images. In order to test all the different methods in an appropriate amount of time we used only five central regions of interest in each image, with a size of 1000×1000 pixels for 10 different brain image datasets. The goal was to eliminate the methods that did not produce the out-of-focus images that had been previously identified via visual inspection. When the testing was completed only three methods produced consistently accurate results for all brain image datasets (a run was labeled as successful when the out-of-focus images that the program identified included all of the out-of-focus images that had been previously identified through visual inspection, even if the program results included images that were not previously identified as OOF from the visual inspe ction). These three methods were the Gaussian derivative [20], the Steerable Filters (SF) method [29, 17] and the Tenengrad method [25].

We repeated the testing with just these three methods for 10 different brain image datasets but this time we included the whole tissue of each image. The method that produced a reliable classification was the SF method. This method synthesizes different Gaussian Filters with several orientations and uses a linear combination of these orientations to calculate a figure of merit that we will later call a focus value.

An image can be considered as a function where $f(x, y)$ is its intensity at position/pixel (x, y). If the image data points are an array of samples of a continuous function of the image intensity, then the gradient is a measure of change of that function. First, a Gaussian mask must be created in order to

filter the image.

The 2D Gaussian kernel is defined as:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{7}$$

The partial derivatives of Eq. (7) are:

$$G_x = \frac{\partial f}{\partial x} = -\frac{x}{\sigma^2} G(x,y) \tag{8}$$

$$G_y = \frac{\partial f}{\partial y} = -\frac{y}{\sigma^2} G(x,y) \tag{9}$$

It can be noted that Eq. (9) is the same function as Eq. (8) rotated by 90°. The previous two filters can be u sed as basis filters in order to construct filters of an arbitrary orientation angle θ. A new filter of an arbitrary orientation can be constr ucted by the linear combination of these two filters.

$$G_\theta = \cos(\theta) G_x + \sin(\theta) G_y \tag{10}$$

In order to find the maximum response of the image, it must to be convolved first. If C_x and C_y are the convolved images of G_x and G_y respectively, then the linear combination of the filters at an arbitrary angle θ can be written as:

$$C_\theta = \cos(\theta) C_x + \sin(\theta) C_y \tag{11}$$

For our purposes, the angles [0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°] are used and the maximum value within all the different angles, $C_{\theta,max}$ for all image pixels is calculated. The final focus value (FV) for each image is the mean value of all the focus values of each individual pixel within that image. This final focus value will serve as the discriminating value in classifying whether an image is in focus or not.

The use of the SF method requires a knowledge of the neighborhood x, y pixel size that must be used to construct the Gaussian. This Gaussian window size (or kernel) was derived from our data. Visual inspection was used to select one OOF image and one sharp image from the same stack for that purpose. The ratio of the focus value (FV) of the two images was compared for several kernels. The goal was to find the optimal kernel size; where the ratio of the FV of a sharp images versus an OOF image was maximized.

The FV ratio of some representative fluorescent, Nissl stained and IHC OOF images versus sharp images of different brain image datasets are displayed in Figure 51 and the optimal kernel based on this plot is found to be a 15×15 pixel window. The F brain image datasets have higher focus value ratios due to the higher contrast compared to the N and IHC image datasets.

4.3 Results

The program was applied to several brain image datasets and the results were then compared to a visual inspection to confirm that the program correctly identifies OOF images. In order to distinguish between the in focus and OOF images the focus value (FV) for each image was used. The method applied requires the results from an entire brain image dataset each time it is employed (approximately 300 images). There is no threshold that can be applied for each image separately. We can not apply a threshold to determine the outliers because of several images

- Finding a threshold would require manual inspection of several images to determine a value that would be appropriate as a threshold for OOF images. Such a task is prone to errors due to the fact that such a threshold has to be empirically defined by inspecting and comparing a substantial number of images. That would be very time consuming and it would introduce a large uncertainty.

- A threshold that will be accurate for all the brain image datasets might not be possible due to the difference in illumination, background and stains between the stacks.

Therefore, the identification of the OOF images was done by locating the outliers of the FV indices dataset for each stack. Figures 52, 53, 54 show images from one particular stack labeled 2888. There are a few image IDs that are missing, possibly removed during the imaging process.

Before we employ the automatic way of locating the OOF images, we visually identify them in the particular stack so that we can compare to the automatic way.The image IDs of the OOF images are:3,21,24,53,63,81,84,102 and 150. In order to find the OOF in an automatic way we will start by estimating the FV (as described in the previous section) of all images in each stack. In our example we are using stack 2888 to demonstrate as an example. The normalized FV versus the image ID for stack 2888 is shown in

Figure 55a. We can notice that there are clearly two groups of data points, the main data group (top) and the outliers (bottom) which are the images we visually indentified as OOF and that we marked with their individual image ID. If we only had to locate the outliers for a few stacks finding the outliers visually from this plot would be an option but because of the massive amount of images and plots that we have to inspect we have to find an automatic way to identify the outliers of the plot which we assume are the OOF images.

In order to trace the data points of the main data group we will use a *moving median* which is the median calculated from a specific subset of points, in our case a small window of neighboring data (Figure 55b). The choice of median is done because the average is influenced a lot by the outliers, the median is more stable. Since all stacks are a little different in illumination, background, injection sites, number of images, sample sizes, we can not built a model to fit the data so the separation of the outliers has to be data-driven in an individual way for each dataset. For every data point we estimate the difference between their FV value (y) and their moving median value (y_m) and compare it to the standard deviation (sd) of the data. We label as "OOF candidates" the data points where the condition $|(y_m - y)| > sd$ is satisfied. This method is similar to the *Hampel identifier (filter)*([35], [12]), which replaces the central value of a window with the median value when that value is not close to the median. The data points that are replaced are considered outliers.

The graph in Figure 55c shows the FV data in black color for the particulat dataset. The moving median is in blue solid line while the points that are in red circles satisfy the above requirement as OOF candidates and were also confirmed as OOF images through visual inspection as we discussed earlier. Our algorithm has confirmed that this brain dataset includes 9 OOF images. Eight of them are shown in Figure 56, it is evident that these images are OOF.

But looking at the section 53 (Figure 57a the image does not look like OOF but zooming in two areas (Figure 57b, 57c) and looking at the closer it it evident that this image is OOF.

The same approach is used for the N and IHC brain image datasets. An example of an N image dataset is shown in Figure 58. In the particular stack there are three images that were confirmed as OOF by the program and through visual inspection (sections 22, 121 and 142). One of the images that was indentified as OOF is shown in Figure 59 but the other two images that were identified as OOF by the program would require from the

person inspecting to magnify in order to find that the images are OOF. To demonstrate this we will show two images from the same stack, one is OOF (section 121) and the other is not OOF (section 122) which we have cropped for convenience. Both these images have been evaluated visually and by the program. Looking at these images as a whole in they both look like they are not OOF (Figure 60). When we zoom in we have a different story, the difference between the two images makes the OOF image seem evident as we see in Figure 61. So we can see how easily an unskilled observer can miss a OOF image while an experienced user who will look at different areas of an image will add a substantial amount of time and effort.

The last OOF image that was identified as OOF by the program (142) has to also be magnified in order to study if it is OOF or not. Looking at these images as a whole they both look like they are not OOF (Figure 62) but their difference is evident when we zoom in(63.

An example of an IHC stack is also shown in Figure 64. The algorithm identifies as OOF images the ones marked with a red $*$ symbol, sections 40,115,116,117,187,188 and 189. All these images do not look out of focus when inspected as a whole but a closer inspection to magnified regions show that the particular images are blurry. Section 40 was identified as OOF, while 41 was not, looking at both images we can not discern a difference in focus (Figure 65). When magnifying an area from each image (we have tried for the areas to be close enough) the difference in focus is evident (Figure 66) so the algorithm correctly has identified section 40 as OOF.

We will demonstrate the other OOF images in the same way. Images from sections 115,116 and 117 were identified by the program as OOF, compared to image 114 that was not identified as OOF from the program , inspecting them as a whole we can not tell a difference in their focus focus (Figures 67, 68, while when we enlarge specific areas (that we tried to be as close as possible between the different images), it is evident that the program was successful in the identification of 114, 115 and 116 as OOF as well (Figure 69).

The last group of images that were identified as OOF from this stack is comprised of sections 187, 188 and 189. We will compare those visually with image 190 that is not OOF. The four images are shown as a whole in Figures 70, 71, while a specific area from each image is magnified and shown in Figure 72.

On a personal note, the brightfield images are more difficult and take longer to inspect visually because they do not have such a high contrast as the fluorescent images, so this method is even more valuable for those images.

4.4 Visual inspections and algorithm comparison

In order to verify that the results from the algorithm are accurate, numerous brain image datasets were tested in parallel to visual inspections. The visual inspection was done in independently before the program results were known.

In Table 4, the results of these brain image datasets during the testing phase are shown. We characterize as *false positive* (FP) images that the program identifies as OOF but visual inspection does not support this characterization. An image will be characterized as *false negative* (FN) if the program does not identify it as an OOF candidate but the visual inspection classifies that image as OOF image. Additionally, an image is characterized as *true positive* (TP) when both the program and the visual inspection characterizes it as OOF while it is a *true negative* (TN) when both the program and the visual inspection characterize it as in focus.

	F images	**N images**
Number of series with 0 false positive	81%	64%
Number of series with 1 false positive	16%	36%
Number of series with 2 false positives	3%	0%
Number of series with 0 false negatives	100%	100%
Number of series with 1 false negative	0%	0%
Number of series with 2 false negatives	0%	0%

Table 4: F and N brain image datasets results from testing phase. Results from a random sample of 31 fluorescent brain image datasets (8527 total images) and 25 Nissl stained brain image datasets (6980 total images) showing false positive and false negatives for each separate dataset.

The results from the initial production runs are shown in Table 5. Those results include brain image datasets with fluorescent (F), Nissl stained (N) and immunohistochemistry (IHC) labels. Table 6, Table 7 and Table 8 show the same results from production runs (as in Table 5) in a confusion matrix to describe the performance of the algorithm to classify whether the images are OOF or not.

Determining whether an image is OOF or not is an example of *binary classification* which is a task of predicting which elements from a dataset belong in one of two groups based on a classification rule. A way to measure the performance of the classifier we use two metrics called *precision* and

	F images	**N images**	**IHC images**
Number of series with 0 false positive	72%	48%	75%
Number of series with 1 false positive	20%	34%	14%
Number of series with 2 false positives	8%	18%	11%
Number of series with 0 false negatives	100%	100%	100%
Number of series with 1 false negative	0%	0%	0%
Number of series with 2 false negatives	0%	0%	0%

Table 5: F, N, IHC brain image datasets results from production. Results from a random sample of 25 fluorescent, 94 Nissl stained and 89 IHC brain image datasets during production runs, showing false positive and false negatives for each separate dataset.

		Visual Observation	
	N = 6812	Out-of-focus	In focus
Algorithm result	Out-of-focus	336 (TP)	9 (FP)
	In focus	0 (FN)	6467 (TN)

Table 6: Confusion matrix from 6812 fluorescent images during production runs.

		Visual Observation	
	N = 25849	Out-of-focus	In focus
Algorithm result	Out-of-focus	693 (TP)	66 (FP)
	In focus	0 (FN)	25090 (TN)

Table 7: Confusion matrix from 25849 N images during production runs.

		Visual Observation	
	N = 24165	Out-of-focus	In focus
Algorithm result	Out-of-focus	401 (TP)	32 (FP)
	In focus	0 (FN)	23732 (TN)

Table 8: Confusion matrix, precision and recall from results from 24165 IHC images during production runs.

recall. The precision and recall are defined as

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN} \tag{12}$$

The precision and recall for the classification of our images is shown in Figure 9, it shows that the recall is excellent in all images which means that the methods identify all OOF images. The precision is better in the F images which means that there are the least false positives. Overall the performance of the method is very good.

	F images	N images	IHC images
Precision	0.97	0.91	0.93
Recall	1.00	1.00	1.00

Table 9: Precision and recall from results from 6812 F, 25849 N and 24165 IHC images during production runs.

4.5 Applications of the method in other fields

We have demonstrated that the SF method can be used to successfully recognize OOF images from the brain image datasets of the MBA project. But this method can be applied outside the particular dataset and the particular field. We can use this method to indentify OOF images in any situation that we have a sequence of images, for example burst images from a still camera or image stills extracted from videos. In order to demonstrate this particular application, we used a built-in demo video that is available in MATLAB [28], the video depicts cars in traffic. We extracted the frames (images) from the video but since the video did not include OOF images we had to artificially create them. We randomly chose a few images from the video and we filtered them using different methods like a Gaussian filter, an average filter, a median filter of varying widths. One of the images was blurred only on the top part as well. Knowing which images are out-of-focus allowed us to confirm whether the method is successful or not. Because the video is very long we chose to use frames 100-281 for this example, the frames of the video are depicted in Figure 73 and Figure 74. After blurring a few images from frames 100-281 of the video we calculated the FV, as we discussed in the previous section, for all these frames of the videos. In order to find the OOF images, we will plot the FV versus the frame number in each video image stack and locate the outliers. Note that the change in FV in the image stack corresponds to the passing of the cars as the camera changes its focus. Figure 75 shows the FV data versus the frame ID for the frames. The moving

window data points are shown in a blue solid line while the points that are in a red asterisk (∗) were confirmed as outliers. The OOF images we created (Figure 76) coincided with the images that were identified as outliers, proving that this method has a wider appplicability than just the brain imaging. Although this example is an ideal situation with some small (adaptive, depending on the situation) changes the method has a variety of applications in several fields.

4.6 Discussion about the method

In order to classify whether images are OOF or not, the data from a whole brain image dataset is used as opposed to imposing a specific threshold to make that classification. The reason is because using an empirical value was not found to be advantageous. Imposing an empirical value might seem an advantage (because the classification of in focus or OOF can be done in real time without having to wait for the analysis of a whole brain image dataset) but having a specific threshold has many drawbacks and can lead to erroneous characterizations. The first drawback is that imposing a threshold would require it to be defined empirically which would mean that a large number of images would have to be visually inspected and compared. That would be very time consuming and would introduce a large uncertainty. Additionally, specific threshold would not be accurate for all the brain image datasets due to changes in illumination and background.

This method identifies outliers which means that the assumption is that the OOF images are the minority in each brain image dataset. If a brain image dataset includes a majority of OOF images, the results will not be reliable but such an image dataset will otherwise have to be discarded because the data would be unusable.

In order to demonstrate that our method can be used in a variety of datasets we have used one built-in demo video available in MATLAB and used a selection of the video frames to simulate OOF images. Contrary to the brain images the method was executed very fast for this data series due to the smaller size of these images.

A drawback of the SF method is that it could be computationally intensive for sizeable images. For example, an image of a tissue size 12000×10000 pixels required 12 seconds to run the analysis with our computational resources. Therefore, it takes roughly an hour to complete the analysis for a brain image dataset of 300 images with our computational resources. However, we can

improve the overall processing time by running several datasets in parallel.

Despite the fact that the analysis could be computationally time consuming for large images, the results are still faster than visual inspection and they can be accelerated by running computations in parallel. Additionally, the results are more reliable than visual inspections.

4.7 Conclusion

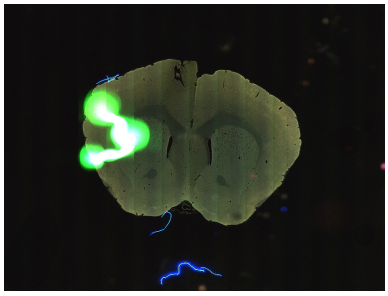
We have presented a method that is used to identify OOF images from a stack of images that have been digitally imaged. Such a method can be used for many different digital scanners. In this project, we have used images that have been taken using the Hamamatsu NanoZoomer 2.0 HT automated slide scanning microscope to demonstrate how to apply the method.

After extensive testing, the method works satisfactorily at identifying OOF brain images and it is an active part of the MBA automated quality control process. For most of the brain image datasets that were analyzed, it accurately identifies only the images that are OOF, while for the remaining datasets, it misidentifies one or two images as OOF where the visual inspections did not support that conclusion. Our goal is to identify and remove the OOF images before they can be used in any further analyses and to replace the OOF images with new, in focus images, and this method clearly satisfies those requirements. The program is currently being used successfully in production.

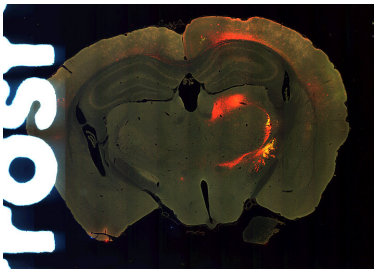
Additionally, we have demonstrated the successful use of the process in other image series where we have inserted simulated OOF images. This demonstrated that the method can have wider applications, for instance in video frame captures, burst images, animated GIFs or any other series of images.

The method has several advantages but some disadvantages too. The main advantage is that the classification of images is not dependent on the skill of the person who does the visual inspection. Additionally, despite the fact that the program could be computationally intensive for very large images, it is faster than inspecting visually all images to identify the OOF images.

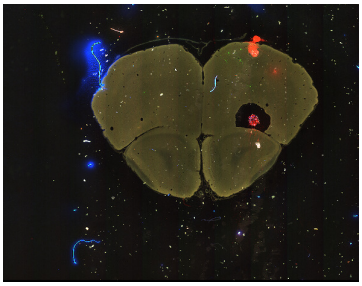
The main disadvantage is that all images in each brain image dataset have to be analysed together because the identification of OOF images comes from the comparison of the image FVs with each other to find the outliers and is therefore not suitable for a real time analysis.



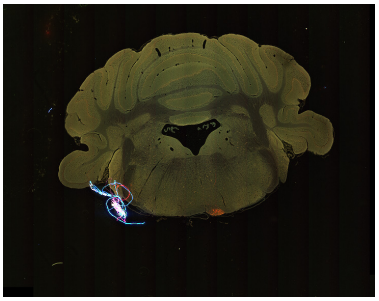
(a) Image with an artifact



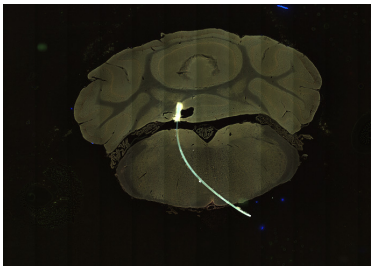
(b) Image with label letters on tissue



(c) Damaged image with artifacts



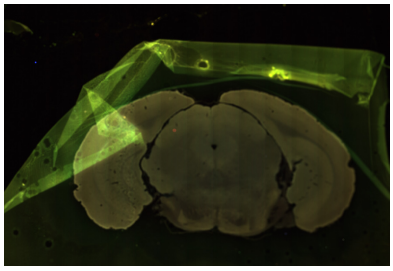
(d) Image with tape on the tissue



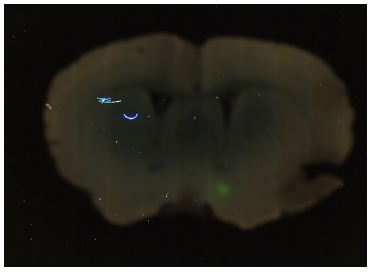
(e) Image with artifact



(f) Damaged image with bubble on tissue

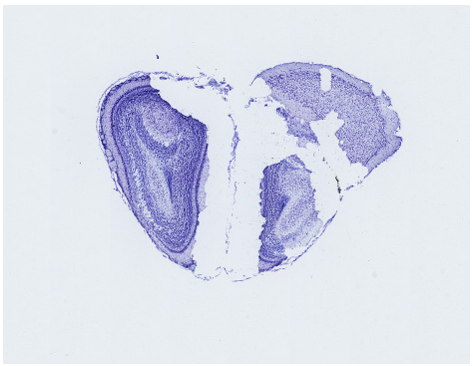


(g) Image with folded tape

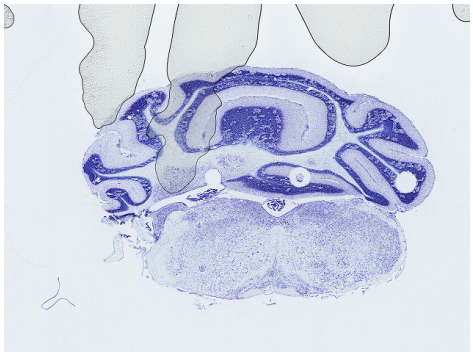


(h) OOF image with artifact

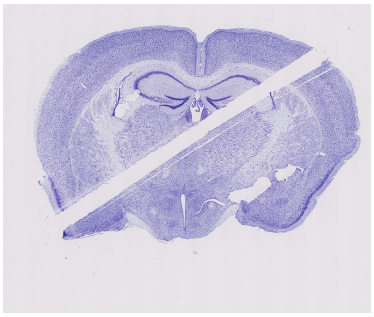
Figure 41: Examples of F images that due to damage, noise, artifacts will be removed from the sample with visual inspection.



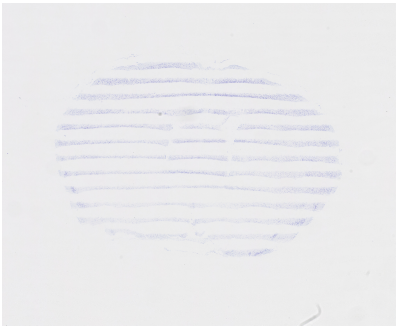
(a) Damaged image



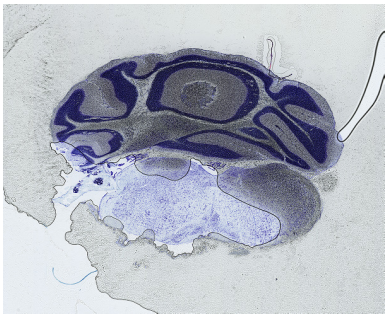
(b) Image with bubbles on tissue



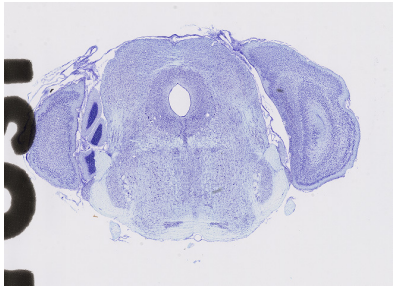
(c) Torn image



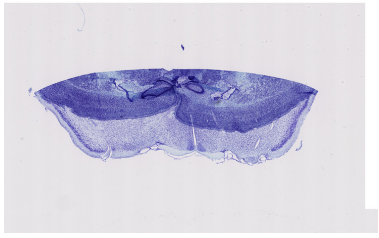
(d) Severely damaged image



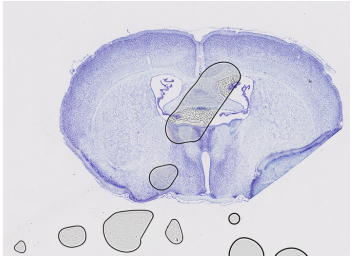
(e) Image with tape and artifacts on tissue



(f) Image with letters on tissue

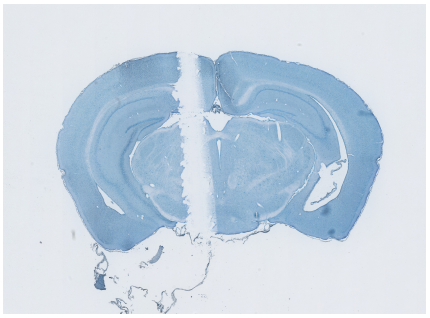


(g) Image with folded tissue

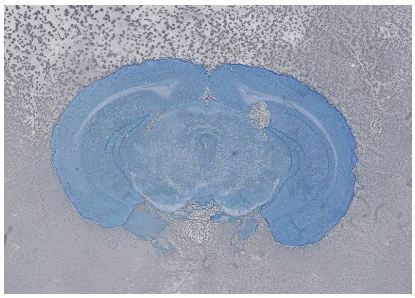


(h) Image with air bubbles on tissue

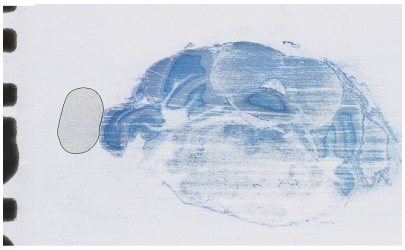
Figure 42: Examples of noisy and damaged Nissl stained images



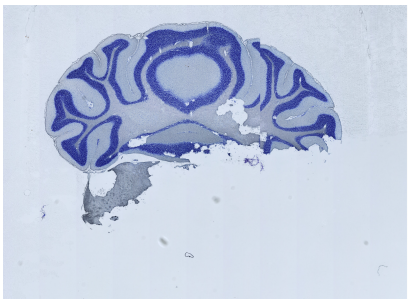
(a) Torn Image



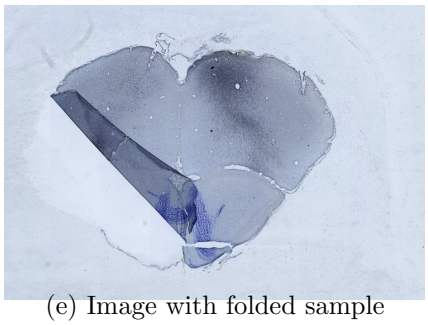
(b) Image with noise on tissue



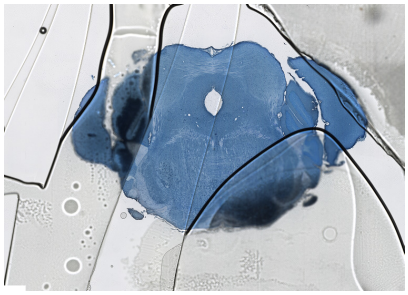
(c) Severely damaged sample with label remnants and bubble



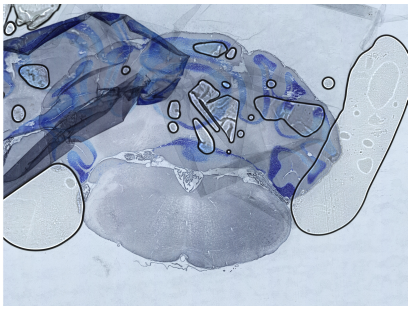
(d) Damaged sample



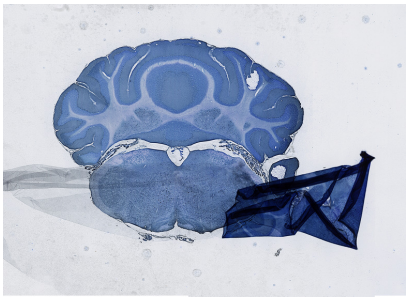
(e) Image with folded sample



(f) Image with plastic tape on the sample



(g) Image with bubbles



(h) Image with tape remnant and an additional sample

Figure 43: Examples of IHC images that due to damage,noise, artifacts would be removed from the sample through visual inspection

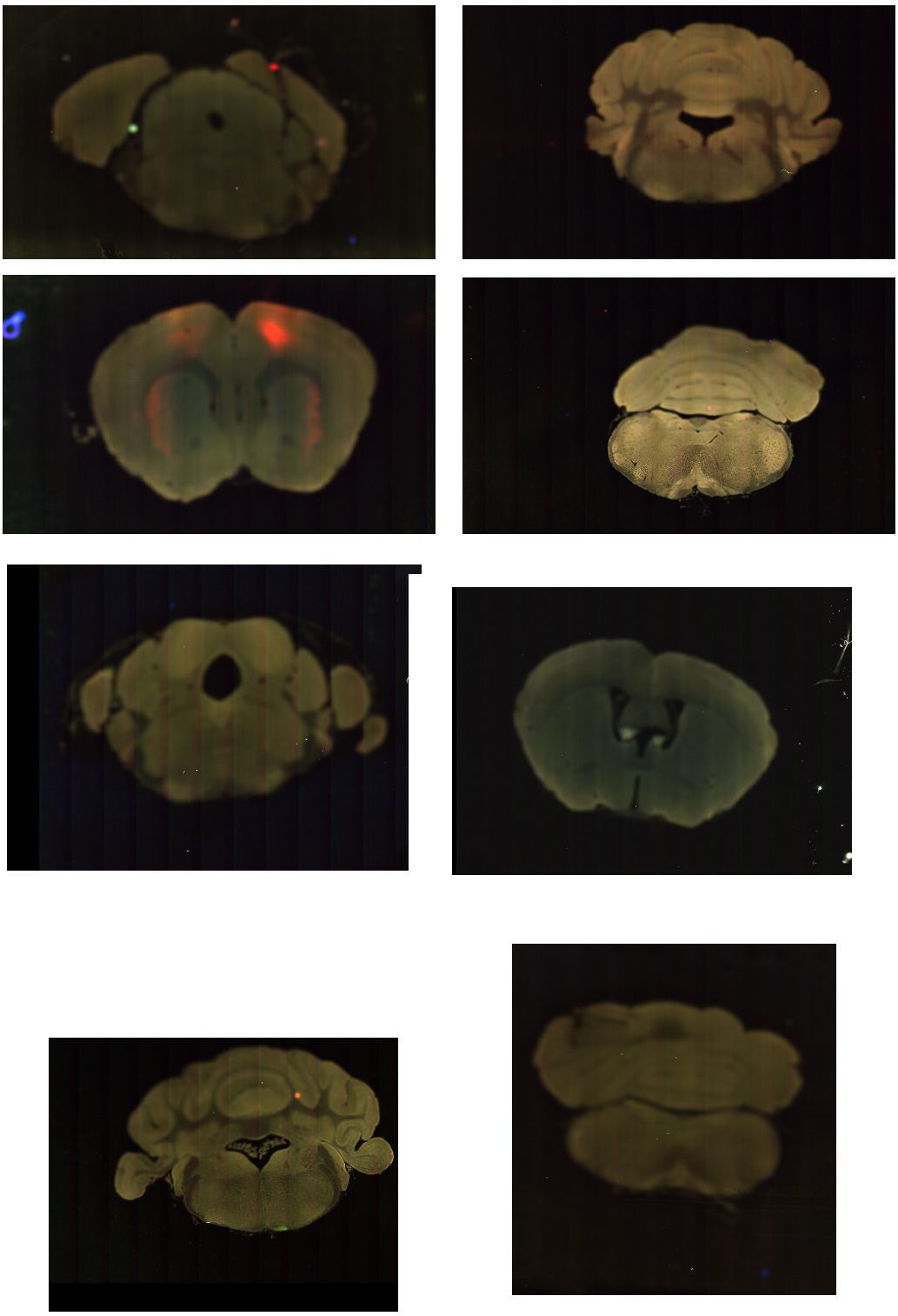


Figure 44: Examples of different fluorescent out of focus images

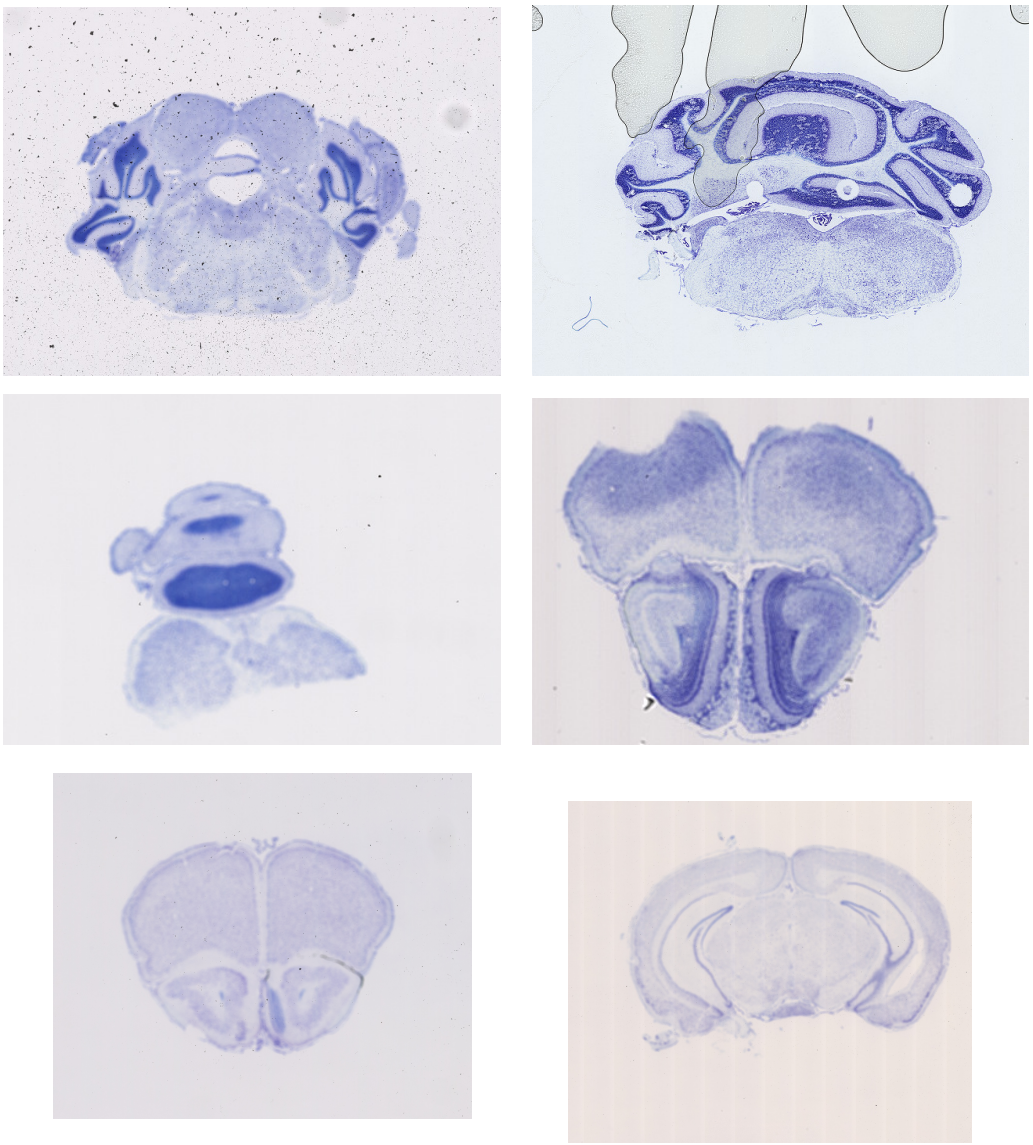


Figure 45: Examples of different Nissl stained out-of-focus images

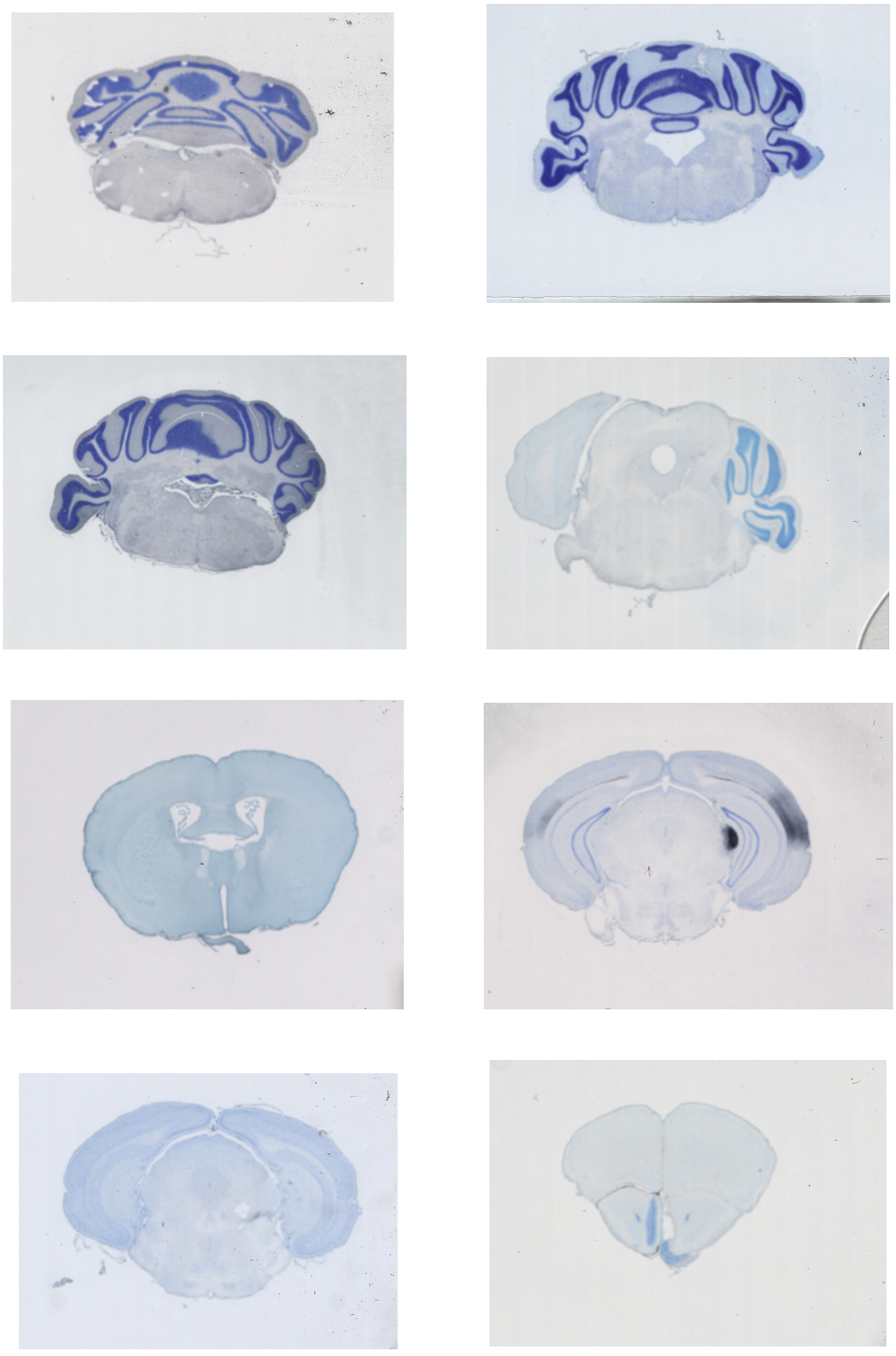


Figure 46: Examples of different IHC out-of-focus images

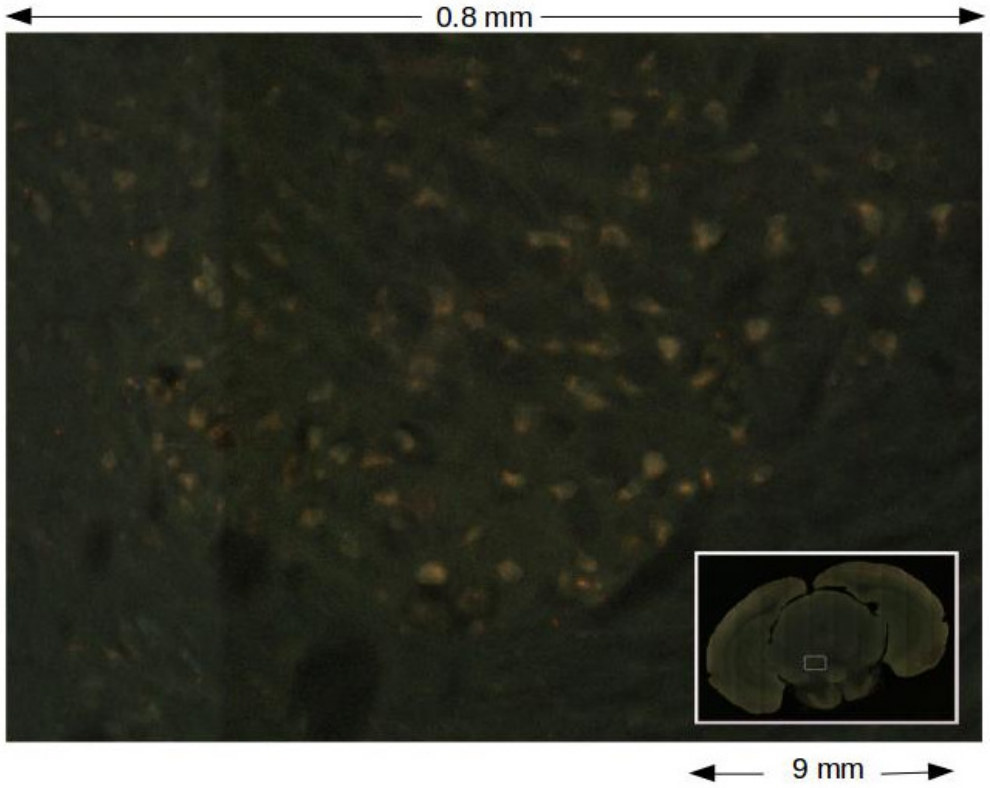
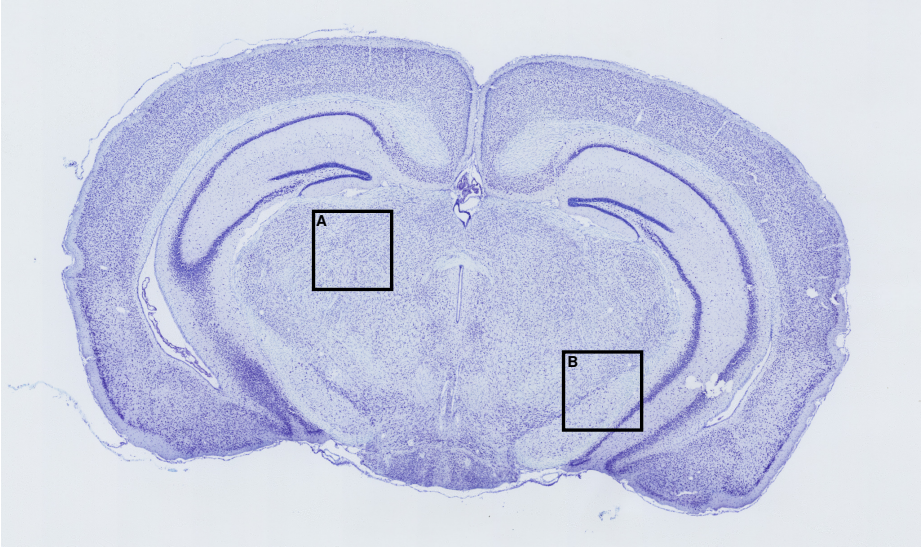
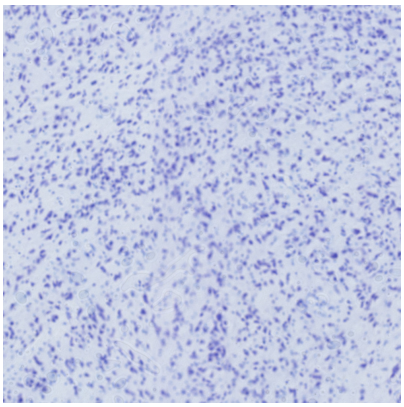
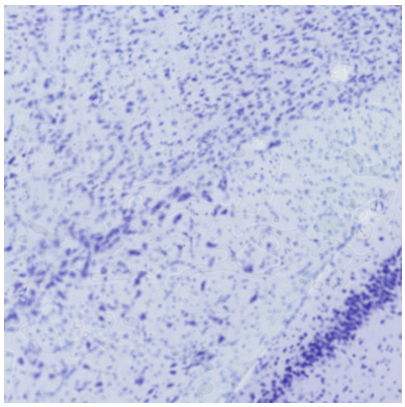


Figure 47: Example of fluorescent (F) image that is out-of-focus when magnified. When the image is in full view it does not look OOF but once the image is magnified the image is clearly OOF.



(a) Section 136 from stack 2681. The size of this cropped image is 13990 × 23540 (the original uncropped image is 19638 × 27928).

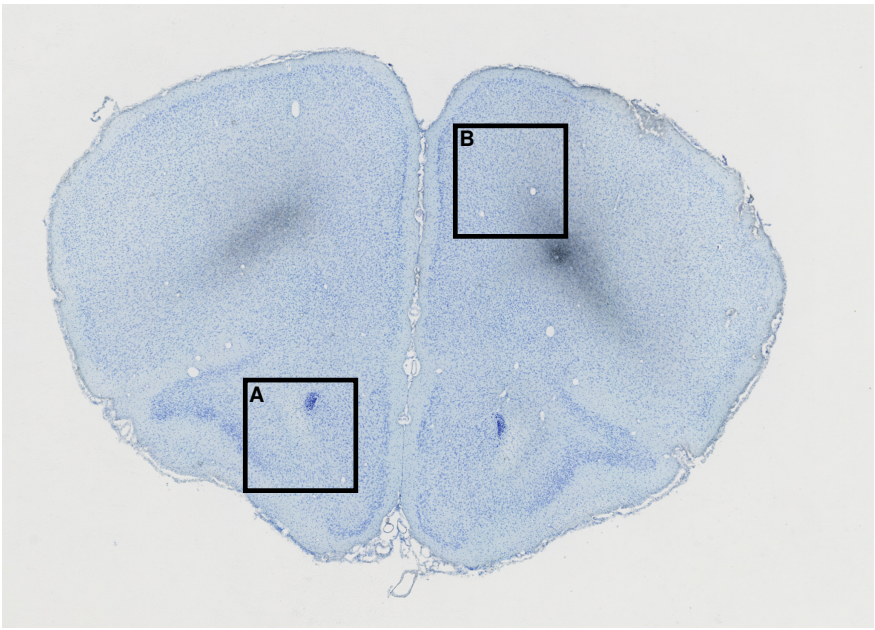




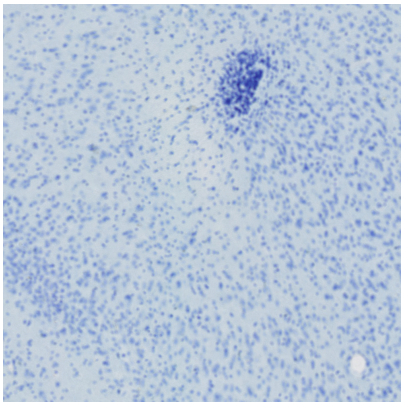
(b) Region A from section 136, stack 2681. The size of the region is 2000 × 2000 pixels.

(c) Region B from section 136, stack 2681. The size of the region is 2000 × 2000 pixels.

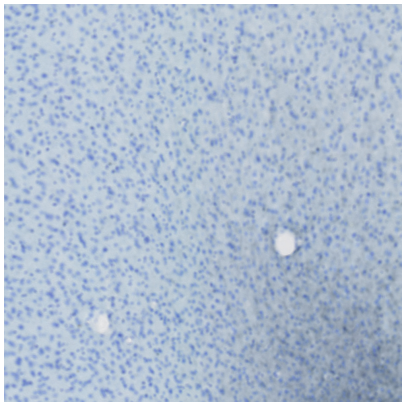
Figure 48: Image of section 136 from stack 2681 does not look like OOF when inspecting the whole image but when we magnify in several regions of the image, it is clear that the image is OOF.



(a) Section 85 from stack 2797. The size of this cropped image is 11276 × 157640 (the original uncropped image is 17261 × 21494).



(b) Region A from section 85, stack 2797. The size of the region is 2000 × 2000 pixels.



(c) Region B from section 85, stack 2797. The size of the region is 2000 × 2000 pixels.

Figure 49: Image of section 85 from stack 2797 does not look like OOF when inspecting the whole image but when we magnify in several regions of the image, it is clear that the image is OOF.

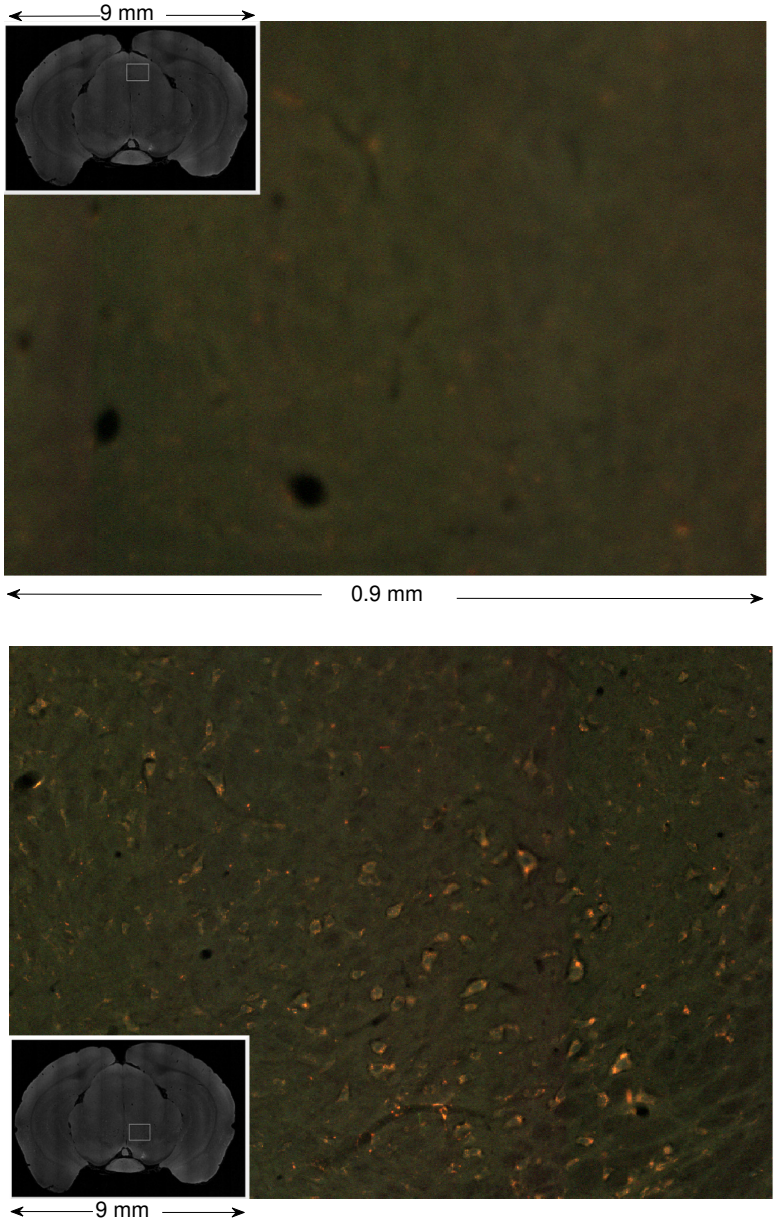
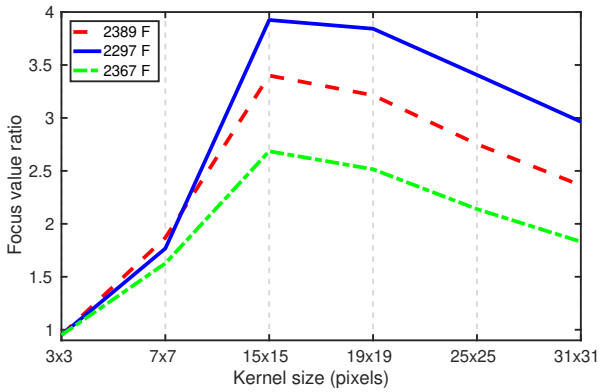
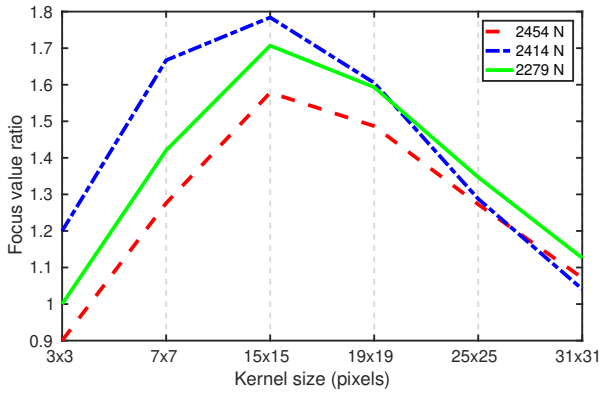


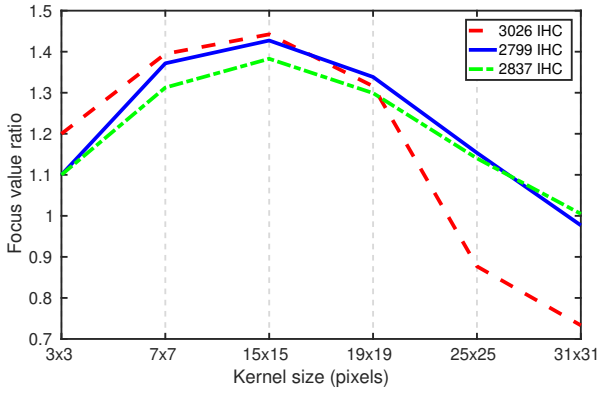
Figure 50: Example of fluorescent (F) image that is partially out-of-focus. The top part of the figure shows a magnified region of interest (region A) while the bottom part of the figure shows another magnified region of interest (region B) of the same image. It is shown that, even though both regions are from the same image, region A is out-of-focus while region B is in focus.



(a) F brain image datasets



(b) N brain image datasets



(c) IHC brain image datasets

Figure 51: Ratio of FV of sharp vs OOF images for different kernels from a sample of different brain image datasets. The numbers displayed in the legend indicate the brain dataset.

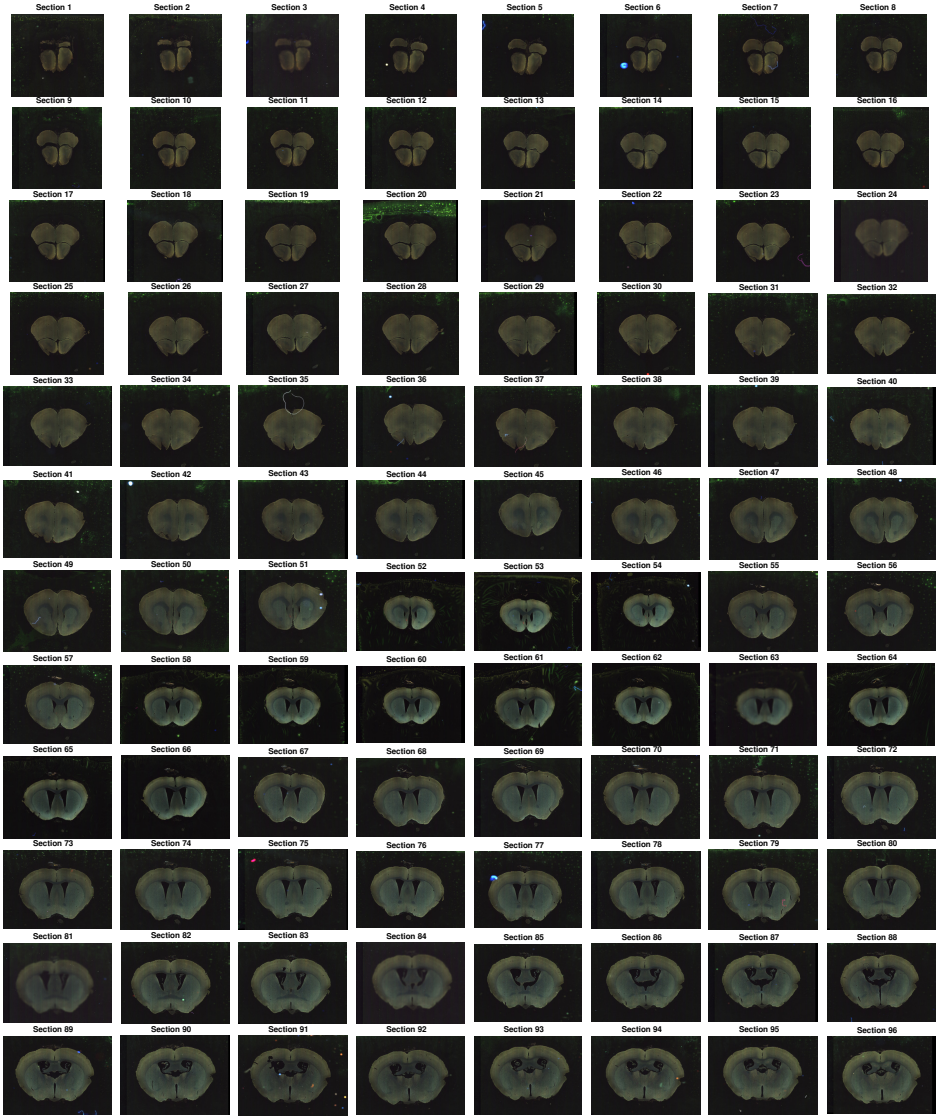


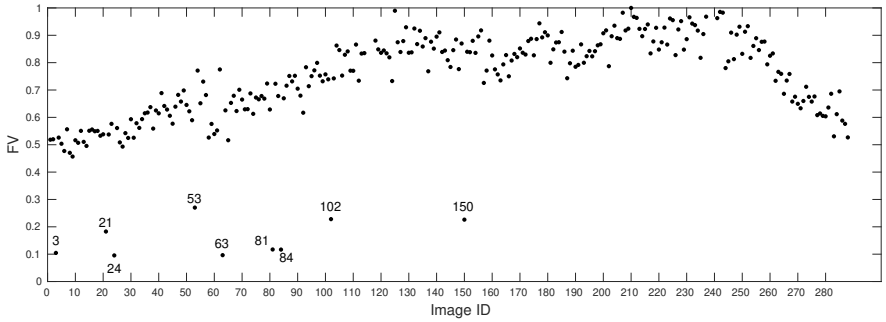
Figure 52: Images from stack 2888



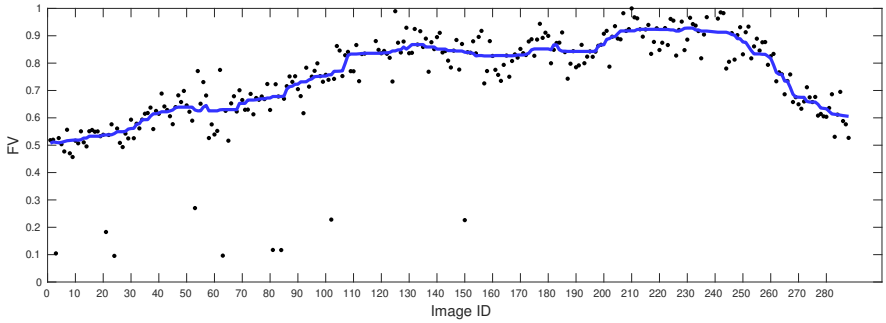
Figure 53: Images from stack 2888



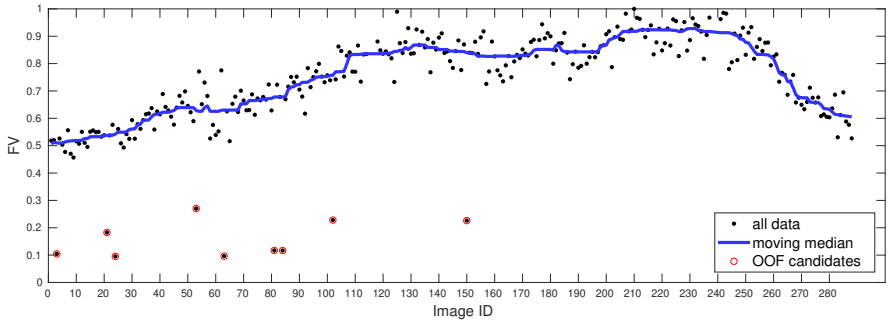
Figure 54: Images from stack 2888



(a) FV versus image ID for stack 2888

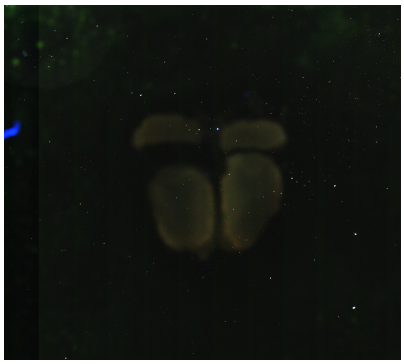


(b) FV versus image ID for stack 2888 with a moving median

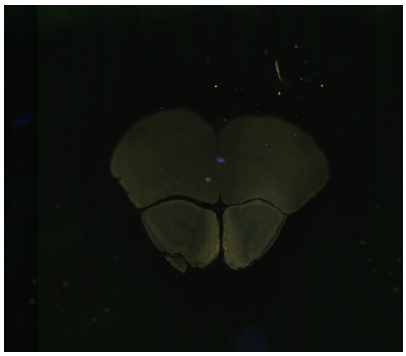


(c) FV versus image ID for stack 2888 with identified outliers

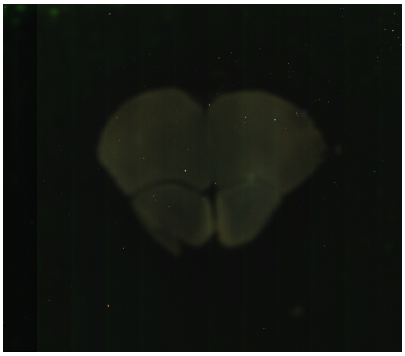
Figure 55: OOF detection for stack 2888



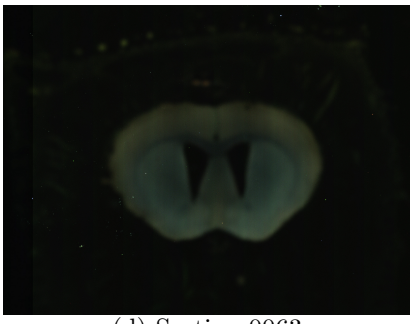
(a) Section 003



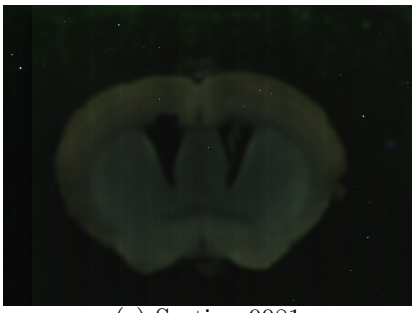
(b) Section 0021



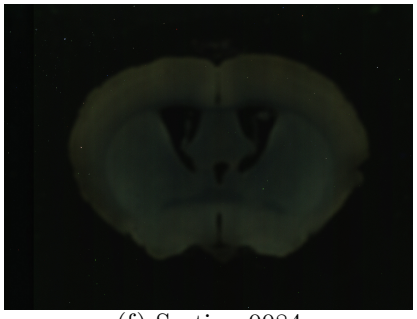
(c) Section 0024



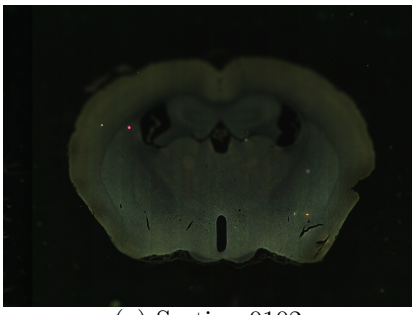
(d) Section 0063



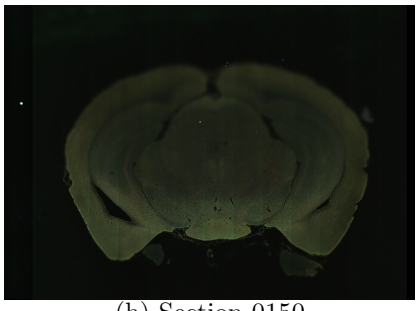
(e) Section 0081



(f) Section 0084

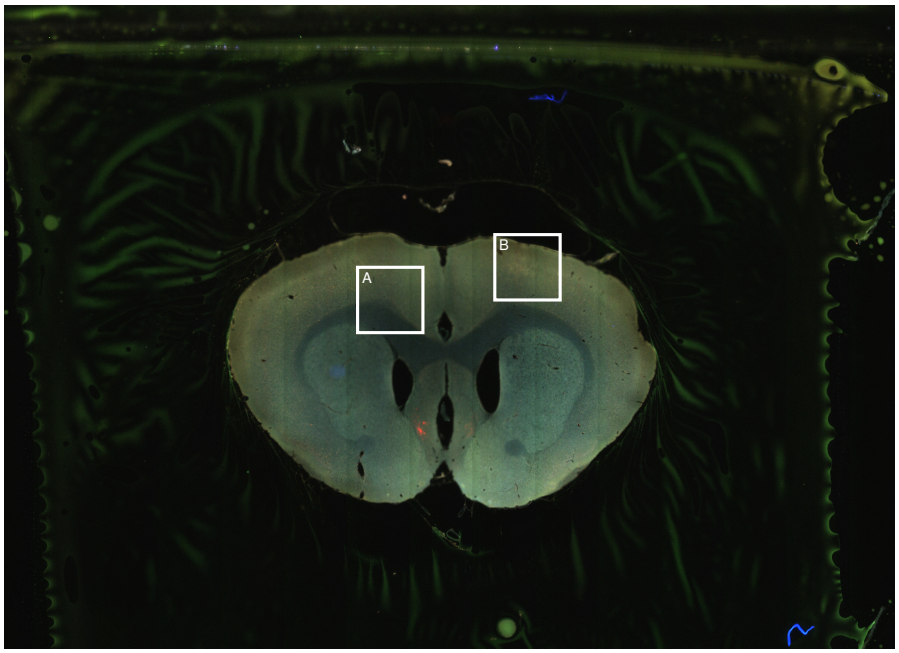


(g) Section 0102



(h) Section 0150

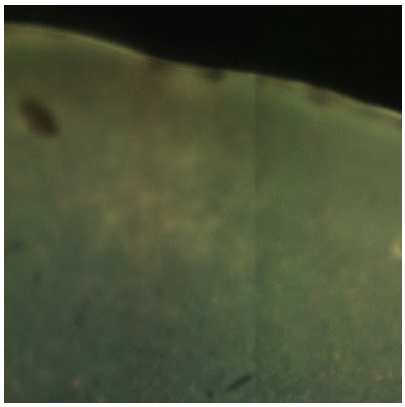
Figure 56: OOF images indentified by the algorithm and visual inspection for series 2888



(a) Section 53



(b) Section 53 Region A



(c) Section 53 Region B

Figure 57: Section 53 identified as OOF. The image does not look like OOF when looking at the whole image but when we zoom in some regions it is clear that it is OOF.

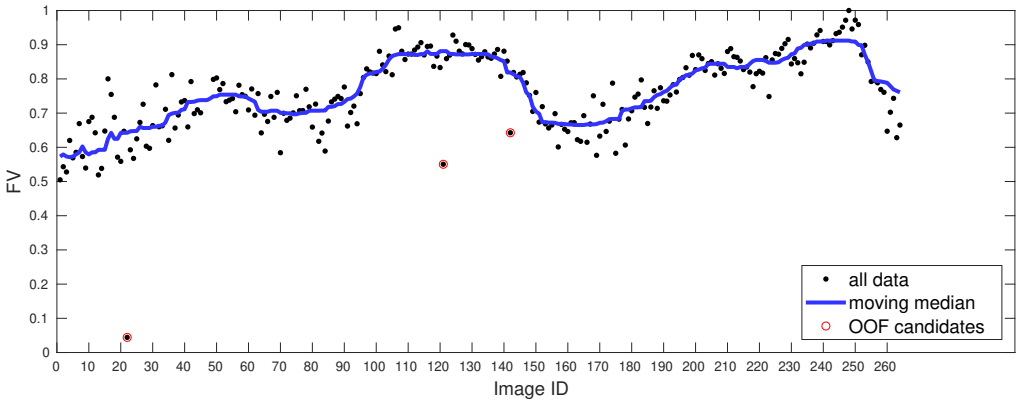


Figure 58: OOF detection of N images in stack 3079

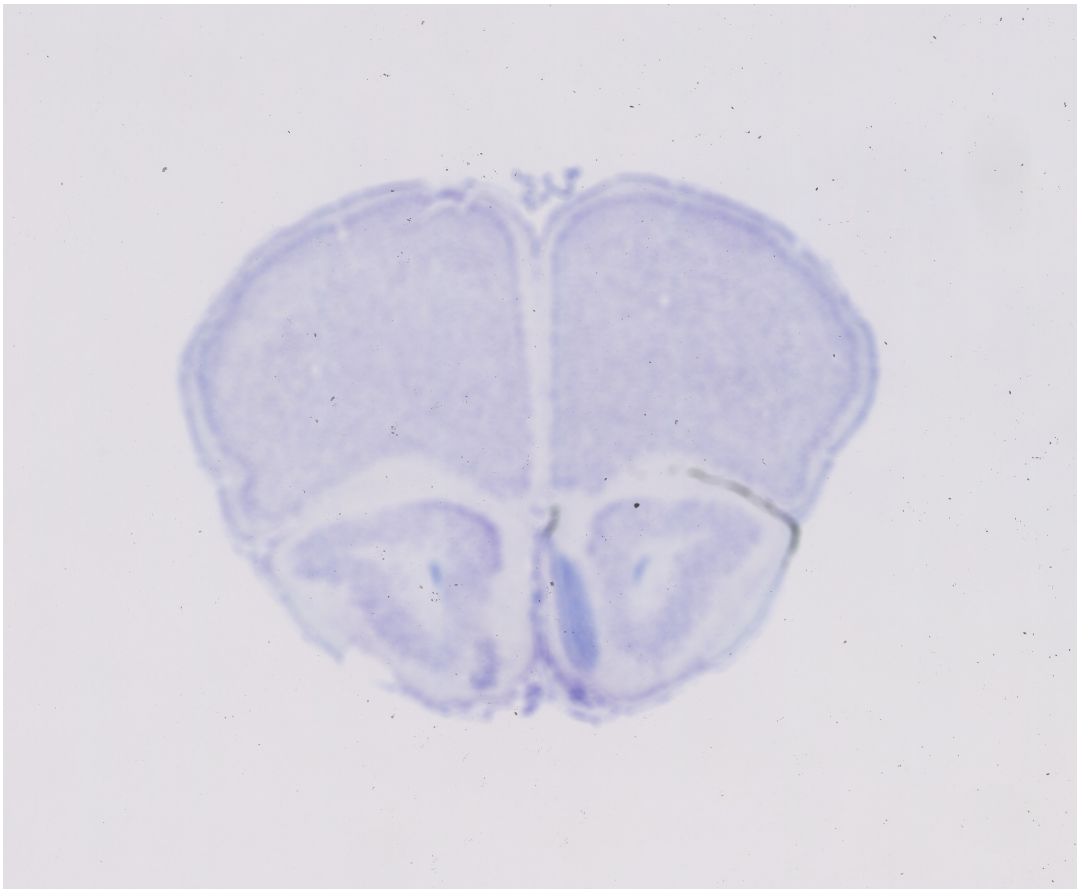
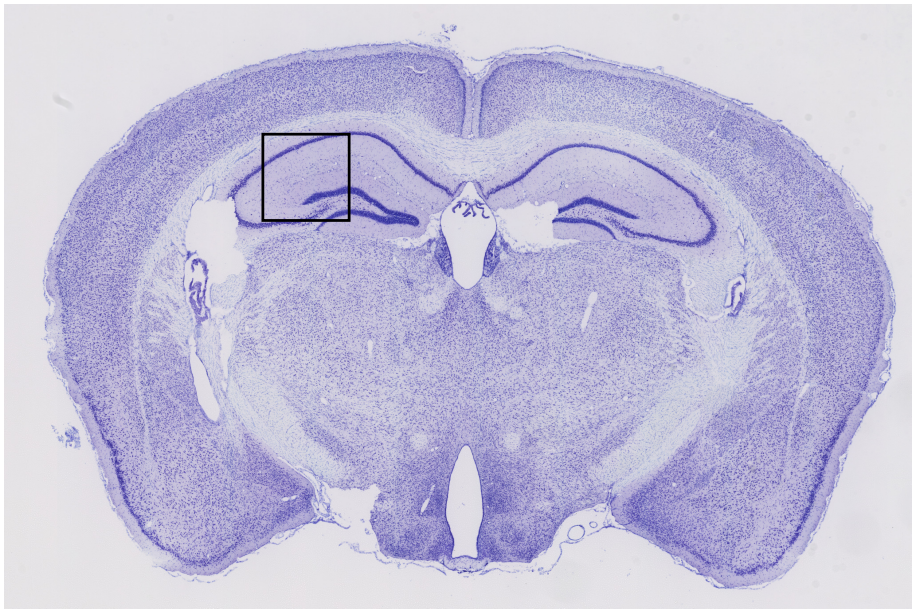
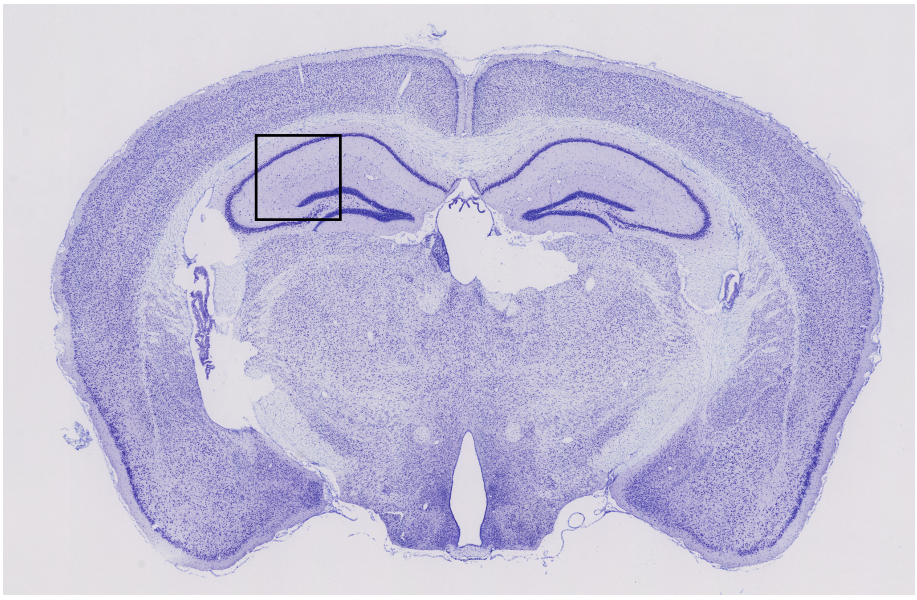


Figure 59: N section 22 from stack 3079 identified as OOF visually and from the program

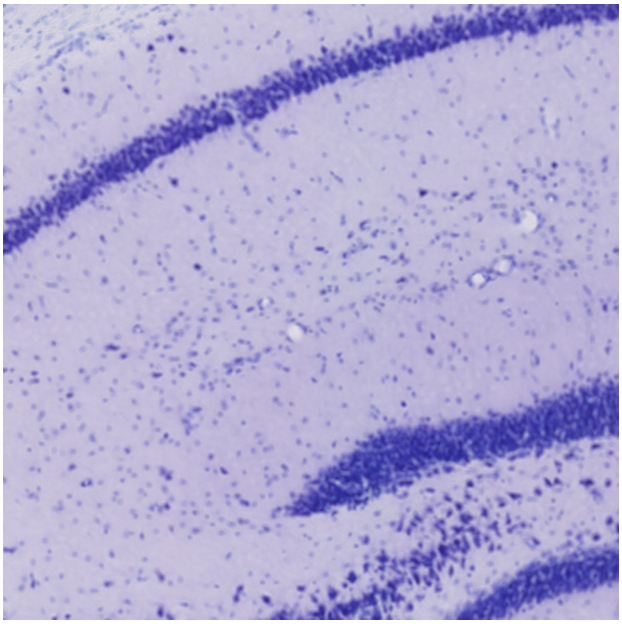


(a) N section 121 from stack 3079, OOF image

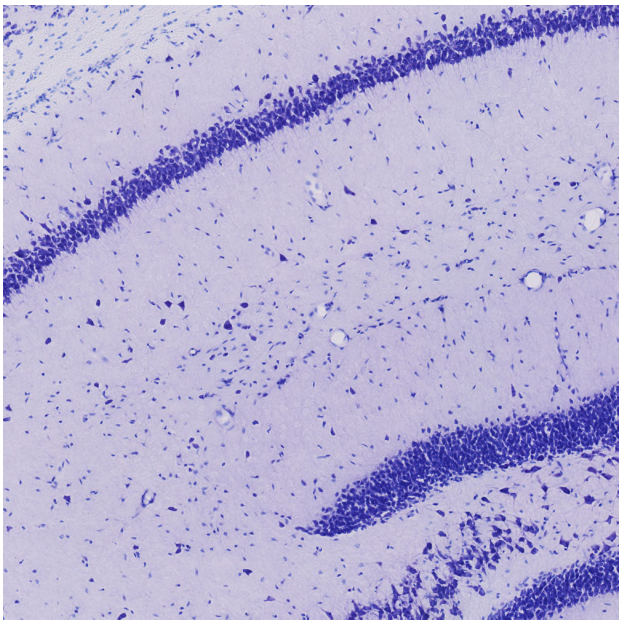


(b) N section 122 from stack 3079, not OOF image

Figure 60: N sections 121(top) and 122(bottom). The top image was identified as OOF by the algorithm while the bottom was not. Looking at both whole images we can not tell the difference.

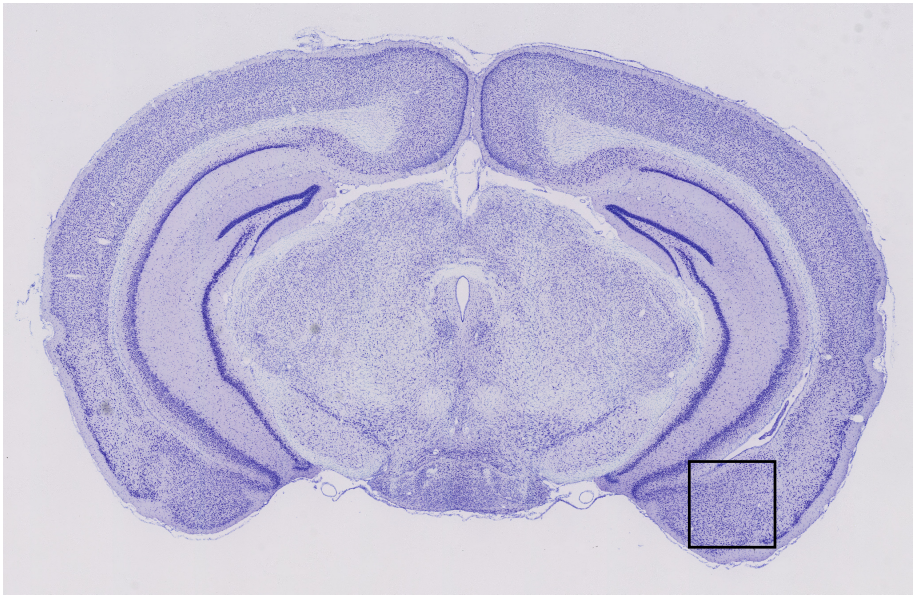


(a) 2000 × 2000 pixel area from N section 121 from stack 3079, OOF image

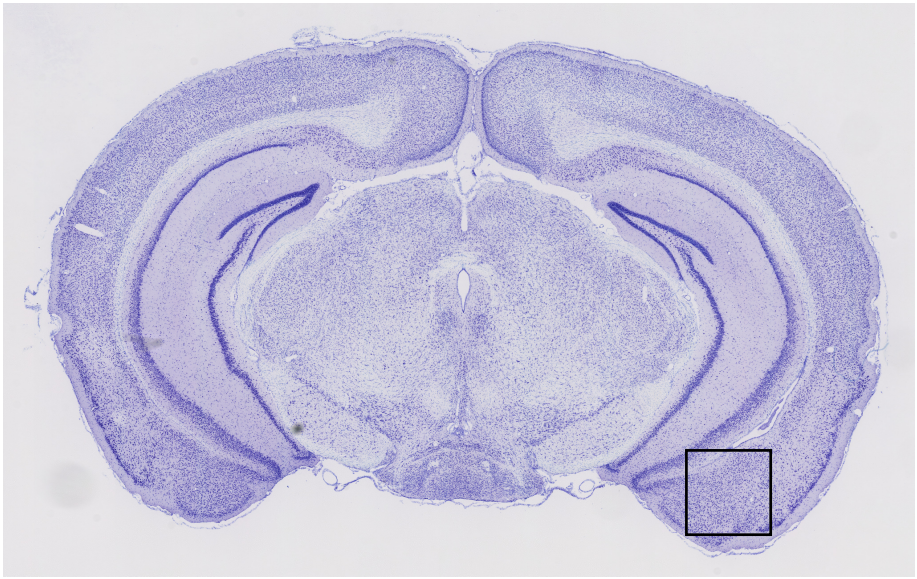


(b) 2000 × 2000 pixel area from Nsection 122 from stack 3079, not OOF image

Figure 61: Zoom in areas from N sections 121(top) and 122(bottom). The top image was identified as OOF by the algorithm while the bottom was not, at this resolution it is evident the top is OOF

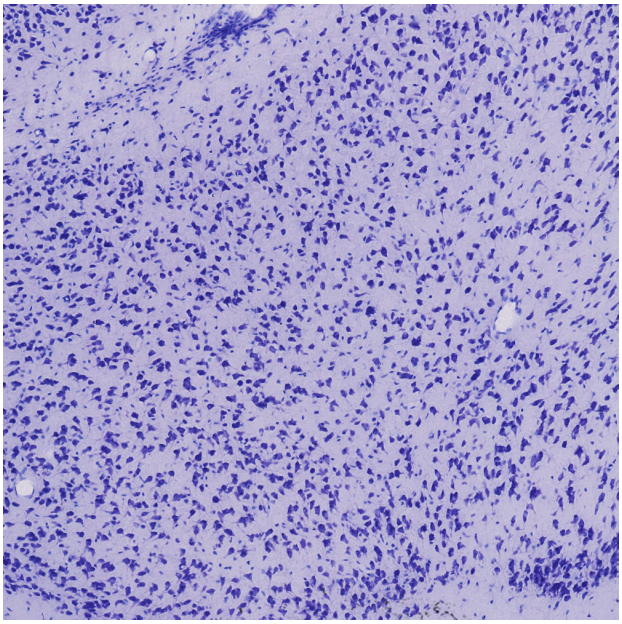


(a) N section 141 from stack 3079, not OOF image

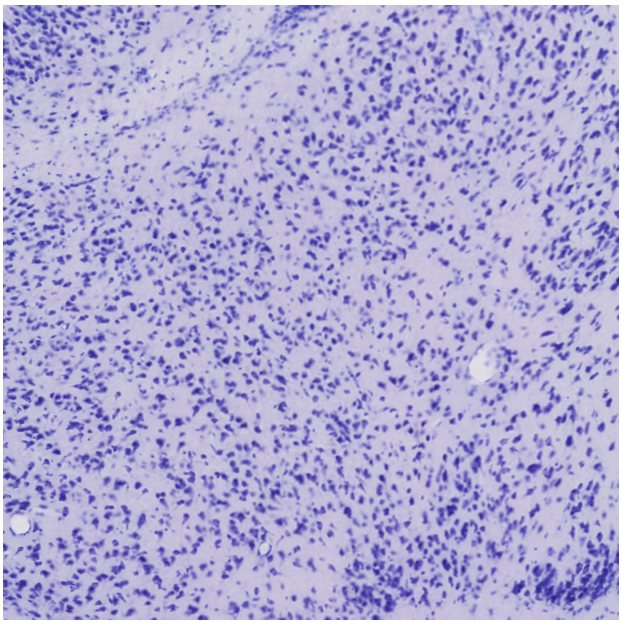


(b) N section 142 from stack 3079, OOF image

Figure 62: N sections 141(top) and 142(bottom). The top image was not identified as OOF by the algorithm while the bottom was identified as OOF. Looking at both whole images we can not tell the difference.



(a) 2000 × 2000 pixel area from N section 141 from stack 3079, not OOF image



(b) 2000 × 2000 pixel area from N section 142 from stack 3079, OOF image

Figure 63: Zoom in areas from N sections 141(top) and 142(bottom). The top image was not identified as OOF by the algorithm while the bottom was identified as OOF, at this resolution it is evident the bottom is OOF.

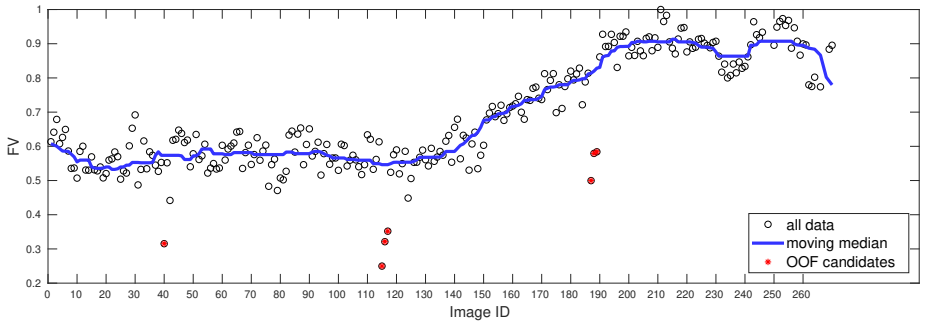
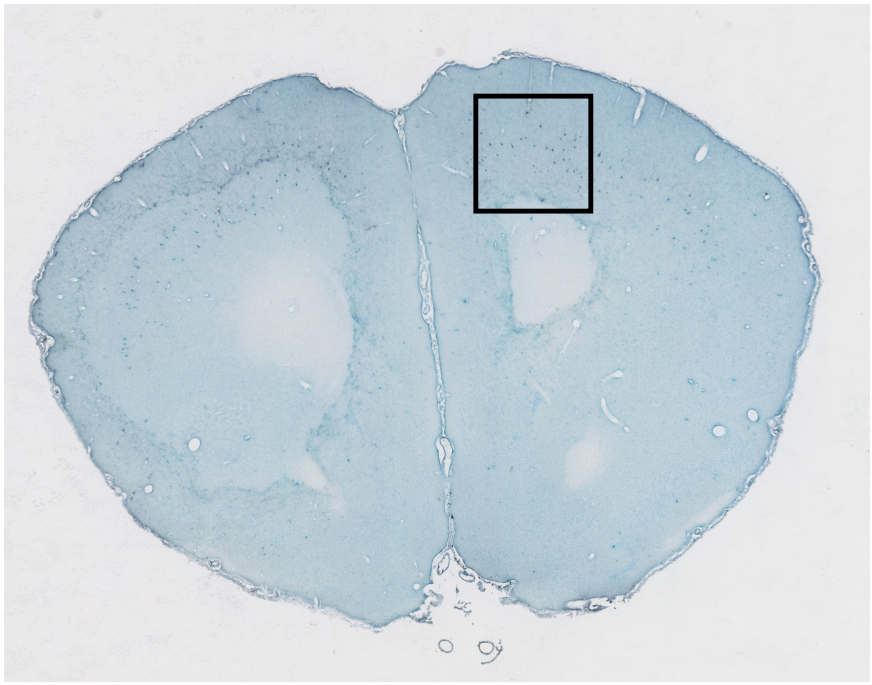
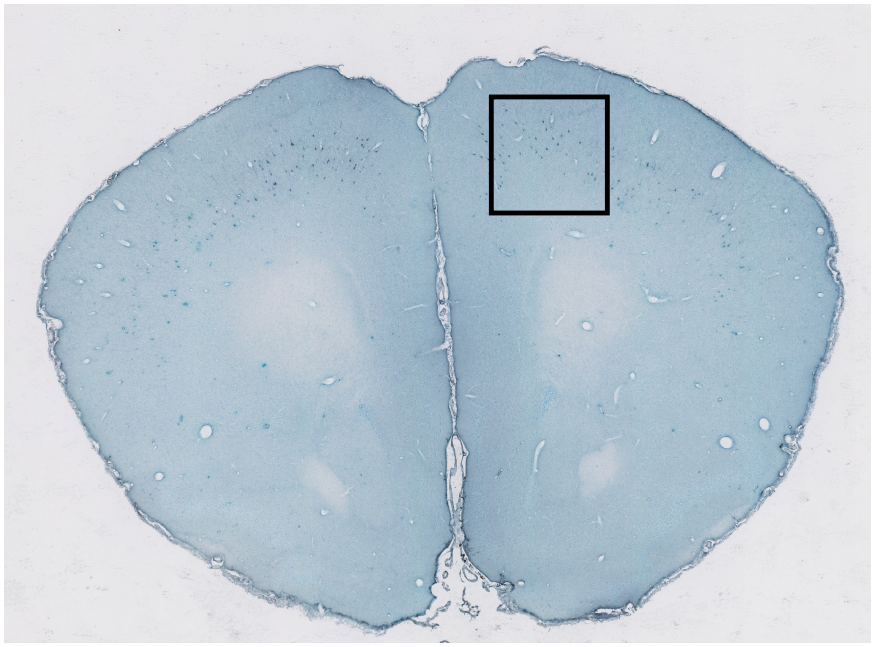


Figure 64: Focus value (FV) versus image ID for different IHC brain image dataset 2695, where the black $\circ$ symbol represents all the data, the straight blue line is the moving median and the OOF candidates are shown as the red $*$ symbol.

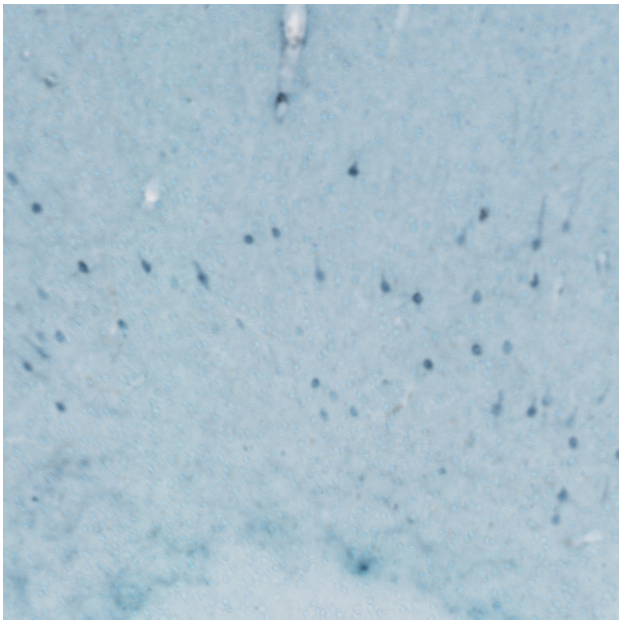


(a) IHC section 40 from stack 2695, OOF image

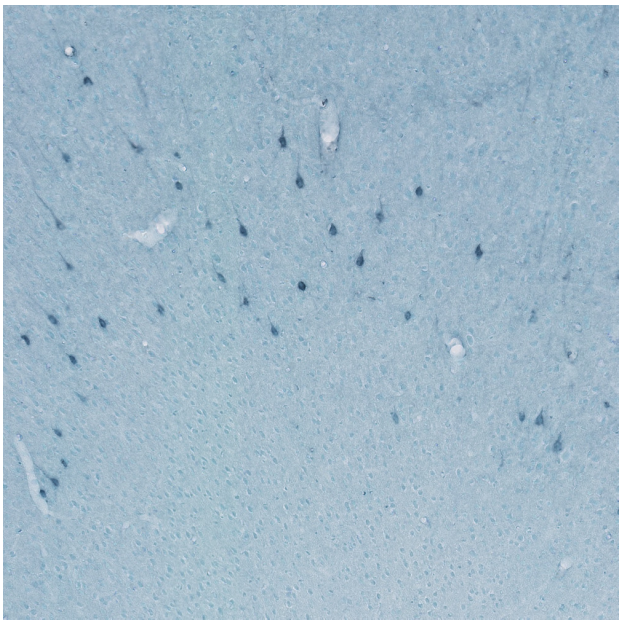


(b) IHC section 41 from stack 2695, not OOF image

Figure 65: IHC sections 40(top) and 41(bottom) from stack 2695. The top image was identified as OOF by the algorithm while the bottom was not. Looking at both whole images at this magnification, we can not tell the difference.

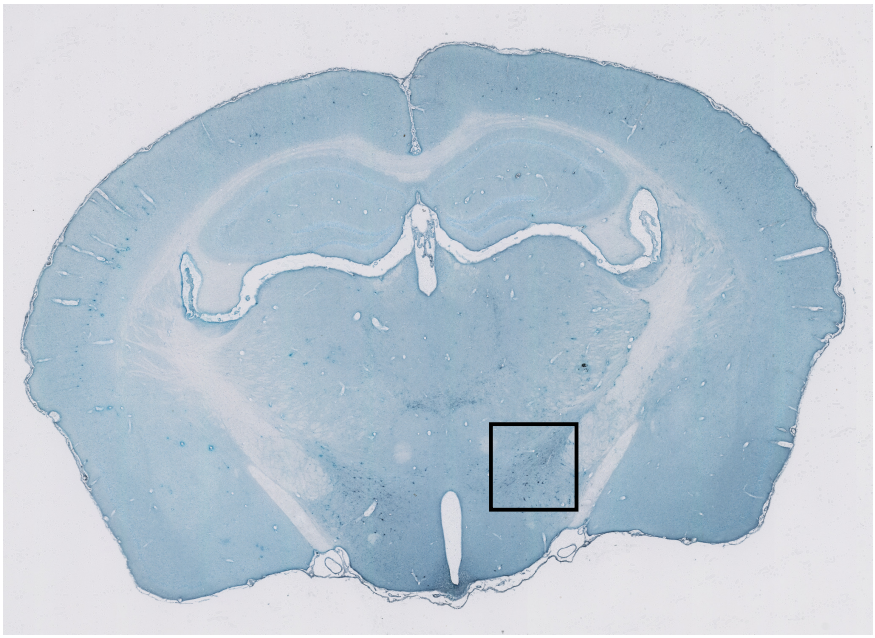


(a) 2000 × 2000 pixel area from IHC section 40 from stack 2695, an OOF image

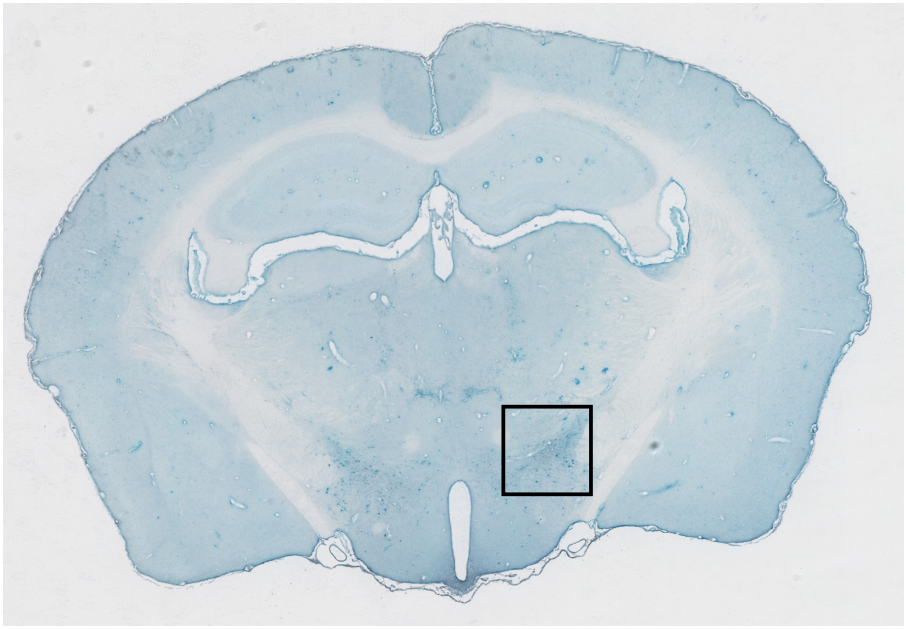


(b) 2000 × 2000 pixel area from IHC section 41 from stack 2695, not an OOF image

Figure 66: Zoom in areas from IHC sections 40(top) and 41(bottom). The top image was identified as OOF by the algorithm while the bottom was identified as OOF, at this resolution it is evident the top is OOF so the algorithm was successful in identifying this OOF image.

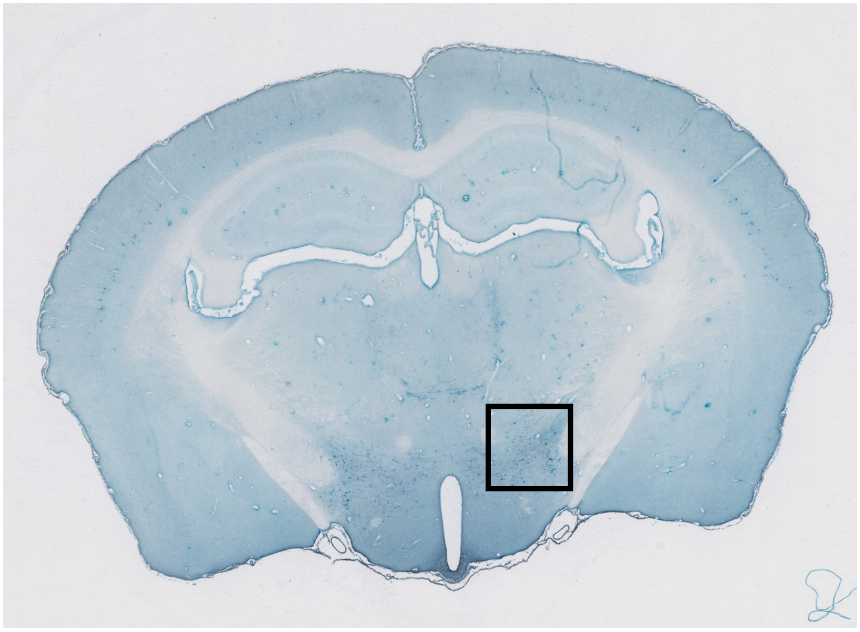


(a) IHC section 114 from stack 2695, not OOF image

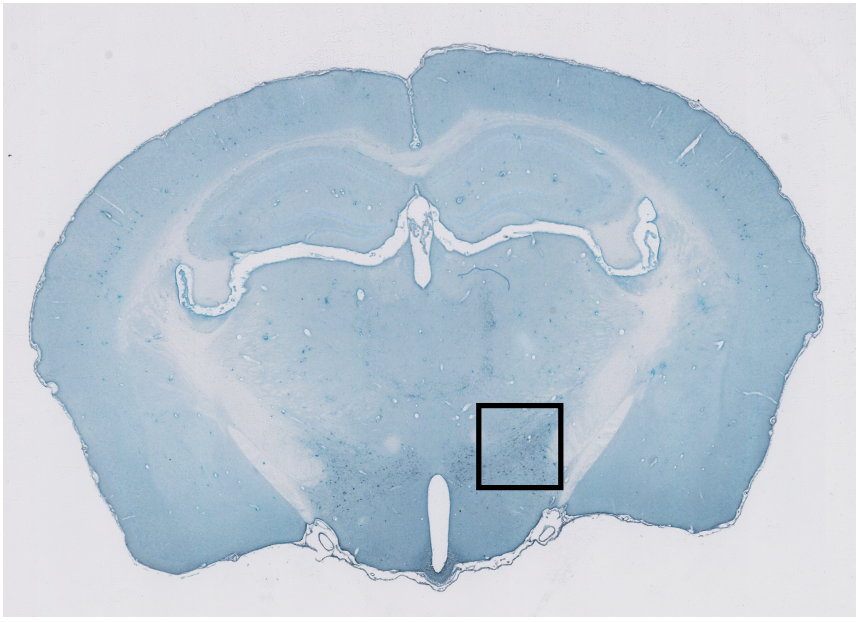


(b) IHC section 115 from stack 2695, OOF image

Figure 67: IHC sections 114(top) and 115(bottom) from stack 2695. The top image was identified as not an OOF by the algorithm while the bottom was identified as an OOF. Looking at both whole images at this magnification, we can not tell the difference.

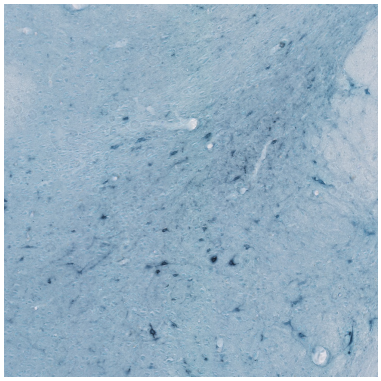


(a) IHC section 116 from stack 2695, not OOF image

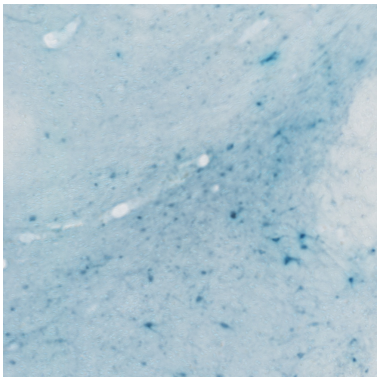


(b) IHC section 117 from stack 2695, OOF image

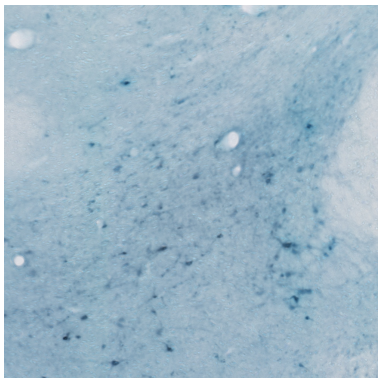
Figure 68: IHC sections 116(top) and 117(bottom) from stack 2695. Both images were identified as an OOF by the algorithm.



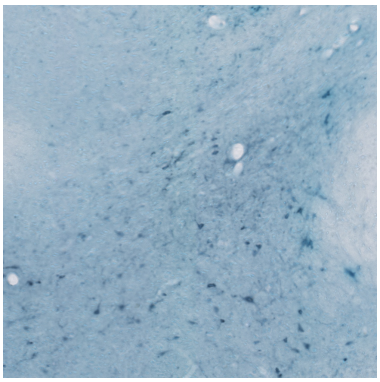
(a) Section 114



(b) Section 115

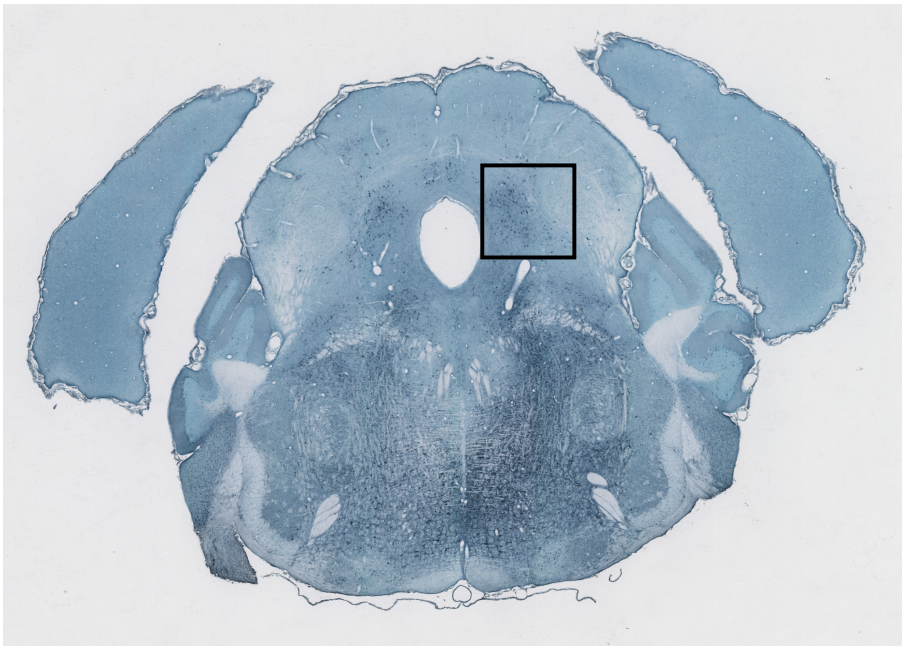


(c) Section 116

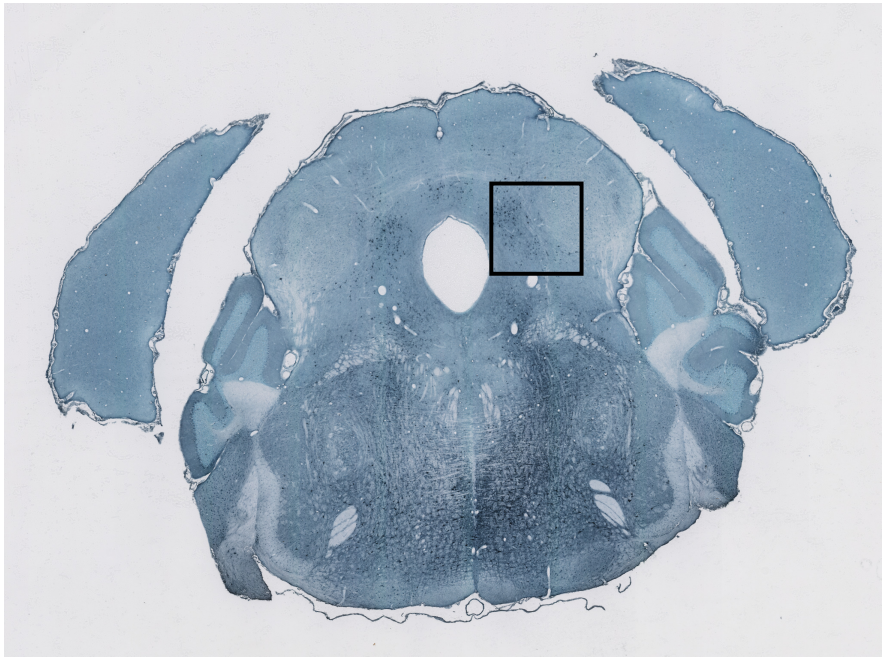


(d) Section 117

Figure 69: Magnified 2000×2000 pixel areas from IHC sections 114, 115, 116 and 117. Images 115, 116 and 117 were identified as OOF by the algorithm while section 114 was identified as not an OOF, at this resolution it is evident the algorithm was successful in identifying the OOF images

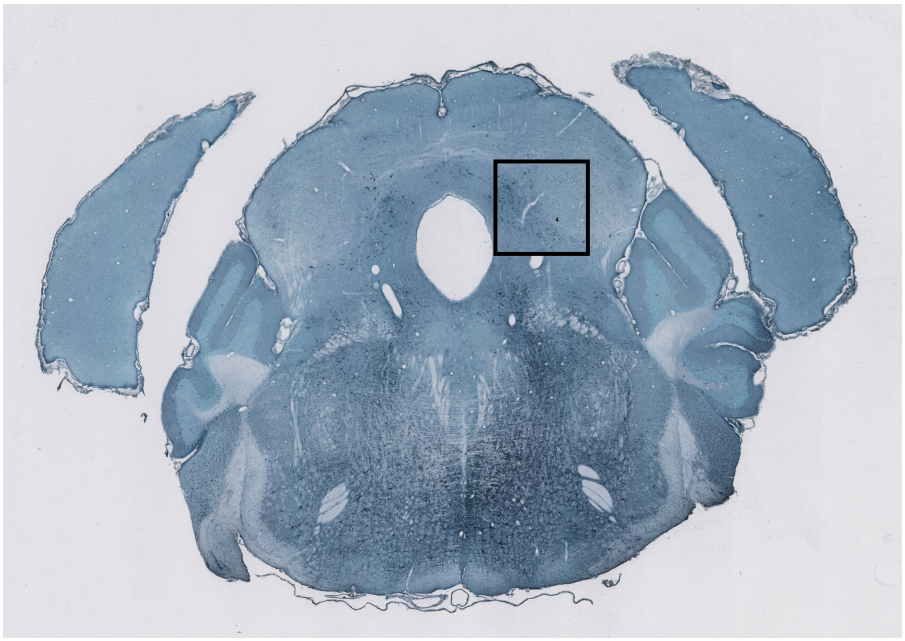


(a) IHC section 187 from stack 2695, OOF image

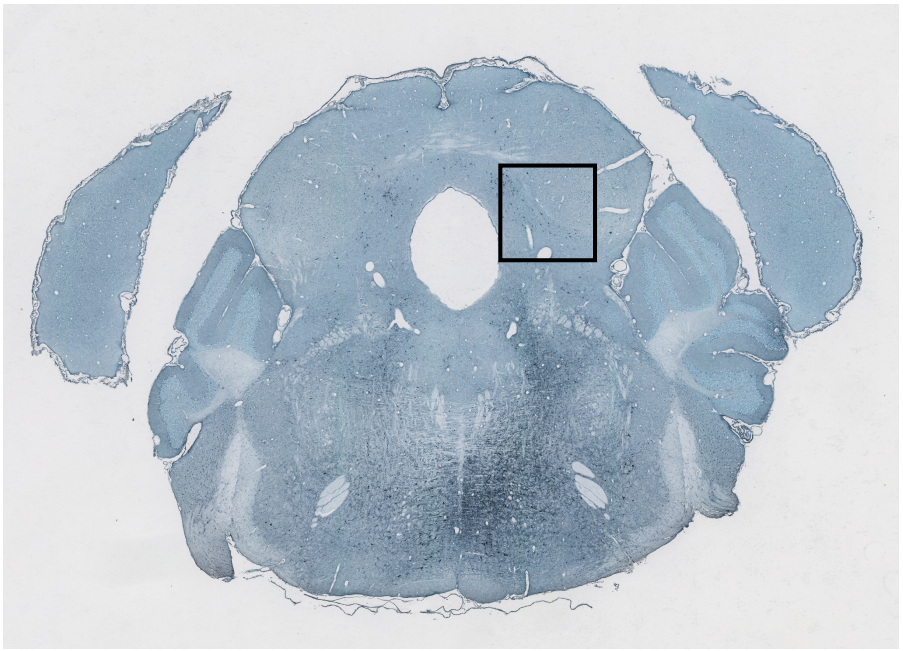


(b) IHC section 188 from stack 2695, OOF image

Figure 70: IHC sections 187 (top) and 188 (bottom) from stack 2695. Both images were identified as OOF but it is not clear at this magnification.

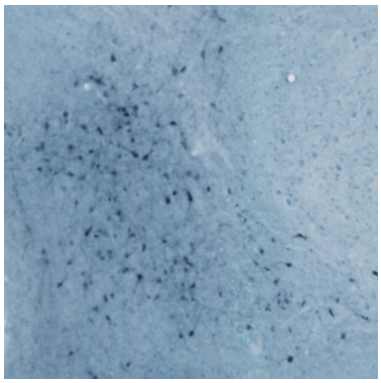


(a) IHC section 189 from stack 2695, OOF image

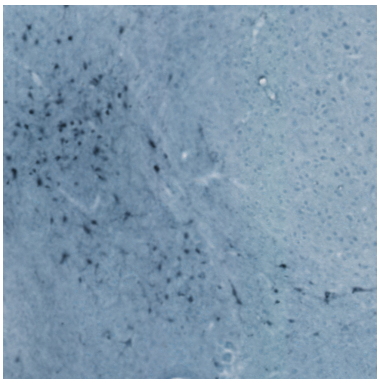


(b) IHC section 190 from stack 2695, not OOF image

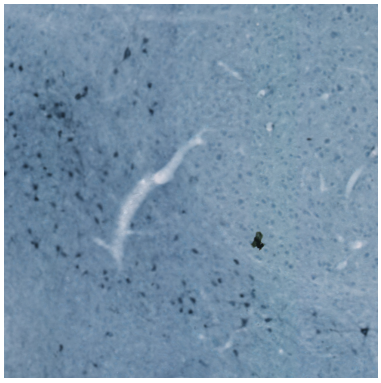
Figure 71: IHC sections 189 (top) and 190 (bottom) from stack 2695. The top image was identified as an OOF by the algorithm while the bottom was identified as not an OOF. Looking at both whole images at this magnification, we can not tell the difference.



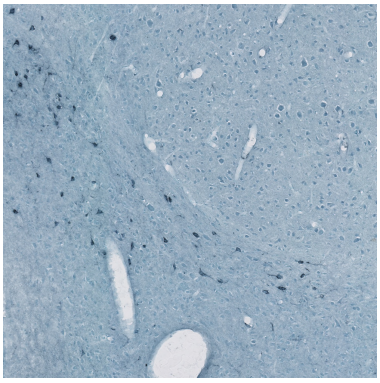
(a) Section 187



(b) Section 188



(c) Section 189



(d) Section 190

Figure 72: Magnified 2000×2000 pixel areas from IHC sections 187, 188, 189 and 190. Images 187, 188 and 189 were identified as OOF by the algorithm while section 190 was identified as not an OOF, at this resolution it is evident the algorithm was successful in identifying the OOF images

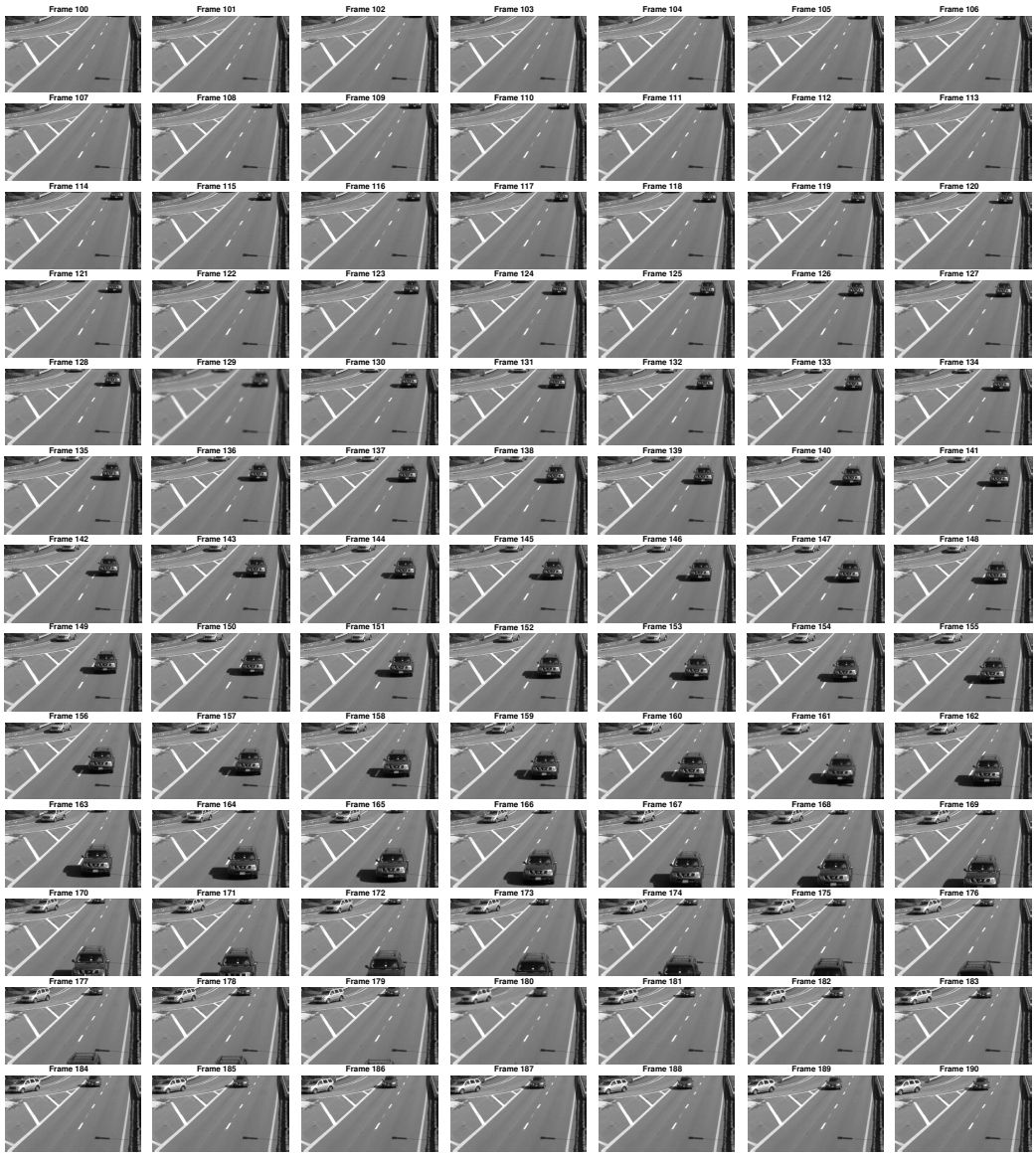


Figure 73: Frames 100-190 from traffic video

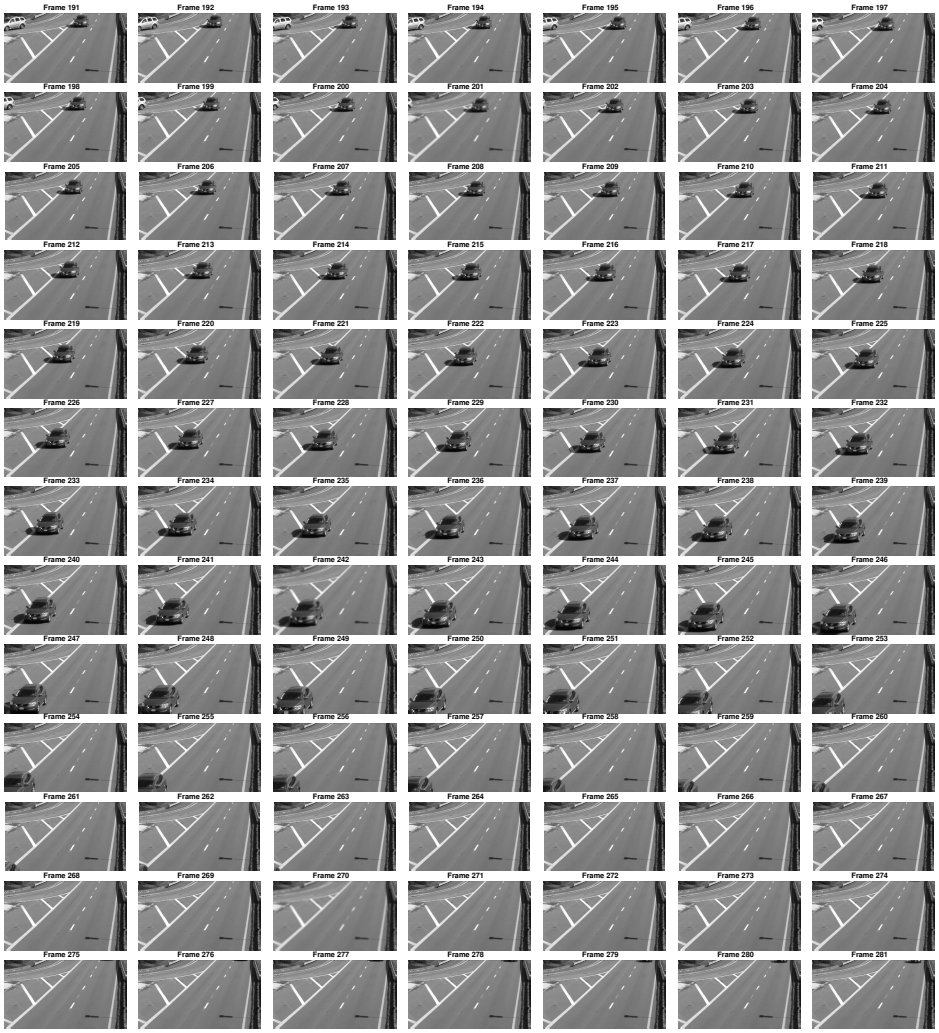


Figure 74: Frames 191-281 from traffic video

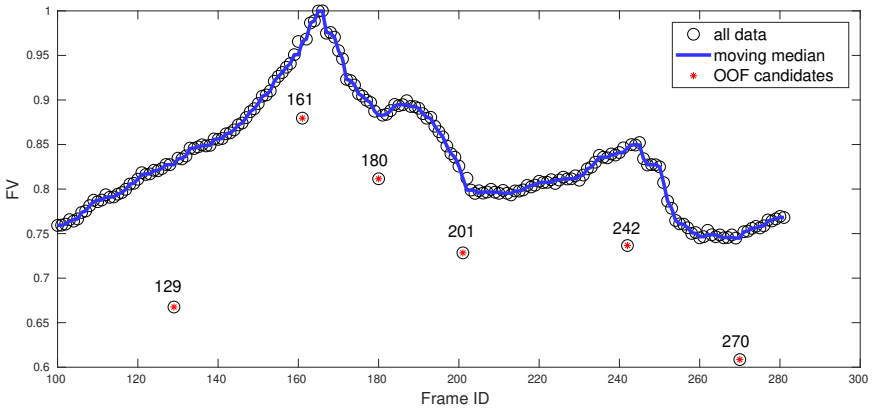


Figure 75: Focus value (FV) versus frame ID for the traffic video. The black $\circ$ symbol represents all the data, the blue solid line is the moving window data points and the OOF candidates are shown as the red $*$ symbol.



(a) Frame 129, Gaussian filter



(b) Frame 161, median filter



(c) Frame 180, median filter (partial)



(d) Frame 201, Gaussian filter



(e) Frame 242, mean filter



(f) Frame 270, circular averaging filter

Figure 76: Blurred traffic images identified as OOF by the algorithm.

5 Striping removal

5.1 Introduction

Brain images taken using optical microscopy can often suffer from imaging artifacts such as horizontal or vertical stripes, those artifacts appear in the biological images taken for the MBA project. These stripes degrade the image quality in fluorescent biological imaging samples and can also limit the accuracy and compromise any subsequent analyses such as neuron segmentation. The *striping* noise effect is the result of the fact that the scanner works in lanes due to the large size of the sample compared to the scanner size. It manifests itself as a pattern of regular, repetitive stripes of the same shape and size throughout entire images.

In order to create a high quality 3D map of the brain an automatic way to segment neurons will be employes. In order to improve the accuracy and quality of the cell segmentation algorithms we first have to remove any damage or artifacts that can negatively affect further data analyses. The striping is such an artifact and it is part of all the images. Unlike the OOF effect that happens in only a few images, the striping happens in all the fluorescent images so we can not remove those images as we did with the OOF images. Since we can not remove those images since the striping is due to the setup, they have to be restored. In order to remove the stripes from all images we have to devise an automated method that does not need user intervention and no parameter or threshold needs to be known or empirically estimated. Such a method will allow us to process large datasets in parallel batching. Another important condition is that the striping removal method has to remove the striping effect but at the same time preserve the image details and quality in order for any method to be successful.

5.2 Previous methods of removing striping effects

Contamination of image data with striping effects is a very common phenomenon in meteorological satellite and atmosphere imagery (such as Moderate Resolution Imaging Spectroradiometer (MODIS)), in spaceborne and airborne multi-detector remote sensing imaging systems (such as Landsat Thematic Mapper (TM)), in astronomy (IRAS images, Lunar Orbiter) and Ocean Remote Sensing (MOS).

The striping problem is approached with many different methods in those

fields. One of the most common method is removing the stripes in the frequency domain. There are several techniques such as Fourier analysis [9, 46, 33], wavelet analysis [8, 51] and a combined wavelet-Fourier filter [31]. The success of those methods is based on the fact that the periodic stripe noise can be identified in a power spectrum and then filtered out. The disadvantage of such methods is that structures of the images that have the same frequency as the stripes would also be removed, which could cause artifacts or blurring in the image. More improved methods have been utilized [34] but they need several filtering parameters to be determined such as the wavelet type, number of decomposition levels and a filtering threshold value which will make these methods unsuitable for automatic removal of stripes from a series of images such as those found in our datasets.

Another category of techniques are based on statistical methods, like moment matching [18] and histogram modifications [55] but they are not particularly efficient for non-linear stripes. In order to overcome this disadvantage, a piece-wise approach is presented that can work for non-linear and irregular MODIS data [43]. These methods would not be successful in destriping our images, because the area that the stripes occupy compared to the whole image is very small.

Apart from the frequency domain, there are methods where some selected stable scenes are used to calculate the relationship between different detectors that are then utilized to correct the other images [2, 16]. The limitation of these methods is that the different scene contents can make the valid coefficients vary which is turn can create artifacts in the corrected images.

Another category of destriping methods is based on finding the location of the stripes but a threshold ([52, 24]) and a filtering window size ([24]) have to be empirically selected first which is not an option for automatic analyses.

Although most of the methods described are from satellite imagery, similar methods are used in microscopy images for the purpose of structural biology. There are methods based on frequency domain analysis [45, 42, 14], smoothing operations based on mathematical morphology [26], and a moving median filter [13].

As we will discuss later, any frequency domain analysis is not appropriate for this analysis due to the nature of the stripes in our images. Using a mathematical morphology would require us to make a choice on the shape and size of the morphological element that will be used. A similar choice has to be made for the median filter method as well, the size of the window

that gives the best results had to be determined. Both determinations have to be done empirically, with human intervention, which is not appropriate for our data analysis since the goal is to create an algorithm that will run automatically for several image stacks. Any choice we make can potentially cause issues because of the possibility that they are not appropriate for all the structures of our images that could have varying sizes. Additionally, any method that requires a reference image will not be applicable in our case since images without stripes for our data samples are not available.

5.3 Data description

One of the preprocessing steps we have to complete is removing the artifacts that manifest as stripes in our images. Because of the large size of the data we need to create preprocessing algorithms that can run automatically, and our algorithm has to be capable of batch and parallel data processing so they can be completed in a realistic amount of time.

A typical image is presented Fig. 77. We notice the visible stripes that split our image into several *panels*, which is how we will refer to the data between stripes. Our stripes are not dense and they seem visually periodic. As we will discuss in the next section, the stripes that are visible are caused by two effects. Initially, there is a slow degradation of the data towards the end of each panel. While this slow drop in intensity happens, the illumination does not return to levels similar to before the intensity drop. The cause of this is the uneven illumination within each panel. This added complication is why the methods that have been mentioned in the previous sections would not have been successful in completely removing the stripes in our data.

Because of the nature of our stripes, any frequency domain methods will not work because our stripes are not an added noise that has a specific frequency which means that they will not appear in the frequency domain as an additional peak on top of the signal data. Additionally, since our stripes are not dense compared to the amount of pixels in the whole image (all stripes combined occupy less than 1% of the whole image), they would not be visible in the frequency domain.

In order to remove the stripes we implemented a procedure that requires five steps in order to remove the stripes and balance the illumination in each image. The steps of the algorithm are:

1. Determine where the brain is located and separate it from the back-

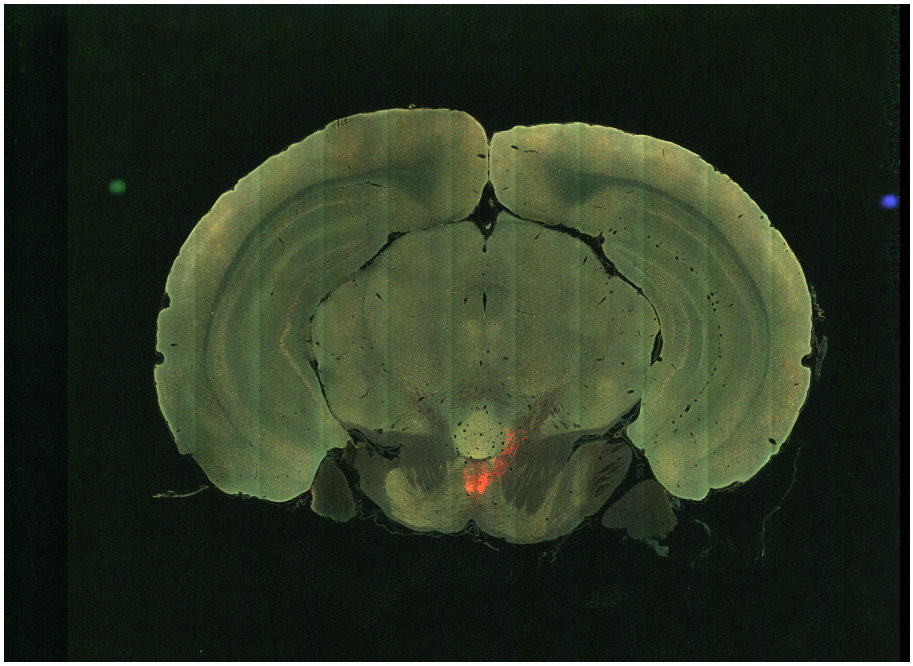


Figure 77: **Original fluorescent color image section 147 from stack 2941 with stripes.** This is an original color image with stripes before any processing, the size of the image is 19726 (rows) ×27156 (columns). There are at least ten visible vertical stripes on the sample, there a few more in the background but not visible due to its color.

ground.

2. Locate the position and size of the stripes using a filter.
3. Modeling and smoothing of the stripes.
4. Balancing the uneven illumination throughout the image.
5. Correction of intensity variations within each of the panels.

Due to the added complication of uneven illumination, all the above steps are necessary in order to create destriped images that are as close to the original image as possible in areas that are not affected by the stripes.

5.4 Proposed Destriping Method

We are going to describe the steps needed to remove the stripes from our images. Our method is based on modeling of the spatial domain. The image that we will use as an example is shown in Fig. 77.

In order to show the structure of the stripes, we chose three random areas from the cropped image, shown in Figure 78. If we zoom in the three area If we zoom in and visually inspect the stripes in this image, we notice that there is no noise added on top of the image but rather a change in illumination as clearly shown in the areas shown in Fig. 79.

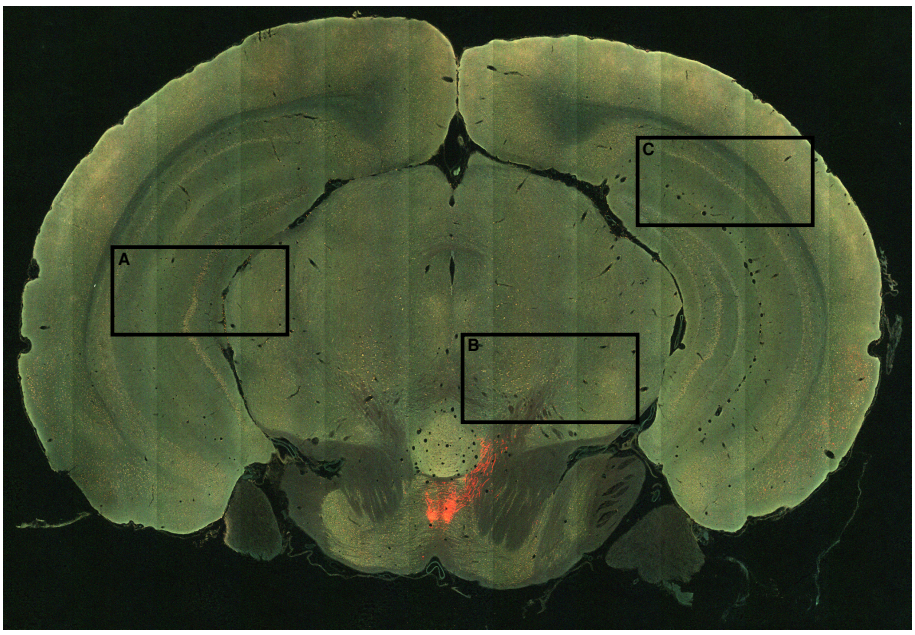


Figure 78: **Cropped original fluorescent color image of section 147 from stack 2941.** Three random areas were picked so that we inspect the structure of the stripes, the whole cropped image is 14231 (rows) $\times$20759 (columns) pixel size

Visual inspection alone is not sufficient to determine how these stripes were created. In the following steps, we will explore these stripes quantitatively so we can design a method to remove them. The nature of these stripes is the reason why a frequency domain method cannot be successful.

In the following sections we will describe each of the five steps needed for a successful removal of the stripes.

5.4.1 Background and Sample tissue image separation

Our images include brain tissue but also contain a large area that does not include any brain sample data, which we label the background. We want to locate the part of the image that includes the tissue without the background. There are important reasons why we want to locate and separate the background. First, by excluding the background and using only the area where the tissue is located, our program runs considerably faster during the analysis because the image is smaller. This is important since our images are very large and the analysis is time intensive, so any steps to minimize the size of the images that we analyze should be taken. The second reason is because in later steps we will use the background for modeling and correction of the illumination of the image so we need it to be separated from the brain tissue.

In order to separate the brain tissue from the background we use an edge detection algorithm as we have described in Chapter 3. The smallest rectangle that includes all the brain tissue will be referred to as the *Sample* from now on. The *Background* image will be the rectangle that has the same number of columns as the Sample image that lies either directly above or below the Sample. Henceforth, any capitalized use of Sample and Background will refer to the specific regions of the original image as we defined above. Both Sample image and Background are shown in Fig. 80.

The evaluation of the Background image will allow us to estimate the location of the stripes in the next step.

5.4.2 Spatial location of stripes

The next step is to uncover the location and the actual size of the stripes in a quantitative way. In order achieve this, we will utilize the Background image that we have separated in the previous step. The reason we are using the Background is because we want to use an image that is as homogeneous as possible where the only variability of the data is the striping (in reality the Background would not be perfectly homogeneous due to other artifacts, but it is the closest we have to a completely smooth area). Several edge detection filters were tested in order to find the one that enhanced the stripes and attenuated the other data points. The ones that worked the best was the

Sobel filter and the top-hat filter. After testing both filters in *Backgrounds* from several images we discovered that the Sobel filter was slightly superior in detecting the lines in images that had very bright injection spots or very bright artifacts, so it is used for all images. If we apply the Sobel filter to the Background image we can visually see the location of the stripes. The Sobel filtered gray scaled Background image is shown in Fig. 81 where we can clearly see the location of the stripes. Although the lines are more clear now, visual inspection will not be very accurate and can not be automated, so we need to find an algorithmic way of calculating the position of the stripes.

The tool we will use to find the exact location of the stripes in the image is the sum of pixel values of each column divided by the number of rows, which we will refer to as the *mean cross-track profile* of the image. The pixel positions of the stripes will be identified once we calculate the local minima of the mean cross-track profile of the gray scale filtered image, as we see in Fig. 82.

Thus, the Sobel filter of the Background image reveals the position and size of the stripes which we will use in the following steps in order to smooth them.

5.4.3 Smoothing the stripes

After we have located the stripes, we will take steps in order to smooth the lines so that the striping is not visible. Since we know the position of the stripes from the Background, we will take the mean cross track profile of the Sample image so that we can examine how the stripes are translated into the numeric data. As we can see in Fig. 83 the reason why we visually register a stripe in the image is because there is a sharp drop of intensity at the location of the same minima we found earlier.

We will enlarge two selected portions of the image so we can see a more detailed structure of the intensity drop. In Fig. 84 we present the pixels in the neighborhood of the second and fourth stripe in the red channel of the Sample image as typical examples. The middle red asterisk is the location of the second minimum and fourth minimum of the image.

Spline interpolation can be used to correct the drop in intensity between the panels. We will fit a second degree polynomial to the mean cross profile data as shown in Eq. (13),

$$y = a_2 x^2 + a_1 x + a_0 \tag{13}$$

By fitting the coefficients a_2, a_1 and a_0 we have an equation that models the data in each stripe. Using this interpolation we can model the data point in each stripe (reference S1 Appendix). After we correct all the data points that create the intensity drop, the mean cross profile around the stripes looks like Fig. 85.

After we model and correct the intensity drop we notice that there is also a discontinuity between the two panels. In order to remove the stripe completely we have to scale each panel to each other so that the discontinuity between the panels is eliminated. We assume that the reason for this gap is because of the instrument which means that the scaling between all panels of the same image should be the same. In order to find the scaling factor in each image we calculate the ratio between 'A' and 'B' (as labeled on the right side of Fig. 85) for all stripes in the image. The scaling factor we will use to multiply each panel is estimated by the median ratio between points A and B for all stripes. If the median ratio is n_{scl} then we multiply the second panel with that number, then the third panel with n_{scl}^2 and so on for all panels. Eventually all intensity drops and gaps are smoothed and because we have only intervened in the areas between the panels thus far, the overall structure of the data in the areas away from the stripes has been preserved.

When all the smoothing and scaling is done, the mean cross track profile does not have any intensity drops anymore, but it has an ascending trend as shown in Fig. 86 due to the uneven illumination of each panel. In the next step we will balance this ascending trend so that the image is not brighter on the right side.

5.4.4 Balancing the uneven illumination

As we see in Fig. 86, when we remove the intensity drops the profile exponentially increases which will result in an image that becomes increasingly brighter on the right side. The reason for this is because the panels have uneven illumination. In order to balance the illumination from the left to the right side of the image, we use a exponential fit for the whole image. In order to do this we can not use the mean cross profile of the Sample image since there is a great deal of structure from the tissue that prevents a good fit. To create a good fit, we can use the Background mean cross profile after we have smoothed the stripes and scaled the panels as shown in Fig. **??** for R, G and B channels. We will model the Background profile using an exponential curve function as shown in Eq. (14)

$$f(x) = ae^{bx} \tag{14}$$

Once we have fit the variables of this exponential we can use the model to alter the data points of the Sample image as described in the S1 Appendix. If our Background is very noisy and the fit can not be done reliably, there is an alternative way to find the variables of the exponential. We can create an independent exponential curve using the middle pixel position from each panel as the x (dependent) data points and the scaling for each panel as the y data points to fit this curve. Both methods resulte d in the same outcome. Fig. 88 shows the exponential curves of both methods on top of the Background profile.

If we correct the data points of the whole image using the exponential fit, our image will have a mean cross profile that looks flat across the image. The corrected Sample image mean cross profiles is shown in Fig. 89. Visually, this will translate into an image that has the same brightness between its two edges. There is one final correction that is needed due to intensity variations within each panel compared to the original image.

5.4.5 Correction of panel intensity variations

There are some differences when we compare the Sample mean cross profile of the initial image to the corrected image. The first is that we no longer see the sharp drop of intensity in the location of the stripes, which is the desired result. But we also observe a difference in each panel in the areas away from the stripes. If we overlay the data as in Fig. 90 we notice a slight rotation of the corrected data compared to the initial image.

In Fig. 91 we view an enlarged view of the mean cross profile of one of the panels so we can see the difference more clearly. The difference is that the panel of the image without stripes has a slight clockwise rotation which in the image will be translated as an uneven illumination, i.e. the right side of each panel will be darker and the left will be slightly lighter. This effect is a remnant of the uneven illumination of the panels that we discussed earlier. Collectively, in the whole image it will visually look like a smooth periodic variation in each panel and in order to correct this variation we will need to smooth each panel individually. We have to build a model that will allow for a smooth change of the data in each side of each panel but in a way that it does not undo any of our previous corrections. We start by estimating the mean cross profile difference between the original image and the corrected

image of each panel. The difference has a linear form but using this function will reintroduce the stripes, so we need to smooth the data with a function that does not have sharp edges like a Fourier model presented in Eq. (15),

$$f(x) = a_0 + a_1 \cos(xw) + a_2 \sin(xw) \tag{15}$$

Once we fit the a_0, a_1, a_2 and the w parameters we will use this model to correct every data point in the image. The results of this correction are shown in Fig. 92.

For this correction we can either use the Background (so we do not have to worry about the structure of the image) or the Sample image, there was no difference in our data since it is based on modeling the difference between the original image and the corrected image. The resulting image after all these steps is shown in Fig. 93. Visually we do not observe any stripes and the structure of the image has not changed.

Fig. 94 shows five smaller regions within the image enlarged in order to visually present the effectiveness of the stripe removal method. Not e that not only have the stripes been smoothed, but the structures in the formerly striped areas are now preserved and their intensity is balanced.

5.5 Results

We will use qualitative and quantitative criteria to show the success of the method in removing the stripes while preserving the quality and the structure of the image. There are several methods that can be used but we are limited by the fact that we do not have an image that has been unaffected by the stripes in order to compare how close the final image of our method resembles an ideal image.

5.5.1 Visual inspection

One of the qualitative ways we will use to confirm that the stripes were removed is a visual inspection of the image. Based on our result image which is shown in Fig. 93 our method can be deemed successful in removing the stripes visually while the structures of the image seem unaffected. Another qualitative method is inspecting the mean cross profiles of the destriped image as shown in Fig. 95.

Looking at the profile and comparing with the initial mean cross profiles in Fig. 83, we notice that the sudden drops in intensity have been removed

and there is no uneven scaling between the panels anymore. Both visual inspections, although a good indication that the method is successful in removing the stripes, are not completely dependable. The reason for that is that the human eye is not reliable in accessing the brightness differences because it is highly adaptable to the surrounding changes. Also, it is a very objective way of determining the success of the method and it depends on the skill of the observer. For those reasons we have to supplement qualitative criteria to show that the quality of the image has not deteriorated due to the removal of the stripes and most of the changes are in the areas were the stripes are located, while the areas away from the stripes were unaffected.

Any metric that we devise has to be a metric that does not need an image that has been unaffected by stripes for comparison because such images are unavailable. Additionally, because the nature of the stripes in our images is unique (as we have discussed in the Section Previous Work), we can not compare our method with other methods used before in a quantitative way.

5.5.2 Mean and standard deviation

There are several metrics we will use to show that the areas away from the stripes were unaffected. The first and most simple quantitative method that we will use to show that the areas away from the stripes were unaffected, is the calculation of the local mean and standard deviation in regions of interest away from the stripes [9, 31]. The smaller the change between those two values, the more the information of the initial image has been preserved. We define eleven regions of interest (ROI) in the image that have a size of 1000×1000 pixels that are away from the stripes in order to demonstrate the fact that the quality of the areas away from the stripes has a minimal change. These ROIs are shown on top of the original image in Fig. 96.

The mean and standard deviation for those regions of the image are presented in Table 10 for comparison and it shows that the mean values do not exhibit more than a 1% change. For simplicity, we are using the values of the gray scale image. These are representative of the changes in each of the color channels.

Region ID	Original Image		DeStriped Image		% change	
	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.
1	32.98	11.80	33.07	11.84	0.27	0.34
2	28.57	7.04	28.79	7.11	0.77	0.99
3	31.18	5.93	31.42	5.97	0.77	0.67
4	32.34	6.52	32.63	6.59	0.90	1.07
5	31.66	9.61	31.89	9.73	0.73	1.25
6	45.21	9.74	45.57	9.79	0.80	0.51
7	31.65	7.03	31.81	7.06	0.51	0.43
8	32.76	6.52	33.02	6.59	0.79	1.07
9	36.71	9.41	37.07	9.44	0.98	0.32
10	32.70	7.99	32.99	8.00	0.89	0.13
11	41.79	10.41	42.17	10.40	0.91	0.10

Table 10: **Mean and standard deviation comparison**. Mean and standard deviations of the different regions of the gray scale image that are away from the original stripes for the original and the destriped image. The last two columns show the percentage change.

5.5.3 Mean Relative Deviation

Another metric we will use is the *mean relative deviation* (MRD) [44, 27, 10]. The metric is defined in Eq. (16)

$$\text{MRD} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{I_D(i,j) - I_O(i,j)}{I_O(i,j)} \tag{16}$$

where I_D and I_O are the regions that have not been affected by the stripes in the destriped and original image respectively and m and n are the rows and columns of these regions. This metric will be used in regions that were unaffected by the stripes. A smaller percentage of this metric is desirable as it shows that the selected ROIs of the image were minimally affected by the corrections. The values of the MRD in percentage are shown in Table 11, our values show that the change of values is generally under 1%.

5.5.4 Structural Similarity Index

The next metric we will use is the Structural Similarity Index (SSIM) [53]. The SSIM is different than the previous metrics in that it is dependent on three different terms: luminance, contrast and structure.

$$\text{SSIM}(x, y) = [l(x, y)]^{\alpha} \cdot [c(x, y)]^{\beta} \cdot [s(x, y)]^{\gamma} \tag{17}$$

Eq (17) shows the three terms, where x, y are two aligned image signals and α, β and γ are parameters we can adjust depending on the relative importance of each term. If one of the two images is considered to be a perfect quality then this measure can be used to measure the quality of the other image. Since we do not have an ideal image without stripes we can only use this index in the areas which are unaffected from the stripes to show that they retained their original structure. The areas away from the stripes in the original image are considered the control area and we will compare them with the image without stripes that has been derived from our method. The indices are also shown in Table 11 and demonstrate how affected the areas were away from the stripes. Two identical images will have a SSIM index equal to 1, so a value as close to 1 as possible is desirable, which is what we have found.

Region ID	**MRD%**	**SSIM**
1	0.30	0.9975
2	0.75	0.9972
3	0.78	0.9967
4	0.88	0.9968
5	0.71	0.9974
6	0.81	0.9973
7	0.52	0.9976
8	0.79	0.9974
9	1.03	0.9971
10	0.92	0.9972
11	0.99	0.9970

Table 11: MRD and SSIM values of the selected ROIs of the image

As we have discussed earlier, the SSIM index is a comparison between two images of which one has to be a perfect image, so any value derived

by comparing the whole original image with the whole destriped image does not have any mathematical meaning. But a map depicting the local values of the SSIM index between the two images can be used to illustrate their differences. Fig. 97 shows that the image is mostly unaffected except the areas close to the stripes.

The darker parts of the image is where we have the largest change of the data. This image shows how the data in the stripes have the most change and as we move further away from the stripes the trend is towards smaller changes.

5.5.5 Image Focus

Another metric we will use is the focus of the image before and after we used the destriping method. Since the image was numerically manipulated we want to ensure that the sharpness of the image has not deteriorated. We will use a focus value (FV) calculated using the Steerable Filters (SF) method as we have explained with detail in the previous chapter. We will use the ROIs of the image unaffected by stripes we used before to calculate the focus and compare to the original ROIs. The results for the different regions are shown in Table 12. We can see that the FV values typically change by less than 1%.

5.5.6 Other metrics

There are several other metrics that are used to quantify the quality of the image after the stripes have been removed but are not appropriate for our method with our particular images. The inverse coefficient variation (ICV) [40, 48, 32, 7] is the ratio of the signal response (average value within a window of a given size) over the noise components (standard deviation within a window). Although this is an non-reference index it can not be used in our case because we can not compare our results with other methods since those other methods do not work for cases of uneven illuminations like our images.

Another metric that is used to compare the effectiveness of a destriping method is the peak signal-to-noise ratio (PSNR) [7, 4] which needs a reference image without stripes, so it is not appropriate for comparison. A quality index called IF1 [4, 11] was also devised as a measure of the improvement factor of the image but it can not be used unless it is compared with other

Region ID	Original FV	Destriped FV	% FV change
1	1.5761	1.5791	0.1903
2	1.4916	1.4808	0.7241
3	1.3713	1.3614	0.7219
4	1.5192	1.5019	1.1388
5	1.6078	1.5978	0.6220
6	1.8099	1.7982	0.6464
7	1.5931	1.5882	0.3076
8	1.5248	1.5116	0.8657
9	1.6044	1.5898	0.9100
10	1.5335	1.5198	0.8934
11	1.6773	1.6631	0.8466

Table 12: Focus values of the different ROIs of the image for the original and destriped image. The last column shows the percentage change in the FV value.

methods, since it is a relative metric.

Additionally as we have discussed before the stripes occupy less than 1% of the whole image,so any metric that uses the whole image might not register the change.

5.6 Discussion

The method presented was successful in removing stripes from biological images. It has been tested for stacks of images that can be close to 1 billion pixels in size and they illustrate a variety of structures down to the individual neurons. The goal is to remove the stripes while at the same time preserving all the formations of the image which have an uneven size and contrast for further analyses. The stripes in the images are artifacts that are created due to an intensity value drop between the panels and an uneven illumination between them. The added complication of the uneven illumination within each panel in the areas between the stripes has made the use of more traditional methods ill-advised. Our stripes are not an added noise on top of the image data, thus their cumulative data values will not be seen as a spike in any of the frequency domain plots, therefore they can not be removed in

this way. Additionally, all our stripes combined occupy less than 1% of the entire image, they are not dense, so any such signal would be weak. Methods like the Fourier analysis [9, 46, 33] and wavelet-Fourier filter method [34, 31] were attempted by the author but the stripes were still clearly visible. Since those methods did not achieve stripe removal even on the visual level we did not compare them on a quantitative level with our method. Additionally, any method that required the knowledge or empirical estimation of several parameters or thresholds was dismissed as it would not allow for automatic image processing or for parallel computation.

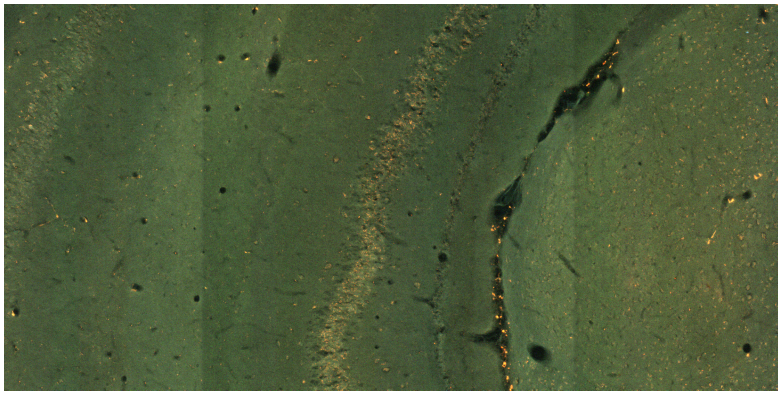
We have used the mean cross profile in order to make the different corrections to the image through several steps that have been described in detail. Our method is based on initially locating the position of the stripes by using a Sobel filter. Once the position of the stripes is known, they are smoothed using a polynomial fit and then each panel is scaled so that any discontinuity is eliminated. The next step requires modeling of the background image that we have extracted from our original image in order to balance the uneven illumination throughout the image. A final correction is needed because of the intensity variations within each of the panels. Since there are no parameters that have to be empirically known or estimated beforehand, this method is appropriate for an automatic run of several series without manual interaction from a user. All fitting and modeling can be done separately for each image to improve accuracy, there is no need for comparison between images.

We have presented several qualitative and quantitative metrics to show that the image was mostly affected in the areas of the stripes while the areas away from the stripes kept the same numeric profile. Visual inspection, although necessary to show that the stripes were removed and that the regions away from the stripes kept their original structure unchanged, is not a objective method to rate the success of the method. In order to calculate the quantitative metrics we defined eleven ROIs that are away from the stripes and evaluated and compared different parameters in the original and the destriped images. All the different tests show minimal changes in the areas away from the stripes which is the goal. Ideally, a clean unstriped image that could be used as a model image with which we can compare our destriped images would be useful, but that is not an option with our current experimental setup. Therefore, we have only used non-reference metrics to show the effectiveness of our method in removing the stripes.

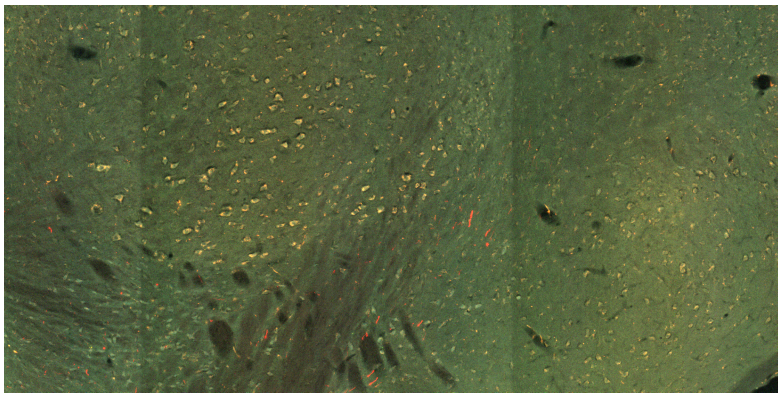
5.7 Conclusion

In this study we have presented an automatic way to remove repeated straight (horizontal or vertical) image striping in fluorescent images. The technique is data-driven so no prior knowledge regarding the microscope behavior is needed. There are five steps that are described: separation of Background and Sample, spatial location of stripes, stripes smoothing, uneven illumination balancing for the whole image and a final intensity variation correction for each panel separately. The method can be repeated automatically for whole series of images and no parameter or threshold needs to be known ahead of time. Although this data analysis can be computationally intensive for large images like ours, because each image can be analyzed separately this problem can be alleviated by running different series of images in parallel.

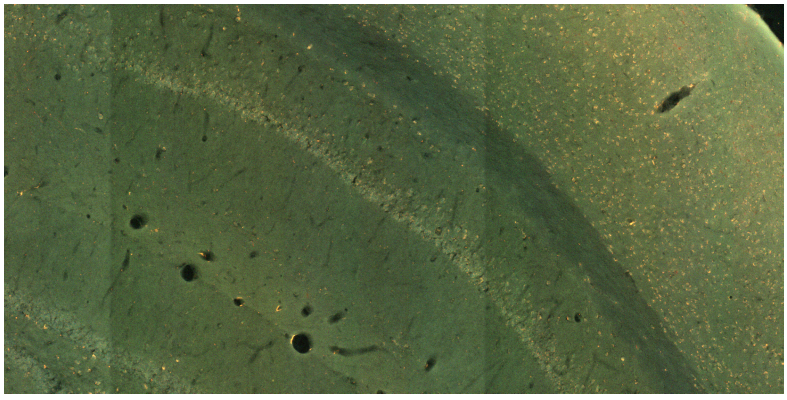
Our method resolves both the striping problem and the uneven illumination that is causing it. Several qualitative and quantitative criteria were used to validate our method of destriping the image without the introduction of any distortions or artifacts. Appropriate metrics were used to show that the regions away from the stripes suffered minimal adjustments. The method could possibly be used in images from other fields that suffer from the same striping problem.



(a) Area A, 2000 × 4000 pixels



(b) Area B, 2000 × 4000 pixels



(c) Area C, 2000 × 4000 pixels

Figure 79: **Details of the fluorescent original color image of section 147 from stack 2941 with stripes**.This is a magnification of three random small sections of the original image that include two clearly visible vertical stripes before any processing.

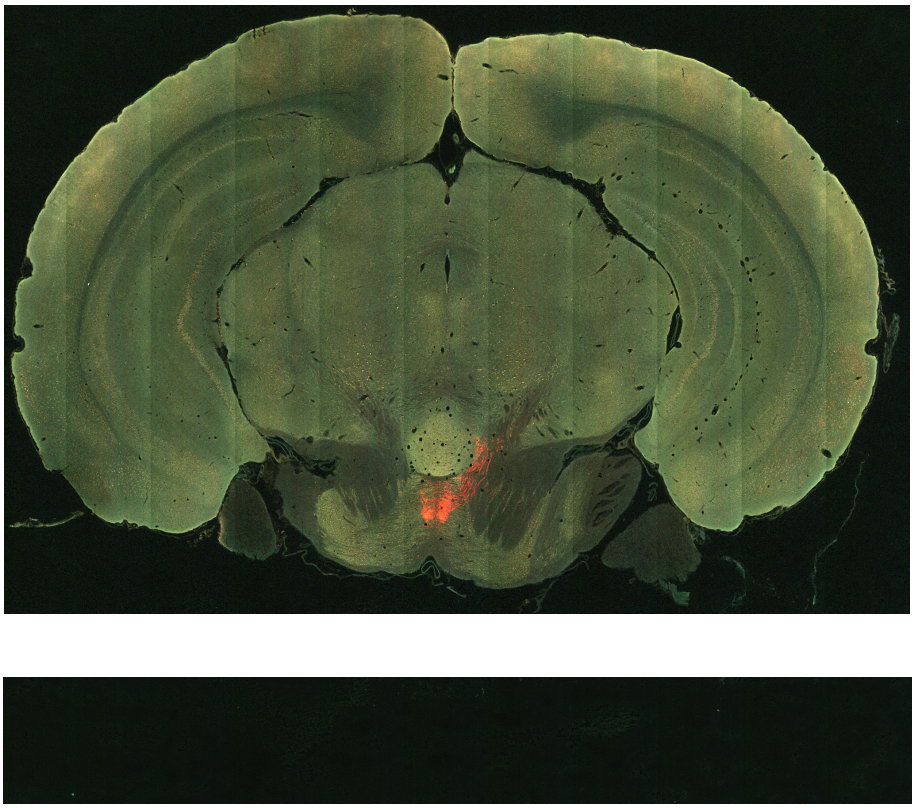


Figure 80: **Color Sample image and Background image.** The top image depicts a full color Sample image and the bottom image shows the Background image as defined in Section 5.3.1.

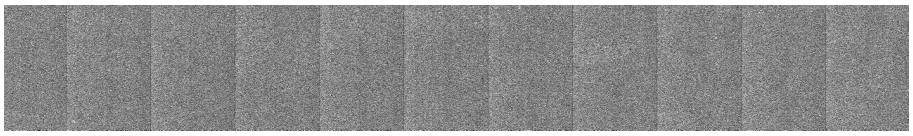


Figure 81: **Sobel filtered Background image.** This is the gray scale Background image after it has been processed with a Sobel filter. The ten stripes of the cropped image are clearly visible. We can see how they split the image into eleven panels.

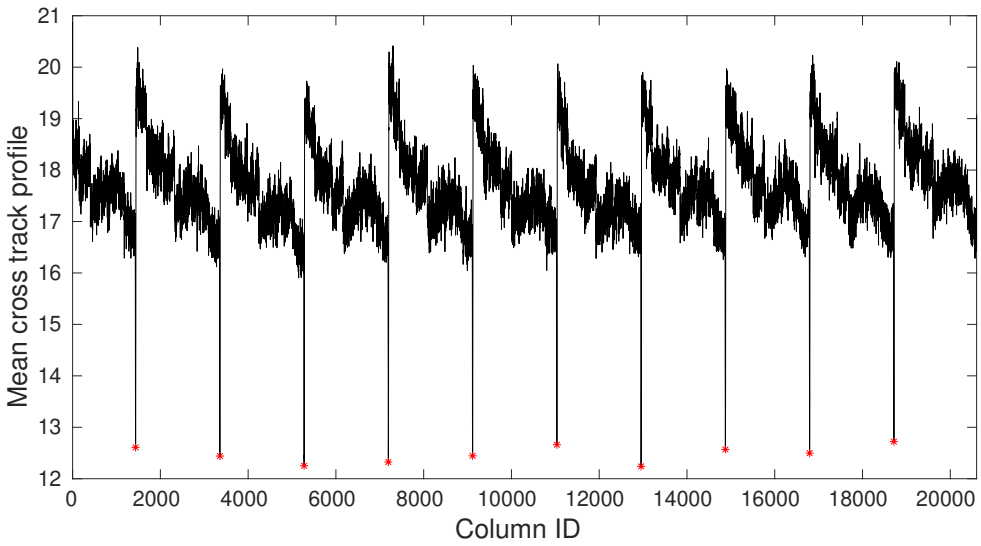


Figure 82: **Mean cross track profile of the Background.** The mean cross track profile of the Sobel Background gray scale image reveals the local minima which are marked in red. This plot allows us to identify the size and exact location of the stripes.

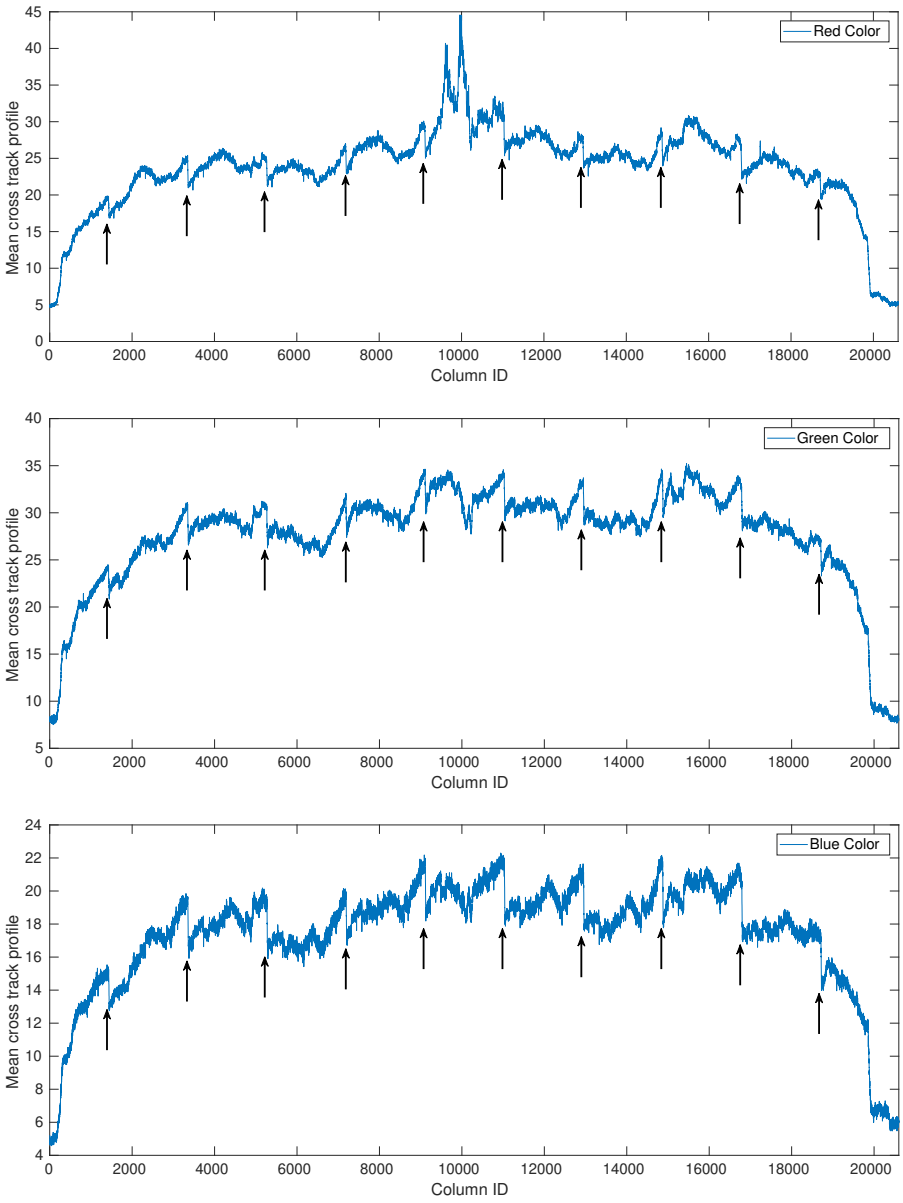


Figure 83: **Mean cross track profiles for the Sample color image.** The mean cross track profile of the Sample image in all three channels (R, G, B). The black arrows show the minima positions that were identified from the Sobel filter of the Background in the previous section. The additional peaks in the center of the red channel are due to the staining with a marker in that area, which can clearly be seen on the bottom of the image in Fig. 80.

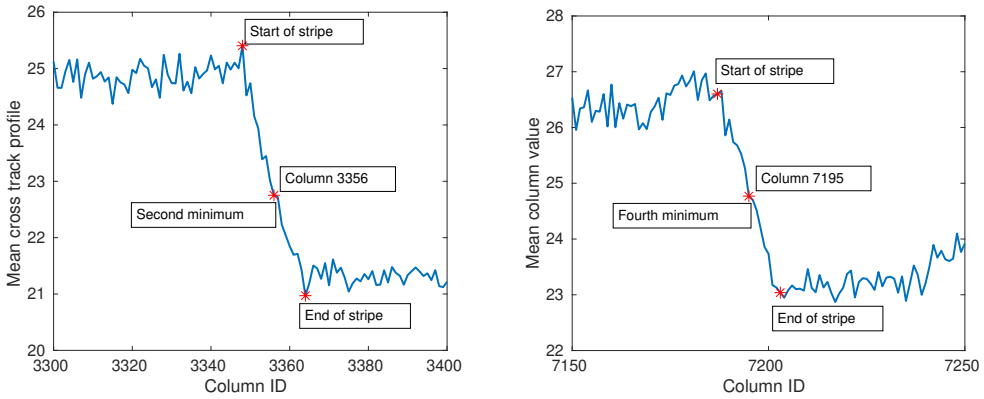
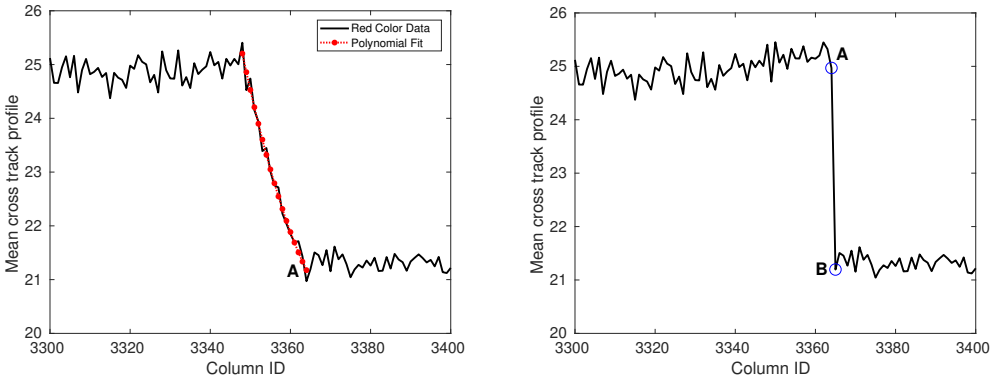


Figure 84: **Profile of the second and fourth stripe of the Sample image.** We are presenting two different stripes from the red channel of the Sample image. The middle red point in both images is the location of the minima while the other two points show the beginning and end of the stripe.



(a) Second stripe with polynomial fit in red. (b) Second stripe after polynomial correction.

Figure 85: The second stripe before and after the polynomial fit. After the correction we need to calculate the scaling factor between points A and B in subfigure (b) so that the discontinuity is removed and the stripe is completely smoothed.

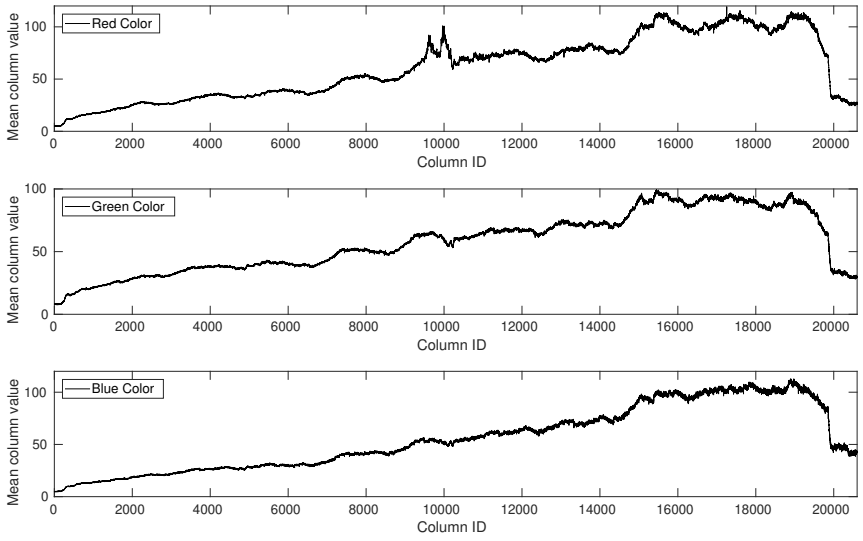


Figure 86: **Mean cross track profiles for R, G, B channels of the image after smoothing and scaling of the stripes.** The rising illumination is the result of an uneven illumination of each panel.

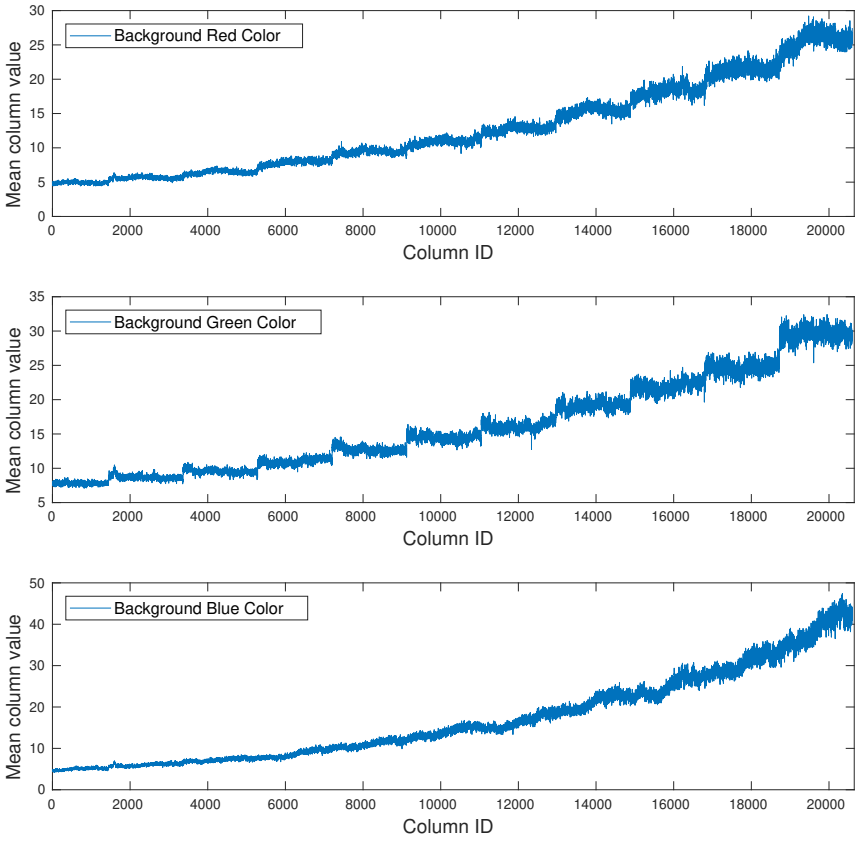


Figure 87: **Cross track profiles for R, G, B Background after smoothing and scaling of the stripes.** These are the mean cross track profiles for the Background image after the stripes were smoothed. Similar to the Sample image, we see the rising illumination towards the right side of the image.

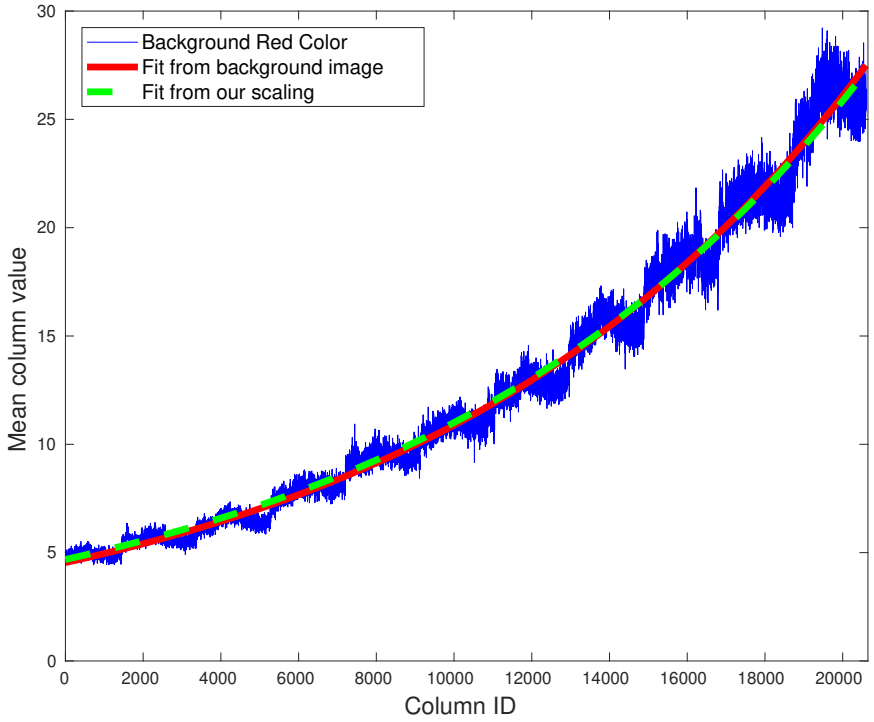


Figure 88: **Mean cross track profile for Red Background with an exponential fit.** The mean cross profile for the Background image in the Red channel, after we have smoothed and scaled the stripes, is shown in blue. The red line is the exponential fit directly from the mean cross profile and the green line is the fit resulting from using our scaling versus panel data points. Based on the g raph, there is an agreement between the two fits.

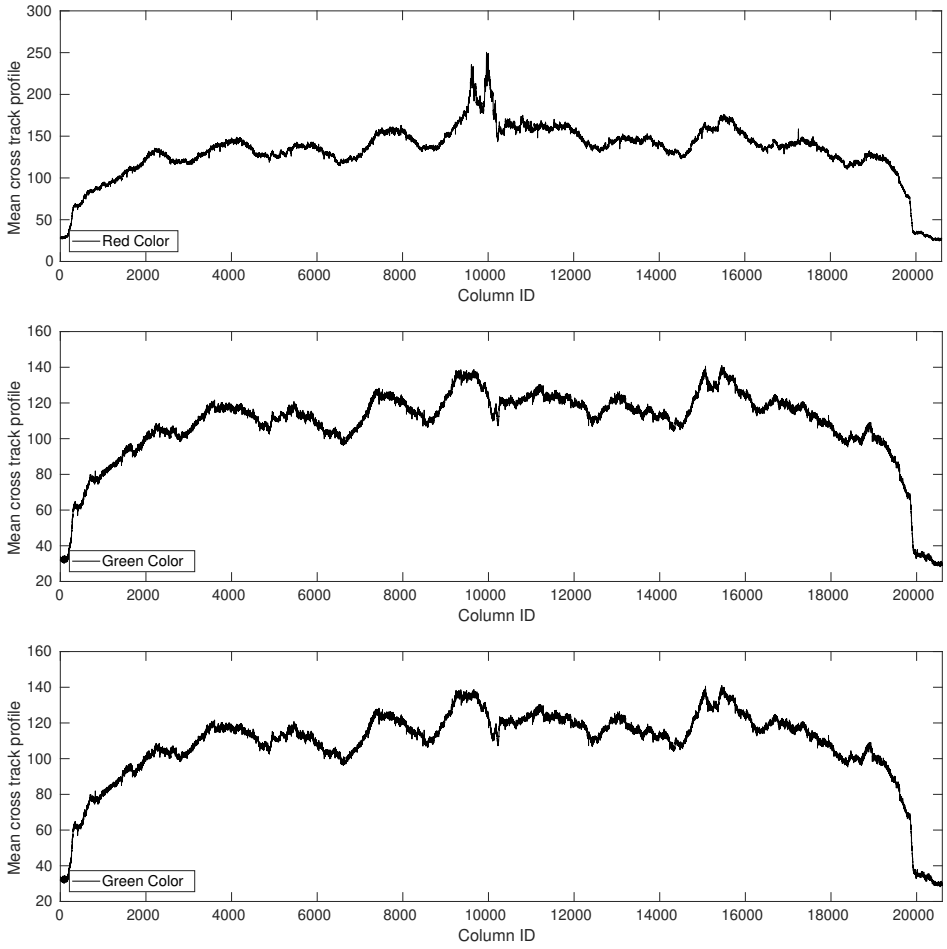


Figure 89: **Mean cross track profiles for the Sample R, G, B channels after balancing the illumination.** After the correction the Background does not have the ascending trend anymore but we see a the individual structure in every panel.

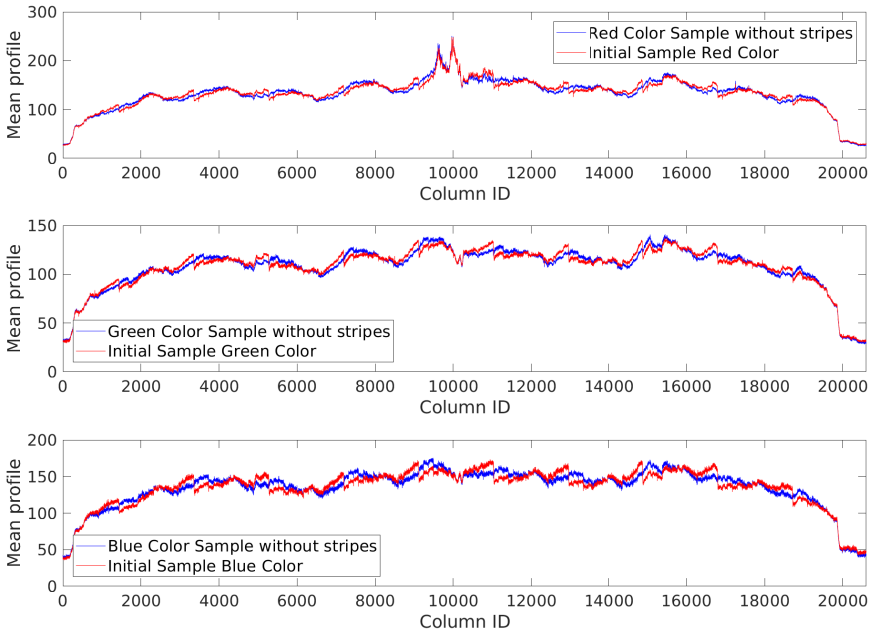


Figure 90: **Mean cross track profiles comparison between original Sample R, G, B and corrected Sample images.** After we complete the smoothing of the stripes and balance the illumination of the image, the mean cross profile of the resulting Sample image is shown in blue and has been overlayed by the original Sample image profile (in red) to highlight the slight rotation between the two sets of data.

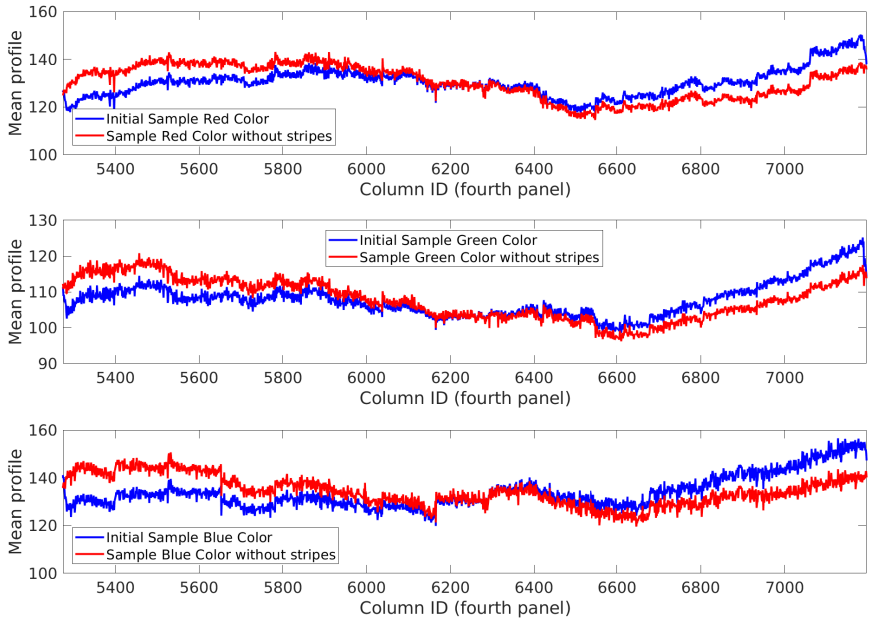


Figure 91: **Mean cross track profiles for Sample R, G, B before and after smoothing the lines.** In order to showcase the slight rotation between the cross track profile of the original image (in red color) and the corrected image (in blue color), we concentrate on the fourth panel of the Sample image.

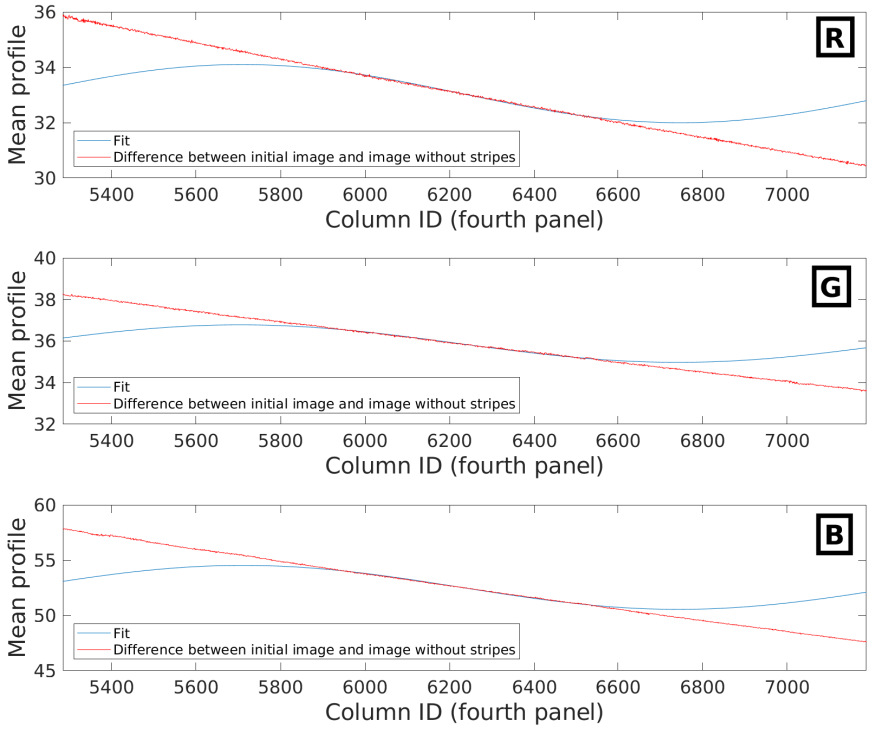


Figure 92: **Modeling the intensity panel variations.** The difference between the image without stripes and the original image has a linear form and is shown in red. The blue color shows the Fourier model we used to model panel four so that the mean cross profile resembles the initial profile.

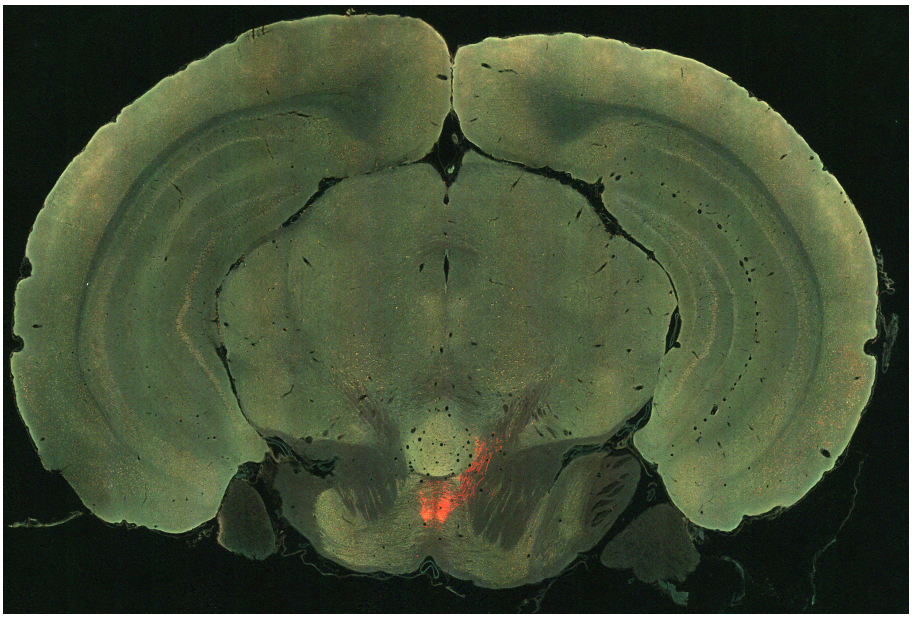


Figure 93: **Color Sample image without stripes.**. This is the resulting color image after all the steps have been completed. The stripes have been removed and the illumination is balanced.

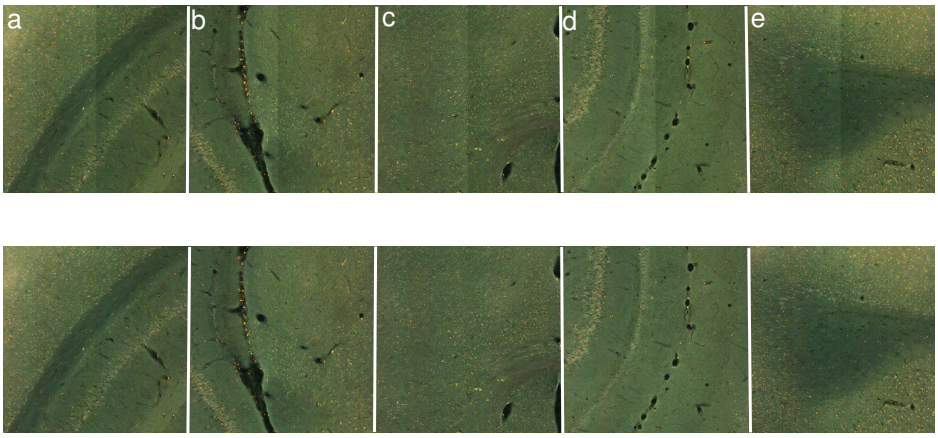


Figure 94: **Comparison of different areas of the image before and after destriping.** We selected five random regions to show the effectiveness of the method in removing the stripes while the quality and structure of the image is preserved.

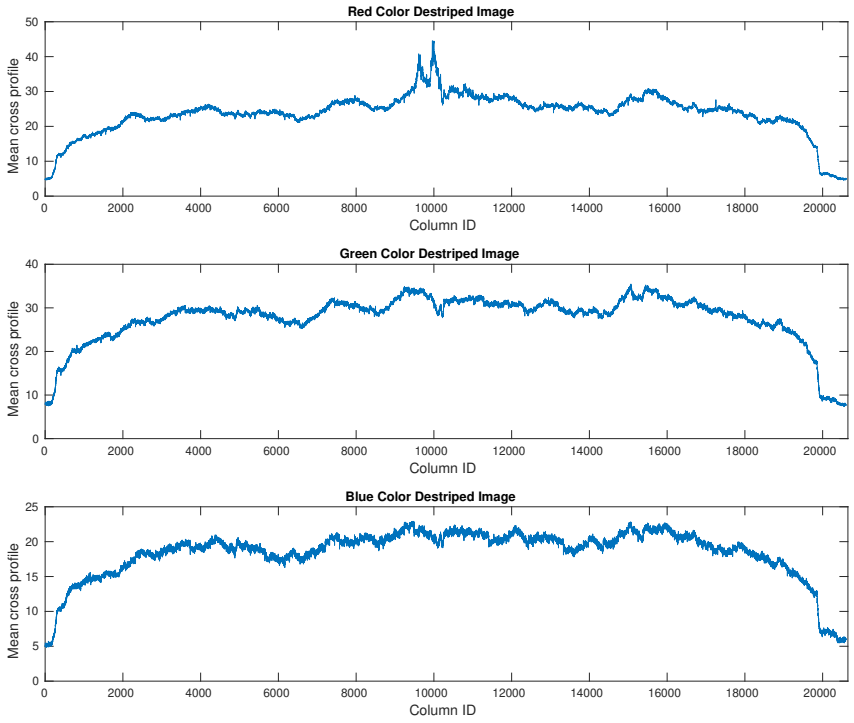


Figure 95: **Mean cross profile of destriped image in R, G, B channels.** The sudden drops in intensity and the uneven scaling of the panels has been removed, while the illumination is balanced throughout the image. The fluctuations we see in the mean cross profile now reflect the structures in the Sample.

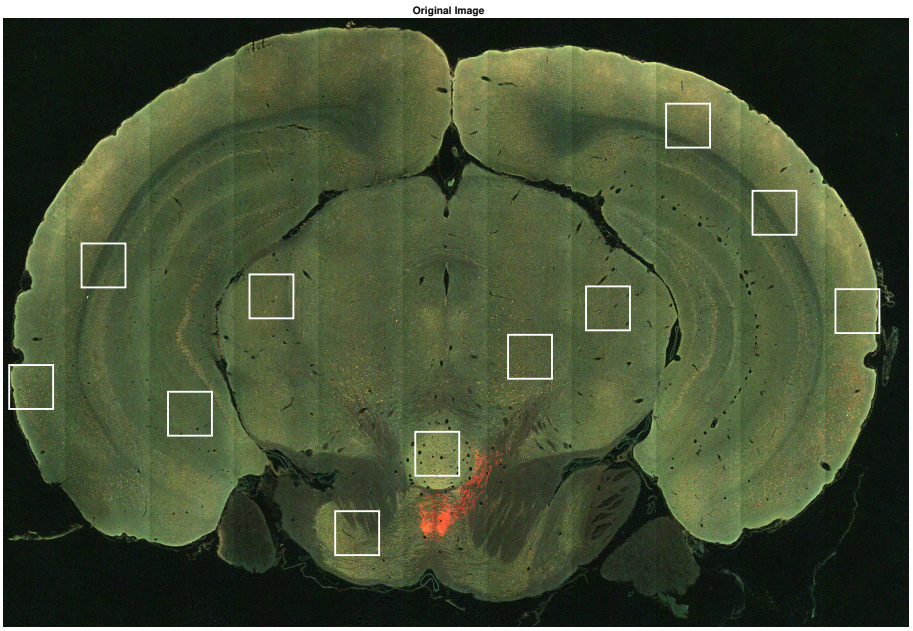


Figure 96: **Original image with selected ROIs for comparison**. Original image with selected 1000 × 1000 pixel ROIs away from the stripes. The ROIs are used to compare the values before and after the destriping of the image in order to assess the quality of the method.

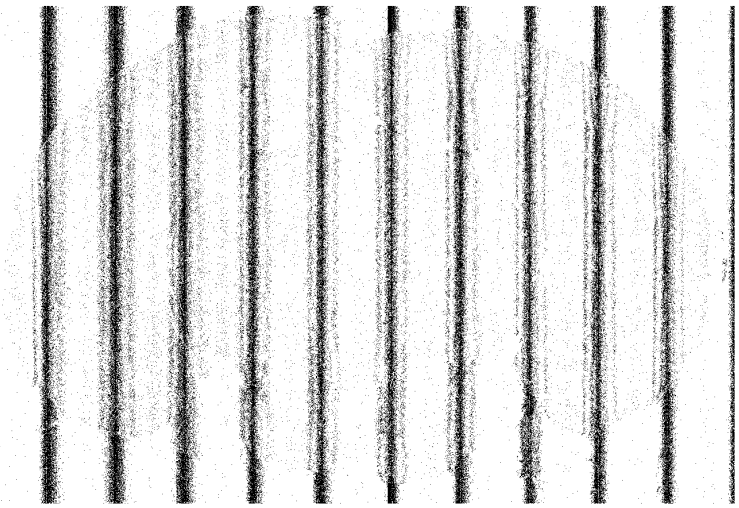


Figure 97: **SSIM index map.** This map shows the local values of the SSIM index that is derived after comparing the initial image with the destriped image. We notice that the largest changes are in the areas of the stripes (darker areas), while further away the changes are smaller (lighter areas).

6 Supporting information

S1 Appendix. Image modeling using the mean cross profile.

Many times during the process we have to use the model of the image from the mean cross profile in order to correct all the data points of the image. In order to do that there are a few steps needed to go from the mean cross profile to correcting all the data points of the image according to that model.

If we assume that the image we want to change is called I, the function we use to fit the mean cross profile is called $f(x)$ and n_r, n_c are the number of rows and columns in I respectively.

When we evaluate the function $f(x)$ for each of the columns, we create a data array called f_n which has one dimension which is equal to the number of columns.

We then create a new array where we replicate the data array f_n by the number of rows of the image, n_r times. This array is then a new image I_m which has the same size as the initial image. In order to find the percent correction of I_m we divide it with the maximum value,

$$I_n = \frac{I_m}{\max(I_m)}. \tag{18}$$

In order to correct our initial image I, we divide it by our model Background I_n.

$$I_{\text{corr}} = \frac{I}{I_n} \tag{19}$$

References

[1] Bear, M., Connors, B., Paradiso, M.: Neuroscience: Exploring the Brain. Wolters Kluwer (2015)

[2] Bindschadler, R., Choi, H.: Characterizing and correcting hyperion detectors using ice-sheet images. IEEE Transactions on Geoscience and Remote Sensing **41**(6), 1189–1193 (2003)

[3] Bohland, J.W., Wu, C., Barbas, H., Bokil, H., Bota, M., Breiter, H.C., Cline, H.T., Doyle, J.C., Freed, P.J., Greenspan, R.J., et al.: A proposal for a coordinated effort for the determination of brainwide neuroanatomical connectivity in model organisms at a mesoscopic scale. PLoS computational biology **5**(3), e1000334 (2009)

[4] Bouali, M., Ladjal, S.: Toward optimal destriping of modis data using a unidirectional variational model. IEEE Transactions on Geoscience and Remote Sensing **49**(8), 2924–2935 (2011)

[5] Canny, J.: A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence **8**(6), 679–698 (1986)

[6] Carter, M., Shieh, J.: Guide to research techniques in neuroscience (Second Edition). Academic Press (2015)

[7] Chang, Y., Fang, H., Yan, L., Liu, H.: Robust destriping method with unidirectional total variation and framelet regularization. Optics express **21**(20), 23307–23323 (2013)

[8] Chen, J., Lin, H., Shao, Y., Yang, L.: Oblique striping removal in remote sensing imagery based on wavelet transform. International Journal of Remote Sensing **27**(8), 1717–1723 (2006)

[9] Chen, J., Shao, Y., Guo, H., Wang, W., Zhu, B.: Destriping cmodis data by power filtering. IEEE Transactions on Geoscience and remote sensing **41**(9), 2119–2124 (2003)

[10] Chen, Y., Huang, T.Z., Zhao, X.L., Deng, L.J., Huang, J.: Stripe noise removal of remote sensing images by total variation regularization and group sparsity constraint. Remote Sensing **9**(6), 559 (2017)

[11] Corsini, G., Diani, M., Walzel, T.: Striping removal in mos-b data. IEEE Transactions on Geoscience and Remote Sensing **38**(3), 1439–1446 (2000)

[12] Davies, L., Gather, U.: The identification of multiple outliers. Journal of the American Statistical Association **88**(423), 782–792 (1993)

[13] Ding, W., Li, A., Wu, J., Yang, Z., Meng, Y., Wang, S., Gong, H.: Automatic macroscopic density artefact removal in a nissl-stained microscopic atlas of whole mouse brain. Journal of microscopy **251**(2), 168–177 (2013)

[14] Fehrenbach, J., Weiss, P., Lorenzo, C.: Variational algorithms to remove stationary noise: applications to microscopy imaging. IEEE transactions on image processing **21**(10), 4420–4430 (2012)

[15] Finn, E.S., Shen, X., Holahan, J.M., Scheinost, D., Lacadie, C., Papademetris, X., Shaywitz, S.E., Shaywitz, B.A., Constable, R.T.: Disruption of functional networks in dyslexia: a whole-brain, data-driven analysis of connectivity. Biological psychiatry **76**(5), 397–404 (2014)

[16] Fischer, A.D., Thomas, T.J., Leathers, R.A., Downes, T.V.: Stable scene-based non-uniformity correction coefficients for hyperspectral swir sensors. In: Aerospace Conference, 2007 IEEE, pp. 1–14. IEEE (2007)

[17] Freeman, W.T., Adelson, E.H.: The design and use of steerable filters. IEEE Transactions on Pattern Analysis & Machine Intelligence **13**(9), 891–906 (1991)

[18] Gadallah, F., Csillag, F., Smith, E.: Destriping multisensor imagery with moment matching. International journal of remote sensing **21**(12), 2505–2511 (2000)

[19] Genetic Science Learning Center, U.o.U.: Neurons transmit messages in the brain. `https://learn.genetics.utah.edu/content/neuroscience/neurons/`

[20] Geusebroek, J.M., Cornelissen, F., Smeulders, A.W., Geerts, H.: Robust autofocusing in microscopy. Cytometry: The Journal of the International Society for Analytical Cytology **39**(1), 1–9 (2000)

[21] Gonzalez, R.C., Woods, R.E.: Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA (2006)

[22] Gross, G.G., Junge, J.A., Mora, R.J., Kwon, H.B., Olson, C.A., Takahashi, T.T., Liman, E.R., Ellis-Davies, G.C., McGee, A.W., Sabatini, B.L., et al.: Recombinant probes for visualizing endogenous synaptic proteins in living neurons. Neuron **78**(6), 971–985 (2013)

[23] Hudspeth, A.J., Jessell, T.M., Kandel, E.R., Schwartz, J.H., Siegelbaum, S.A.: Principles of neural science (5th Edition). McGraw-Hill (2013)

[24] Jung, H.S., Won, J.S., Kang, M.H., Lee, Y.W.: Detection and restoration of defective lines in the spot 4 swir band. IEEE Transactions on Image Processing **19**(8), 2143–2156 (2010)

[25] Krotkov, E., Martin, J.P.: Range from focus. In: Robotics and Automation. Proceedings. 1986 IEEE International Conference on, vol. 3, pp. 1093–1098. IEEE (1986)

[26] Leischner, U., Schierloh, A., Zieglgänsberger, W., Dodt, H.U.: Formalin-induced fluorescence reveals cell shape and morphology in biological tissue samples. PloS one **5**(4), e10391 (2010)

[27] Lu, X., Wang, Y., Yuan, Y.: Graph-regularized low-rank representation for destriping of hyperspectral images. IEEE transactions on geoscience and remote sensing **51**(7), 4009–4018 (2013)

[28] MATLAB: version 9.3 (R2017b). The MathWorks Inc., Natick, Massachusetts (2017)

[29] Minhas, R., Mohammed, A.A., Wu, Q.J., Sid-Ahmed, M.A.: 3d shape from focus and depth map computation using steerable filters. In: International Conference Image Analysis and Recognition, pp. 573–583. Springer (2009)

[30] Mitra, P.P.: The circuit architecture of whole brains at the mesoscopic scale. Neuron **83**(6), 1273–1283 (2014)

[31] Münch, B., Trtik, P., Marone, F., Stampanoni, M.: Stripe and ring artifact removal with combined waveletfourier filtering. Optics express **17**(10), 8567–8591 (2009)

[32] Nichol, J., Vohora, V.: Noise over water surfaces in landsat tm images. International Journal of Remote Sensing **25**(11), 2087–2094 (2004)

[33] Pan, J.J., Chang, C.I.: Destriping of landsat mss images by filtering techniques. Photogrammetric engineering and remote sensing **58**(10), 1417–1423 (1992)

[34] Pande-Chhetri, R., Abd-Elrahman, A.: De-striping hyperspectral imagery using wavelet transform and adaptive frequency domain filtering. ISPRS journal of photogrammetry and remote sensing **66**(5), 620–636 (2011)

[35] Pearson, R.K.: Outliers in process modeling and identification. IEEE Transactions on control systems technology **10**(1), 55–63 (2002)

[36] Pertuz, S., Puig, D., Garcia, M.A.: Analysis of focus measure operators for shape-from-focus. Pattern Recognition **46**(5), 1415–1432 (2013)

[37] Pinskiy, V., Jones, J., Tolpygo, A.S., Franciotti, N., Weber, K., Mitra, P.P.: High-throughput method of whole-brain sectioning, using the tape-transfer technique. PloS one **10**(7), e0102363 (2015)

[38] Pratt, W.K.: Introduction to Digital Image Processing. CRC Press, Taylor and Francis Group, Boca Raton, FL, USA (2013)

[39] Prewitt, J.M.: Object enhancement and extraction. Picture processing and Psychopictorics **10**(1), 15–19 (1970)

[40] Rakwatin, P., Takeuchi, W., Yasuoka, Y.: Stripe noise reduction in modis data by combining histogram matching with facet filter. IEEE Transactions on Geoscience and Remote Sensing **45**(6), 1844–1856 (2007)

[41] Roberts, L.G.: Machine perception of three-dimensional solids. Ph.D. thesis, Massachusetts Institute of Technology (1963)

[42] Salili, S.M., Harrington, M., Durian, D.J.: Note: Eliminating stripe artifacts in light-sheet fluorescence imaging. Review of Scientific Instruments **89**(3), 036107 (2018)

[43] Shen, H., Jiang, W., Zhang, H., Zhang, L.: A piece-wise approach to removing the nonlinear and irregular stripes in modis data. International journal of remote sensing **35**(1), 44–53 (2014)

[44] Shen, H., Zhang, L.: A map-based algorithm for destriping and inpainting of remotely sensed images. IEEE Transactions on Geoscience and Remote Sensing **47**(5), 1492–1502 (2009)

[45] Shu-wen, W.C., Pellequer, J.L.: Destripe: frequency-based algorithm for removing stripe noises from afm images. BMC Structural Biology **11**(1), 7 (2011)

[46] Simpson, J.J., Gobat, J.I., Frouin, R.: Improved destriping of goes images using finite impulse response filters. Remote sensing of environment **52**(1), 15–35 (1995)

[47] Skudlarski, P., Jagannathan, K., Anderson, K., Stevens, M.C., Calhoun, V.D., Skudlarska, B.A., Pearlson, G.: Brain connectivity is not only lower but different in schizophrenia: a combined anatomical and functional approach. Biological psychiatry **68**(1), 61–69 (2010)

[48] Smith, G.M., Curran, P.J.: Methods for estimating image signal-to-noise ratio (snr). Advances in remote sensing and GIS analysis pp. 61–74 (1999)

[49] Sobel, I., Feldman, G.: A 3x3 isotropic gradient operator for image processing. a talk at the Stanford Artificial Project in pp. 271–272 (1968)

[50] Soille, P.: Morphological Image Analysis: Principles and Applications, 2 edn. Springer-Verlag, Berlin, Heidelberg (2003)

[51] Torres, J., Infante, S.O.: Wavelet analysis for the elimination of striping noise in satellite images. Optical Engineering **40**(7), 1309–1315 (2001)

[52] Tsai, F., Chen, W.W.: Striping noise detection and correction of remote sensing images. IEEE Transactions on Geoscience and remote sensing **46**(12), 4122–4131 (2008)

[53] Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)

[54] Wass, S.: Distortions and disconnections: disrupted brain connectivity in autism. Brain and cognition **75**(1), 18–28 (2011)

[55] Wegener, M.: Destriping multiple sensor imagery by improved histogram matching. International Journal of Remote Sensing **11**(5), 859–875 (1990)