A Comparison of Image Processing Techniques for Bird Detection

A Thesis

presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree of

Master of Science in Electrical Engineering

by

Elsa Reyes

June 2014

© 2014
Elsa Reyes

ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE:	A Comparison of Image Processing Techniques for Bird Detection
AUTHOR:	Elsa Reyes
DATE SUBMITTED:	June 2014
COMMITTEE CHAIR:	Jane Zhang, PhD Professor, Electrical Engineering
COMMITTEE MEMBER:	Bridget Benson, PhD Professor, Electrical Engineering
COMMITTEE MEMBER:	John A. Saghri, PhD Professor, Electrical Engineering

ABSTRACT

A Comparison of Image Processing Techniques for Bird Detection

Elsa Reyes

Orchard fruits and vegetable crops are vulnerable to wild birds and animals. These wild birds and animals can cause critical damage to the produce. Traditional methods of scaring away birds such as scarecrows are not long-term solutions but short-term solutions. This is a huge problem especially near areas like San Luis Obispo where there are vineyards. Bird damage can be as high as 50% for grapes being grown in vineyards. The total estimated revenue lost annually in the 10 counties in California due to bird and rodent damage to 22 selected crops ranged from $168 million to $504 million (in 2009 dollars).

A more effective and permanent system needs to be put into place. Monitoring systems in agricultural settings could potentially provide a lot of data for image processing. Most current monitoring systems however don't focus on image processing but instead really heavily on sensors. Just having sensors for certain systems work, but for birds, monitoring it is not an option because they are not domesticated like pigs, cows etc. in which most these agricultural monitoring systems work on. Birds can fly in and out of the area whereas domesticated animals can be confined to certain physical regions.

The most crucial step in a smart scarecrow system would be how a threat would be detected. Image processing methods can be effectively applied to detecting items in video footage. This paper will focus on bird detection and will analyze motion detection with image subtraction, bird detection with template matching, and bird detection with the Viola-Jones Algorithm. Of the methods considered, bird detection with the Viola-Jones Algorithm had the highest accuracy (87%) with a somewhat low false positive rate. This image processing step would ideally be incorporated with hardware (such as a microcontroller or FPGA, sensors, a camera etc.) to form a smart scarecrow system.

Keywords: bird detection, bird recognition, Viola Jones, template matching, motion detection

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor Dr. Jane Zhang for the continuous support of my studies and research related to my Masters in Electrical Engineering, for her patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Bridget Benson, and Dr. John Saghri, for their encouragement, insightful comments, and hard questions.

My sincere thanks also goes to Dr. James Crenshaw, Dr. Senthil Periaswamy, and Dr. Qiong Liu, for offering me the summer internship opportunities in their groups and leading me working on diverse exciting projects.

A special thanks to my family for their continual support and to my boyfriend Marcos Ramirez for your continual support when even I did not believe in myself.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0

Introduction

Traditional methods of scaring away birds such as scarecrows are only a short-term fix. Scarecrows, raptor models, eye balloons (raptor models and eye balloons are usually shaped to look like hawks and owls), and predator models of cats and snakes are only short term solutions as discussed in [1]. This is a huge problem especially near areas where there are vineyards. Bird damage can be as high as 50% for grapes being grown in vineyards [2]. "The total estimated revenue lost annually in the 10 counties [in California] due to bird and rodent damage to 22 selected crops ranged from $168 million to $504 million (in 2009 dollars)" [3]. Because of this some California counties have even seen an increase in employment for pest control [3].

A more effective and permanent system needs to be put into place. As said before solutions like scarecrows only temporarily scare away the birds until the birds become comfortable with habitat again. Therefore in the long term these methods essentially have an almost 0% success rate. The use of chemicals and netting as a method of bird management in fruit crops is discussed in [4, 5]. They concluded chemicals gave mixed results. They found that the non-treated control system had about 65% undamaged food fruit and the crops with the netting had 80% undamaged fruit. Therefore if we assume that 45% of that damage is from birds, of that 45% netting decreased the amount of damaged fruits by 30% ([80%- 65%] /45%).

Ideally farmers would want a much higher amount of undamaged fruit. Although netting has been proven to be successful, the article [4] mentions how netting can be

expensive. With the combination of labor to put on (and take off) and material costs of wire the cost is over $500 a year for a 10 year period. Netting costs thousands of dollars. A smart scarecrow system would potentially be much cheaper and much more accurate.

Monitoring systems in agricultural settings could potentially provide a lot of data for image processing. Most current monitoring systems however don't focus on image processing but instead really heavily on sensors [6-8] such as temperature/humidity sensors and motion sensors. Just having sensors for certain systems work, but for bird monitoring it is not enough because they are not domesticated like pigs, cows etc. in which most of the current agricultural monitoring systems work on.

The end product to really be able scare away birds would be to setup a smart scarecrow system. The system would have an IP-camera monitoring an agricultural field, a processing component (which is what this paper focuses on) would determine if there are birds in a scene. This would most likely run on an FPGA or microcontroller (out near the agricultural area and receiving data from the IP-camera). The FPGA or microcontroller would then scare away the bird, by sending a message to sound an alarm, shoot water at the area where the birds are located, "walk" toward the birds (like by attaching the FPGA/microcontroller to an autonomous robot) etc. The system diagram of this proposed system is shown in Figure 1.

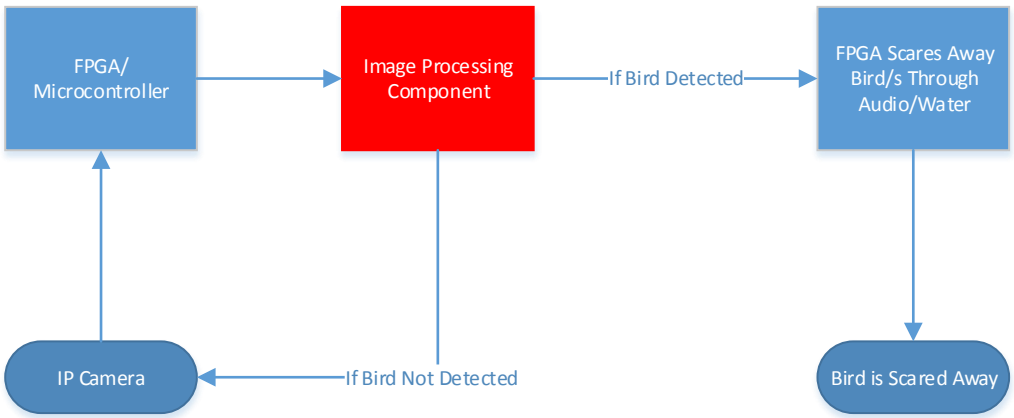

Figure 1: System Diagram of Smart Scare Crow System

The most crucial step in a smart scarecrow system would be how a threat would be detected. Image processing methods can be effectively applied to detecting items in video footage. This paper will focus on bird detection and will analyze motion detection with image subtraction, bird detection with template matching, and bird detection with the Viola-Jones Algorithm. This is why in Figure 1, the image processing component is red because this is what this paper focuses on.

This paper will give a brief overview of what has already been done in this area (this current section). Then it will discuss image difference subtraction, then template matching, and finally the Viola-Jones Algorithm. Next a brief overview of the experimental setup will be given followed by the results and conclusions for the topics previously mentioned.

Some smart scarecrow systems that have been developed in the past are [9, 10]. These systems either did not contain an image processing step or the image processing step was not fully developed. Past scarecrow systems also did not have training data that had backgrounds similar to agricultural backgrounds. As talked about in [11, 12],

pigeons, swallows, starlings, and crows are a threat to central coast agriculture.

The Motion Detection section (Chapter 2) used videos as data because the video frames needed to come from a static video camera and have to be video sequences. It was difficult to find videos of birds which is why the dataset is so small compared to the other two chapters Template Matching (Chapter 3) and Bird Detection Using the Viola-Jones Algorithm (Chapter 4). Both the template matching and Viola-Jones algorithm section used the same data sets as far as bird images. The Viola-Jones algorithm requires additional images of non-birds (unlike the other two methods) as well. The run times shown in this paper do not include the time it takes to load needed images or the time it takes to display figures to Matlab. To insure the run times were correct, the algorithm was run 20 times each time recording the run times. The 20 run times were then average together for an average run time of the algorithm.

2.0

Motion Detection

2.1. Introduction

One of the traditional motion detection algorithms used for tracking objects (usually humans) has been the Kalman Filter [13]. Optical flow and Mean Shift tracking has also been used for bird detection [14, 15]. These methods however are much more computationally intensive than other simpler methods such as background subtraction, or difference subtraction because of all the extra calculations in trying to predict the object's next future positions. Both of these studies', [14, 15], main goal was to be able to track birds. For our case, tracking the bird is not necessary because bird detection should be enough for a smart scarecrow to operate properly. In our case we can avoid storing past information and calculations when new image frames come in from video footage.

Background subtraction would not be a good solution for outdoor video footage. The lighting in the scene would cause the background image to be potentially inaccurate of what the current background is. Motion detection can instead be accomplished by subtracting back to back images as shown in equation 1. The absolute value of this difference then becomes a difference image that can be used to find blobs which are objects in the scene.

$$d_{i,j}(x,y) = \begin{cases} 1, & if \ \left|f(x,y,t_i) - f\left(x,y,t_j\right)\right| > T \\ x, & otherwise \end{cases} \quad (1)$$

This method can be further improved by adding up the accumulative differences. An accumulative difference image (ADI) is formed by comparing this reference image with every subsequent image in the sequence. A counter for each pixel location in the accumulative images is incremented every time a difference occurs at that pixel location

between the reference and an image in the sequence. Often useful is consideration of three types of accumulative difference images: absolute, positive, and negative ADIs. These equations are shown below.

$$A_k(x,y) = \begin{cases} A_{k-1}(x,y) + 1, & if\ |R(x,y) - f(x,y,k)| > T \\ A_{k-1}(x,y), & otherwise \end{cases} \quad (2)$$

$$P_k(x,y) = \begin{cases} P_{k-1}(x,y) + 1, & if\ [R(x,y) - f(x,y,k)] > T \\ P_{k-1}(x,y), & otherwise \end{cases} \quad (3)$$

$$N_k(x,y) = \begin{cases} N_{k-1}(x,y) + 1, & if\ [R(x,y) - f(x,y,k)] < -T \\ N_{k-1}(x,y), & otherwise \end{cases} \quad (4)$$

2.2. Experimental Setup

Four videos of birds moving with a still non-changing background were used to adjust the threshold value that came from [16]. It's important to make sure minimal movement in the background isn't detected as a moving object. Sample images of the videos used are showed in Figure 1 below.

Figure 2: Sample Images from Videos Used for Thresholding

Ten videos of various birds of different backgrounds were used for the actual testing of the motion detection. Most videos only contained one bird in the footage but some contained multiple birds in the footage. Like the videos used for testing, the background is still (static) in the whole footage.

To see if any improvements could be made in motion detection, mathematical morphology operators, opening and closing, were done on the binary image. The reason these morphology operators were added to this experiment were to see if it removed noise and imperfections to the potential "blobs" that represent moving objects. The morphology operators do a good job of region filling in some cases on binary images.

Thinning and/or thickening might better allow for birds to be detected as separate birds and it might better detect the whole bird instead of multiple regions of the bird moving.

Seven different cases were run for testing. The first case did not contain mathematical morphology operators. Two cases contained the opening operator, one with the structuring element being a disk shape and the other a diamond shape. Two cases contained the closing operator, one with the structuring element being a disk shape and the other a diamond shape. Another case used the opening operator and then the closing operator (both with the structuring element being a disk shape). The last case used the closing operator and then the opening operator (both with the structuring element being a disk shape).

As far as other parameters when creating the morphological structuring element, the value 10 was used. This seemed like a number not too big or too small for most of these extra parameters. The code used to make these templates with morphology is in Appendix B4. The code running the templates on bird images is shown in Appendix B5.

For the ADI testing, 5 of the 10 videos used for motion detection were also used. The threshold was set to the optimal threshold found from the initial motion detection testing. As shown from equations 2-4, the testing was done for Absolute, Positive, and Negative cases with varying k values.

2.3. Results

The results for motion detection did not have a column for the average amount of false positives because there were no false positives found in any of tests run. Because the background of the videos were still and the only motion in the videos were birds,

there were not false positives for any of the results in this chapter. Table 1 shows the overall results of motion detection via image subtraction. The cases with the highest accuracies were the case with no morphology operators, both of the closing operators, and the closing then opening operators. The possible reason closing performed better could have been that dilation must be done first to fill in holes in the image. Beginning with erosion makes the blobs in the binary image into smaller blobs causing the object to not be detected as moving.

Table 1: Motion Detection w/Morphology Results

No.	Technique	Accuracy	Run Time in Seconds
1	No Morphology	81.8%	0.18
2	Opening (Disk)	63.6%	0.2948
3	Opening (Diamond)	54.5%	0.1022
4	Closing (Disk)	81.8%	0.1144
5	Closing (Diamond)	81.8%	0.1024
6	Opening then Closing	63.6%	0.1382
7	Closing then Opening	81.8%	0.1336

The image below shows the video sample video at the top and the ADI for technique number 1 (no morphology). The outline of the bird can be seen and ideally morphology will enhance the ADI.

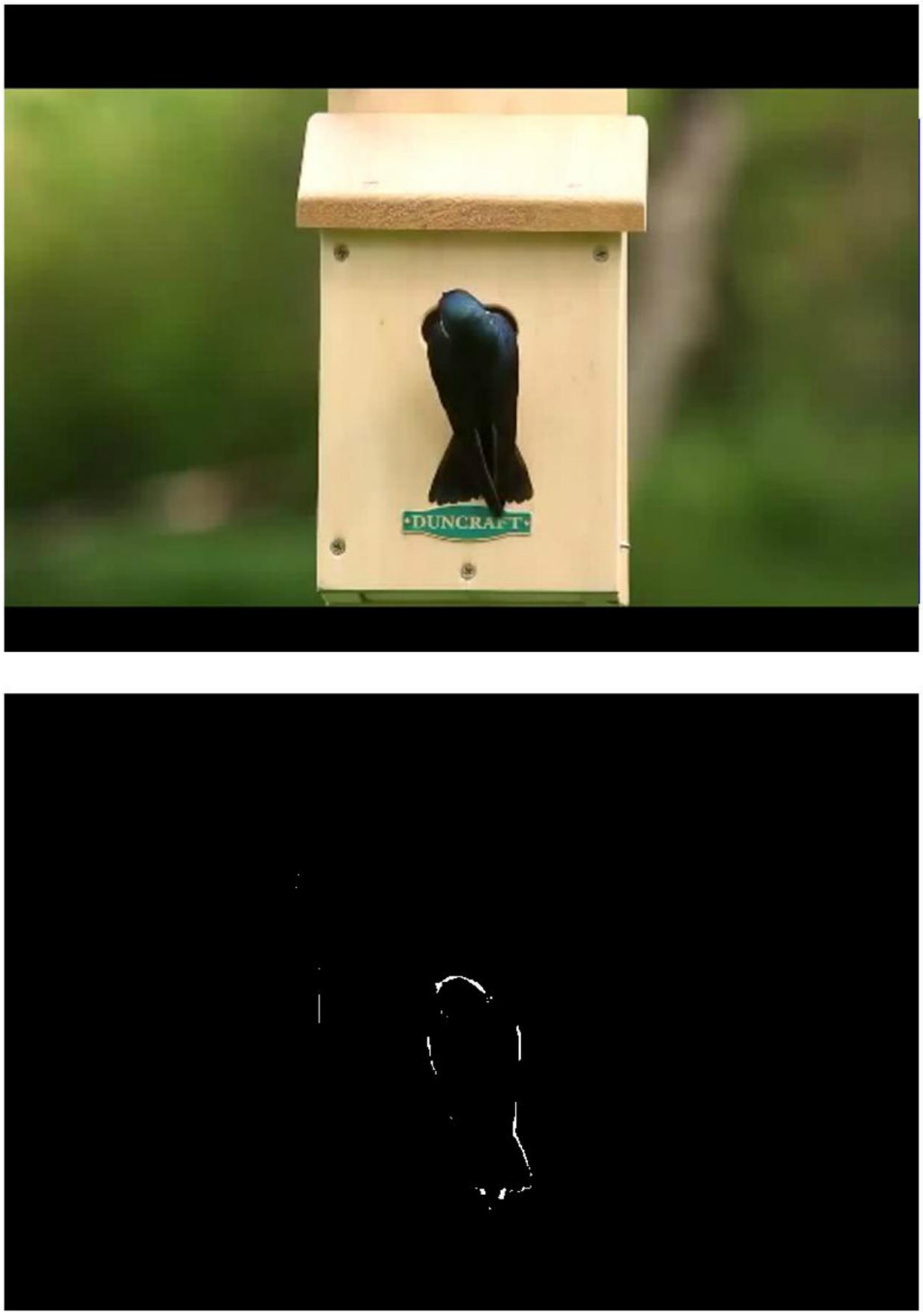

Figure 3: Sample Video Frame for Morphology Operators

As shown in the figure below, only morphology operators number 4 and 5 (the two type of closing) were able to properly find the bird for this sample video frame.

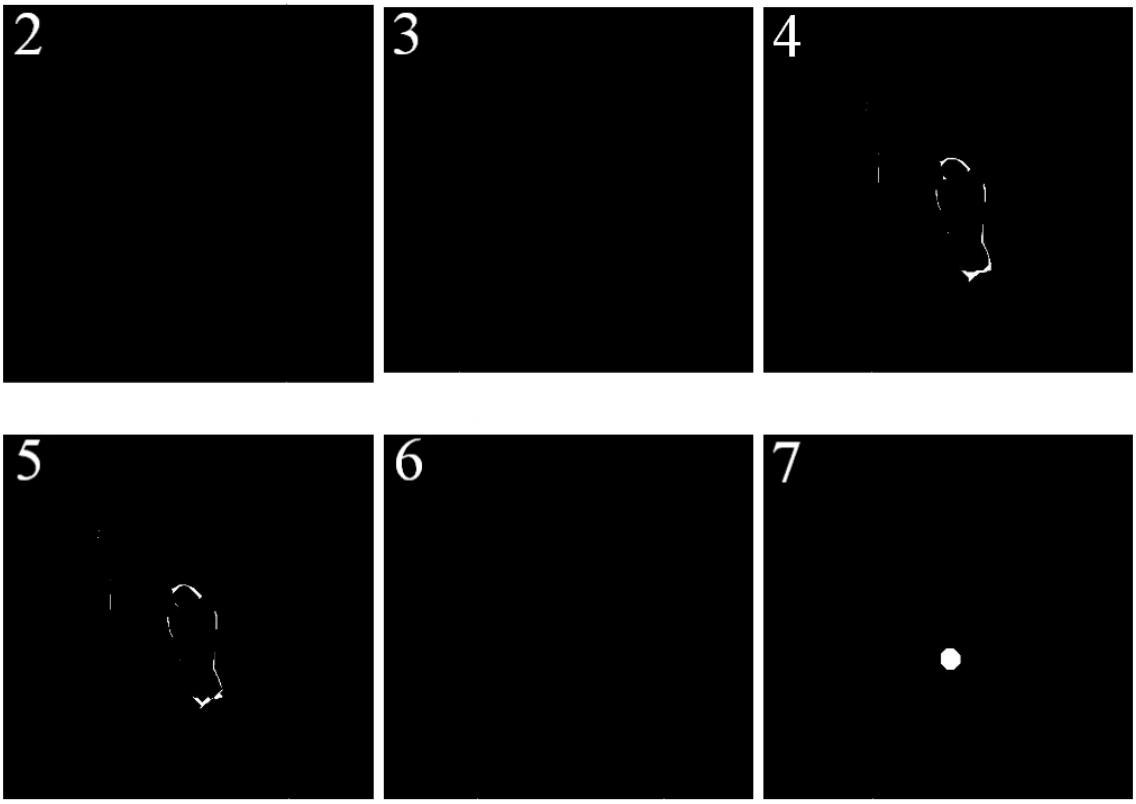

Figure 4: Background Subtraction Results for Morphology Operator for Sample Video Frame

The next table shows the results for the Absolute ADI, Positive ADI and Negative ADI with various k values. K represents the number of frames accumulated to form the reference image. Therefore the kth frame is then compared with the reference. The Absolut ADI seemed to in general have the best results. As the k value decreased, the accuracy seemed to increase. Therefore the Absolute ADI with a k value of 10 had the best results.

Table 2: ADI Testing Results

k	10	15	20	25	30	35	40
Absolute ADI	100%	100%	99.57%	99.50%	99.67%	100%	98.82%
Positive ADI	100%	100%	99.39%	99.15%	99.55%	100%	99.05%
Negative ADI	96.29%	97.59%	96.53%	100%	98.32%	95.54%	95.27%

The table below shows the run times of the ADI testing. Overall the run times were very similar to the background subtraction run times shown in Table 1. The negative ADI consistently had lower run times. This could be because the equation for calculating the ADI is somehow computationally faster than the other ADI equations.

Table 3: ADI Testing Run Times in Seconds

k	10	15	20	25	30	35	40
Absolute ADI	0.1592	0.1484	0.1438	0.1288	0.1134	0.1144	0.1140
Positive ADI	0.1708	0.1720	0.1196	0.1450	0.1610	0.1360	0.1672
Negative ADI	0.0684	0.0743	0.0737	0.0710	0.0737	0.0808	0.782

The errors in accuracy were caused when multiple birds were in the scene. For example if two birds were moving closely to each other, they were detected as one object moving as shown in Figure 2 and 3 below. Overall this error would not be too bad if developing a smart scarecrow system. Since at least one bird would be detected, the system would still alert that there is a threat.

The figure below shows an example of the ADI at the same frame location all with the same k value of ten. The upper-left corner shows the video frame. The upper-right corner shows the absolute ADI. The lower-left corner shows the negative ADI. The lower-right corner show the negative ADI. The absolute ADI is a combination of both the positive ADI and negative ADI which is why it mostly likely had a higher success rate.

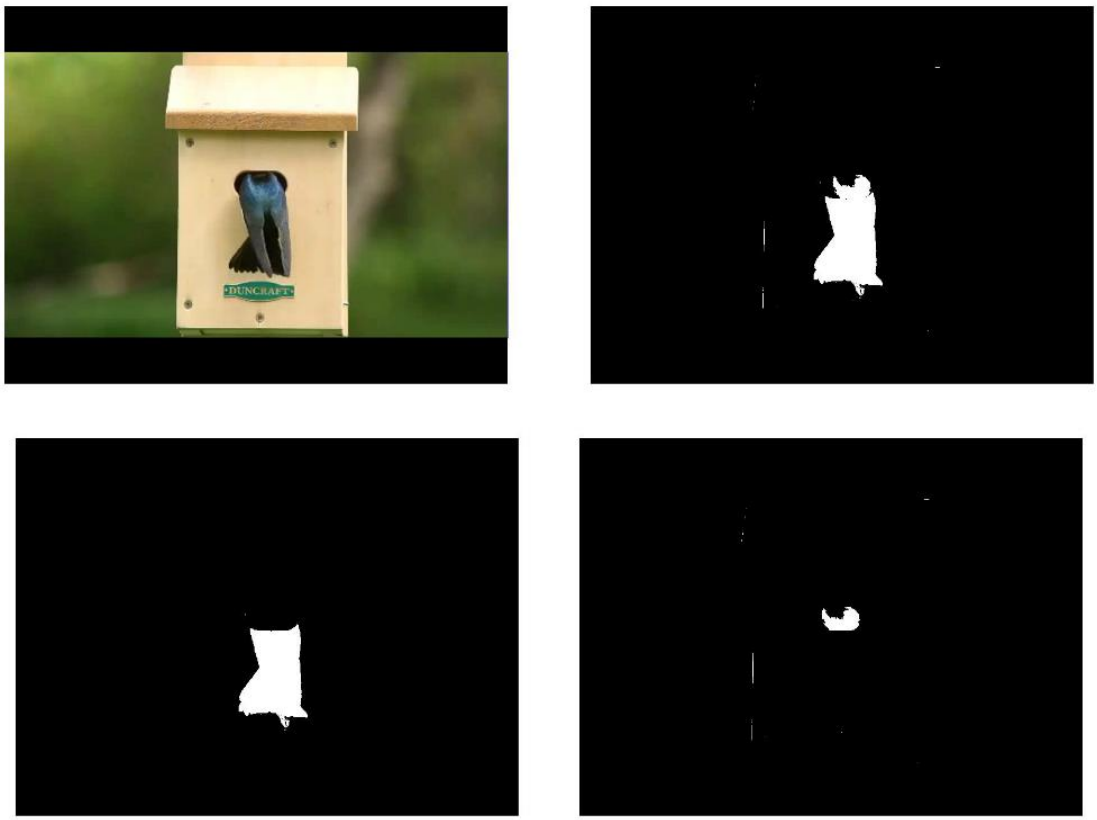

Figure 5: ADI Implementation Comparisons

The figure below shows one example of that the absolute ADI looks like for varying k values. These ADIs are from the same video frame as shown in the previous figure. The upper-left image is with a k value of 10. The upper-right image is with a k value of 20. The lower-left image is with a k value 30. The lower-right image has a k value of 40. The two lower images are starting to pick up more pixels from the background. Lower k value are less likely to pick up minimal background noised.

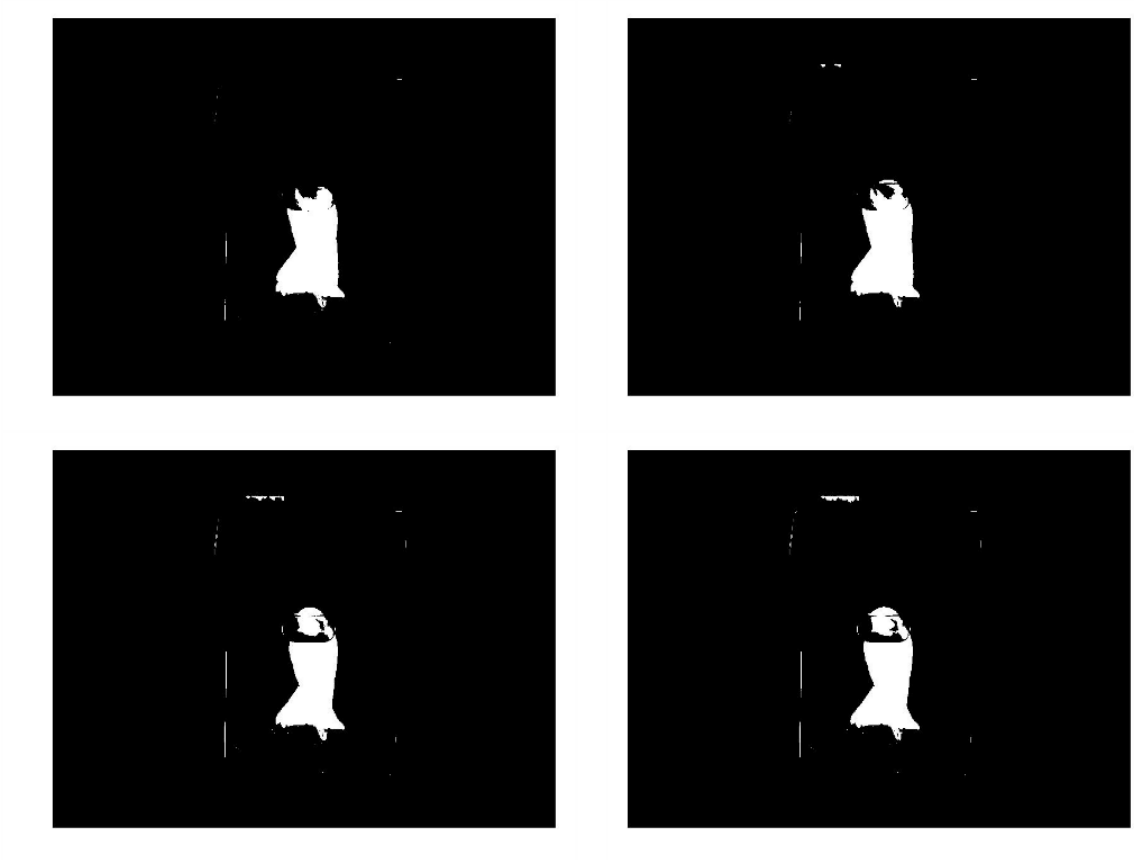

Figure 6: Absolute ADI with Varying K Values

Figure 7: Binary image with Two Birds Blending

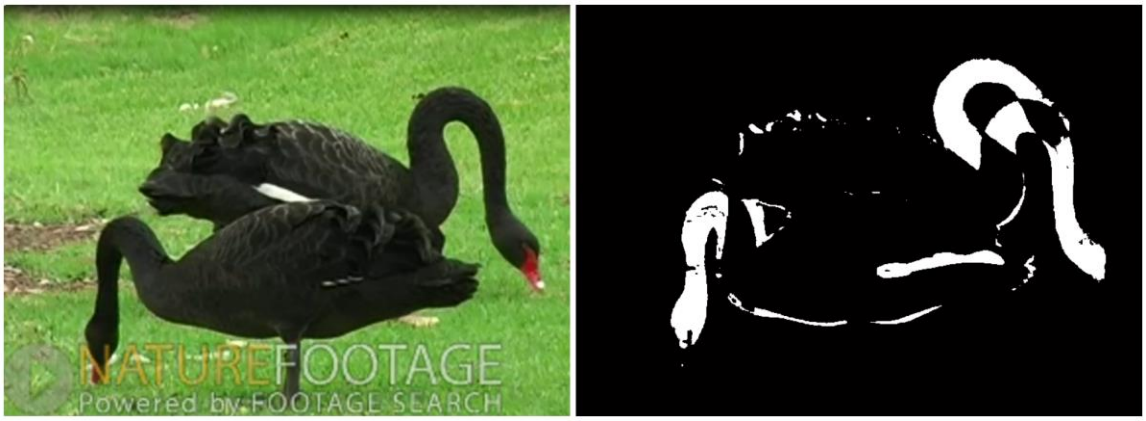

Figure 8: Binary image with Two Birds Blending

Although motion detection gives really high accuracy results, motion detection needs to be supplemented with another image processing technique (such as template matching and bird detection using the Viola-Jones algorithm) because anything moving in the scene will be detected as a bird. This is why the next chapters go into object detection tailored to birds specifically.

3.0

Template Matching

3.1. Introduction

Object recognition and object detection has had an increased importance with many fields such as biometrics, robotics, and other image processing applications. One of the oldest methods of object recognition is template matching. Template matching consists of sliding the template over the search area (usually an image in which we are trying to locate an object in) and, at each position, calculating a "distortion" or "correlation" measure that estimates the degree of dissimilarity or similarity, between the template and the candidate. Then the minimum distortion or maximum correlation position (depending on the implementation) is taken to represent the instance of the template into the image under examination. There are various ways of calculating the degree of dissimilarity or similarity like Sum of Absolute Differences (SAD) and the Sum of Squared Differences (SSD). The Normalized Cross Correlation (NCC) is by far one of the most widely used correlation measures [17, 18]. In the paper [19] they compared various image processing techniques for bird recognition. One of the techniques was template matching and they were able to in some cases get high accuracies.

With Normalized Cross Correlation the template sub image is located into the image under examination by searching for the maximum of the NCC function:

$$\mathrm{NCC}(x, y) = \frac{\sum_{j=1}^{N} \sum_{i=1}^{M} I(x+i,y+j) \cdot T(i,j)}{\sqrt{\sum_{j=1}^{N} \sum_{i=1}^{M} I(x+i,y+j)^2} \cdot \sqrt{\sum_{j=1}^{N} \sum_{i=1}^{M} T(i,j)^2}} \quad (5)$$

where x and y are the coordinates, i and j are integers, I and T denote the image and the template array matrix, and the original image is of dimensions MxN. Since there might

be multiple matches between the template and the image, the points with a normalized cross correlation above a certain threshold are considered matches.

Another way the NCC can be calculated in a way more tailored to the image that is discussed in [20] and is as follows:

$$\rho(x,y) = \frac{\varphi(x,y)}{\sigma_1 \sigma_T} \tag{6}$$

where the numerator $\varphi(x, y)$ represents the cross correlation between the template and the current sub-window of position (x, y) in the input image. It can be calculated as following:

$$\varphi(x,y) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[I(x+i, y+j) - \bar{I}][T(i,j) - \bar{T}] \tag{7}$$

And the terms σ_1 and σ_T represent standard deviation of the current sub-window of the input image and the template.

$$\sigma_1 = \sum_{j=1}^{n}\sum_{i=1}^{n}\left[I(x+i, y+j) - \overline{I(x,y)}\right]^2 \tag{8}$$

$$\sigma_T = \sum_{j=1}^{N}\sum_{i=1}^{N}[T(i,j) - \bar{T}]^2 \tag{9}$$

where $\overline{I(x,y)}$ and $\overline{T}$ represent the mean of the current sub-window of the input image and the template as follows:

$$\overline{I(x,y)} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} I(x + i, y + j) \tag{10}$$

$$\overline{T} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} T(i,j) \tag{11}$$

The standard deviation and mean of the current sub-window allow the NCC to be less affected by the lighting conditions in the image. The value of the NCC is between -1 and 1.

While template matching is typically carried out in the spatial domain, it was found in [21-23] that template matching calculations are much faster in the Fourier Domain. [20, 24] show how the NCC specifically could be calculated much quicker and more robustly in the Fourier domain.

Below are the main steps in implementing the algorithm for each image that needs to be analyzed.

I. Load the original image and template.

II. The next steps are then done independently on each color space (the first one run

for red, the second run for green, and –the third run for blue).

a. Calculate the Fast Fourier Transform (FFT) of both the image and template
b. Pad the image on all the sides with zeroes so that the center of the template falls on the very first pixel of the main image when kept on the top-left corner
    i. Calculate the size of the template.
    ii. Pad rows of zeroes on the top and bottom of main image. The number of rows is equal to the size of template in y-direction divided by 2.
c. Now, move the mask over the entire image and simultaneously multiply both the padded image and the template and store it in an array.
d. Normalize NCC values such that they lie in the range from 0 (template and image are not similar at all) to 1 (the template is exactly the same as the current area of the template) using mean compensation.
e. Unpad the array where the NCC values are being stored.

III. Combine R, G, and B pieces by averaging the NCC results.

IV. Find the positions where the value is above the selected threshold. These coordinates will be the locations where potential desired objects, similar to the template, are located.

V. Combine coordinate points that are clustered near each other

VI. Repeat previous steps with scaled (resized) template.

Although template matching can potentially give better accuracy results if more templates are added, template matching is not a reasonable solution for real time processing. Many templates have to be passed though the image which would make the processing time on one image too long for the results to be relevant to the current scene. For example, if the images are coming from a live video stream, by the time the template matching solution would determine whether there is a bird in the image or not, the birds could have left or entered the scene. Ideally the template is also scaled and rotated about each coordinate point in the image making template matching highly computationally intensive. In this template matching implementation, the templates are scaled to various sizes but not rotated about different degrees at each coordinate point.

3.2. Experimental Setup

After looking at various template matching implementations, Dr. Kroon's fast robust template matching [25] was selected. He implemented a template matching algorithm that performed the normalized cross correlation in the Fast Fourier Transform. Dr. Kroon's implementation is also considered fast and robust because the implementation allows for the storage of temporary variables from the image and faster search for multiple templates in the same image. The implementation also padded the image with zeroes to allow for normalized cross correlation to be calculated on the edge pixels. After the normalized cross correlation values are calculated, the results are then "unpadded".

In our implementation, two templates in the RGB color space are used. One template was an average image of all birds facing right (35 images) and the other

template was an average image of all birds facing left (24 images). The images were averaged by summing all of the images of the same bird orientation and then dividing by the number of images. The images were resized to be the same length and width. The lengths/widths chosen were 480, 240, and 120 (and 60 was also tested with). The two templates are shown below in Figure 1. Some sample images used to make the templates can be seen below in Figure 2 and 3.

Figure 9: Left Oriented Template

Figure 10: Right Oriented Template

As seen in the previous figures, it is hard to see the pixel value spread. The two figures below show the histogram equalized versions of the templates and the histogram of where the red, green, and blue values lie. Most pixels are clustered around the color dark brown (red: 65, green: 65, blue: 65).

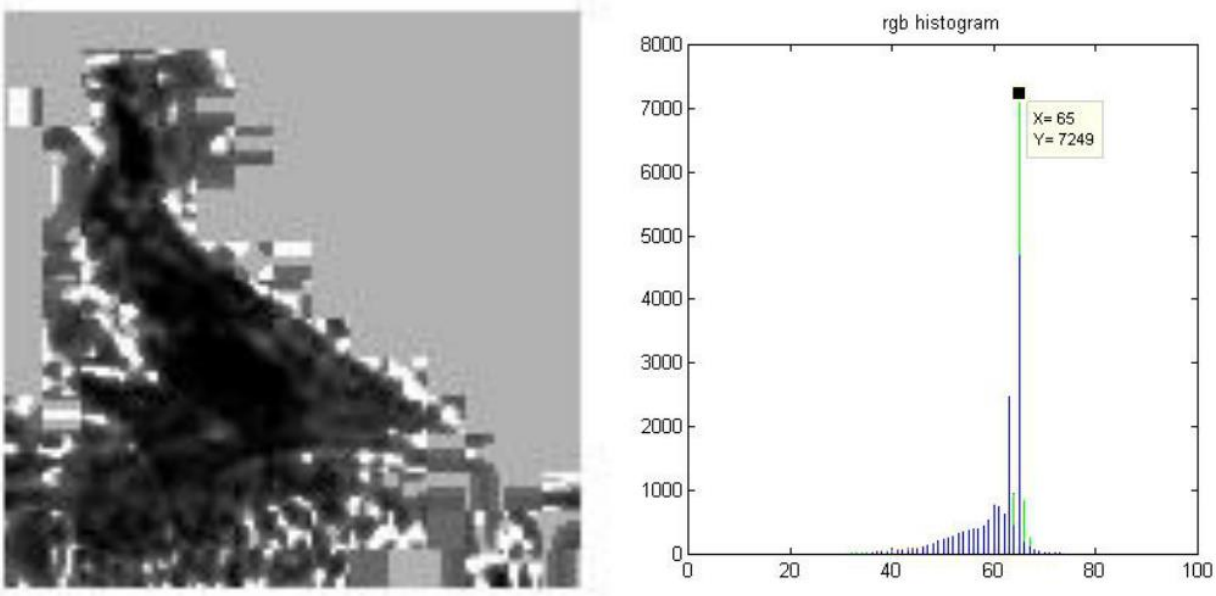

Figure 11: Equalized Image and Histogram of Left Template

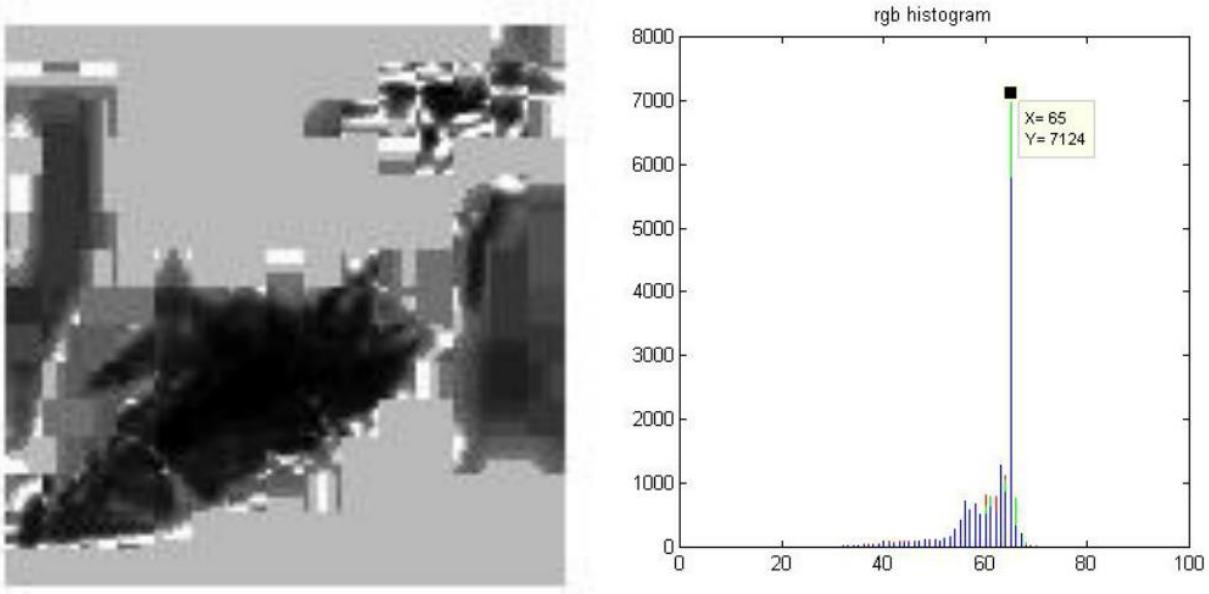

Figure 12: Equalized Image and Histogram of Right Template

After looking at the sample images below, most of the birds (for this specific project) are dark brown so it makes sense that this is where most of the pixel values in the

templates are in the dark brown region. Because various kinds of backgrounds were used, the backgrounds seem to cancel each other out in the templates. Although the templates could have been constructed with images of simple uniform backgrounds, this would potentially affect the accuracy because the majority of the bird images do not have simple uniform backgrounds. The majority of the images are of birds in an outdoor environment.

Figure 13: Sample Images Used to Make Right Template

Figure 14 Sample Images Used to Make Left Template

The templates were resized and the implementation was tested using 3 different sizes of the template and then 4 different sizes of the templates to see if this would improve the accuracy. The four image sizes were 60x60, 120x120, 240x240, and 480x480. The images were resized using the imresize function available in Matlab. The testing with only 3 different sizes excluded the 60x60 templates. 188 images containing various numbers of birds were used to test and these images were resized to be 640x480 because this is the size the camera uses that will be eventually used for the project switching over to real time data. An area in the image is considered a bird if the normalized cross correlation is above the specified threshold. The threshold can range from zero to one with one meaning the template and the specified image area are exactly the same and zero that they are total opposites. The specific threshold values tested were

0.7, 0.725, 0.75 and 0.8. These numbers were selected after doing quick testing with this threshold value on a dozen images.

Whenever a new area in the image passed the threshold, its normalized cross correlation value was compared with other areas considered nearby and the area with the lower normalized cross correlation value would be discarded. Two areas are merged if the center points of these areas' distance was less than or equal to the new area's radius. The radius would be the length/width of the current template being used. Figure 4 shows two images on the left that are unaltered (unpruned) showing all potential birds found. The red dot is the center pixel and the blue square is supposed to roughly enclose the area around the "found bird". The images on the right show the "pruned" results in which neighboring points are combined into one point.

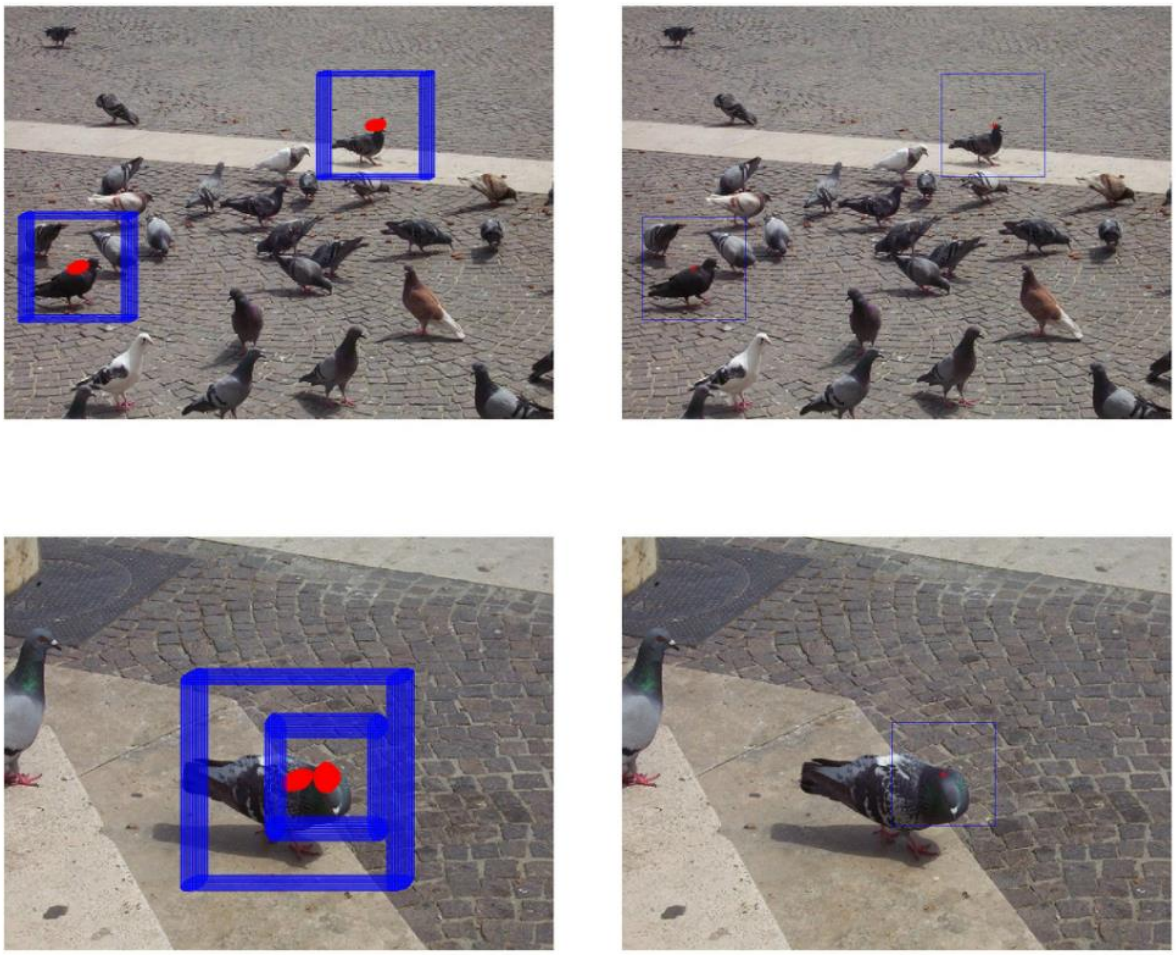

Figure 15: Example of Unpruned Image and Pruned Image

Most of the images used for template matching and the Viola-Jones Algorithm are from the Caltech-UCSD Birds 200 (CUB-200), [26], an image dataset and the University of California Berkeley's CalPhotos (an image database), [27].

3.3. Results

From the 188 images used to test both the threshold and number of template sizes for each sizes, the accuracy and false positives were recorded. The Accuracy was

calculated by summing the total number of birds the algorithm was able to find and then dividing by the total number of actual birds in these 188 images. The false positive value recorded in the table was the average amount of false positives found in the images. The false negatives were not added in the table because the false negative percentage is just 100% minus the accuracy percentage. A high percentage in accuracy means a low percentage in false positives and vice versa.

Table 4: Accuracy and False Positives Results

Threshold	Accuracy w/4 template sizes	False positives w/4 template sizes	Accuracy w/3 template sizes	False Positives w/3 template sizes
0.7	0.59	4.65	0.58	2.46
0.725	0.54	3.62	0.43	0.9
0.75	0.46	2.26	0.35	0.38
0.8	0.11	0.37	0.03	0.02

As shown in the table below, in general, higher thresholds resulted in higher run times. For an algorithm to be able to keep up in run time, it definitely needs to keep the detection under a second like motion detection (and as we will later see bird detection using the Viola-Jones algorithm).

Table 5: Run Time for Template Matching with Varying Template Sizes

Threshold	Run Time in Seconds w/4 template sizes	Run Time in Seconds w/4 template sizes
0.7	979.297	367.89
0.725	224.06	44.87
0.75	44.21	8.64
0.8	6.32	5.49

As shown in Table 4, as the threshold increases, the accuracy decreases and the false positives per frame decrease as well. Adding an extra template size slightly increased both the accuracy and the number of false positives. Images below show the output of some of the successful and unsuccessful runs of the algorithm.

Figure 16: Images Showing Successful or Partially Successful Bird Recognition

Images Showing Unsuccessful or Partially Unsuccessful Bird Recognition

Figure 17: Images Showing Unsuccessful or Partially Unsuccessful Bird Recognition

On Figure 7 below, the upper left image was the result of using a threshold value 0.7. The upper right image was the result of using a threshold value 0.725. The lower left image was the result of using a threshold value 0.75. The lower right image was the result of using a threshold value 0.8. The Figure shows that as the threshold increases, the accuracy decrease and the number of false positives decrease.

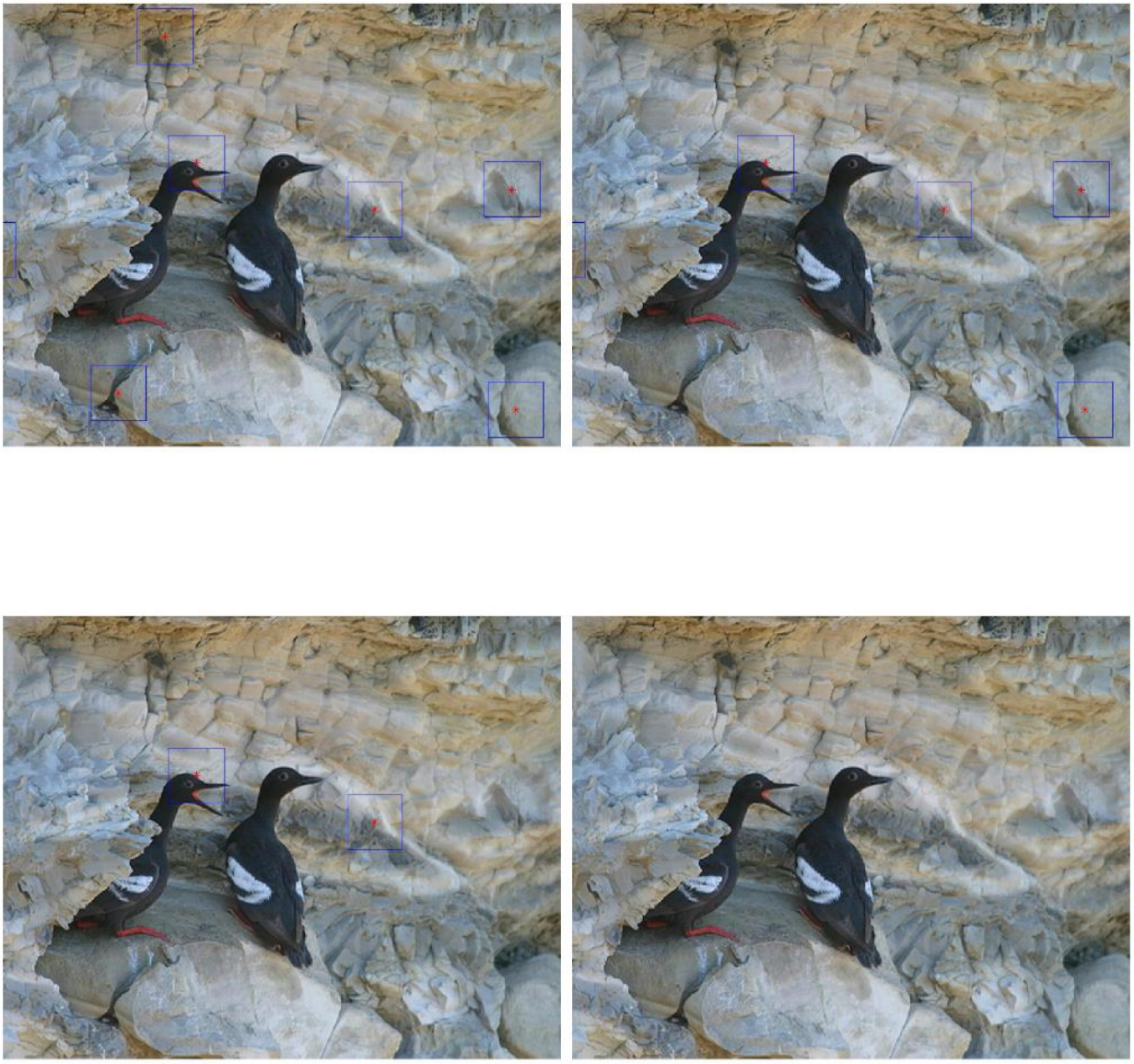

Figure 18: Various Thresholds on Same Image

As shown in Table 6 the accuracy and the number of false positives (FPs) increases as the number of bird images averaged together increases. The first column shows how many bird templates were averaged together that were oriented to the left and

the second column shows how many bird templates were averaged together that were oriented to the right.

Table 6: Template Averaging Results

Avg. Amount of Left	Avg. Amount of Right	Accuracy	Avg. Number of FPs
1	1	0.39	1.78
2	2	0.2	0.92
3	4	0.29	1.03
5	8	0.43	2.990
24	35	0.54	3.62

After analyzing where some of the false positives were located, some false positives would be near the edge of the image. This meant the way the image was padded for calculating normalized cross correlation at those edge points were causing false positives. The original method used to pad image was to pad with zeroes (which would look like a black frame around the image). Run times were not shown in Table 6 because changing what the template looks like has minimal change in the run time.

Other methods used to try to decrease the number of false positives was a circular border, replicate border, a symmetric border, and a padding of "255" (which would look like a white frame around the image). The white padding seemed to have the lowest numbers of false positives (FPS). This is most likely due to the fact that the templates are

on the darker side of the spectrum and therefore white padding makes it very unlikely for an edge pixel have a high cross correlation.

Table 7: Padding Techniques Results

Method	Number of FPs	Avg. Number of FPs
Original (0 Value Padded)	353	3.76
Padded in White (255 Value Padded)	345	3.67
Circular	348	3.7
Replicate	350	3.72
Symmetric	348	3.7

The resolution of both the templates and the images were decreased as further experimentation in trying to increase accuracy. The original resolution used for this project was 640x480. The resolution was reduced in half for both the templates and the images (the images therefore where 320x240). Another test was run with half the resolution of this previous resolution for both the templates and the images (the images where therefore 160x120). The table below shows the results. For all three cases the same threshold was used (0.725).). Run times were not shown in Table 7 because changing the padding technique has minimal change in the run time.

Table 8: Resolution Change Results

	Accuracy	Average Number of FPs	Run Time in Seconds
160x120	0.49	5.04	491.55
320x240	0.49	4.88	417.30
640x480	0.54	3.64	224.06

As shown in the table, the smaller resolutions had a lower accuracy than the higher resolution. The average number of false positives also increased, as the resolution decreased. Template matching therefore works best with the original (and highest) resolution.

3.4. Analysis and Future Work

With further testing the accuracy could be increased to a higher percentage by adding more templates however, each template added to the system also greatly slows down the system because of the extra computations needed for each additional template. Also it is important to note that in images with multiples birds, the algorithm was able to usually find a least one bird in the frame. For a smart scarecrow this would be enough because scaring off one bird has the same procedure as scaring off multiple birds so the accuracy can be lower than expected to still perform successfully. A lot of the false positives were caused by branches, water backgrounds, and some birds occlude other birds.

For this project high false positives is not as damaging as false negatives. It is

more fatal to not catch a bird attacking produce then the smart scarecrow indicating there is a bird when there isn't one. As the threshold increases, the false positives decrease but the accuracies decrease as well. Therefore a lower threshold is optimal for this project because the accuracy of the system is more important than how many false positives are detected.

More templates could be added which would increase the accuracy and might or might not increase the false positives. Possible templates that could be added to the collection are an average of birds that are facing straight (are centered) and templates with birds with their wings open. Further testing could lead to a more fine-tuned optimal threshold that would most likely increase the accuracy of the algorithm.

4.0

Bird Detection Using the Viola-Jones Algorithm

4.1. Introduction

After template matching, more research was done on more modern face detection techniques that could possibly be applied to bird detection. Template matching works best with many templates and this is not optimal for a real time solution. A popular face recognition algorithm that has competitive detection rates in real-time, is the Viola-Jones algorithm [32], The Viola-Jones algorithm has shown to be successful especially in face detection. Although the training for the object classifier is slow, the actual detecting is fast which is why there have been some web browser implementations like [33] and mobile implementations like [34]. The Viola-Jones algorithm is also traditionally used for finding human body parts in a scene [35-37] and not just faces. The Viola-Jones algorithm can be trained for almost any object as long as there is enough similar positive images that can be used for training the classifier.

The classifier learns (via machine learning) from labeled data. Images with the desired object labeled are called positive data. Ideally the training data should be large (thousands of images). The labeled data is encouraged to have many variations across the desired object to detect such as variations in the pose of the object, the illumination in the image, etc. Images not containing the desired object are also needed and are considered negative images.

Then the AdaBoost algorithm is used. A set of weak classifiers are made and then iteratively combines classifiers. The final classifier is a linear combination of weak classifiers. Image features are used by AdaBoost that are “rectangle filters” similar to

Haar wavelets. The integral image is also used by AdaBoost. The integral image computes a value at each pixel (x, y) that is the sum of the pixel values above and to the left of (x, y), inclusive. This can quickly be computed in one pass through the image.

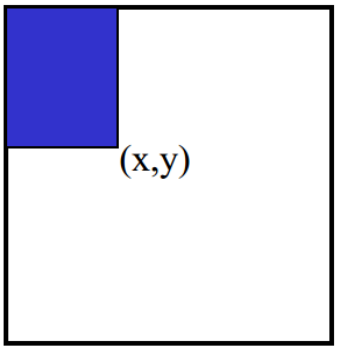

Figure 19: Integral Image Computation at x, y

These classifiers are then cascaded for even higher accuracies. Usually the higher cascade level (higher feature classifier) will have a lower false positive rate.

4.2. Experimental Setup

Matlab r2013a introduced Viola-Jones object detection training functions in its Computer Vision System Toolbox. The Cascade Training GUI made by Shoelson [38] was used to train the cascade detector. This project helps you set up all the parameters that go into the trainCascadeObjectDetector function. This function is used to train a cascade object detector model. The first step of the Viola-Jones algorithm, is to gather positive images and label the desired object (in this project's case, birds) as the region of interest (ROI) as shown in the figure below.

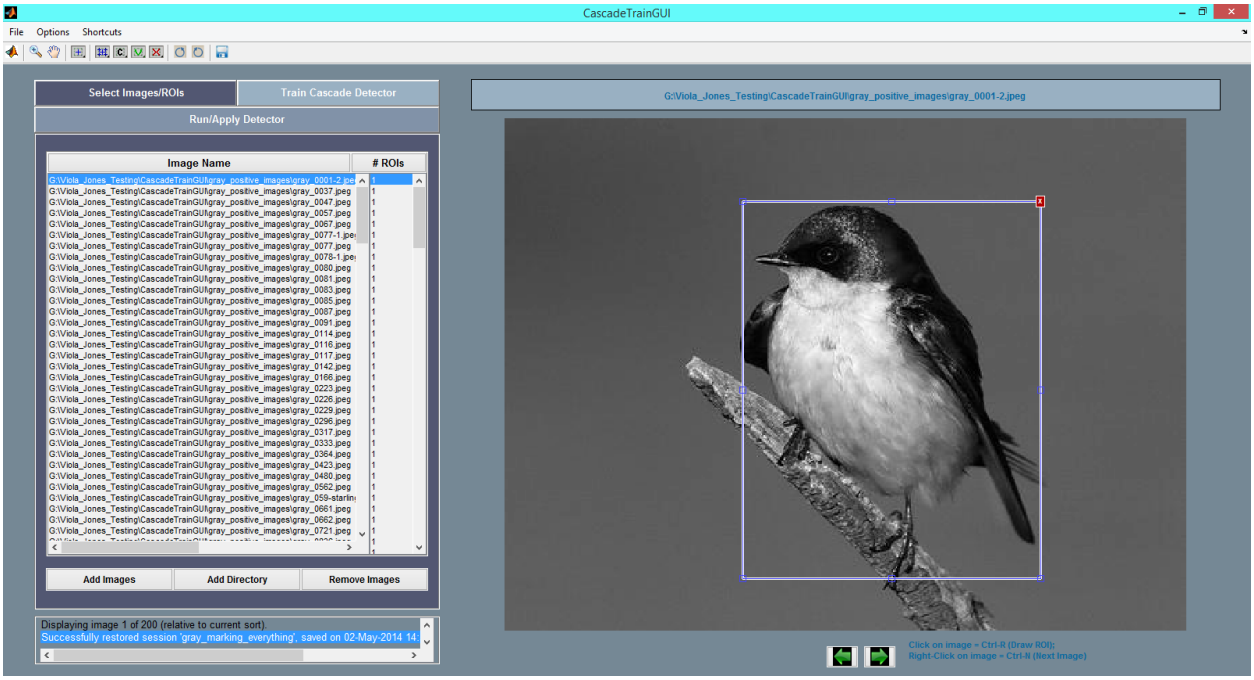

Figure 20: Selecting Images/ROIs

After all of the positive images have been labeled ROIs, a folder with negative images is made. Images of trees, branches, and other animals such as rabbits and deer were used as negative images. It is important for the detector to know the difference between for example of an image of a branch and an image of a bird on a branch. All of these negative images were put into the same folder. Below are sample images of the positive images used for this experiment.

Figure 21: Horizontal Positive Samples

Figure 22: Vertical Positive Samples

Below are examples of samples used in the negative folder (non-bird images).

Figure 23: Horizontal Negative Samples

Figure 24: Vertical Negative Samples

There is an option to add the positive images in the negative images pool with the ROIs deleted. This is the first step needed for running the filter as shown in in the figure below. The next figure is zoomed in on step two which is to select parameters for the training detector.

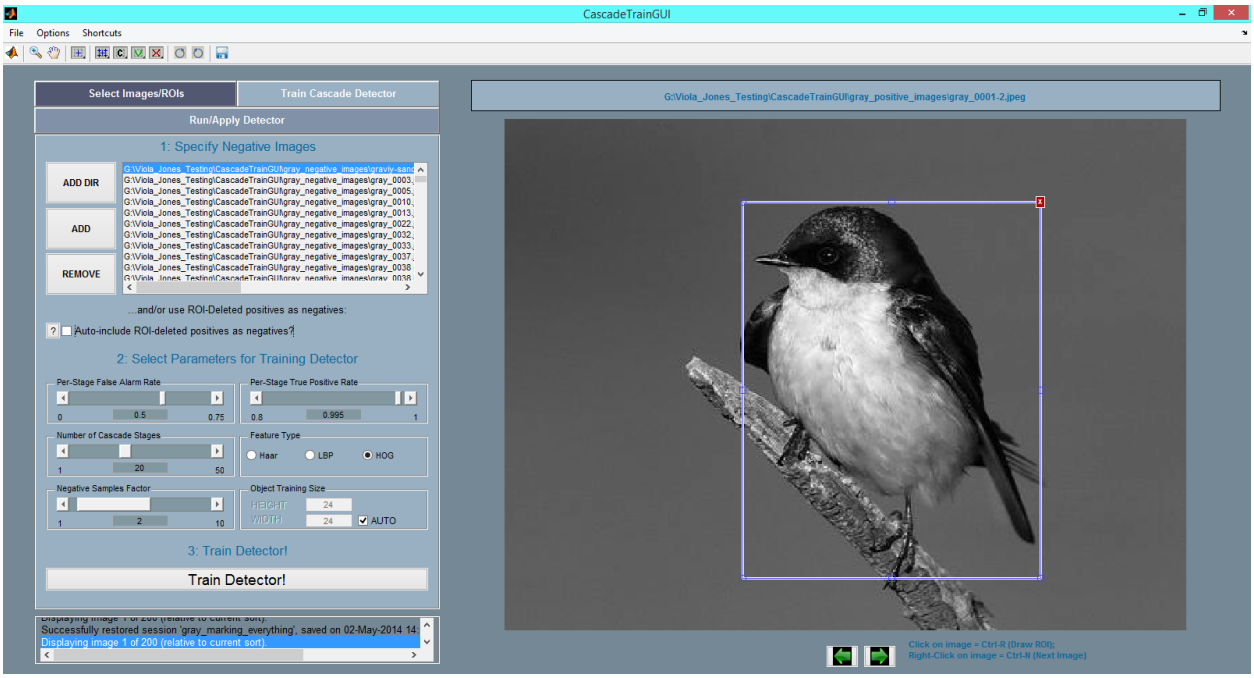

Figure 25: Training a Cascade Detector

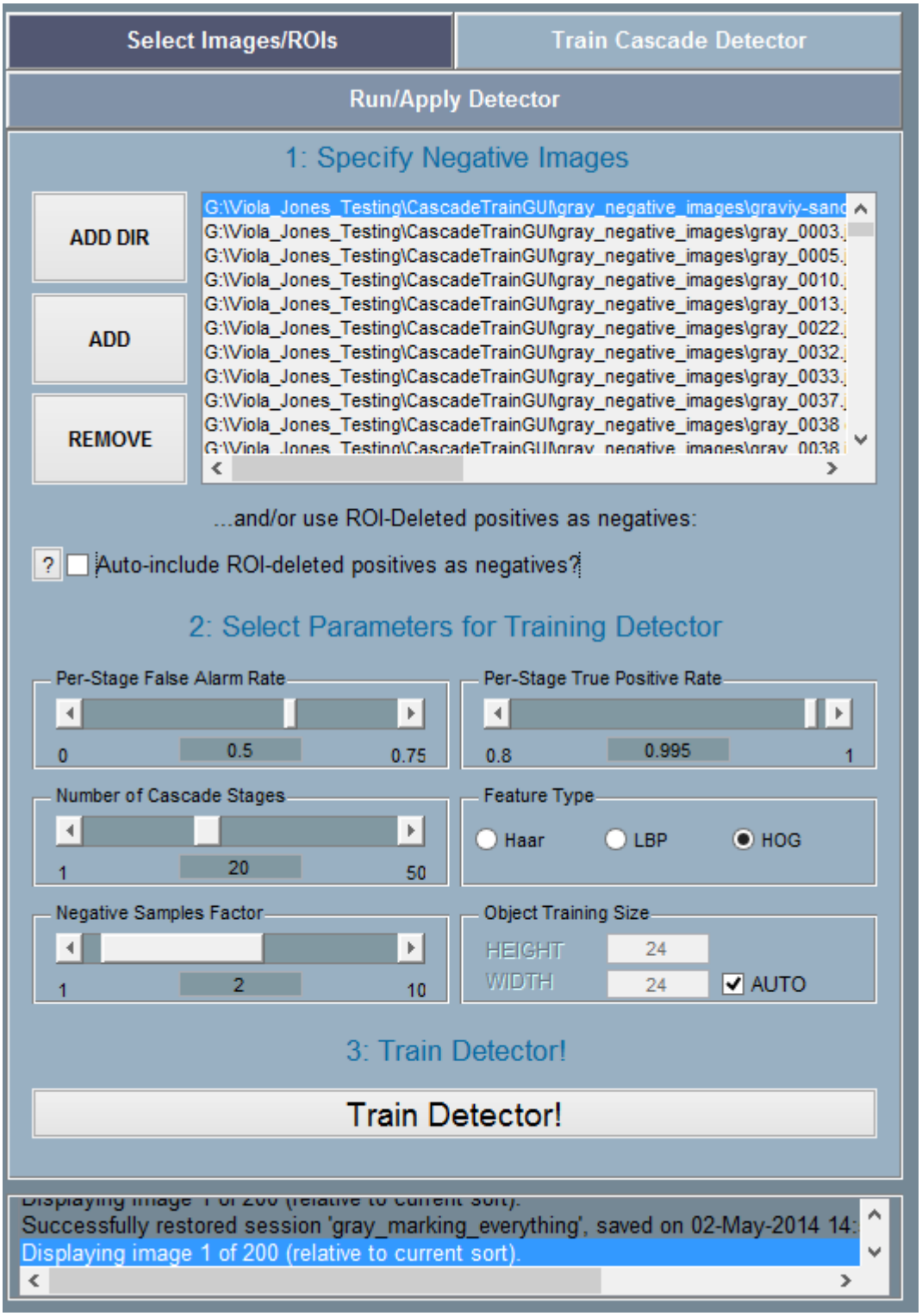

Figure 26: Variables in Viola-Jones Algorithm

All the parameter settings shown in the figure above are the default settings. The first parameter seen in the top figure is the per-stage false alarm rate, which is the acceptable false alarm rate at each stage. The false alarm rate is the fraction of negative training samples incorrectly classified as positive samples. The overall false alarm rate is calculated using the false alarm rate per stage and the number of case stages. The lower this number is, the higher the complexity of each stage.

The next parameter is the per-stage true positive rate. Minimum true positive rate required at each stage. The overall resulting target positive rate is calculated using the true positive rate per stage and the number of cascade stages. The higher the value this variable has, the higher the complexity of each stage. As [39] explains this increased complexity could result in longer training and detection times.

The next parameter is the number of cascade stages. The false alarm rate and the true positive rate need to be adjusted accordingly when the number of cascade stages are changed. Increasing the number of stages may result in a more accurate detector but also increases training time. More stages may require more training images, because at each stage, some number of positive and negative samples may be eliminated. Increasing the number of stages also increases the false negative rate.

The next parameter is the feature type. The Viola-Jones algorithm Matlab version feature types are shown in table below.

Table 9: Feature Types

'Haar'	Haar-like features
'LBP'	Local Binary Patterns
'HOG'	Histogram of Oriented Gradients

As the Mathworks documentation explains in [39], historically, Haar and LBP features have been used for detecting faces. The HOG features have been used for detecting objects such as people (whole body instead of just a person's face) and cars. They are useful for capturing the overall shape of an object. The results will show whether bird detection will perform better with Haar, LBP, or HOG feature type. These feature types have been used in the past for birds as shown in [40, 41].

The next parameter is the negative samples factor. This is the number of negative sample to use at each stage. This number is calculated by multiplying the negative samples factor by the number of positive samples used at each stage. The object training size can also be adjusted to a certain size (by specifying the height and width) or the "auto" option then the algorithm will determine the size automatically based on the median width-to-height ratio of the positive instances.

One last parameter specific to this CascadeTrainGui is that there is an "auto-include ROI-deleted positives as negatives?" option. If selected this means that the positive images will also be used as negative images with the ROI/s specified area/s removed.

When the function trainCascadeObjectDetector is run, as shown in the figure below, the automatic training size, the maximum amount of positive samples per stage, and the maximum amount of negative samples per stage it outputted. The function then goes through the number of cascade stages set and outputting the amount of positive and negative samples it actually used. An xml file is outputted which is then used to detect birds into images.

```
Automatically setting ObjectTrainingSize to [ 32, 35 ]
Using at most 302 of 309 positive samples per stage
Using at most 604 negative samples per stage

Training stage 1 of 5
[..........................................................................]
Used 302 positive and 604 negative samples

Training stage 2 of 5
[..........................................................................]
Used 302 positive and 604 negative samples

Training stage 3 of 5
[..........................................................................]
Used 302 positive and 604 negative samples

Training stage 4 of 5
[..........................................................................]
Used 302 positive and 604 negative samples

Training stage 5 of 5
[..........................................................................]
Used 302 positive and 604 negative samples

Training complete
```

Figure 27: Output of trainCascadeObjectDetector Function

The figure below shows how to use the xml file from training and how to detect birds in the image. First a bird detector object is made from the xml file, the image to be tested is read, and then birds are detected. The last step allows you to visually the bird being annotated as the figure below the code.

```
% Create a detector object.
birdDetector = vision.CascadeObjectDetector('bird_detector.xml');

%Read input image.
I = imread('bird1.jpeg');

%Detect bird.
bboxes = step(birdDetector, I);

%Annotate detected bird/s.
index = 1;
IBirds = insertObjectAnnotation(I, 'rectangle', bboxes, 'bird');
figure(index), imshow(IBirds), title('Detected birds');
```

Figure 28: Matlab Code on How to Detect Bird

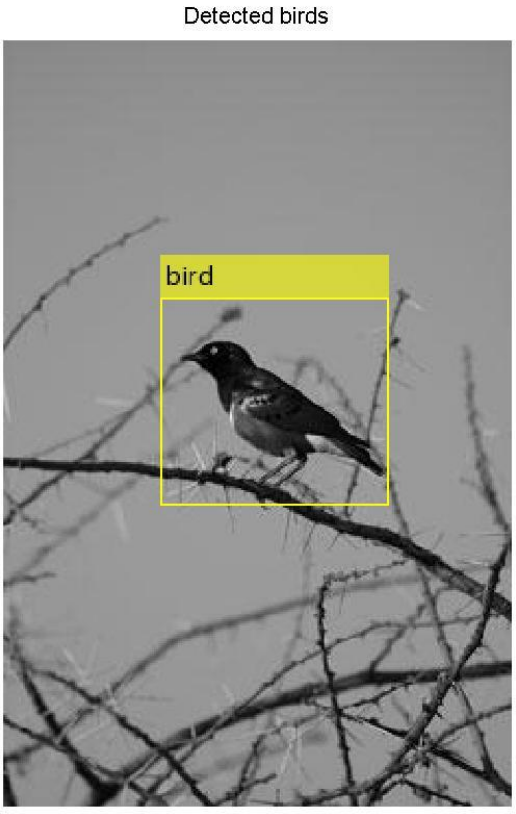

Figure 29: Detected Bird Example

As seen in the figure above, all of the images were converted to grayscale. The reason the bird detector was trained in grayscale was so that it would not be overly sensitive to variations in bird color. 860 images were used as negative images and 208 images were used as positive images. 98 bird images were used as test images. The test images are a different set of images used from the training stage.

Only the images in the positive folder (contained the positive images) were used for training. The majority of the test images and positive images came from the same two databases used in template matching (UC Berkeley's CalPhotos and the Caltech-UCSD Birds 200). Negative images were acquired from the following image databases [28-31] and Google search. The reason there was so many negative images compared to positive images was because there are more image databases of nature scenes without birds than without birds. One potential task for future work would be to try to find more bird image databases such that the positives images and/or test images pile could be larger. Another reason for such a high amount of negative images is to ensure that the cascaded training stages don't get truncated prematurely. In early stages of testing, when only about 200-400 images were used, these amounts of negative images were not enough for certain parameter changes. In one run for example, a 25 cascaded stage detector was truncated to 19 cascaded stage detector.

4.3 Results

The first parameter that was experimented was the feature type because it is the parameter that is the most diverse. The rest of the parameters have a number range

whereas the feature type directly changes how computationally the bird detector is trained.

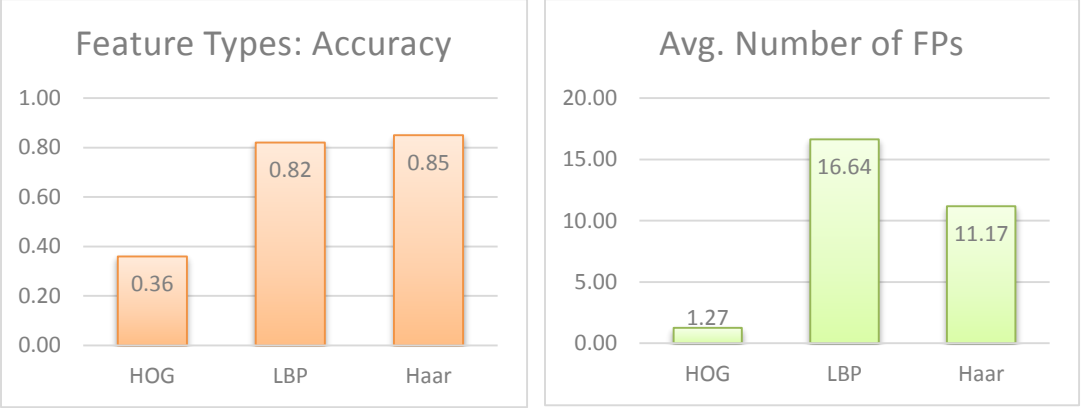

Figure 30: Feature Type Testing

As shown in the figure above. Haar and LBP were in the lower 80s as far as percentage in accuracy while HOG was about half of that. However, when looking at the average number of false positives (FPs) in the images, LBP had the highest average while Haar and HOG were somewhat low. The next parameter to be experimented with was the number of cascade stages. Because LBP and Haar had such better accuracies than HOG, HOG was eliminated as a good match for bird detection.

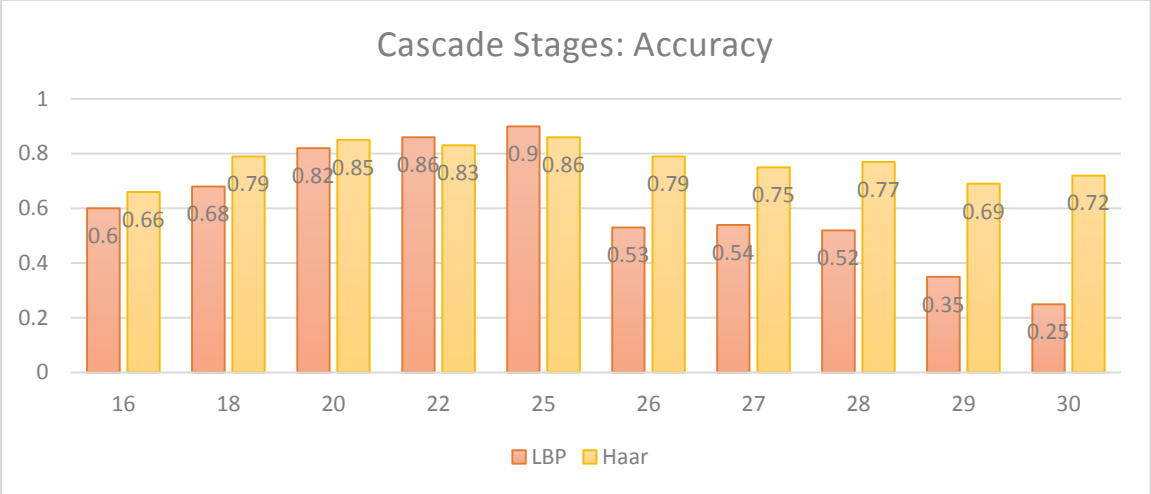

Figure 31: Chart Showing Accuracies as Cascade Stage Number Changes

In general Haar had a higher accuracy regardless of the amount of cascade stages (25 cascade stages was the exception). Twenty-five cascade stages resulted in the highest accuracy for both LBP and Haar. Although usually increase the number of cascade stages will increase the accuracy, it seems like after 25 cascade stages, the accuracy starts to decrease. This could be because after that point, the accuracy becomes so rigorous that many birds are overlooked that are slightly different from the birds in the training data.

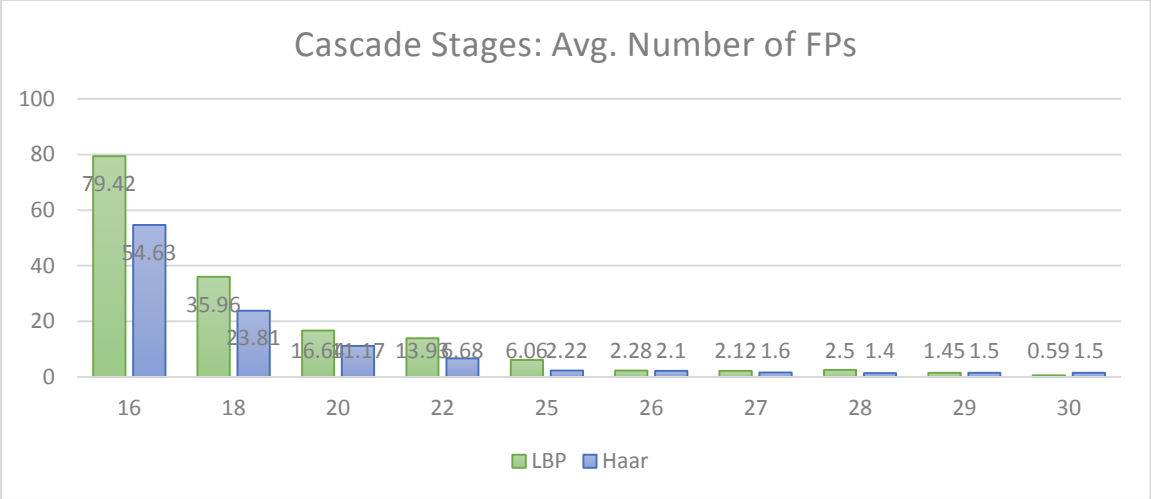

Figure 32: Chart Showing the Average Number of FPS as Cascade Stage Changes

As the number of cascade stages increased, the average number of FPs decreased. The more cascade stages the detector has, the more the average number of FPs decrease. This is because as the number of cascade stages gets increased, the bird detector becomes more complex and more rigorous because the false alarm rate (FAR) decrease as the number of cascade stages get increased. In further testing only Haar was used with a 25 cascade stages. For bird images, Haar almost consistently gave a better accuracy and a lower value of false positives compared to LBP. Because the amount of images used for testing, 25 cascade stages seemed to be the most optimal number. If a more extensive positive bird images database was found and added to this project, the optimal number of cascade stages would increase.

A low FAR number will result in less FPs. The FAR parameter was the next parameter that was changed in order to find the optimal setting. This optimal number seemed to be 0.5 therefore the rest of the testing used this value for the FAR parameter.

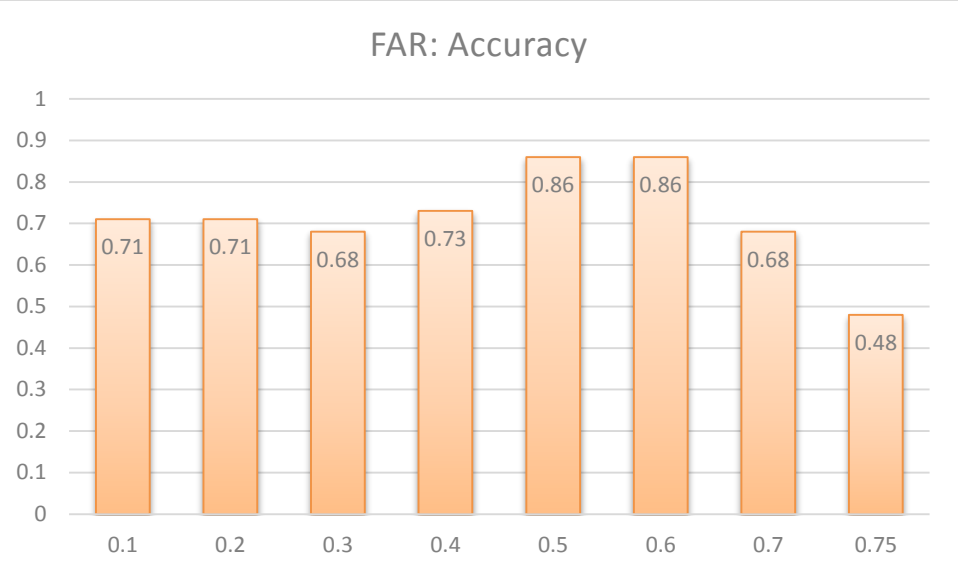

Figure 33: Chart Showing the Accuracy as Per-Stage False Alarm Rate (FAR) Changes

As expected, the figure below shows the increasing the false alarm rate, increases the number of false positives. The false alarm rate is the percentage of "allowable false positives" therefore it makes sense that increasing it would lead to a higher amount of false positives.

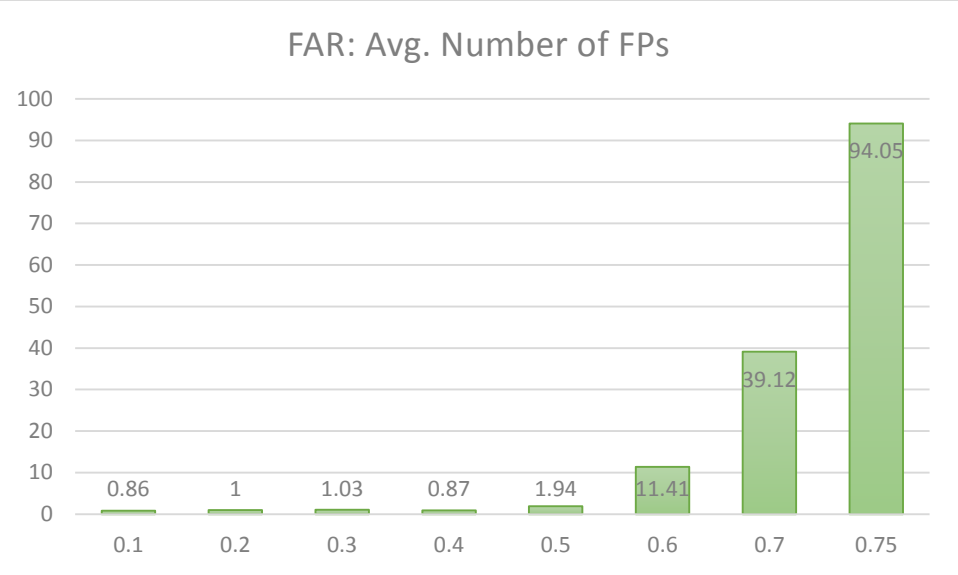

Figure 34: Chart Showing the Average Number of False Positives as Per-Stage False Alarm Rate (FAR) Changes

The next parameter that experimented with to find the optimal value was the per-stage true positive rate (TPR). As expected, for the most part as the TPR value was increased, the accuracy increased. However the optimal value seems to be 0.995 because after this value the accuracy starts to drop off. Therefore the rest of the testing is done with this TPR value.

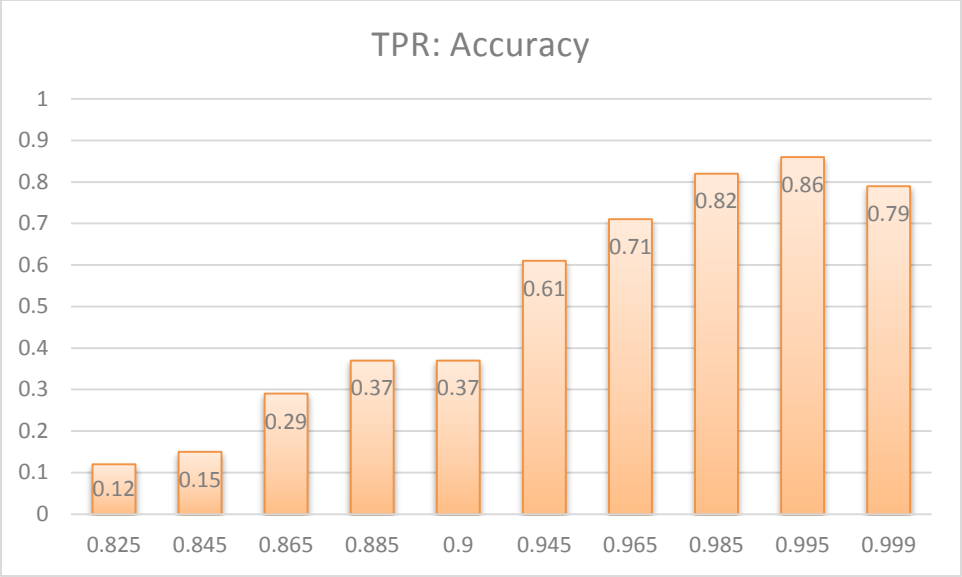

Figure 35: Chart Showing the Accuracy as Per-Stage True Positive Rate (TPR) Changes

The number of false positives increase as the true positive rate increases. This could be because as the bird detector tries to increase the accuracy, it finds as many birds as it can resulting in more false positives (non-bird areas detected as birds).

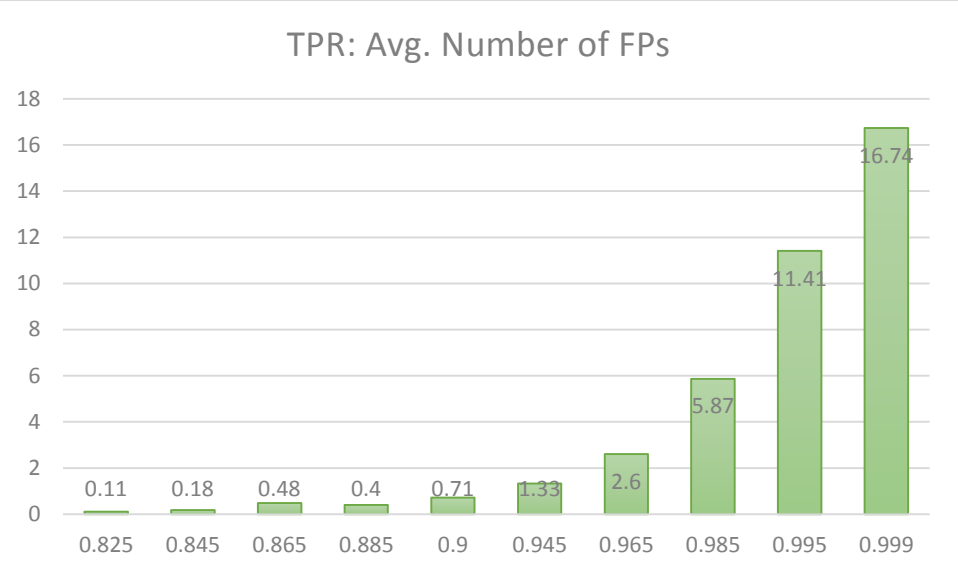

Figure 36: Chart Showing the Average Number of False Positives as Per-Stage True Positive Rate (TPR) Changes

The last parameter to be modified is the negative samples factor (NSF). Besides the feature type, this the only other parameter that is not dependent on other parameter settings. This is why this was the last parameter to be modified is the NSF. For the most part changing the NSF parameter only slightly affected both the accuracy and average number of FPs. The only exception being the value one (because this value had a considerably lower accuracy and a considerably higher average number of FPs compared to the other values). In further testing an NSF value of 7 was used since this gave the best accuracy.

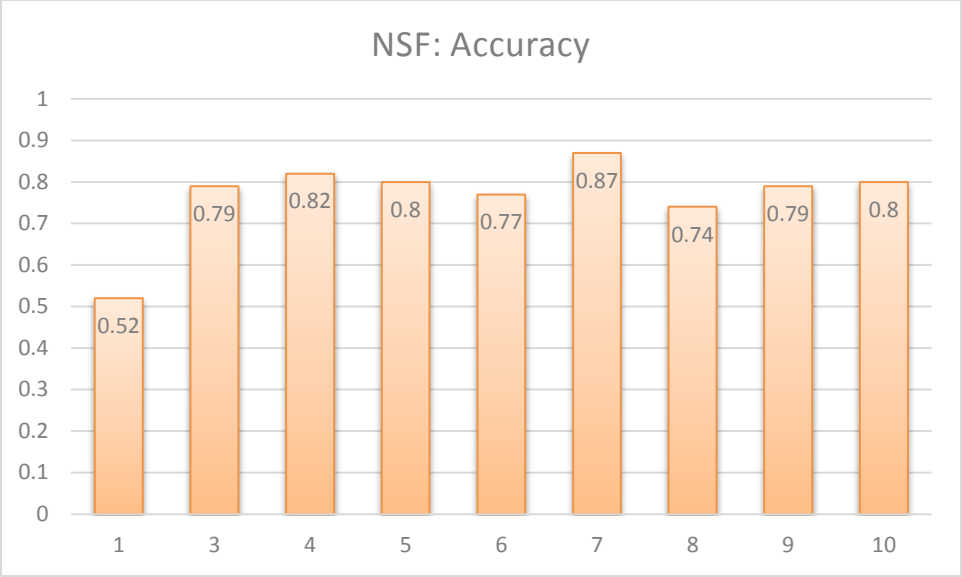

Figure 37: Chart Showing the Accuracy as the Negative Samples Factor (NSF) Changes

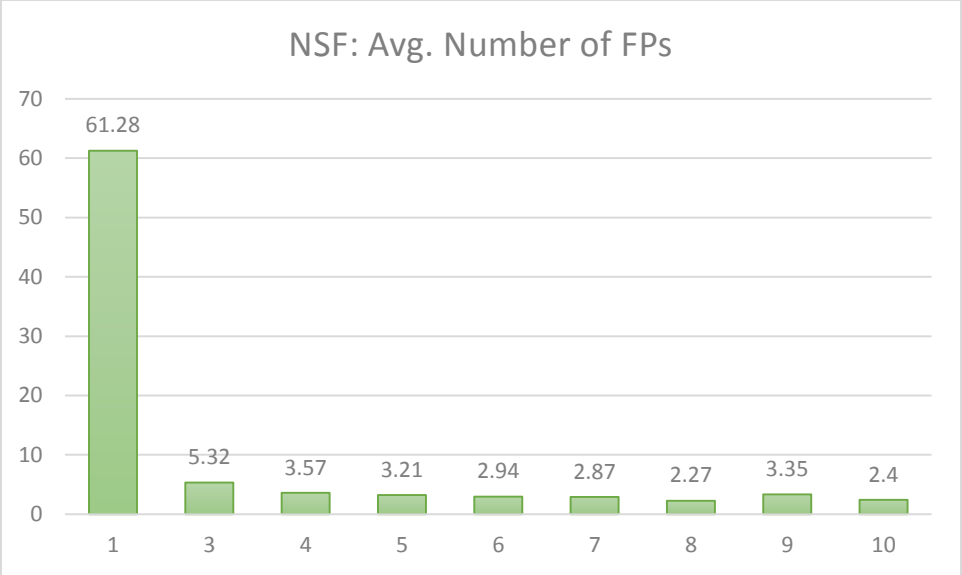

Figure 38: Chart Showing the Average Number of False Positives as the Negative Samples Factor (NSF) Changes

The figure below shows the accuracy difference of with and without the ROI-deleted positives as negatives. In theory the bird detector should have performed better with the ROI-deleted positives as negatives because those backgrounds should have

helped the detector better train what wasn't a bird. The results show however, the bird detector worked best when the ROI-deleted positives as negatives were not used as training data. The average amount of false positives was also lower for the case without the ROI-deleted positives as negatives.

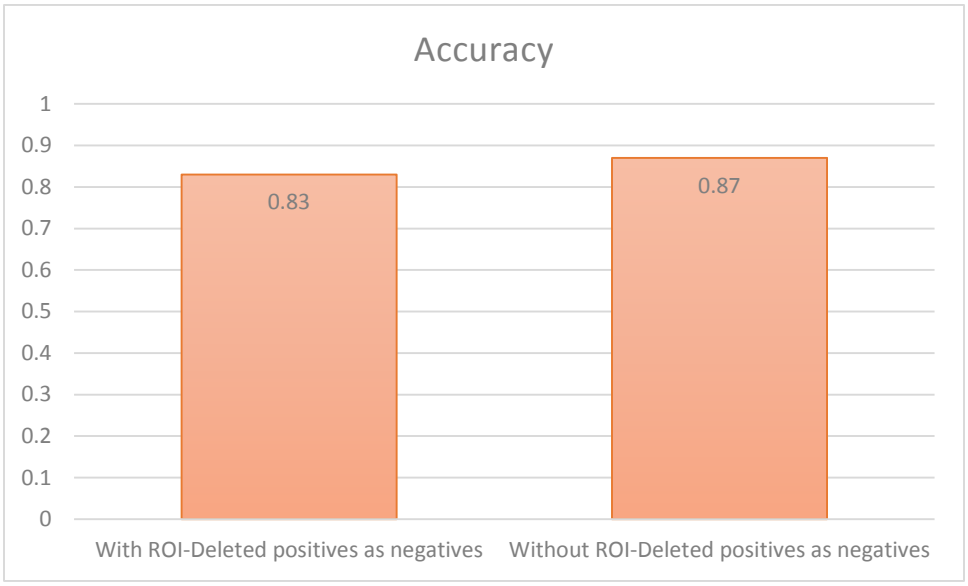

Figure 39: Chart Showing the Accuracy with and without ROI-Deleted Positives as Negatives

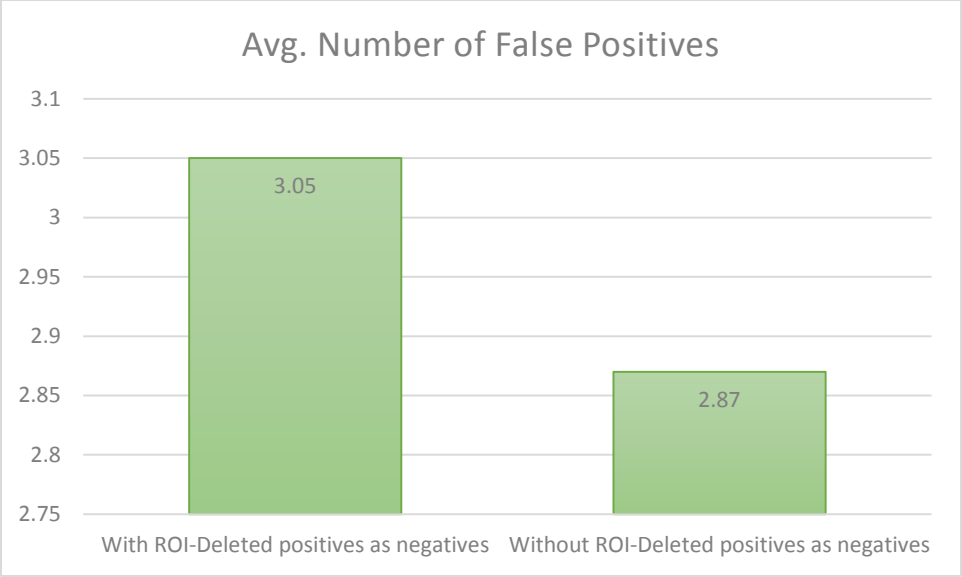

Figure 40: Chart Showing the Average Number of False Positives with and without ROI-Deleted Positives as Negatives

Once the parameters where modified, the next parameters that were modified to hopefully decrease the number of false positives without majorly affecting the accuracy of the bird detector. The Matlab function vision.CascadeObjectDetector has three parameters that could potentially help decrease the number of false positives which are MinSize, ScaleFactor, and MergeThreshold.

MinSize sets the minimum size a bird detection can be. The [height width] must be greater than or equal to [32 35] because this is the object size used to train the classification model. Various lengths/widths were tested however none had a better accuracy than the default value which was to not have a minimum size restriction on detected objects (birds). Because the bird images have birds of various sizes, it is hard to set a minimum size a bird can be without affecting the accuracy. Therefore, it is better to leave the default setting.

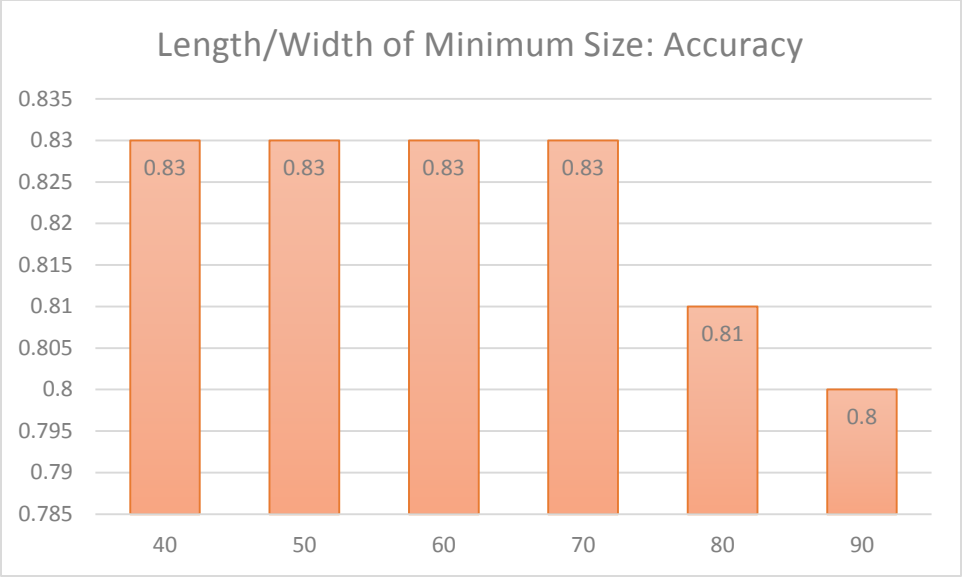

Figure 41: Chart Showing the Accuracy as the Length and Width Changes of the Minimum Size

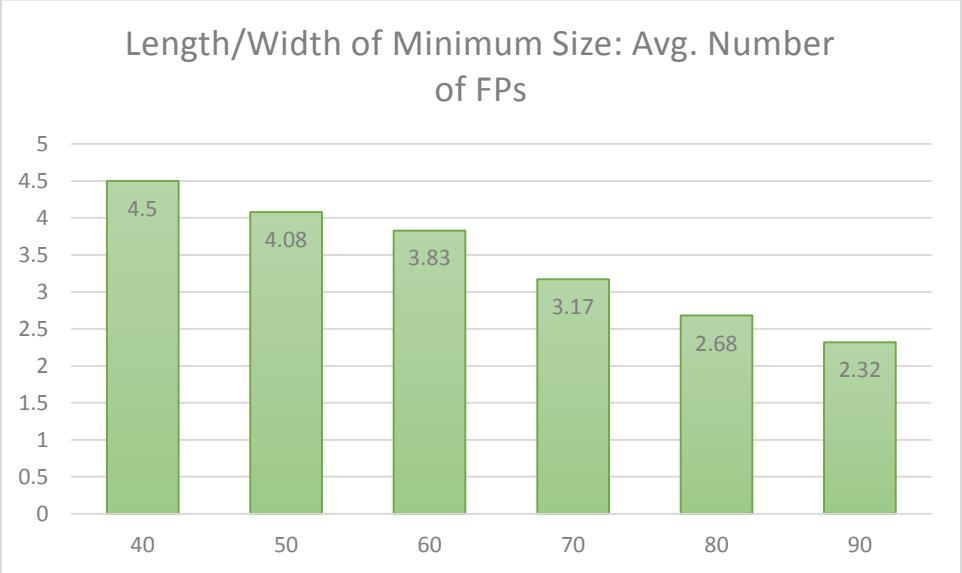

Figure 42: Chart Showing the Average Number of False Positives as the Length and Width Changes of the Minimum Size

The MergeThreshold parameter controls the number of detections required before combining or rejecting the detections. As shown in the figure below.

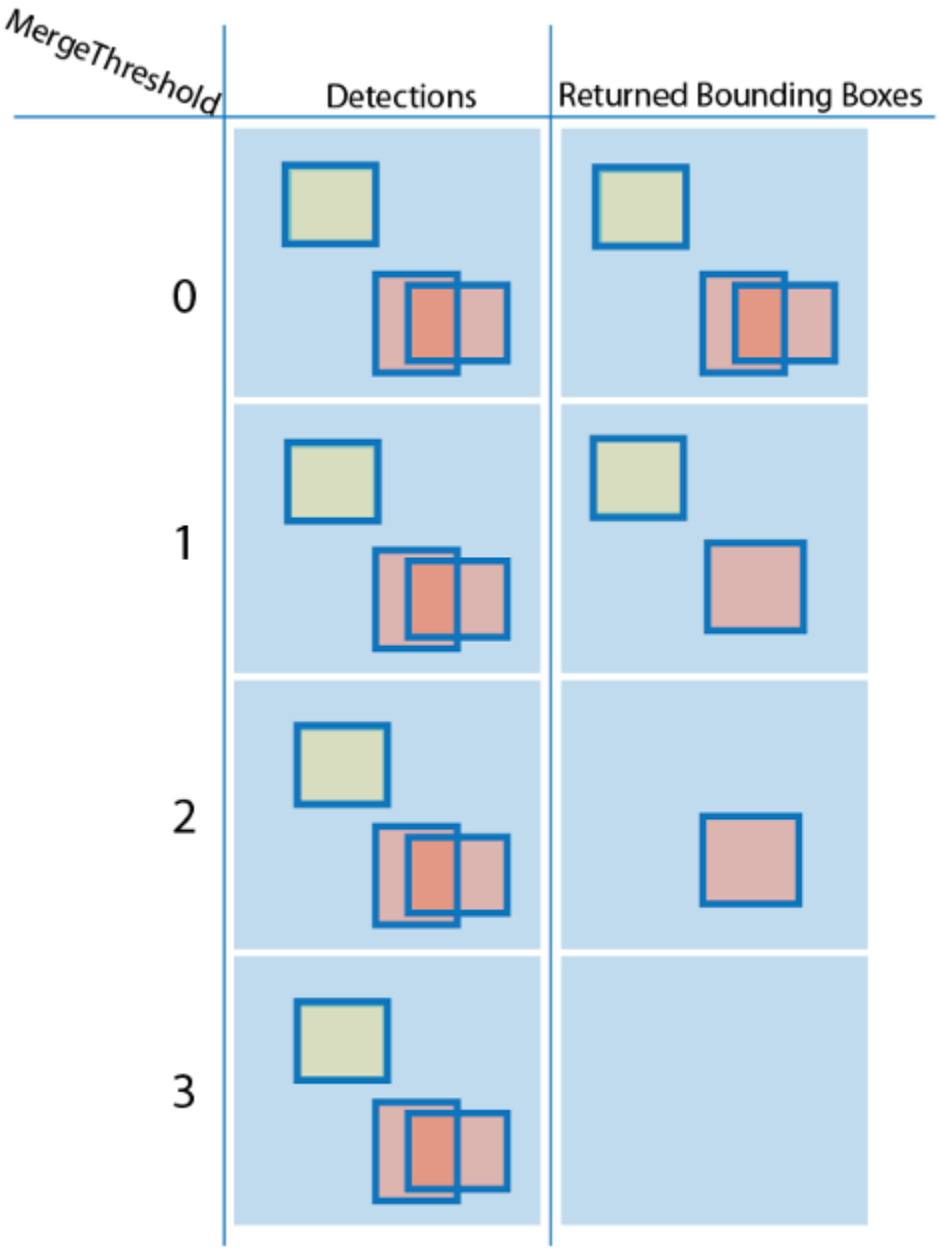

Figure 43: MergeThreshold Visual from Matlab Documentation

When trying different values of the merge threshold parameter, the accuracy in general increased as this parameter decreased. However as the merge threshold

parameter decreased, the number of false positives increased. Therefore the optimal value for the merge threshold is very subjective. If false positives are not wanted, merge threshold should be set to 4 or greater (but this will not give you the highest possible accuracy). If accuracy is the only goal, a value below 4 is needed (a value of 2 seems most optimal), but will have a high number of false positives.

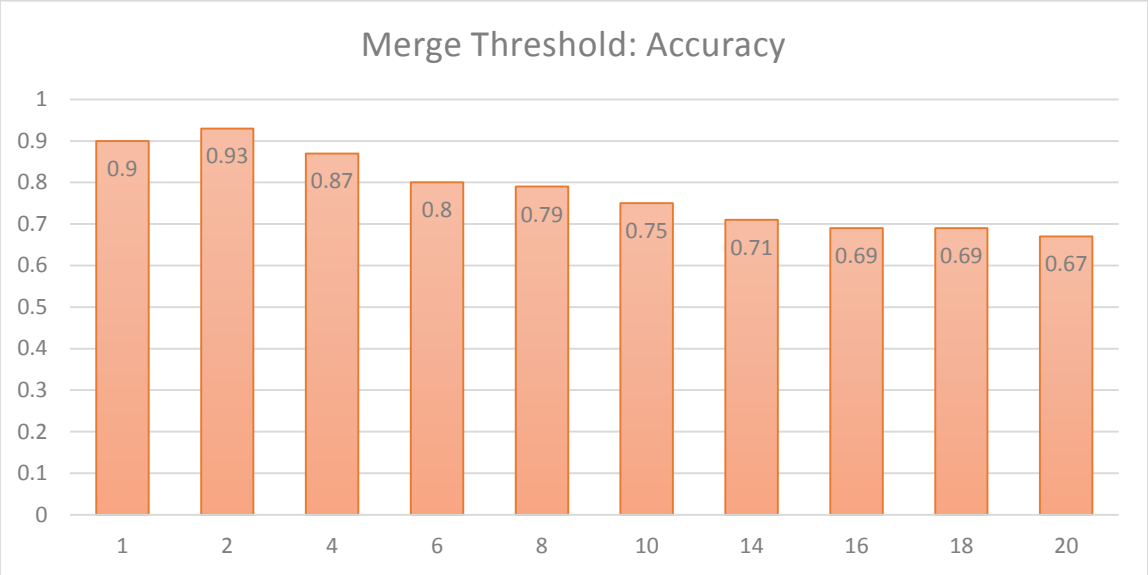

Figure 44: Chart Showing Accuracy as the Merge Threshold Changes

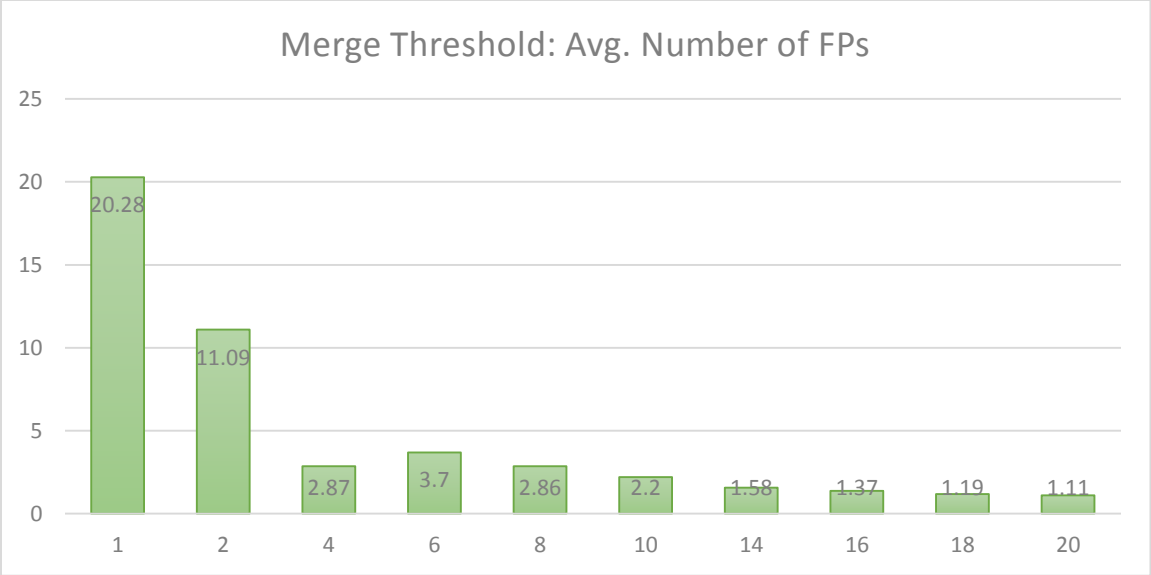

Figure 45: Chart Showing the Average Number of False Positives as the Merge Threshold Changes

The detector incrementally scales the input image to locate target objects (in this case birds). At each scale increment, a sliding window, whose size is the same as the training image size, scans the scaled image to locate objects. The ScaleFactor as explained in [42], the Matlab documentation for detecting objects using the Viola-Jones algorithm, determines the amount of scaling between successive increments.

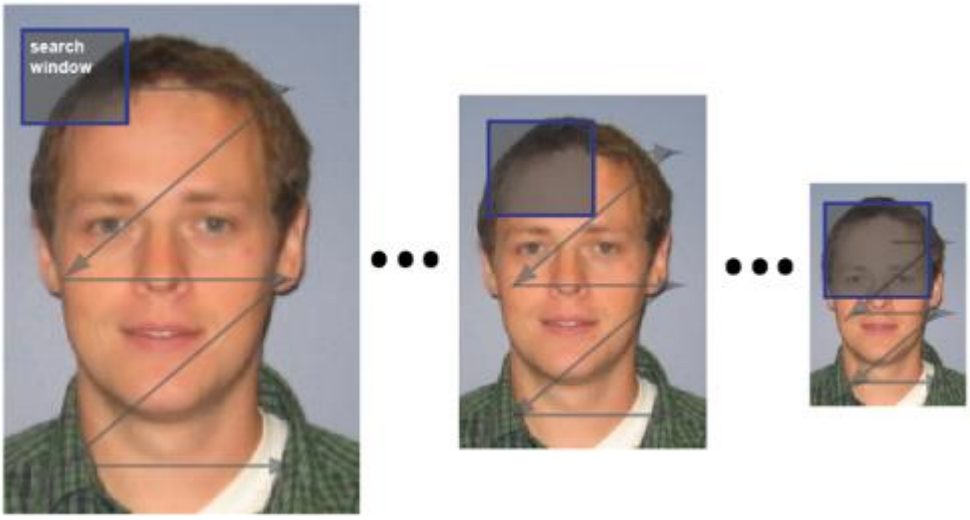

Figure 46: Search Window Example from Matlab Documentation

As with the previous two parameters the default value (1.1) turned out to be already the most optimal parameter. In general the accuracy and the number of false positives seemed to decrease as the scale factor increased.

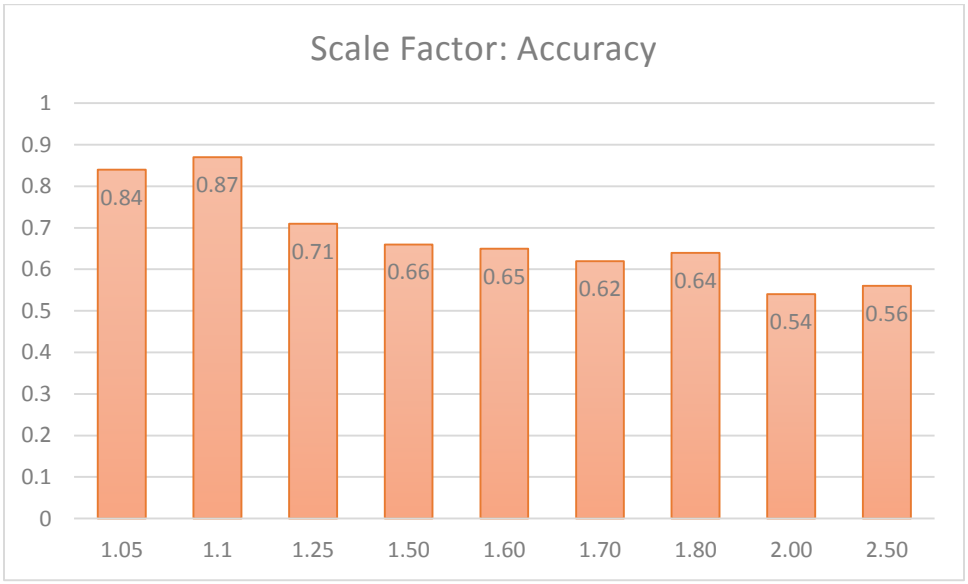

Figure 47: Chart Showing the Accuracy as the Scale Factor Changes

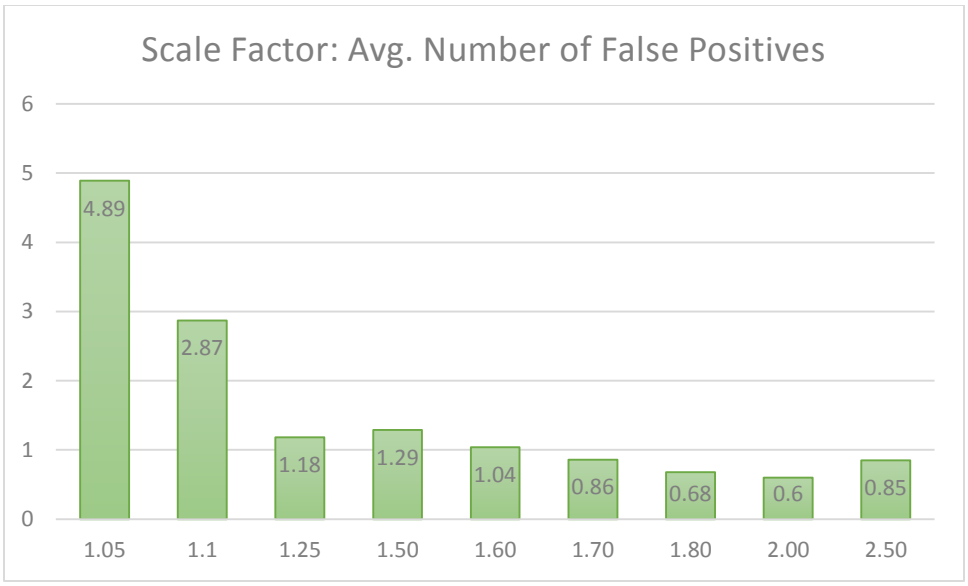

Figure 48: Chart Showing the Average Number of False Positives as the Scale Factor Changes

Run times for training the detector varied anywhere from 2-7 hours depending on the parameter settings. Increasing or decreasing certain parameter adds complexity and more time on the run time. The bird detector that produced an accuracy of 87% took 4 hours to train. The average run time for the actual detection of birds is similar to results in the Motion Detection section, 0.1950 seconds. In the figure below there are examples of successful bird detections.

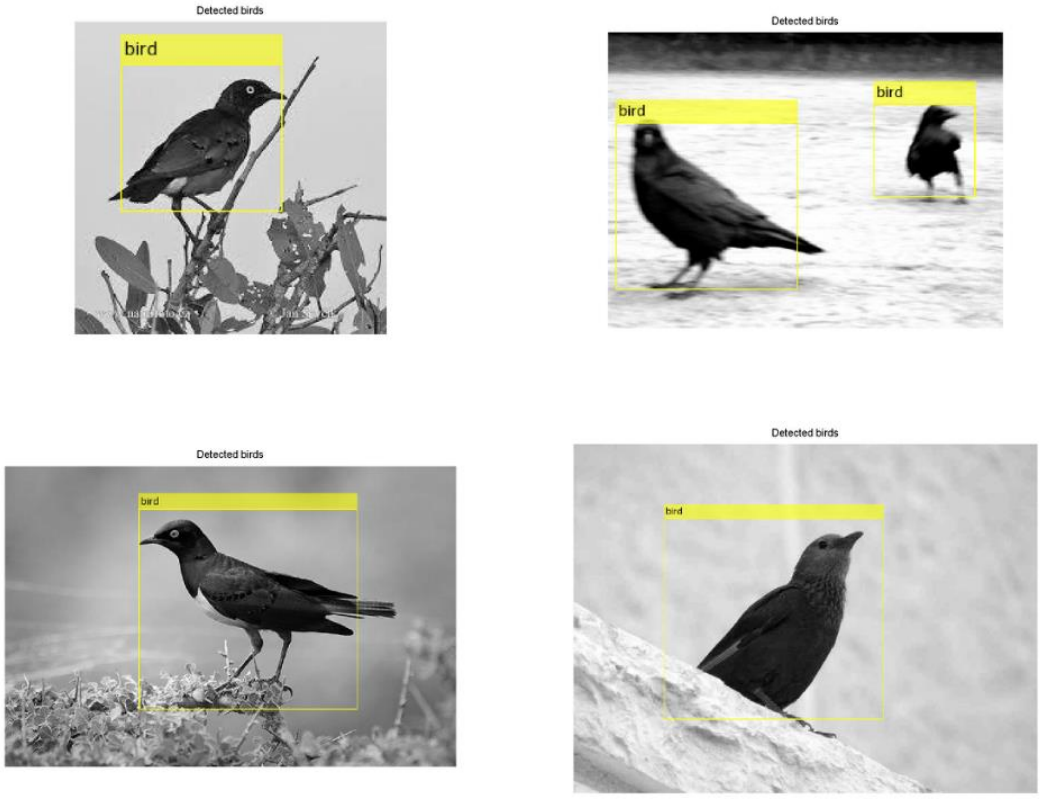

Figure 49: Examples of Successful Bird Detection

The figure below shows images in which the bird detection was unsuccessful or partially unsuccessful. The upper left image shows a bird being detected by one bird. Although it was able to detect the bird in the image successfully. The upper right image shows an example of a bird not being detected at all by the bird detector. In general bird images with water as part of the background usually had detection problems like this image. The lower left image shows a bird being detected correctly and a branch incorrectly being detected as a bird. The lower right image shows some of the birds being detected in the image and some false positives in the image. When birds are very close together, it is harder for the bird detector to properly detect birds.

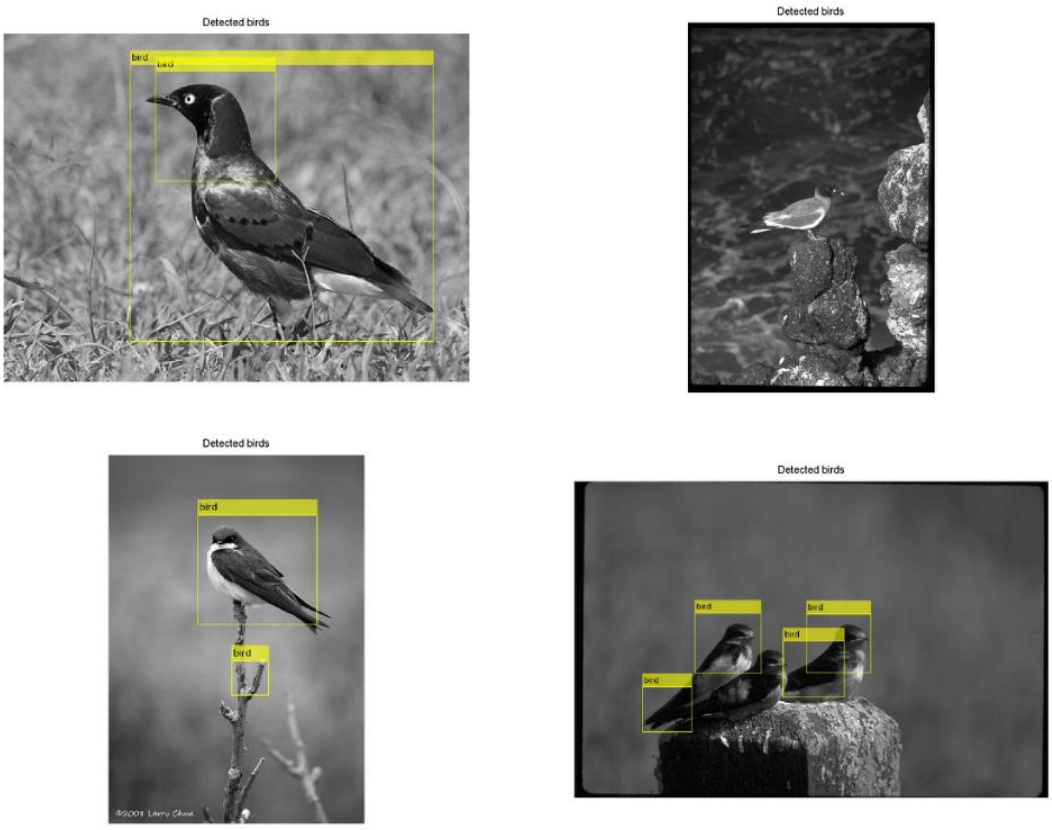

Figure 50: Examples of Unsuccessful/Partially Unsuccessful Changes

4.4 Analysis and Future Work

Overall an accuracy of 87% percent with the average number of false positives below 3 is quite useful in real world applications. Some of the difficulties in finding bird in a scene are that birds overlap each other, there are various positions birds can be in (flying, standing, etc.), lighting can vary depending on the camera and time of the day, etc. Overall it would be important for a scarecrow to be able to detect at least one bird. It does not necessarily need to detect all of the birds on the scene to effectively scare them away. In this case the accuracy is more important than the number of false positives

because there is a bigger penalty in not capturing a bird than incorrectly guessing a bird is in the scene.

As far as future work the parameters of the bird detector could be further fine-tuned and experimented with. What might make a bigger potential impact however is more positive and negative images. With more positive images to train the bird detector, the accuracy might potentially greatly improve. Negative images of natural scenes could also potentially help the accuracy of the bird detector. The parameters MinSize, ScaleFactor, and MergeThreshold could have been potentially combined for the possibility of finding a combination with a higher accuracy and lower false positives then when tested all individually.

The bird detector could also be tested in the RGB color space to see if the accuracy of the bird detector would improve. Although, there is also the possibility that the accuracy could worsen instead. Another possible modification to this project would be to have multiple bird detectors (one for left oriented birds, one for right oriented birds, etc.) like there are multiple templates in template matching. This option was not tested because of the small amount of bird images each bird detector would potentially have. This modification was suggested by the Matlab documentation, however the Viola-Jones algorithm should be able handle images with the object of different orientation so this was another reason this option was not testes. This would also further complicate how to merge false positives and multiple detections on the same bird. Another possibility in taking advantage of using the RGB color space is using color as a possible way to remove some of the false positives that are not in the typically bird color range (such as brown, black, white, dark blue etc.).

The bird detector could also be combined with a motion detection algorithm (like the one mentioned in this paper). This option was not tested because of the minimal amount of eligible footage found of birds. The background needs to be for the most part still (therefore the camera capturing the footage should be stationary) in order for motion detection algorithms to work properly. Most available footage of birds are not taken with a video camera.

Future students can further add to this project in various ways. Many papers focused on bird detection through audio instead of images. A whole thesis can be written on using signal processing techniques to identify birds via audio. An Audio component could be added to this system to further improve the accuracy and/or lower the number of false positives. There are also other popular face detection algorithms that could be explored and experimented with.

This project could also be made into a full system with the addition of an FPGA or microcontroller that could react when birds are detected in real time by scaring away the birds. The system could possibly move towards the bird and try to throw water at it or make alarming noises.

5.0

Summary and Future Work

Using the Viola-Jones algorithm as a bird deterrent is cheaper than netting, a long-term solution unlike the traditional scarecrow, and will deter birds at a rate of 87% (if one bird of many is selected this is still for a real world implementation so in a way the accuracy is actually higher) which is higher than what netting (30%) and scarecrows (0% long term) can accomplish. The motion detection algorithm that had the highest accrucacy was the absolute ADI with a k value of 10. It was able to able to find objects in a scene at a rate of 100%. However as mentioned before, a motion detection algorithm is not enough alone for detecting birds in the scene. Template matching at a threshold of 0.7 (with scaling the templates to four different sizes) achieved the highest accuracy (for template matching specifically) of 59% with a false positive rate per frame of 4.65. Padding the image with white borders instead of black borders lowered the false positive rate per frame to 3.67. The template matching technique would not be suitable for the smart scarecrow system because of the fact that it cannot be run in real time the absolute ADI and the Viola-Jones algorithm.

In this paper, three different image processing techniques were implemented for bird detection. Overall this paper addressed the image processing component that would be part of an overall smart scarecrow system. This component is the most crucial step in the system because the image processing component is what determines whether or not the bird gets detected as a threat or not.

As far as future work, ideally the motion detection component (the absolute ADI) and the bird detection with the Viola-Jones algorithm component should be combined.

These two components can be run in real time. Although template matching could also be potentially used, it would slow down the system too much to be able to keep up in real time. Once a physical smart scarecrow system is implemented, the image processing component (the absolute ADI algorithm combined with the Viola-Jones algorithm) would have to trained and tailored with images and videos from the locations in which the camera will be placed. This will allow for optimal parameters and higher accuracies when setup to run autonomously.

Bibliography

[1] R. E. Marsh, W. A. Erickson, and T. P. Salmon, “SCARECROWS AND PREDATOR MODELS FOR FRIGHTENING BIRDS FROM SPECIFIC AREAS,” pp. 3–6, 1992.

[2] C. M. Somers and R. D. Morris, “Birds and wine grapes: foraging activity causes small-scale damage patterns in single vineyards,” J. Appl. Ecol., vol. 39, no. 3, pp. 511–523, Jun. 2002.

[3] K. Gebhardt, “The Economic Impact of Bird and Rodent Damage to,” 2009.F

[4] Fuller-Perrine, L.D, and M.E Tobin. "A Method for Applying and Removing Bird-exclusion Netting in Commercial Vineyards." *Wildlife Society. Wildlife Society Bulletin,* 21.1 (1993): 47-51.

[5] Curtis, P.D, D.V Peterson, M.P Pritts, and I.A Merwin. "Chemical Repellents and Plastic Netting for Reducing Bird Damage to Sweet Cherries, Blueberries, and Grapes." *HortScience : A Publication of the American Society for Horticultural Science,* 29.10 (1994): 1151-1155.

[6] M. Lusby and I. Al-Bahadly, “An integrated approach for automatic farm monitoring system,” *2009 4th International Conference on Autonomous Robots and Agents*, pp. 21–26, Feb. 2000.

[7] A. Kim, S. Rhee, S. Seok, H. Lee, and Y. Kim, “CCIS 254 - A Surveillance System for a Farm,” pp. 273–284, 2011.

[8] H. Lee, “Design and implementation of pig farm monitoring system for ubiquitous agriculture,” *2010 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 557–558, Nov. 2010.

[9] Richard Wotiz, " Electronic Scarecrow," *Circuit Cellar*, vol. 186, pp. 18-24, January 2006.

[10] C. Pornpanomchai, M. Homnan, N. Pramuksan, and W. Rakyindee, "Smart Scarecrow," *2011 Third International Conference on Measuring Technology and Mechatronics Automation*, pp. 294–297, Jan. 2011.

[11] Animal Control Solutions, "Santa Maria, CA Animal Removal, Wildlife Control" Animal Control Solutions, 2013, [Online]. Available: http://www.animalcontrolsolutions.com/california/santa-maria-animal-removal.html [Accessed: February 2013].

[12] Mummert, D. P., Baines, L., & Tietje, W. D. (2002). Cavity-nesting Bird Use of Nest Boxes in Vineyards of Central-Coast California 1, 335–340.

[13] X. Yun and E. R. Bachmann, "Design , Implementation , and Experimental Results of a Quaternion-Based Kalman Filter for Human Body Motion Tracking," vol. 22, no. 6, pp. 1216–1227, 2006.

[14] Y. Li, Y. Yin, and B. Xu, "Bird Objects Detection and Tracking on the Wild Field Circumstance," *2009 WRI World Congress on Computer Science and Information Engineering*, no. 40771151, pp. 533–536, 2009.

[15] K. Maruyama, T. Saitoh, and R. Konishi, "Bird Tracking and Flapping Motion Recognition for Monitoring System," pp. 3613–3618, 2009.

[16] Bird Video and Film Stock Footage HD High Definition Nature Videos Nature Wildlife Stock Video Footage HD Royalty Free Rights Managed Red 4K Stock Footage, *Nature Footage*, [online] 2013,

http://www.naturefootage.com/stock_footage/bird_footage.html (Accessed: 20 May 2013).

[17] Di Stefano, L.; Mattoccia, S.; Mola, M., "An efficient algorithm for exhaustive template matching based on normalized cross correlation," *Image Analysis and Processing, 2003.Proceedings. 12th International Conference on,* vol., no., pp.322, 327, 17-19 Sept. 2003.

[18] Zhuo Yang, "Fast Template Matching Based on Normalized Cross Correlation with Centroid Bounding," *Measuring Technology and Mechatronics Automation (ICMTMA), 2010 International Conference on* , vol.2, no., pp.224,227, 13-14 March 2010.

[19] U. D. Nadimpalli, R. R. Price, S. G. Hall, and P. Bomma, "A comparison of image processing techniques for bird recognition," *Biotechnology Progress*, vol. 22, no. 1, pp. 9–13, 2006.

[20] Qiuze Yu; Yufan Wang; Yan Zhang; Guangzhou Qu, "Highperformance local-texture-information weighted SAR template image matching," *Radar Conference 2013, IET International* , vol., no., pp.1,5, 14-16 April 2013.

[21] Brunelli, Roberto, *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley ed, Vol . .2009.

[22] Madhu, M.; Amutha, R., "An Improved method for face authentication and verification system," *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on* , vol., no., pp.396,403, 30-31 March 2012.

[23] Aboutajdine, D.; Essannouni, F., "Fast block matching algorithms using frequency domain," *Multimedia Computing and Systems (ICMCS), 2011 International Conference on* , vol., no., pp.1,6, 7-9 April 2011.

[24] Yufan Wang; Qiuze Yu; Wenxian Yu, "An improved Normalized Cross Correlation algorithm for SAR image registration," Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International , vol., no., pp.2086,2089, 22-27 July 2012.

[25] D. Koon, "Fast/Robust Template Matching" Mathworks, August 04, 2009, [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/24925-fastrobust-template-matching [Accessed: February 2014].

[26] P. Welinder and C. Wah, "Caltech-UCSD Birds 200," vol. 200, pp. 1–15.

[27] CalPhotos: Brows BirdCommon Names, *CalPhotos,* [online] 2013, http://calphotos.berkeley.edu/fauna/com-Bird.html (Accessed: 20 May 2013).

[28] Jianxiong Xiao; Hays, J.; Ehinger, K.A.; Oliva, A.; Torralba, A., "SUN database: Large-scale scene recognition from abbey to zoo," *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* , vol., no., pp.3485,3492, 13-18 June 2010.

[29] Olmos, A., Kingdom, F. A. A. (2004). A biologically inspired algorithm for the recovery of shading and reflectance images, *Perception*, 33, 1463 - 1473.

[30] E. C. Larson and D. M. Chandler, "Most apparent distortion: full-reference image quality assessment and the role of strategy," *Journal of Electronic Imaging*, 19 (*1*), March 2010.

[31] J. Z. Wang, J. Li, G. Wiederhold, ``SIMPLIcity: Semantics-sensitive integrated matching for picture libraries,'' *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(9):947-963, 2001.

[32] Viola, P.; Jones, M., "Robust real-time face detection," *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol.2, no., pp.747, 747, 2001.

[33] Ephraim, T.; Himmelman, T.; Siddiqi, K., "Real-Time Viola-Jones Face Detection in a Web Browser," Computer and Robot Vision, 2009. CRV '09. Canadian Conference on, vol., no., pp.321,328, 25-27 May 2009.

[34] Jianfeng Ren; Kehtarnavaz, N.; Estevez, L., "Real-time optimization of Viola -Jones face detection for mobile platforms," *Circuits and Systems Workshop: System-on-Chip - Design, Applications, Integration, and Software, 2008 IEEE Dallas* , vol., no., pp.1,4, 19-20 Oct. 2008.

[35] Kolsch, M.; Turk, M., "Analysis of rotational robustness of hand detection with a Viola-Jones detector," *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on* , vol.3, no., pp.107,110 Vol.3, 23-26 Aug. 2004.

[36] Krerngkamjornkit, R.; Simic, M., "Enhancement of human body detection and tracking algorithm based on Viola and Jones framework," *Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), 2013 11th International Conference on* , vol.01, no., pp.115,118, 16-19 Oct. 2013.

[37] Navarathna, R.; Lucey, P.; Dean, D.; Fookes, C.; Sridharan, S., "Lip detection for audio-visual speech recognition in-car environment," *Information Sciences Signal*

Processing and their Applications (ISSPA), 2010 10th International Conference on , vol., no., pp.598,601, 10-13 May 2010.

[38] B. Shoelson, "Cascade Training GUI: Specify Ground Truth" Mathworks, March 08, 2013, [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/39627-cascade-training-gui--specify-ground-truth [Accessed: February 2014].

[39] Mathworks "Train a Cascade Object Detector" Mathworks, 2014, [Online]. Available: http://www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html [Accessed: February 2014].

[40] Mihreteab, K.; Iwahashi, M.; Yamamoto, M., "Crow birds detection using HOG and CS-LBP," *Intelligent Signal Processing and Communications Systems (ISPACS), 2012 International Symposium on* , vol., no., pp.406,409, 4-7 Nov. 2012.

[41] Chih-Cheng Huang; Chun-Yi Tsai; Horng-Chang Yang, *Signal Processing, Image Processing and Pattern Recognition*. Springer; vol. 260.2011.

[42] Mathworks "Detect objects using the Viola-Jones algorithm" Mathworks, 2014, [Online]. Available: http://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html [Accessed: February 2014].

Appendix A: Motion Detection Code

A1: absolute_adi_testing.m

```
function [ numbirds] = absolute_adi_testing( file1, maxframes )
   %example input
   %maxframes = represents the k value from the ADI equations;
   %file1 = 'Duncrafts Swallow Bird House 2107.flv(360p_H.264-AAC).mpg';

   %base code is from:
   %http://www.mathworks.com/matlabcentral/fileexchange/36786-kalman-filter-in-
tracking/content/tracking.m
   %file1:  name of video for which the motion will be looked for

   %% Read video into MATLAB using mmreader
   % user tag set to 'myreader1'.
   readerobj = mmreader(file1, 'tag', 'myreader1');

   % Read in all video frames.
   vidFrames = read(readerobj);

   % Get the number of frames.
   nframes = get(readerobj, 'NumberOfFrames');

   % Create a MATLAB movie struct from the video frames.
   images = cell(maxframes,1);

   for k = 1 : (nframes - 10)
       video(k).cdata = vidFrames(:,:,:,k);
       video(k).colormap = [];
   end
%
     temp1 = zeros(size(video(1).cdata));
    [M,N] = size(temp1(:,:,1));
      for i = 1:maxframes
        %images{i} = im2bw(zeros(M,N));
        images{i} = double(video(k).cdata);
      end

   counter = 1;
   imbkg = images{counter};

   % Calculate the background image by averaging the first 5 images
   temp = zeros(size(video(1).cdata));
   [M,N] = size(temp(:,:,1));
   for i = 1:10
```

```
    temp = double(video(i).cdata) + temp;
  end

  th = 100;
  th_area = 23000;
  total_adi = im2bw(zeros(M,N));
  for i=1:nframes-10
   imshow(video(i).cdata);
   hold on
   imcurrent = double(video(i).cdata);

   cur_adi = (abs(imcurrent(:,:,1)-imbkg(:,:,1))>th) ...
      | (abs(imcurrent(:,:,2)-imbkg(:,:,2))>th) ...
      | (abs(imcurrent(:,:,3)-imbkg(:,:,3))>th);

   images{counter} = cur_adi;

   if counter == maxframes,
      counter = 1;
      total_adi = cur_adi;
   else
      counter = counter + 1;
   end
  total_adi = im2bw(images{counter}) +  total_adi;

  total_adi_thresholded = im2bw(total_adi);
  markimg = regionprops(total_adi_thresholded,['basic']);
  [MM,NN] = size(markimg);

   % The larger regions are objects
   for nn = 1:MM
      if markimg(nn).Area > th_area%markimg(1).Area
         tmp = markimg(1);
         markimg(1)= markimg(nn);
         markimg(nn)= tmp;
          % Get the upper-left corner, the measurement centroid and bounding window
size
         bb = markimg(1).BoundingBox;
         xcorner = bb(1);
         ycorner = bb(2);
         xwidth = bb(3);
         ywidth = bb(4);
         cc = markimg(1).Centroid;
         centroidx(i)= cc(1);
         centroidy(i)= cc(2);
         numbirds = numbirds + 1;
```

```
            if counted == 0;
              loopcounter = loopcounter + 1;
              counted = 1;
            end

            % Plot the rectangle of background subtraction algorithm -- blue
            hold on
            rectangle('Position',[xcorner ycorner xwidth ywidth],'EdgeColor','b');
            hold on
            plot(centroidx(i),centroidy(i), 'bx');
            drawnow;
        end
    end
  end
end
```

A2: negative_adi_testing.m

```
function [ ] = negative_adi_testing( file1, maxframes )
   %example input
   %maxframes = represents the k value from the ADI equations;
   %file1 = 'Duncrafts Swallow Bird House 2107.flv(360p_H.264-AAC).mpg';

   %base code is from:
   %http://www.mathworks.com/matlabcentral/fileexchange/36786-kalman-filter-in-
tracking/content/tracking.m

   %file1:  name of video for which the motion will be looked for
   close all;

   %% Read video into MATLAB using mmreader
   readerobj = mmreader(file1, 'tag', 'myreader1');

   % Read in all video frames.
   vidFrames = read(readerobj);

   % Get the number of frames.
   nframes = get(readerobj, 'NumberOfFrames');

   % Create a MATLAB movie struct from the video frames.
   images = cell(maxframes,1);

   for k = 1 : (nframes - 10)
       video(k).cdata = vidFrames(:,:,:,k);
       video(k).colormap = [];
   end
%
     temp1 = zeros(size(video(1).cdata));
    [M,N] = size(temp1(:,:,1));
      for i = 1:maxframes
        images{i} = double(video(k).cdata);
      end

   counter = 1;
   imbkg = images{counter};

   % Calculate the background image by averaging the first 5 images
   temp = zeros(size(video(1).cdata));
   [M,N] = size(temp(:,:,1));
   for i = 1:10
      temp = double(video(i).cdata) + temp;
   end
```

```
th = 100;
th_area = 23000;
total_adi = im2bw(zeros(M,N));
for i=1:nframes-10
 imshow(video(i).cdata);
 hold on
 imcurrent = double(video(i).cdata);

 cur_adi = ((imcurrent(:,:,1)-imbkg(:,:,1))<-th) ...
    | ((imcurrent(:,:,2)-imbkg(:,:,2))<-th) ...
    | ((imcurrent(:,:,3)-imbkg(:,:,3))<-th);

 images{counter} = cur_adi;

 if counter == maxframes,
    counter = 1;
    total_adi = cur_adi;
 else
    counter = counter + 1;
 end
 total_adi = im2bw(images{counter}) +  total_adi;

 total_adi_thresholded = im2bw(total_adi);
  markimg = regionprops(total_adi_thresholded,['basic']);
 [MM,NN] = size(markimg);

 % The larger regions are objects
 for nn = 1:MM
    if markimg(nn).Area > th_area%markimg(1).Area
       tmp = markimg(1);
       markimg(1)= markimg(nn);
       markimg(nn)= tmp;
       % Get the upper-left corner, the measurement centroid and bounding window
size
       bb = markimg(1).BoundingBox;
       xcorner = bb(1);
       ycorner = bb(2);
       xwidth = bb(3);
       ywidth = bb(4);
       cc = markimg(1).Centroid;
       centroidx(i)= cc(1);
       centroidy(i)= cc(2);

       % Plot the rectangle of background subtraction algorithm -- blue
       hold on
```

```
            rectangle('Position',[xcorner ycorner xwidth ywidth],'EdgeColor','b');
            hold on
            plot(centroidx(i),centroidy(i), 'bx');
            drawnow;
        end
    end
  end
end
```

A3: positive_adi_testing.m

```
function [ numbirds] = positive_adi_testing( file1, maxframes )
   %example input
   %maxframes = represents the k value from the ADI equations;
   %file1 = 'Duncrafts Swallow Bird House 2107.flv(360p_H.264-AAC).mpg';

   %base code is from:
   %http://www.mathworks.com/matlabcentral/fileexchange/36786-kalman-filter-in-
tracking/content/tracking.m
   %file1:  name of video for which the motion will be looked for

   %% Read video into MATLAB using mmreader
   % user tag set to 'myreader1'.
   readerobj = mmreader(file1, 'tag', 'myreader1');

   vidFrames = read(readerobj);

   % Get the number of frames.
   nframes = get(readerobj, 'NumberOfFrames');

   % Create a MATLAB movie struct from the video frames.
   images = cell(maxframes,1);

   for k = 1 : (nframes - 10)
       video(k).cdata = vidFrames(:,:,:,k);
       video(k).colormap = [];
   end

     temp1 = zeros(size(video(1).cdata));
    [M,N] = size(temp1(:,:,1));
      for i = 1:maxframes
         images{i} = double(video(k).cdata);
      end

   counter = 1;
   imbkg = images{counter};

   % Calculate the background image by averaging the first 5 images
   temp = zeros(size(video(1).cdata));
   [M,N] = size(temp(:,:,1));
   for i = 1:10
      temp = double(video(i).cdata) + temp;
   end
```

```
th = 100;
th_area = 23000;
total_adi = im2bw(zeros(M,N));
for i=1:nframes-10
 imshow(video(i).cdata);
 hold on
 imcurrent = double(video(i).cdata);

  cur_adi = ((imcurrent(:,:,1)-imbkg(:,:,1))>th) ...
    | ((imcurrent(:,:,2)-imbkg(:,:,2))>th) ...
    | ((imcurrent(:,:,3)-imbkg(:,:,3))>th);

 images{counter} = cur_adi;

 if counter == maxframes,
    counter = 1;
    total_adi = cur_adi;
 else
    counter = counter + 1;
 end
total_adi = im2bw(images{counter}) +  total_adi;

total_adi_thresholded = im2bw(total_adi);
  markimg = regionprops(total_adi_thresholded,['basic']);
 [MM,NN] = size(markimg);

 % The larger regions are objects
 for nn = 1:MM
    if markimg(nn).Area > th_area%markimg(1).Area
       tmp = markimg(1);
       markimg(1)= markimg(nn);
       markimg(nn)= tmp;
       % Get the upper-left corner, the measurement centroid and bounding window
size
       bb = markimg(1).BoundingBox;
       xcorner = bb(1);
       ycorner = bb(2);
       xwidth = bb(3);
       ywidth = bb(4);
       cc = markimg(1).Centroid;
       centroidx(i)= cc(1);
       centroidy(i)= cc(2);
       numbirds = numbirds + 1;
       if counted == 0;
         loopcounter = loopcounter + 1;
         counted = 1;
```

```
            end

            % Plot the rectangle of background subtraction algorithm -- blue
            hold on
            rectangle('Position',[xcorner ycorner xwidth ywidth],'EdgeColor','b');
            hold on
            plot(centroidx(i),centroidy(i), 'bx');
            drawnow;
        end
    end
  end
end
```

Appendix B: Template Matching Code

B1: template_matching.m

```
function [I_SSD,I_NCC,Idata]=template_matching(T,I,IdataIn)
% TEMPLATE_MATCHING is a cpu efficient function which calculates matching
% score images between template and an (color) 2D or 3D image.
% It calculates:
% - The sum of squared difference (SSD Block Matching), robust template
%   matching.
% - The normalized cross correlation (NCC), independent of illumination,
%   only dependent on texture
% The user can combine the two images, to get template matching which
% works robust with his application.
% Both measures are implemented using FFT based correlation.
%
%   [I_SSD,I_NCC,Idata]=template_matching(T,I,Idata)
%
% inputs,
%   T : Image Template, can be grayscale or color 2D or 3D.
%   I : Color image, can be grayscale or color 2D or 3D.
%  (optional)
%   Idata : Storage of temporary variables from the image I, to allow
%           faster search for multiple templates in the same image.
%
% outputs,
%   I_SSD: The sum of squared difference 2D/3D image. The SSD sign is
%          reversed and normalized to range [0 1]
%   I_NCC: The normalized cross correlation 2D/3D image. The values
%          range between 0 and 1
%   Idata : Storage of temporary variables from the image I, to allow
%           faster search for multiple templates in the same image.
%
% Example 2D,
%   % Find maximum response
%    I = im2double(imread('lena.jpg'));
%   % Template of Eye Lena
%    T=I(124:140,124:140,:);
%   % Calculate SSD and NCC between Template and Image
%    [I_SSD,I_NCC]=template_matching(T,I);
%   % Find maximum correspondence in I_SDD image
%    [x,y]=find(I_SSD==max(I_SSD(:)));
%   % Show result
%    figure,
%    subplot(2,2,1), imshow(I); hold on; plot(y,x,'r*'); title('Result')
%    subplot(2,2,2), imshow(T); title('The eye template');
```

```
%    subplot(2,2,3), imshow(I_SSD); title('SSD Matching');
%    subplot(2,2,4), imshow(I_NCC); title('Normalized-CC');
%
% Example 3D,
%   % Make some random data
%    I=rand(50,60,50);
%   % Get a small volume as template
%    T=I(20:30,20:30,20:30);
%   % Calculate SDD between template and image
%    I_SSD=template_matching(T,I);
%   % Find maximum correspondence
%    [x,y,z]=ind2sub(size(I_SSD),find(I_SSD==max(I_SSD(:))));
%    disp(x);
%    disp(y);
%    disp(z);
%
% Function is written by D.Kroon University of Twente (February 2011)

if(nargin<3), IdataIn=[]; end

% Convert images to double
T=double(T); I=double(I);
if(size(T,3)==3)
   % Color Image detected
%     disp('got in color');
   [I_SSD,I_NCC,Idata]=template_matching_color(T,I,IdataIn);
else
   % Grayscale image or 3D volume
%     disp('got in here?');
   [I_SSD,I_NCC,Idata]=template_matching_gray(T,I,IdataIn);
end

function [I_SSD,I_NCC,Idata]=template_matching_color(T,I,IdataIn)
if(isempty(IdataIn)), IdataIn.r=[];  IdataIn.g=[]; IdataIn.b=[];  end
% Splite color image, and do template matching on R,G and B image
[I_SSD_R,I_NCC_R,Idata.r]=template_matching_gray(T(:,:,1),I(:,:,1),IdataIn.r);
[I_SSD_G,I_NCC_G,Idata.g]=template_matching_gray(T(:,:,2),I(:,:,2),IdataIn.g);
[I_SSD_B,I_NCC_B,Idata.b]=template_matching_gray(T(:,:,3),I(:,:,3),IdataIn.b);
% Combine the results
I_SSD=(I_SSD_R+I_SSD_G+I_SSD_B)/3;
I_NCC=(I_NCC_R+I_NCC_G+I_NCC_B)/3;


function [I_SSD,I_NCC,Idata]=template_matching_gray(T,I,IdataIn)
% Calculate correlation output size  = input size + padding template
T_size = size(T); I_size = size(I);
```

```
outsize = I_size + T_size-1;

% calculate correlation in frequency domain
if(length(T_size)==2)
   FT = fft2(rot90(T,2),outsize(1),outsize(2));
   if(isempty(IdataIn))
      Idata.FI = fft2(I,outsize(1),outsize(2));
   else
      Idata.FI=IdataIn.FI;
   end
   Icorr = real(ifft2(Idata.FI.* FT));
else
   FT = fftn(rot90_3D(T),outsize);
   FI = fftn(I,outsize);
   Icorr = real(ifftn(FI.* FT));
end

% Calculate Local Quadratic sum of Image and Template
if(isempty(IdataIn))
   Idata.LocalQSumI= local_sum(I.*I,T_size);
else
   Idata.LocalQSumI=IdataIn.LocalQSumI;
end

QSumT = sum(T(:).^2);

% SSD between template and image
I_SSD=Idata.LocalQSumI+QSumT-2*Icorr;

% Normalize to range 0..1
I_SSD=I_SSD-min(I_SSD(:));
I_SSD=1-(I_SSD./max(I_SSD(:)));

% Remove padding
I_SSD=unpadarray(I_SSD,size(I));
% I_SSD = 0;

if (nargout>1)
   % Normalized cross correlation STD
   if(isempty(IdataIn))
      Idata.LocalSumI= local_sum(I,T_size);
   else
      Idata.LocalSumI=IdataIn.LocalSumI;
   end

   % Standard deviation
```

```
    if(isempty(IdataIn))
        Idata.stdI=sqrt(max(Idata.LocalQSumI-(Idata.LocalSumI.^2)/numel(T),0) );
    else
        Idata.stdI=IdataIn.stdI;
    end
    stdT=sqrt(numel(T)-1)*std(T(:));
    % Mean compensation
    meanIT=Idata.LocalSumI*sum(T(:))/numel(T);
    I_NCC= 0.5+(Icorr-meanIT)./ (2*stdT*max(Idata.stdI,stdT/1e5));

    % Remove padding
    I_NCC=unpadarray(I_NCC,size(I));
end

function T=rot90_3D(T)
T=flipdim(flipdim(flipdim(T,1),2),3);

function B=unpadarray(A,Bsize)
Bstart=ceil((size(A)-Bsize)/2)+1;
Bend=Bstart+Bsize-1;
if(ndims(A)==2)
    B=A(Bstart(1):Bend(1),Bstart(2):Bend(2));
elseif(ndims(A)==3)
    B=A(Bstart(1):Bend(1),Bstart(2):Bend(2),Bstart(3):Bend(3));
end

function local_sum_I= local_sum(I,T_size)
% Add padding to the image
% Original packing technique padded image with zeroes but since the
% template leans more toward lower intensity values, padding with
% 255 (white border) gave better results
% B = padarray(I,T_size);
B = padarray(I,T_size, 255);


% Calculate for each pixel the sum of the region around it,
% with the region the size of the template.
if(length(T_size)==2)
    % 2D localsum
    s = cumsum(B,1);
    c = s(1+T_size(1):end-1,:)-s(1:end-T_size(1)-1,:);
    s = cumsum(c,2);
    local_sum_I= s(:,1+T_size(2):end-1)-s(:,1:end-T_size(2)-1);
else
    % 3D Localsum
    s = cumsum(B,1);
```

```
    c = s(1+T_size(1):end-1,:,:)-s(1:end-T_size(1)-1,:,:);
    s = cumsum(c,2);
    c = s(:,1+T_size(2):end-1,:)-s(:,1:end-T_size(2)-1,:);
    s = cumsum(c,3);
    local_sum_I  = s(:,:,1+T_size(3):end-1)-s(:,:,1:end-T_size(3)-1);
end
```

B2: template_matching_testing.m

```
function [] = template_matching_testing( thresh )

%using the first template_matching.m
%function [I_SSD,I_NCC,Idata]=template_matching(T,I,IdataIn)
%http://www.mathworks.com/matlabcentral/fileexchange/24925-fastrobust-template-matching
close all;
%thresh = 0.75;
x_max = 480;
y_max = 640;
points = zeros(x_max, y_max);
% This excel list consists a list of images in the same directory as this
% code that will be used to test template matching.
[~,data, ~] = xlsread('list_of_images.xlsx');
[M,~] = size(data);
T1_left_120 = im2double(imread('left_template_120.jpg'));
T1_right_120 = im2double(imread('right_template_120.jpg'));
T1_left_60 = im2double(imread('left_template_60.jpg'));
T1_right_60 = im2double(imread('right_template_60.jpg'));
T1_left_480 = im2double(imread('left_template_480.jpg'));
T1_right_480 = im2double(imread('right_template_480.jpg'));
T1_left_240 = im2double(imread('left_template_240.jpg'));
T1_right_240 = im2double(imread('right_template_240.jpg'));


for index = 1:M
    I = im2double(imresize(imread(data{index,1}), [x_max y_max]));
    [~, I_NCC_L_120, Idata]=template_matching(T1_left_120,I);
    [~, I_NCC_R_120]=template_matching(T1_right_120,I, Idata);
    [~, I_NCC_L_60, Idata]=template_matching(T1_left_60,I);
    [~, I_NCC_R_60]=template_matching(T1_right_60,I, Idata);
    [~, I_NCC_L_480, Idata]=template_matching(T1_left_480,I);
    [~, I_NCC_R_480]=template_matching(T1_right_480,I, Idata);
    [~, I_NCC_L_240, Idata]=template_matching(T1_left_240,I);
    [~, I_NCC_R_240]=template_matching(T1_right_240,I, Idata);

    [x,y]=find((I_NCC_L_120 >= thresh) |(I_NCC_L_60 >= thresh) | (I_NCC_L_240 >= thresh) | ((I_NCC_L_480) >= thresh)...
            | (I_NCC_R_120 >= thresh) |(I_NCC_R_60 >= thresh) | (I_NCC_R_240 >= thresh) | ((I_NCC_R_480) >= thresh));

   final_120 = ones(size(x));
    radius = 120;
    for i=1:length(x),
```

```
        for j=1:length(x),
            distance = sqrt((x(i,1) - x(j,1)) .^ 2 + (y(i,1) - (y(j,1))) .^ 2);
            if distance <= radius,
                max_vali = find_max(I_NCC_L_120(x(i, 1), y(i, 1)), I_NCC_L_60(x(i, 1), y(i,
1)), I_NCC_L_240(x(i, 1), y(i, 1)), I_NCC_L_480(x(i, 1), y(i, 1)),...
                I_NCC_R_120(x(i, 1), y(i, 1)), I_NCC_R_60(x(i, 1), y(i, 1)),
I_NCC_R_240(x(i, 1), y(i, 1)), I_NCC_R_480(x(i, 1), y(i, 1)));
                max_valj = find_max(I_NCC_L_120(x(j, 1), y(j, 1)), I_NCC_L_60(x(j, 1), y(j,
1)), I_NCC_L_240(x(j, 1), y(j, 1)), I_NCC_L_480(x(j, 1), y(j, 1)),...
                I_NCC_R_120(x(j, 1), y(j, 1)), I_NCC_R_60(x(j, 1), y(j, 1)),
I_NCC_R_240(x(j, 1), y(j, 1)), I_NCC_R_480(x(j, 1), y(j, 1)));
                if(max_vali < max_valj)
                    final_120(i, 1) = 0;
                end
            end
        end
    end

    figure(index),
    imshow(I);
    hold on;
    for i=1:length(x),
        if(final_120(i, 1) == 1)
            plot(y(i, 1),x(i,1),'r*');
            points(y(i, 1),x(i,1)) = 1;
            max_val = find_max(I_NCC_L_120(x(i, 1), y(i, 1)), I_NCC_L_60(x(i, 1), y(i,
1)), I_NCC_L_240(x(i, 1), y(i, 1)), I_NCC_L_480(x(i, 1), y(i, 1)),...
                I_NCC_R_120(x(i, 1), y(i, 1)), I_NCC_R_60(x(i, 1), y(i, 1)),
I_NCC_R_240(x(i, 1), y(i, 1)), I_NCC_R_480(x(i, 1), y(i, 1)));
            if((max_val == max(I_NCC_L_120(x(i, 1), y(i, 1)))) || (max_val ==
max(I_NCC_R_120(x(i, 1), y(i, 1)))));
                rectangle('Position',[(y(i,1) - 60) (x(i,1) - 60) 120 120],'EdgeColor','b');
            elseif((max_val == max(I_NCC_L_60(x(i, 1), y(i, 1)))) || (max_val ==
max(I_NCC_R_60(x(i, 1), y(i, 1)))));
                rectangle('Position',[(y(i,1) - 30) (x(i,1) - 30) 60 60],'EdgeColor','b');
            elseif((max_val == max(I_NCC_L_240(x(i, 1), y(i, 1)))) || (max_val ==
max(I_NCC_R_240(x(i, 1), y(i, 1)))));
                rectangle('Position',[(y(i,1) - 120) (x(i,1) - 120) 240 240],'EdgeColor','b');
            elseif((max_val == max(I_NCC_L_480(x(i, 1), y(i, 1)))) || (max_val ==
max(I_NCC_R_480(x(i, 1), y(i, 1)))));
                rectangle('Position',[(y(i,1) - 240) (x(i,1) - 240) 480 480],'EdgeColor','b');
            end
        end
    end
    saveas(index, sprintf('withright%d_1size_2temp_%dthresh.jpg',index, thresh));
end
```

B3: find_max.m

```
function [ max7 ] = find_max( a, b, c, d, e, f, g, h )
%UNTITLED3 Summary of this function goes here
  max1 = max(a, b);
  max2 = max(c, d);
  max3 = max(e, f);
  max4 = max(g, h);
  max5 = max(max1, max2);
  max6 = max(max3, max4);
  max7 = max(max5, max6);

end
```

B4: making_morphology_templates

```
%This code shows how the templates with morphology were used

% All of the different posibilities in using the strel function (which creates
% morphological structuring element)below.
% Some of these options below cannot be used because some use nonflat
% structuring Elements
% this code only makes templates of size 240x240

% SE = strel('arbitrary', NHOOD)
% SE = strel('arbitrary', NHOOD, HEIGHT)
% SE = strel('ball', R, H, N)
% SE = strel('diamond', R)
% SE = strel('disk', R, N)
% SE = strel('line', LEN, DEG)
% SE = strel('octagon', R)
% SE = strel('pair', OFFSET)
% SE = strel('periodicline', P, V)
% SE = strel('rectangle', MN)
% SE = strel('square', W)

 left = im2double(imread('left_template_edge_240.jpg'));
 right = im2double(imread('right_template_edge_240.jpg'));
close all;


%
% left = im2double(imread('left_template_edge_240_solo_1.jpg'));
% right = im2double(imread('right_template_edge_240_solo_1.jpg'));
R = 10; H = 10; N = 10; NHOOD = 10;
 LEN = 10; DEG = 10; OFFSET = 10; P = 10;
 V = 10; MN = [10 10]; W = 10; HEIGHT = 10;

 SE1 = strel('arbitrary', NHOOD);
 SE2 = strel('diamond', R);
 SE3 = strel('disk', R, 8);
 SE4 = strel('line', LEN, DEG);
 SE5 = strel('octagon', 12);
 SE6 = strel('rectangle', MN);
 SE7 = strel('square', W);

eroded_left1     = imerode(left,    SE1 );
eroded_left2     = imerode(left,    SE2 );
eroded_left3     = imerode(left,    SE3 );
eroded_left4     = imerode(left,    SE4 );
```

```
eroded_left5    = imerode(left,    SE5 );
eroded_left6    = imerode(left,    SE6 );
eroded_left7    = imerode(left,    SE7 );


eroded_right1   =imerode(right,    SE1 );
eroded_right2   =imerode(right,    SE2 );
eroded_right3   =imerode(right,    SE3 );
eroded_right4   =imerode(right,    SE4 );
eroded_right5   =imerode(right,    SE5 );
eroded_right6   =imerode(right,    SE6 );
eroded_right7   =imerode(right,    SE7 );

dilate_left1    =imdilate( left,   SE1 );
dilate_left2    =imdilate( left,   SE2 );
dilate_left3    =imdilate( left,   SE3 );
dilate_left4    =imdilate( left,   SE4 );
dilate_left5    =imdilate( left,   SE5 );
dilate_left6    =imdilate( left,   SE6 );
dilate_left7    =imdilate( left,   SE7 );

dilate_right1   =imdilate( right,  SE1 );
dilate_right2   =imdilate( right,  SE2 );
dilate_right3   =imdilate( right,  SE3 );
dilate_right4   =imdilate( right,  SE4 );
dilate_right5   =imdilate( right,  SE5 );
dilate_right6   =imdilate( right,  SE6 );
dilate_right7   =imdilate( right,  SE7 );

index = 1;
figure(index)
imshow(dilate_left1)
saveas(index, sprintf('left_template_combo_240_arbritary.jpg' ));
index = index + 1;

figure(index)
imshow(dilate_left2)
saveas(index, sprintf('left_template_combo_240_diamond.jpg' ));
index = index + 1;

figure(index)
imshow(dilate_left3)
saveas(index, sprintf('left_template_combo_240_disk.jpg' ));
index = index + 1;

figure(index)
```

```
imshow(dilate_left4)
saveas(index, sprintf('left_template_combo_240_line.jpg' ));
index = index + 1;

figure(index)
imshow(dilate_left5)
saveas(index, sprintf('left_template_combo_240_octagon.jpg' ));
index = index + 1;

figure(index)
imshow(dilate_left6)
saveas(index, sprintf('left_template_combo_240_rectangle.jpg' ));
index = index + 1;

figure(index)
imshow(dilate_left7)
saveas(index, sprintf('left_template_combo_240_square.jpg' ));
index = index + 1;

figure(index)
imshow(dilate_right1)
saveas(index, sprintf('right_template_combo_240_arbritary.jpg' ));
index = index + 1;

figure(index)
imshow(dilate_right2)
saveas(index, sprintf('right_template_combo_240_diamond.jpg' ));
index = index + 1;

figure(index)
imshow(dilate_right3)
saveas(index, sprintf('right_template_combo_240_disk.jpg' ));
index = index + 1;

figure(index)
imshow(dilate_right4)
saveas(index, sprintf('right_template_combo_240_line.jpg' ));
index = index + 1;

figure(index)
imshow(dilate_right5)
saveas(index, sprintf('right_template_combo_240_octagon.jpg' ));
index = index + 1;

figure(index)
imshow(dilate_right6)
```

```
saveas(index, sprintf('right_template_combo_240_rectangle.jpg' ));
index = index + 1;

figure(index)
imshow(dilate_right7)
saveas(index, sprintf('right_template_combo_240_square.jpg' ));
index = index + 1;
```

B5: template_matching_testing_morphology

```
function [] = template_matching_testing_morphology( thresh , num)
% this is the code that tested various templates that used various kinds of
% image morphology
%using the first template_matching.m
%function [I_SSD,I_NCC,Idata]=template_matching(T,I,IdataIn)
%http://www.mathworks.com/matlabcentral/fileexchange/24925-fastrobust-template-
matching
close all;
% thresh = 0.75;
% num = 1;
x_max = 480;
y_max = 640;
points = zeros(x_max, y_max);
[~,data, ~] = xlsread('list_of_images.xlsx');
[M,~] = size(data);

if(num == 1)
   T1_left_120 =
rgb2gray(im2double(imread('left_template_combo_120_arbritary.jpg')));
   T1_right_120 =
rgb2gray(im2double(imread('right_template_combo_120_arbritary.jpg')));
   T1_left_480 =
rgb2gray(im2double(imread('left_template_combo_480_arbritary.jpg')));
   T1_right_480 =
rgb2gray(im2double(imread('right_template_combo_480_arbritary.jpg')));
   T1_left_240 =
rgb2gray(im2double(imread('left_template_combo_240_arbritary.jpg')));
   T1_right_240 =
rgb2gray(im2double(imread('right_template_combo_240_arbritary.jpg')));
elseif(num == 2)
   T1_left_120 =
rgb2gray(im2double(imread('left_template_combo_120_diamond.jpg')));
   T1_right_120 =
rgb2gray(im2double(imread('right_template_combo_120_diamond.jpg')));
   T1_left_480 =
rgb2gray(im2double(imread('left_template_combo_480_diamond.jpg')));
   T1_right_480 =
rgb2gray(im2double(imread('right_template_combo_480_diamond.jpg')));
   T1_left_240 =
rgb2gray(im2double(imread('left_template_combo_240_diamond.jpg')));
   T1_right_240 =
rgb2gray(im2double(imread('right_template_combo_240_diamond.jpg')));
elseif(num == 3)
   T1_left_120 = rgb2gray(im2double(imread('left_template_combo_120_disk.jpg')));
```

```
    T1_right_120 = rgb2gray(im2double(imread('right_template_combo_120_disk.jpg')));
    T1_left_480 = rgb2gray(im2double(imread('left_template_combo_480_disk.jpg')));
    T1_right_480 = rgb2gray(im2double(imread('right_template_combo_480_disk.jpg')));
    T1_left_240 = rgb2gray(im2double(imread('left_template_combo_240_disk.jpg')));
    T1_right_240 = rgb2gray(im2double(imread('right_template_combo_240_disk.jpg')));
elseif(num == 4)
    T1_left_120 = rgb2gray(im2double(imread('left_template_combo_120_line.jpg')));
    T1_right_120 = rgb2gray(im2double(imread('right_template_combo_120_line.jpg')));
    T1_left_480 = rgb2gray(im2double(imread('left_template_combo_480_line.jpg')));
    T1_right_480 = rgb2gray(im2double(imread('right_template_combo_480_line.jpg')));
    T1_left_240 = rgb2gray(im2double(imread('left_template_combo_240_line.jpg')));
    T1_right_240 = rgb2gray(im2double(imread('right_template_combo_240_line.jpg')));
elseif(num == 5)
    T1_left_120 = rgb2gray(im2double(imread('left_template_combo_120_octagon.jpg')));
    T1_right_120 =
rgb2gray(im2double(imread('right_template_combo_120_octagon.jpg')));
    T1_left_480 = rgb2gray(im2double(imread('left_template_combo_480_octagon.jpg')));
    T1_right_480 =
rgb2gray(im2double(imread('right_template_combo_480_octagon.jpg')));
    T1_left_240 = rgb2gray(im2double(imread('left_template_combo_240_octagon.jpg')));
    T1_right_240 =
rgb2gray(im2double(imread('right_template_combo_240_octagon.jpg')));
elseif(num == 6)
    T1_left_120 =
rgb2gray(im2double(imread('left_template_combo_120_rectangle.jpg')));
    T1_right_120 =
rgb2gray(im2double(imread('right_template_combo_120_rectangle.jpg')));
    T1_left_480 =
rgb2gray(im2double(imread('left_template_combo_480_rectangle.jpg')));
    T1_right_480 =
rgb2gray(im2double(imread('right_template_combo_480_rectangle.jpg')));
    T1_left_240 =
rgb2gray(im2double(imread('left_template_combo_240_rectangle.jpg')));
    T1_right_240 =
rgb2gray(im2double(imread('right_template_combo_240_rectangle.jpg')));
elseif(num == 7)
    T1_left_120 = rgb2gray(im2double(imread('left_template_combo_120_square.jpg')));
    T1_right_120 =
rgb2gray(im2double(imread('right_template_combo_120_square.jpg')));
    T1_left_480 = rgb2gray(im2double(imread('left_template_combo_480_square.jpg')));
    T1_right_480 =
rgb2gray(im2double(imread('right_template_combo_480_square.jpg')));
    T1_left_240 = rgb2gray(im2double(imread('left_template_combo_240_square.jpg')));
    T1_right_240 =
rgb2gray(im2double(imread('right_template_combo_240_square.jpg')));
end
```

```
for index = 1:M
   I = rgb2gray(im2double(imresize(imread(data{index,1}), [x_max y_max])));
   I   =  edge(   I   ,'roberts'  );
   I = 255 * uint8(I);
   [~, I_NCC_L_120]=template_matching(T1_left_120,I);
   [~, I_NCC_R_120]=template_matching(T1_right_120,I);
   [~, I_NCC_L_480]=template_matching(T1_left_480,I);
   [~, I_NCC_R_480]=template_matching(T1_right_480,I);
   [~, I_NCC_L_240]=template_matching(T1_left_240,I);
   [~, I_NCC_R_240]=template_matching(T1_right_240, I);

   [x,y]=find((I_NCC_L_120 >= thresh) | (I_NCC_L_240 >= thresh) | ((I_NCC_L_480)
>= thresh)...
          | (I_NCC_R_120 >= thresh) | (I_NCC_R_240 >= thresh) | ((I_NCC_R_480) >=
thresh));

      final_120 = ones(size(x));

   radius = 120;
   for i=1:length(x),
      for j=1:length(x),
         distance = sqrt((x(i,1) - x(j,1)) .^ 2 + (y(i,1) - (y(j,1))) .^ 2);
         if distance <= radius,
            max_vali = find_max(I_NCC_L_120(x(i, 1), y(i, 1)), I_NCC_L_120(x(i, 1),
y(i, 1)), I_NCC_L_240(x(i, 1), y(i, 1)), I_NCC_L_480(x(i, 1), y(i, 1)),...
            I_NCC_R_120(x(i, 1), y(i, 1)), I_NCC_R_120(x(i, 1), y(i, 1)),
I_NCC_R_240(x(i, 1), y(i, 1)), I_NCC_R_480(x(i, 1), y(i, 1)));
            max_valj = find_max(I_NCC_L_120(x(j, 1), y(j, 1)), I_NCC_L_120(x(j, 1),
y(j, 1)), I_NCC_L_240(x(j, 1), y(j, 1)), I_NCC_L_480(x(j, 1), y(j, 1)),...
            I_NCC_R_120(x(j, 1), y(j, 1)), I_NCC_R_120(x(j, 1), y(j, 1)),
I_NCC_R_240(x(j, 1), y(j, 1)), I_NCC_R_480(x(j, 1), y(j, 1)));
            if(max_vali < max_valj)
               final_120(i, 1) = 0;
            end
         end
      end
   end

   figure(index),
   imshow(I);
   hold on;
   for i=1:length(x),
      if(final_120(i, 1) == 1)
         plot(y(i, 1),x(i,1),'r*');
         points(y(i, 1),x(i,1)) = 1;
```

```
        max_val = find_max(I_NCC_L_120(x(i, 1), y(i, 1)), I_NCC_L_120(x(i, 1), y(i,
1)), I_NCC_L_240(x(i, 1), y(i, 1)), I_NCC_L_480(x(i, 1), y(i, 1)),...
            I_NCC_R_120(x(i, 1), y(i, 1)), I_NCC_R_120(x(i, 1), y(i, 1)),
I_NCC_R_240(x(i, 1), y(i, 1)), I_NCC_R_480(x(i, 1), y(i, 1)));
        if((max_val == max(I_NCC_L_120(x(i, 1), y(i, 1)))) || (max_val ==
max(I_NCC_R_120(x(i, 1), y(i, 1)))));
            rectangle('Position',[(y(i,1) - 60) (x(i,1) - 60) 120 120],'EdgeColor','b');
        elseif((max_val == max(I_NCC_L_240(x(i, 1), y(i, 1)))) || (max_val ==
max(I_NCC_R_240(x(i, 1), y(i, 1)))));
            rectangle('Position',[(y(i,1) - 120) (x(i,1) - 120) 240 240],'EdgeColor','b');
        elseif((max_val == max(I_NCC_L_480(x(i, 1), y(i, 1)))) || (max_val ==
max(I_NCC_R_480(x(i, 1), y(i, 1)))));
            rectangle('Position',[(y(i,1) - 240) (x(i,1) - 240) 480 480],'EdgeColor','b');
        end
      end
   end
   saveas(index, sprintf('edge%d_1size_3temp_%dthresh.jpg',index, thresh));
end
```

g

Appendix C: Viola-Jones Algorithm

C1: detecting_birds.m

```
% Create a detector object.
% xml created from the training code CascadeTrainGUI.m from
% http://www.mathworks.com/matlabcentral/fileexchange/39627-cascade-training-gui--specify-ground-truth
birdDetector =
vision.CascadeObjectDetector('gray_marking_everything_haar_NSF7_0.995TPR_25stages_0.6FAR.xml');

%go to the directory where the test images in grayscale
cd('gray_test_images');

%the excel file has a list of all the test images in grayscale
[~,data, ~] = xlsread('gray_test_images.xlsx');
[M,~] = size(data);
index = 1;


for index_image = 1:M
   close all;
   I = (imread(data{index_image,1}));
   %Detect bird.
   bboxes = step(birdDetector, I);

   %Annotate detected birds.
   IBirds = insertObjectAnnotation(I, 'rectangle', bboxes, 'bird');

   figure(index), imshow(IBirds), title('Detected birds');

   saveas(index,     sprintf('testing_%d_188.jpg', index_image));
end
disp('finished');
cd('..');
```