Project in Computer Vision

Bird localization and activity score

Authors:
Johan Rodhe
Gustaf Oxenstierna

Supervisor:
Mikael Nilsson

December 10, 2017

Abstract

Analysing the behaviour of birds when exposed to different magnetic fields would give insight as to their migratory patterns. To enable quantitative analysis of the movement of birds in their cages we use a combination of techniques from computer vision and machine learning to automatically track their location and by extension, their behaviour.

For videos recorded by a thermal camera the results were promising with regards to accuracy. Real time evaluation was not achieved. For videos recorded by a regular RGB web camera, the results were not as good; accuracy was poor and the runtime per frame was high. Usually the classifier would locate objects of similar colour of the bird but with different shape.

Larger activity spikes are detected by the system, such as flight. However, activity in place, such as the bird moving its head is not accurately detected.

1 Introduction

In this project we aim to develop software for locating birds in images taken from a video camera. The video data was provided by a research project at OriLAB. The idea is that by being able to determine the position of the birds the activity can then be inferred by considering positions in previous frames. The data available is four different videos. Two videos recorded with a thermal camera, one in daylight and one at night. Two videos recorded by a regular rgb camera, one in daylight and one at night. Initially, we use the thermal imaging footage to train and evaluate a classifier. Then we do the same for the rgb footage and compare the results.

2 Theory/Method

We train a classifier using logistic regression. To develop a model a set of training data is required. The first step is therefore to create the training data for the classifier. Video's of the birds were converted to images. Then we created a script for manually clicking the image to locate the location of the birds and also three negative examples per cage. All in all this gave us four positive and twelve negative data points per image. A training set of

102 images was created using the script. This means that the size of our training set is $102 \cdot 16 = 1632$.

Having fitted a logistic model to our training data, we can now use this model to predict new data that was not previously seen. This is done using the logistic function where the input is a large vector. Let β_0 denote the constant term in the logistic model and $\boldsymbol{\beta}$ denote the weights of the model. The input vector, $\boldsymbol{x}$, is a large vector containing $C \times S \times S$ elements where C is the number of channels in the image and S denotes the patch size. S is chosen to be an odd number since we are looking for the centre pixel in a given patch. Thus, the logistic function is given by the following.

$$F(\boldsymbol{x}) = \frac{1}{1 + e^{-(\beta_0 + \boldsymbol{\beta} \cdot \boldsymbol{x})}} \text{ where } \boldsymbol{x}, \boldsymbol{\beta} \in \mathbb{R}^{C \times S \times S} \tag{1}$$

Where $\boldsymbol{\beta} \cdot \boldsymbol{x}$ denotes the dot product between the vectors $\boldsymbol{x}$ and $\boldsymbol{\beta}$. Since the dot product between two vectors of equal size returns a scalar; coupled with the fact that β_0 is a scalar, the output of equation 1 must be a scalar in the interval $[0, 1]$.

In practice, the $\boldsymbol{x} \in [0, 1]$ because we normalize the input images before doing calculations to improve numeric precision.

This classifier is then evaluated on a test set that was not seen during training. An overview of the workflow can be seen in figure 1.

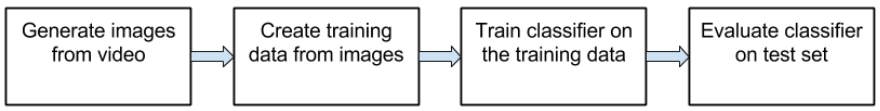

Figure 1: Overview of workflow.

Because it is known that the location of the birds in the image will be in a specific region of interest which is their cages it is not necessary to go through the pixels outside of the cages. Therefore the cages are masked out so the classifier will go through each pixel inside each cage and not the ones outside(see figure 2). In practice, we do not check every pixel inside the cage. Due to some implementation details we only check one pixel for every 32 which saves processing time; when the classifier gives a response higher

than 0.6, the step size is reduced so that more pixels are explored. This leads to increased accuracy while keeping the processing time in check. When the output of the classifier is less than 0.6, the step size is reverted to original size. The way the skipping works is by skipping pixels along the y-axis of the image. If a bird is found, the step size is reduced appropriately. The reason for the skipping along the y-axis is a Matlab implementation detail due to how it handles linear indexing. By experimentation we found the step size of 32 to be ideal for precision and speed.

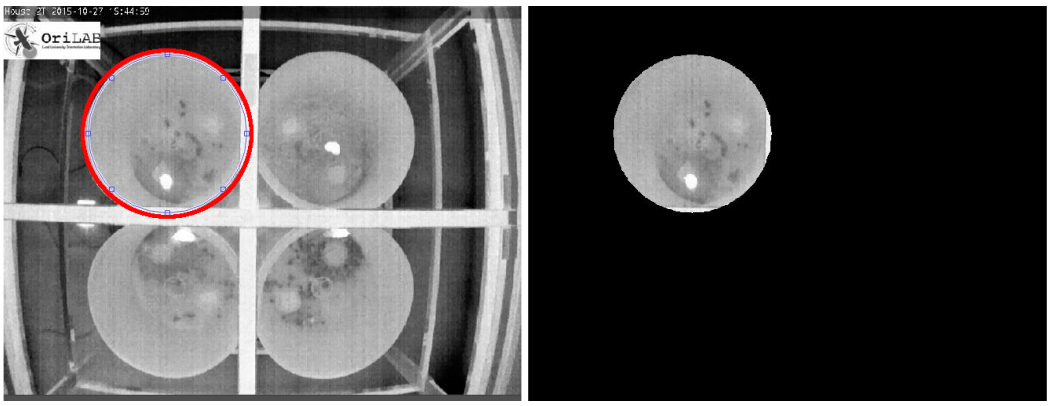

Figure 2: Shows the region of interest for the first cage.

For each pixel inside the cages the classifier will give a value in the range $[0, 1]$ corresponding to the probability of there being a bird in a $S \times S$ patch around this pixel.

From the values given by the classifier a heatmap is generated. The peak of this heatmap gives the location of the bird. Selecting the peak is done by simply picking the pixel with the highest value. Empirically, the maximum pixel value coincided well with the location of the bird.

The accuracy is evaluated by first creating a test set where we manually click on the birds true location. For this test set we annotated 19 frames giving us a total of 76 positive examples. Then the classifier is run on this set and the euclidean distance between the classifiers result and the ground truth is calculated and given as a measurement of accuracy.

2.1 Channels/Feature Extraction

2.1.1 Gradient magnitude

We use the Sobel method for computing gradient magnitude. This is a computationally cheap albeit crude approximation of the gradient magnitude. It works by applying the convolution operator using two different 3×3 kernels on the source image, thereby obtaining the approximate partial derivative in the x and y direction. The gradient magnitude is then given by taking the absolute value of the partial derivatives [1]. For an example of how it looks when applied to an image, see the right image in figure 3.

2.1.2 Gabor

First the images are converted to grayscale form. Then the gabor filter is applied. Gabor filters has been used successfully in different applications in image processing such as face recognition and texture analysis [2].

A Gabor filter is a band-pass linear filter where the impulse response is defined by a sinusoidal wave multiplied by a Gaussian function. In a 2-D Gabor filter it is a plane wave multiplied with a Gaussian function[2].

The Gabor filter in the 2-D case has the following form:

$$G_{\theta_k, f_i, \sigma_x, \sigma_y}(x, y) = exp\left(-\left[\frac{x_{\theta_k}^2}{\sigma_x^2} + \frac{y_{\theta_k}^2}{\sigma_y^2}\right]\right) \dot{cos}\left(2\pi f_i x_{\theta_k} + \phi\right) \tag{2}$$

where $x_{\theta_k} = xcos\theta_k + ysin\theta_k$, $y_{\theta_k} = ycos\theta_k - xsin\theta_k$, f_i is the frequency of the plane wave at angle θ_k with respect to the x-axis. σ_x and σ_y represent the standard deviations of the Gaussian envelope along the two axes and ϕ is the phase offset[3].

Each orientation is computed with $\theta_k = \frac{k\pi}{n}$

where $k = \{0, ..., n-1\}$.

An example of the gabor filter applied to a patch with a bird in the centre can be seen in figure 4.

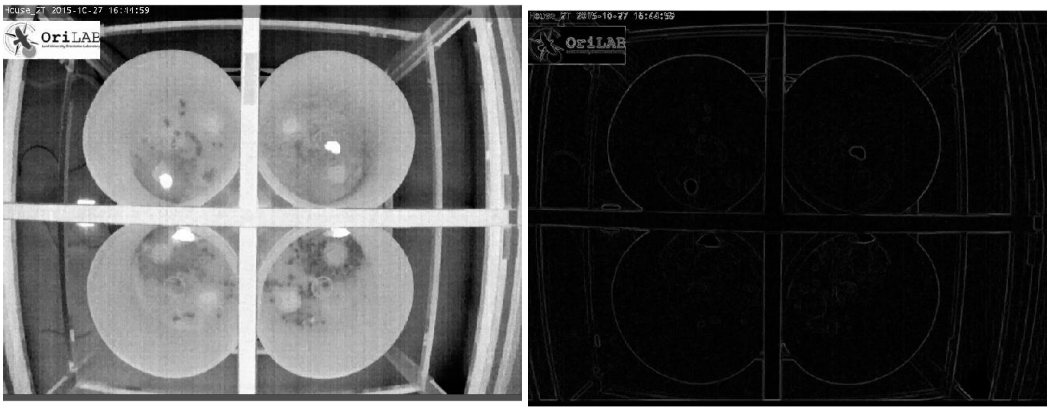

Figure 3: Left: The image in grayscale, right: gradient magnitude filter applied to the image.

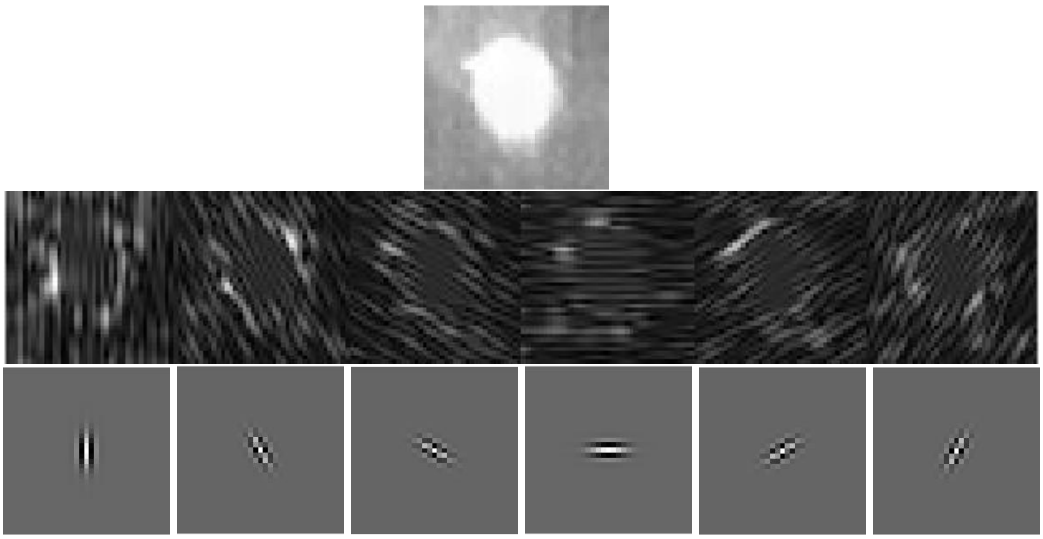

Figure 4: Original patch (top) and the result of the gabor filter (mid) with the gabor convolution kernel of each filter (bottom).

2.2 Activity

To measure the activity we calculate how much the position of each of the bird changes each frame. A spike in distance will imply that the bird is active.

3 Results

GM = Gradient Magnitude, IM = image in grayscale, GBR(d) = Gabor filter where d is number of directions. For the RGB results, GS refers to grayscale transformation of the image.

The error is measured by calculating the euclidean distance from the point which we manually tagged as the birds location to the result from the classifier. An illustration of this can be seen in figure 5.

Figure 5: An illustration of the result of the classifier. The blue cross is the manually tagged point and the red is the result.

3.1 Thermal

$S \times S$	C	Channels	Mean error	Max error	Runtime/frame
41x41	6	GM, IM, GBR(4)	3.94px	8.20px	1.47s
31x31	8	GM, IM, GBR(6)	3.95px	8.20px	1.55s
31x31	6	GM, IM, GBR(4)	3.86px	8.20px	1.18s
27x27	8	GM, IM, GBR(6)	3.98px	8.20px	1.41s
27x27	6	GM, IM, GBR(4)	3.90px	8.20px	1.25s
19x19	8	GM, IM, GBR(4)	4.29px	8.43px	1.39s

Table 1: Classifier results for the thermal dataset.

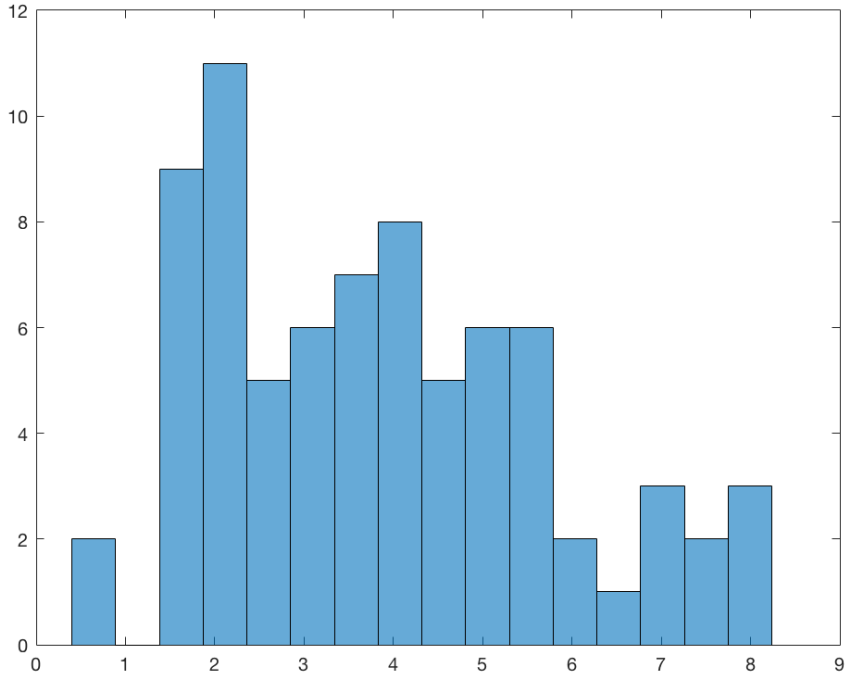

Figure 6: Histogram of the errors for the test set with $S = 31$ and $C = 6$.

The mean and max error for all four cages can be obtained from table 1. The error for each cage can be seen in figure 7 for the test set and in figure 8 for

another test set with 80 consequent images. The x-axis is the image number and the y-axis is the error. The blue line is the first cage, red second, green third and red is the fourth.

A box plot of the errors for each cage on the test set can be seen in Figure 9. The red line is the median for each box, the horizontal blue lines are the 25th and 75th percentile. The black whiskers represent the minimum and maximum data points that are not considered outliers. As we can see in the figure 9 we have no outliers for this data set [4].

A histogram showing the distribution of the errors can be seen in figure 6.

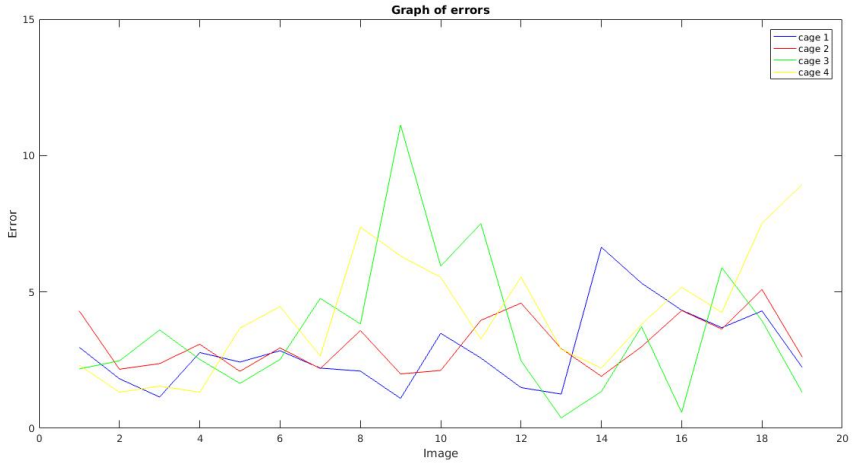

Figure 7: Graph showing the error for each cage for the test set with $S = 31$ and C=6.

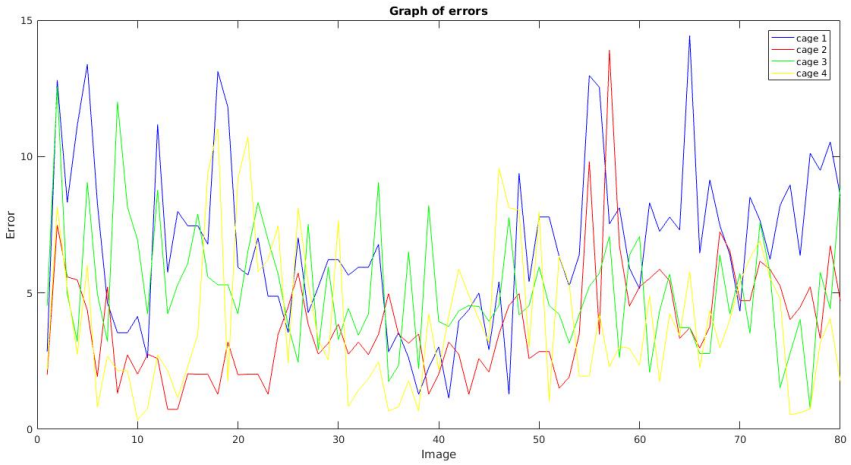

Figure 8: Graph showing the error for each cage for 80 consequent images with $S = 31$ and $C = 6$.

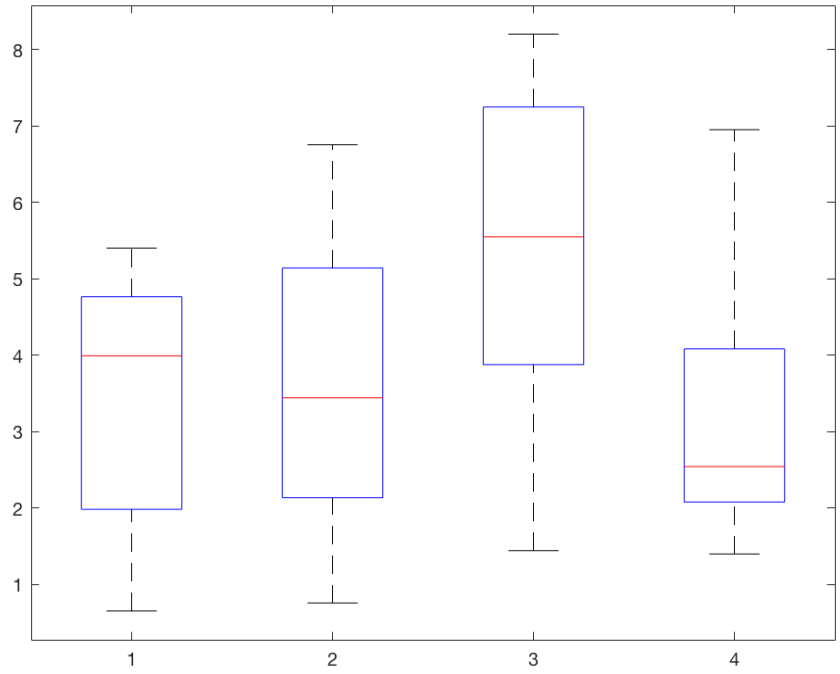

Figure 9: Box plot of the errors in each cage for the test set where $S = 31$ and $C = 6$.

3.2 RGB

$S \times S$	C	Channels	Mean error	Max error	Runtime/frame
31x31	8	GM, GS, GBR(6)	386.21px	834.96px	11.48s

Table 2: Classifier result for the RGB dataset.

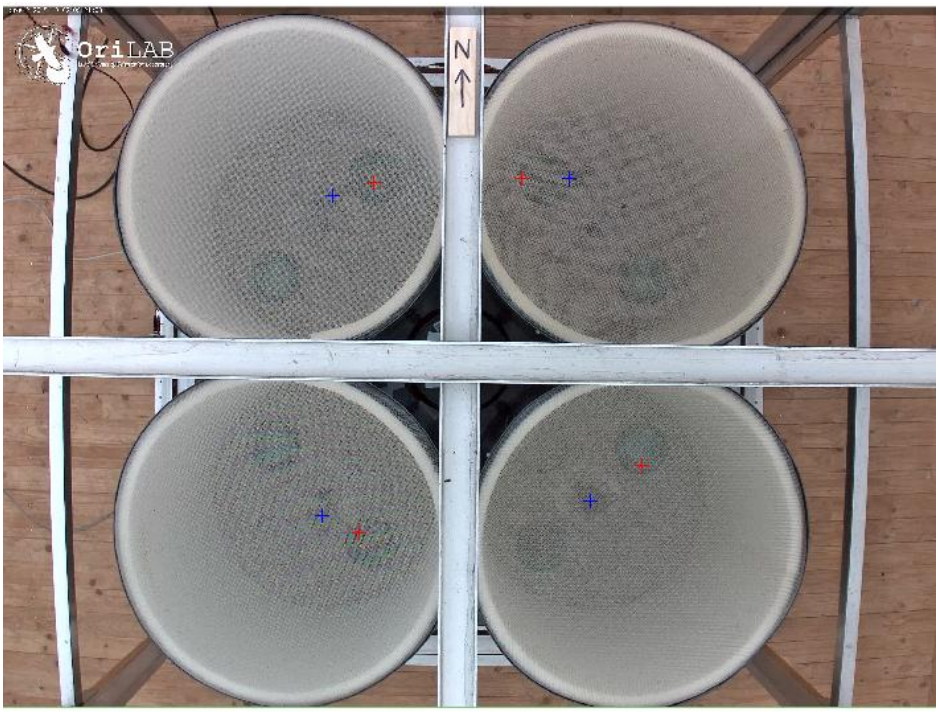

Figure 10: Sample output from the localizer together with the ground truth.

When the classifier is run on a set of consequent images the activity for each of the birds can be analysed. A result of this can be seen in figure 11. The birds in cage 1 and 2 are still so the graphs are portraying noise in the result from the classifier. Since we know that the bird around 20 pixels large we chose to interpret movement under this value as noise. The bird in cage 3 makes two flights. First early on in the image sequence and once more towards the end. The bird in cage 4 is active and makes several flights during the image sequence.

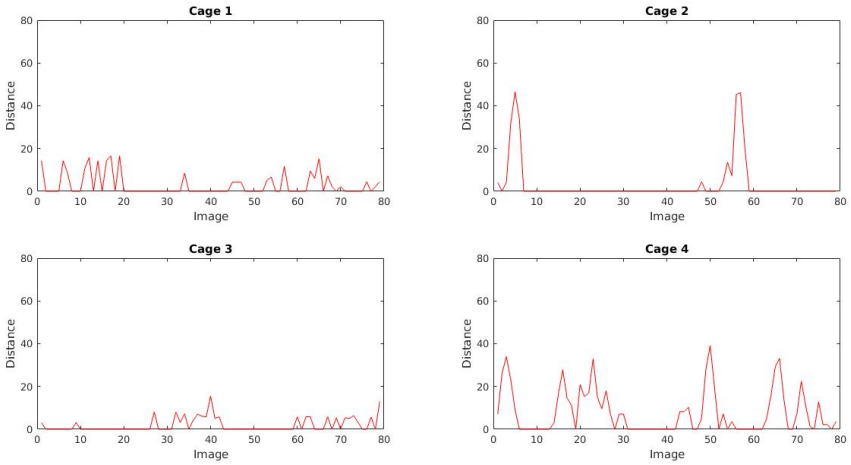

Figure 11: Plots showing a measurement of activity for the birds.

4 Discussion

4.1 Cage difference

Looking at figure 9 we see that the results differ between cages. The classifier seems to perform best on cage four and worst on cage three. An explanation for this could be that the training data for the cages differ. In the dataset the bird in cage four is moving around frequently, providing a varied training set. But the bird in cage three on the other hand is mostly stationary, giving less variation in the training data.

4.2 RGB versus Thermal

Attempts to train the classifier on RGB images show that the it needs to be tweaked for RGB to give good results. One issue is that the RGB frames are significantly larger having roughly six times more pixels which leads to increased processing time. In addition, since we trained with $S = 31$ it is very likely that these patches were too small to accurately identify the shape of the bird. As can be seen in figure 10 the classifier seems to mostly go by colour. As a result, the classifier often mistakes the water bowl for the bird. The problem with increasing the patch size is that the processing time increases quadratically. Doubling the patch size (S) quadruples the processing time.

Another caveat is that our current channel features are tuned for thermal images which have different characteristics, such as better contrast and less noise.

The result is not good, as can be observed in table 2.

4.3 Accuracy versus Runtime

Different experiments were made with the classifier using different values for S and C. The results from these experiments can be obtained from table 1. Comparing for example patch size $S_1 = 31$ and $C_1 = C_2 = 8$ and $S_2 = 19$ we see that the error is lower for the bigger value on S. S seems to have a bigger influence than C on the error. E.g, the difference in error between $S_1 = S_2 = 31$, $C_1 = 8$ and $C_2 = 6$ is $0.047px$. On the other hand, C does affect the runtime. Comparing $C_1 = 8$ and $C_2 = 6$ we see that the runtime is $0.54s$ faster per frame with C_2.

The reason that the error increases for $S = 19$ could be that the bird doesn't always fit into a patch.

5 Conclusion

The goal is to make the localisation as fast as possible while maintaining a low error. Looking at table 1 we can conclude that $S = 31$ and $C = 6$ runtime-to-error ratio.

To have any hope of computing the location in near real time we suspect the software would have to be ported to something like openCV to enable hardware acceleration. Running the classifier concurrently on the GPU should yield significant performance gains.

For the RGB images, some sort of noise removal will likely be necessary to obtain good results. Currently, the mesh above the cages pollute the image to the point where annotating them is difficult. Given the difficulty for even a human to find the bird, it seems unrealistic to expect a computer to do well given that humans are very good at recognizing patterns in images. Background / foreground segmentation might be useful to reduce the noise in the image.

For this project we have chosen to measure activity as when the birds location changes over time. But the birds could still move in place, by flapping

their wings or moving their heads for example. This could also be seen as activity but is not something our implementation will pick up. For further improvement one should first try to define what is meant by activity and try to look at when the birds move in place.

References

[1] Irwin Sobel, (2014), *An Isotropic 3x3 Image Gradient Operator*, https://www.researchgate.net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator

[2] Fogel, I. & Sagi, D. Biol. Cybern. (1989) 61: 103. doi:10.1007/BF00204594

[3] Tudor, BARBU, Institute of Computer Science, Romanian Academy, Iaşi, Romania, *Gabor Filter-Based Face Recognition Technique*, Proceedings of the Romanian Academy, Series A, vol. 11, number 3/2011, pp. 277-283. http://www.acad.ro/sectii2002/proceedings/doc2010-3/12-Barbu.pdf

[4] MathWorks, http://se.mathworks.com/help/stats/boxplot.html