MiRo Detector
Consequential Robotics

Matthew Whelan

November 27, 2017

Abstract

Deliverable D1.1, part of Work Package 1 in the InnovateUK project 103690, describes the need for the "detection of ... other robots" to be implemented in MiRo. This report details work done towards accomplishing this deliverable. Specifically, a MiRo-Detector has been built, behaving as a *perceptual filter*, which gives not only detection but also orientation, scale and confidence values for detection. To build the detector, extraction of HOG features are combined with an SVM classifier, performing simple one-vs-one discrimination for a multi-class case, in order to classify for orientation differences (left, right and back). MiRo face detection is also performed via a separate binary classifier. Detection accuracy is acceptable, and on a small test sample, achieved a rate of 100% correct classifications overall if ignoring orientation, and a total of 92% correct classification when orientation is accounted for. However, this is true only for distances of $<$ 0.7m; beyond this, MiRo can no longer successfully detect. Improvements are suggested for optimisation, through reduction in the HOG feature descriptor size in order to reduce the computational requirements in the SVM, as well as running the algorithm on lower resolution images.

1 Introduction

As part of the InnovateUK project 103690; Deliverable D1.1, part of Work Package 1, describes the need for "software for scene analysis segmentation". This deliverable includes, amongst others, development of software for the "detection of ... other robots". Although this statement has some ambiguity, it would seem sensible that the first 'robot of detection' that MiRo can perform would be the detection of other MiRo robots. This report therefore details the work conducted by the author over the work period 1st September 2017 - 30th November 2017 in order to achieve the goal of having a *MiRo-detecting perceptual filter* embedded into MiRo.

1.1 Report Outline

Chapter 2: The report begins by discussing, briefly, the history of the MiRo detector, including an introduction to the original algorithms developed as part of an MSc project. This is done for two reasons: one is that the processes used in each could, in some form, be of benefit for use in any future developmental work done on the current MiRo detector. The second is that it proves useful for comparison, highlighting the necessities of generating more than one detector for varying conditions (complexity vs speed, for instance).

Chapter 3: The report then moves onto discussing the current state of the MiRo detector, including a description of the features it extracts, the machine learning approach taken to achieve multi-class classification, and other smaller algorithms implemented to gain characteristics such as MiRo depth estimation, confidence levels of detection, and others intended to improve the speed of the overall algorithm.

Chapter 4: The results of testing are presented following the description of the algorithm. Here, measurements such as detection accuracy and computational load (processing speed and memory requirements) are given.

Chapter 5: An analysis of the test results is then discussed, which is followed by concluding remarks highlighting areas of future improvement, as well as possibilities for merging current/proposed vision strategies with the MiRo detector.

2 History of the MiRo-Detector

The development of a MiRo-Detector has gone through a number of stages, due primarily to the difficulty of the detection task, as well as the need to balance computational load with complexity and detection accuracy.

The author began their MSc project work with the intention of developing a MiRo-following strategy; that is, an ability for one MiRo to follow closely behind another MiRo. The following task itself was the simplest aspect of this project, and indeed was accomplished in a relatively short time. This was performed using colour segmentation only which, when conditions are controlled for, makes detection a trivial problem.

The difficulty lied wholly in the detection task. When MiRo is placed, as is, in an uncontrolled environment, detection becomes non-trivial. Detecting MiRo in uncontrolled environments therefore became the primary objective of the MSc project, and two solutions were found during to achieve this.

The reader is referred to the MSc dissertation for an account of the MiRo-following strategy implemented, as well as for a fuller account of the two detection algorithms (Whelan, 2017).

2.1 Solution 1 - ANN and Greyscale Histogram

After some analysis of the greyscale histograms for an image of MiRo, it became apparent that there existed two major sources of correlation, independent of MiRo's orientation in the frame. There was MiRo's white outer body and head, then there was the darker grey colour of the ears and underbelly. One of perhaps the simplest techniques for taking advantage of such correlations in data is to use the sample data to train a binary classifier, in this instance a single-layered perceptron ANN was used. Thus, discretising the histogram into 256 bins (for an 8-bit image), this would provide the feature vector used to train the perceptron, which inevitably contained 256 input neurons, with a single output. Technical details are avoided here, but can be found in (Whelan, 2017).

2.2 Solution 2 - SURF and Bayesian Estimation

Although the first algorithm worked with some success, it was essentially nothing more than a slightly more complex colour segmenter. As such, the algorithm was susceptible to failing with lighting variations and objects of similar colouring - particularly white objects would be classified as false-positives in a large number of instances.

Algorithm two was developed then to collect features that were more distinctly belonging to a MiRo. A number of the most popular feature extractors were investigated, including the corner-detecting SIFT algorithm (Lowe, 2004) and the open-source ORB algorithm, but it was the blob-detecting SURF algorithm (Bay, Tuytelaars, and Van Gool, 2006) that was ultimately chosen for extracting features.

A collection of SURF features were gathered for a number of sample MiRo images, along with a vast number of features for sample negative images. This collection of MiRo features plus sample negative features were used as evidence

sets in a Bayesian estimator. The estimator gave, therefore, a posterior probability that a set of features may belong to a MiRo.

2.3 Test Results

Testing was performed on the two detectors using the classic validation technique of sample removals; i.e. take N sample images, remove from the sample set V images which are used for validation purposes, train the algorithms using the remaining N-V images, then test the algorithms with the V validation images.

Testing under such conditions gave the results shown in Table 1. Whilst those classification rates of near 90% seem like remarkable successes, it must be noted that the validation images were closely similar (i.e. similar lighting levels, image backgrounds and overall environments). As such, the results are likely specialised cases for the operating environment of MiRo at that time.

Table 1: Comparison of the histogram/perceptron algorithm vs SURF with Bayes' inference. All times should be taken relative to one another.

Algorithm	Total time (ms) (processing + classifying)	Correctly classified (%) (false-negatives / false-positives)
Histogram/perceptron	0.271 (0.146 + 0.125)	90.1 (5.4 / 4.5)
SURF with Bayes	84.729 (0.729 + 84.0)	87.8 (11.4 / 0.8)

2.4 MiRo-Detector History: Concluding Remarks

Both algorithms did offer some success in MiRo detection, and were at the very least a step in the right direction. The issue with algorithm 1, as mentioned above, is that it was perhaps overly simplistic, and would often classify objects that were simply white. Further, it did not give a rigorous prediction of confidence levels of there being a MiRo in the same way algorithm 2 could.

Algorithm 2, on the other hand, was perhaps overly complex, and required large computational loads (memory to store every sample feature and large processing times). Hence, it was not feasible to run in real-time on MiRo.

One advantage of both however, which has just come to the authors attention, is that they are rotationally invariant. As will be shown in the upcoming chapter on the current MiRo detector, it is not robust to rotation in the same way the above approaches are. This shall be discussed further in the concluding remarks of this report.

3 Current MiRo-Detector

We turn now to the current state of the MiRo-detector. The detector, simply put, is an implementation of a human-detector algorithm developed by Dalal and Triggs, (2005). The approach they took was to build a feature descriptor using the histograms of gradients of image intensities. They then applied an SVM classifier to discriminate between images containing humans and non-humans. Their methodology was given the name *Histograms of Oriented Gradients* (HOG).

Although OpenCV contains open source libraries for computing both HOG features and training/predicting using SVMs, it is always worth considering the technical details of each first, least so that one can be more certain that each method would work under the conditions here, but also so that parameters can be set appropriately. Therefore, the following two sub-sections aim to give an overview of the inner-workings of the HOG feature extraction technique and the SVM, with particular emphasis on how the SVM has been used for multi-classification, as opposed to its standard operation as a binary classifier.

Subsequent to this, the overall algorithm implemented will be described, and will detail the fashion in which HOG and SVM are incorporated into the overall algorithm. Other, smaller sub-algorithms are run alongside HOG and SVM in order to extract information such as scale, orientation and confidence levels, as well as speeding up processing. This is all described in Section 3.3.

3.1 Histogram of Oriented Gradients (HOG)

As mentioned above, HOG was introduced by Dalal and Triggs, (2005) in order to achieve human detection. The principles of HOG will be discussed in this section, and a typical example image of MiRo will be used to guide this description, as well as proving the relevance of this approach for MiRo-detection.

Starting with a typical image of a scene that contains an image of MiRo, Figure 1. The image was captured using MiRo's own camera, at an image resolution of 320x240. In Dalal's original work, a window size of 64x128 was used as this was ideal for the dimensions of an upright human. In this case, however, a window size of 64x64 is taken to be a more suitable option considering MiRo's body shape. Figure 1 then gives an example window, aligned perfectly with MiRo's body (how this is done is described in Section 3.3). Therefore, the square region is transformed into a window size of 64x64, which is further discretised into 8x8 pixels cells. A typical cell is expanded further in Figure 1.

We start then by building a description of the 8x8 cell. Without any form of smoothing, the gradients of the cell pixels are computed using a centred mask as $[-1, 0, 1]$ for both x- and y-directions. Each cell pixel will therefore have a gradient magnitude and direction, given as,

$$M = \sqrt{g_x^2 + g_y^2} \tag{1}$$

$$\theta = \arctan \frac{g_x}{g_y} \tag{2}$$

where g_x and g_y are the gradients in the x- and y-directions for a pixel, respec-

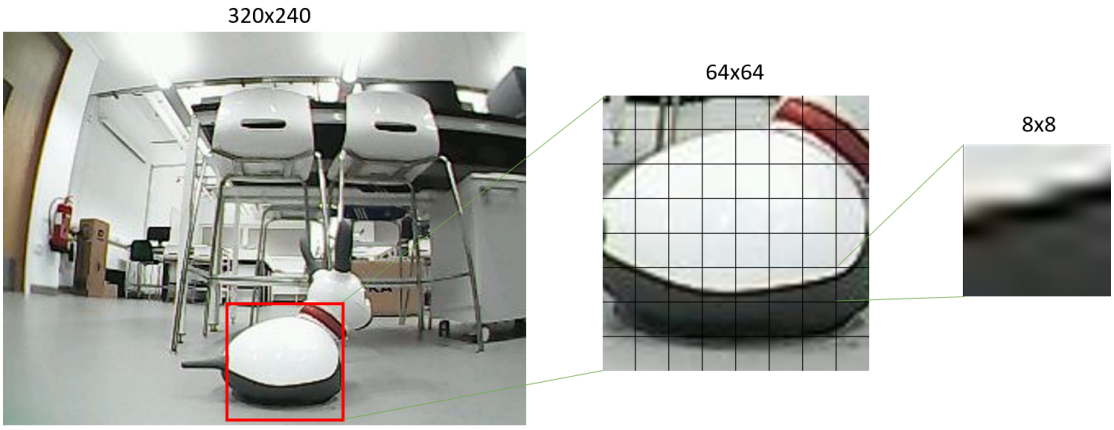

Figure 1: Typical image of a scene containing a MiRo. A square window is taken for MiRo's body, transformed to size 64x64, then discretised into 8x8 cells.

tively.[1] Given there are $8 \times 8 = 64$ pixels in each cell, there will be $64 \times 2 = 128$ values representing the gradients of a cell.

It is at this point that the histograms are introduced, as follows. Using a 9-bin histogram, with *unsigned angles*[2] corresponding to the series of angles 0, 20, 40, ... 160. This then forms the histogram set. The histogram is built by adding a gradient magnitude for a particular gradient into each bin. So, say one of the pixels in the 8x8 cell of Figure 1 has a gradient magnitude of 12 and a direction of 50^o. 50^o is midway between 40^o and 60^o, and as such 6 is placed into the 40^o bin with the remaining 6 placed into the 60^o bin. This is repeated for all 64 pixels, and the resultant histogram then builds a descriptor of size 9 for the cell. This is then repeated for all cells in the window.

The final stage is a normalisation process, done to accommodate variations in lighting effects. For this, blocks of size 16x16 are taken and slid over the image in steps of 8, thus causing a 50% overlap of each block (see Figure 2). Each block thus contains 4 cells, having a total descriptor size of $9 \times 4 = 36$. The L^2 norm is then computed for each block, and the original 36-length feature descriptor is divided by the L^2 norm to give the resultant feature descriptor for this block. For the window in Figure 2, there will be 7×7 blocks in total, each with a 32-length feature descriptor, giving the final window feature descriptor with length $36 \times 7 \times 7 = 1764$. This completes the description of how HOG features are extracted and described for a window size of 64x64.

In this way, it should now make clearer sense when we talk of "winSize", "blockSize", "cellSize", "nBins", etc. when discussing the application of HOG for the MiRo detector.

Figure 3 gives an example plot for the normalised histograms for each cell of the window in Figure 1. Note that each 'line' in each 'star' is perpendicular to its gradient direction, with the length of the line proportional to the magnitude. Notice how the gradient plots form themselves approximately around the shape of MiRo. It is these signals that the HOG descriptor uses so effectively in describing the overall shape and characteristics of an object. The most striking

[1] Alternatively, this operation on each pixel can be done beforehand for the whole image; the result is the same.

[2] I.e. an angle of $240^o = -120^o$, which is unsigned to give an angle of 120^o.

HOG characterstics for MiRo in Figure 3 seems to be the lower portion, at the point in which MiRo's white body meets its grey underbelly, then where the grey underbelly meets the ground.

Figure 2: Blocks of size 16x16 are passed over the image in steps of 8. As such, each block contains 4 of the 8x8 cells. These blocks are used to normalise the histograms of gradients computed for each cell, in order to account for possible variations in local lighting levels.

Figure 3: Cell gradients plotted for the normalised histograms. Each 'star' depicts the 9 angles of 0-180^o in the histogram, with line length proportional to the value of the angle bin.

3.2 Support Vector Machine (SVM)

The Support Vector Machine, or sometimes generalised as a kernel machine, is a powerful machine learning approach used for binary classification, proposed originally by Vapnik, (1995). Vapnik suggested that one should not go about solving a harder problem in order to solve a simpler one. That is, rather than attempting to predict class densities, $P(\mathbf{x}|C_i)$, or posteriors $P(C_i|\mathbf{x})$, it is simpler to predict only the boundary separating the classes (the C_i term above) determined by their features (the $\mathbf{x}$ term).[3]

Almost all of the mathematical details will be neglected here, but the principals of SVM will be discussed with the necessary formulas to clarify qualitative ideas, as necessary.

The term "kernel machine" refers to the fact that the machine uses a *kernel function*, which is a function of a sub-set (the support vectors, as shall be seen) of training data, used to compute the machine's output. The *discriminant*, or the dividing hyperplane, is optimised so that the margins between the support vector and the discriminant is maximised. As simple as this concept sounds, the mathematics is a little more complex.

A linear kernel is taken here, (which is the kernel type used in the MiRo detector). As such, the output, $g(\mathbf{x})$, is computed in the standard fashion,

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \epsilon \tag{3}$$

and classification is determined based on the sign of the output – since it is a binary classifier, positive sign indicates one class and negative sign indicates the other.

To find the weight vector and constant ϵ term, the SVM requires a minimisation process with a constraint,[4] necessitating the introduction of Lagrange multipliers to solve for the constrained minimisation case.[5] Whilst the derivation is not covered here, the important principle to understand after minimising is that most of the Lagrange multipliers actually equate to zero. The remaining non-zero terms are the coefficients for the Support Vectors, or those training data points lying on the margins! These then make up the

The SVM has been shown to be remarkably effective in a number of classification purposes, particularly in the work performed by Dalal and Triggs, (2005) for human detection, in which only a linear SVM was needed to achieve remarkable accuracies.

The obvious question remains, however, as to how SVM, designed as a binary classifier, can be expanded for multi-class classification. This will be the discussion for the following section.

[3]Note that this is contrary to the Bayesian estimation approach taken in (Whelan, 2017), in which posteriors were predicted. Whilst it is immediately true that the SVM performs substantially quicker than the Bayesian estimator, it awaits to be seen whether accuracy is improved upon also.

[4]A topic not worth covering here due to the mathematical investment required to do so.

[5]Note that it is a convex optimisation problem, so there is a definite max/min point.

3.2.1 Multi-Class Classification

As seen above, the original SVM is only a binary classifier at its core. This means, for instance, that although Dalal and Triggs, (2005) did achieve high classification rates, their algorithm was not able to offer information on orientation of detected humans (although it did offer scale values and confidence levels).

To accomplish multi-class classification, OpenCV's SVM class implements a one-vs-one procedure. Hence, for N classes the total number of output functions (or decision functions) is $\frac{N(N-1)}{2}$. Thus we see that as the number of classes is increased, the number of computations required increases quadratically.

For a 4-class case, as is the one taken in the MiRo detector (relating to left side, right side, back and negative), there will therefore be 6 decision functions. Upon each classification, a "winner" is declared, which is then carried forward to the next decision function. The resultant winner after all 6 decisions will be the declared class. Of course, this also requires the training of 6 separate kernel machines. In this way, therefore, should it be clear as to how SVM is applied in a multi-class scenario.

3.2.2 Confidence Levels

With the knowledge as to the workings of the SVM now in place, one obvious method for predicting confidence levels for a certain class could be to use the data point's distance away from the separating hyperplane. The margins would be at distances of -1 and 1 from the hyperplane. Thus, any data point with its absolute distance being greater than 1 could be said to be almost certain to belong to its corresponding class, whilst uncertainties arise for absolute distances below 1.

This is a rather common approach for extracting the confidence level of detection (see Dalal and Triggs, (2005) for instance), and as such is the approach taken here. It must be noted, however, that an SVM distance is not equivalent to a posterior probability, and shouldn't be taken as such. SVM wasn't designed to solve posteriors, but only aimed to separate distinct classes, and this should be kept in mind when the SVM distances are given from the MiRo detector.

Additionally, it must be considered as to how SVM distances relate to confidence levels for a multi-class classifier. If, as should be the case in the MiRo detector application, all that is needed is a binary confidence level (confidence that it is a MiRo regardless of orientation), then the SVM distance for the MiRo-left-side view vs MiRo-back view would be insufficient, since its SVM gives only an indicative confidence that the region contains MiRo's left side view vs the back view, not a confidence level that gives a general confidence level as to whether the region contains a MiRo or not. Instead, it's highly important to recognise that the operational SVM distance is that taken for the *classified-MiRo-side vs negative-regions*.

To ensure this is so, one may notice in the source code (miroDetection, 2017) that the SVM classification stage has an explicit miro-vs-negative SVM classification operation, from which the distance is taken. I.e. there is a binary classifier of MiRo-vs-negative alongside the multi-class classifier for MiRo's orientation.

With all principles for HOG features and SVM classifying in place, the next section will give the details and logic for the final MiRo-Detector.

3.3 Building the MiRo-Detector

Figure 4 displays the basic logic of the MiRo-Detection algorithm. Each major operation shall be discussed here in turn.

1. The first stage is to initialise the HOG extractor and SVM classifier. The main paramaters to note here are the following:

   - HOG:
     - winSize = 64x64
     - cellSize = 8x8
     - blockSize = 16x16
     - blockStride = 8x8 (50% overlap)
     - nBins = 9
   - SVM:
     - kernel type = Linear
     - number of classes = 4 (left=1, right=2, back=3 and negative=-1)

2. **Region of Interest (ROI) Extraction**: Exploiting the observation that MiRo's body is of a largely white make-up, ROIs are extracted by applying an adaptive threshold over the image. This is done using a minor modification of an adaptive threshold developed by Sauvola and Pietikäinen, (2000), as follows,

$$Thresh = m(img) \cdot \left[1 - k\left(\frac{\sigma(img)}{R} - 1\right)\right] \tag{4}$$

   where $m(img)$ is the mean for a given image, $\sigma(img)$ is the standard deviation for the image, $R = 128$ for an 8-bit greyscale image, and k is constant. It thus uses the image's mean and variance, in order to account for variations in lighting. The total percentage of the image thresholded is set as T_{area}.

3. **Histogram Threshold**: If more than 30% of the image has been thresholded, the image greyscale histogram is used to set the threshold so that the lightest 15% of the image is thresholded. On most occasions, however, the adaptive threshold successfully thresholds less than 15% of the image. Between 15% and 30% threshold, experimentation showed to be quicker to apply an iterative approach, by incrementing k by 0.05 until the thresholded region is less than 15%.

4. **Computing and Classifying**: After performing the thresholding operation, there will be N number of subregions in the image. For each subregion, the HOG features are extracted, followed by classification using SVM to classify for MiRo's body orientation and frontal face. The orientations and SVM confidences (distance from SVM hyperplane) are stored for all subregions, and collected into a list named *detectedZones*.

5. **Overlapping zones**: Many of the detected regions overlap. This method finds those overlapping regions, counts the detection orientations, and returns a smaller set of zones, along with the coordinates for each zone. The final output then gives the coordinates, orientation prediction, scale and confidence level for each zone.

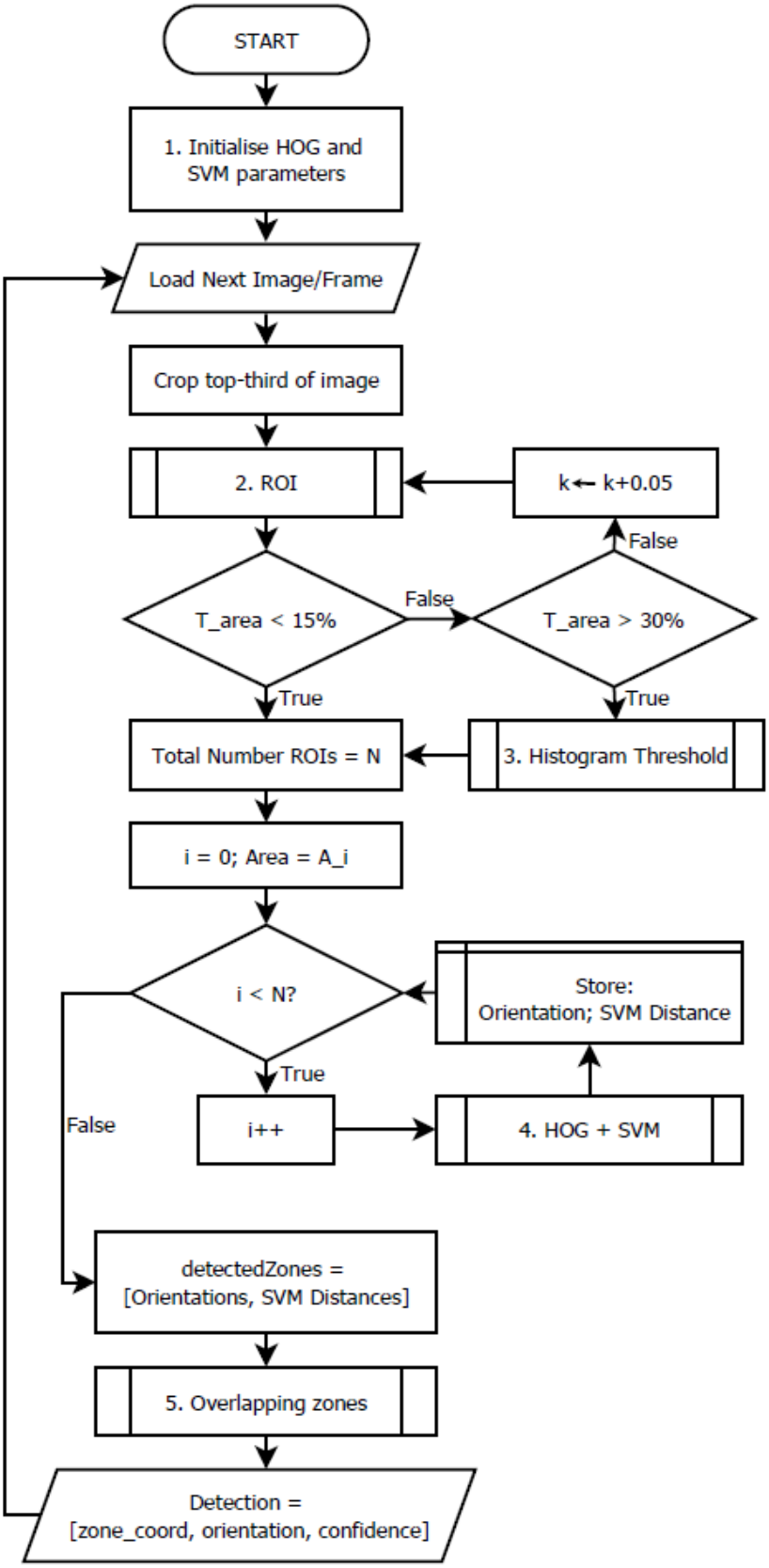

Figure 4: Algorithm logic in the form of a flowchart. The major operations, numbered 1-5, are detailed fuller on the previous page.

3.4 Distance Estimation

It is worth discussing briefly the method in which distance is estimated. The goal is to find the relationship between detected window area and distance. Geometrically speaking, the object (MiRo) does not get smaller, but rather the area it consumes within the image space reduces in proportion. I.e. MiRo's detected window area is related to the arc tangent of the viewing angle. For small distances, we can approximate this via an inverse relationship,

$$d = \frac{C}{A} \tag{5}$$

where C is a constant and A is the window area. To solve for C, a large number of experiments were performed with known distance measurements, with their corresponding detected window size stored (see Figure 5).

Unfortunately, performing regression with an inverse fit did not give a satisfactory response, suggesting the distances were not suitable for inverse approximating. Rather, a logarithmic fit seems more suitable, as shown in Figure 5. Solving the exponential for d gives,

$$d = -\frac{1}{3.6} \log\left(\frac{A}{19000}\right) \tag{6}$$

and is thus the relationship used for distance estimating.

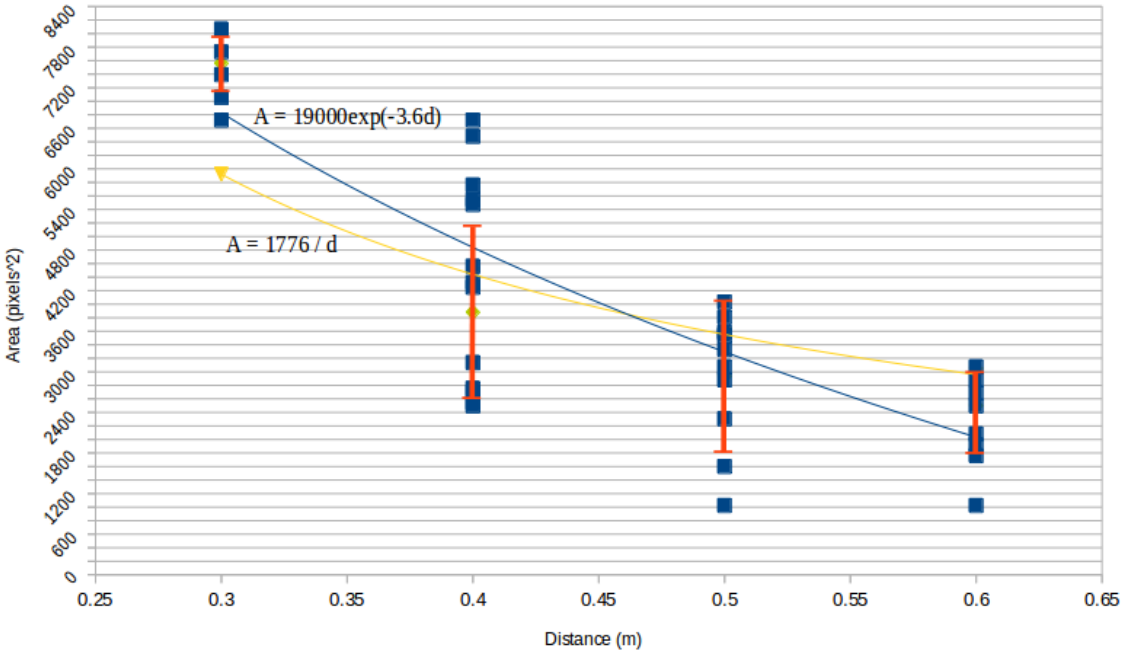

Figure 5: Relationship between detected window area and distance taken from experiment. The blue points indicate area measurements at set distances (0.3m - 0.6m in steps of 0.1m), with the red bars showing the standard deviations. Yellow plot is the result of regression fitting with an inverse relationship. Blue plot is the result of a logarithmic (or exponential) regression fitting. The logarithmic fit gives the least amount of error in the regression.

4 Testing

4.1 Methodology

Test results are designed to be indicative only, and as such testing was performed on a (relatively) small number of samples, done for varying orientations, scales and lighting.

The orientations and scales are first noted, and then the MiRo detector gives values indicating the orientation likelihood and confidence level (or SVM hyperplane distance). These results are shown in Table 2. A full discussion on these results are given in Section 5.

4.2 Results

Table 2: Indicative results for the MiRo detector in 15 example scenarios. See below for the notation descriptions.

Conditions		**Detector Predictions**	
Distance (m)	Orientation	Orientation	Confidence (SVM distance)
0.35	R	R	0.3
0.35	L	L	0.35
0.35	B	B	0
0.35	RB	R	0.25
0.35	LB	U	0.2
0.5	R	R	0.7
0.5	L	L	0.3
0.5	B	B	0.2
0.5	RB	0.66R; 0.33B	0.35
0.5	LB	0.8L; 0.2B	0.4
0.6	R	R	0.45
0.6	L	L	0.3
0.7	R	N	-
0.7	L	N	-
0.7	B	N	-

Notations:

- R – right orientation
- L – left orientation
- B – back orientation
- LB/RB – right-back/left-back orientations (i.e. 45^o on MiRo's back)
- U – unknown orientation, yet still detected
- N – no MiRo detected

5 Discussion and Conclusions

5.1 Analysis of the Results

We start the analysis by focusing on the false-negative results, which consistently show to appear when the distance between detecting MiRo and detectable MiRo is $\geq$ 0.7m, as is noticeable with the latter three results in Table 2. Upon closer inspection, it appeared that once MiRo's distance was sufficiently far that the detection window dropped below a 50x50 resolution, which occurred at a distance of around 0.7m, than detection rates would diminish. This result should not be a surprise, given that the HOG feature algorithm requires that each window be of size 64x64. For resolutions larger than 64x64, scaling down isn't a problem and all features are usually preserved. Scaling up, however, causes some issues with feature loss as, and is the most likely explanation for the failure to detect beyond a distance of 0.7m.

Between distances of 0.35m and 0.7m however, the detector gives relatively good detection results, giving correctly, most of the time, the orientation with positive confidence levels (recall these are SVM distances, not posteriors, and should not be treated therefore as probabilities). There is one exception, and that is the LB orientation at distance 0.35m, in which the orientation was unknown here. To clarify, this means MiRo is still detected, yet the orientation prediction gives equal weight to all three orientations and is thus the orientation is undecided. This can occur due to two possibilities: 1) the detector has detected an equal amount of windows for the three orientations within the detected zone, or 2) no orientations have been detected, but the separate binary classifier has detected that the zone contains a MiRo. In the case of the test results, case 2 is the reason for the unknown orientation. For distances of up to 0.7m, therefore, the detector correctly detected for all cases, and classified correctly the orientation for 11/12 examples (92%). Beyond 0.7m however, the detector failed to detect MiRo at all.

Finally, for the interested reader, 4 example videos of MiRo detection can be found in the miroDetection GitHub repository. All the source code, training images and further documentation is also stored here (miroDetection, 2017).

5.2 Running On-board (Raspberry Pi)

Initial trials were run for testing the computational load of the algorithm when running on a Raspberry Pi 3 (1.2GHz, quad-core). Included in the miroDetection package are a number of benchmarking images, all taken at a resolution of 240x320 (located in folder "Test_images"). These images are selected as testing images due to them being particularly challenging to detect on (extreme lighting and backgrounds).

When running these images on-board the Raspberry Pi, the computational load was very high, taking over a minute in some cases, with none performing within a reasonable time of at least 2s.

Unfortunately, the current state of the detector is not in an optimised state. Possible areas of exploration for optimising is suggested in the next section.

5.3 Conclusions and Further Work

The report has discussed a methodology for building a MiRo-Detector. This methodology includes extraction of HOG features and the training of an SVM classifier. Multi-class classifying has been accomplished with the SVM by performing one-vs-one classifications. The detector gives satisfactory results for a small test sample, detecting for all cases in which MiRo was at a distance less than 0.7m away, and predicting the orientation correctly for 92% of these detections. Beyond a distance of 0.7m, however, MiRo can no longer successfully detect MiRo. This is due mainly to the restriction that the HOG algorithm requires window resolutions of 64x64 or greater, and at a distance of 0.7m, the resolution drops below this requirement.

5.3.1 Optimising the Algorithm

To return to the point on optimisation, it is noticeable that the majority of the processing time occurs during the SVM classification stage. Altering the parameters of the SVM itself seems unlikely to speed up processing – its parameters are already at their most optimal with regards to speed (linear kernel). Unless, of course, one wants to remove orientation detections and move from multi-class to binary classification only.

Alternatively, reducing the size of the HOG feature descriptors seems more reasonable, which would reduce the size of the dot products required in the SVM computation. This can be done a number of ways, by increasing window/cell/block size, reducing the number of bins, or increasing the length of the block strides. Any number of these could be searched for optimality. Of course, one should keep in mind detection accuracies whilst optimising for computation loads.

Finally, one may want to run the algorithm on various image resolution sizes. The author found, for instance, that running the algorithm on a resolution of 144x192 reduced the computation times, but the SVM required retraining with new sample images taken from the reduced resolution (the resolution change, done as part of MDK update 171012, also altered the colours somewhat, adding an orange/sepia hue to the image).

5.3.2 Combining with other MiRo sub-projects

It has also come to the author's attention that the face recognition algorithm currently proposed to be implemented in the MiRo (by Daniel Camilleri) is based upon Baltrusaitis, Robinson, and Morency, (2016)'s OpenFace package. Their paper mentions that they also extract HOG features, having final feature length vectors of size 4464 (2.5x larger than those used in the MiRo detector). It may be that integrating the MiRo detector with the OpenFace package would not have a satisfying outcome due to differences in cell and window sizes (specific details of which are given in (Baltrusaitis, Robinson, and Morency, 2016)), however, MiRo detection using HOG features seemed to be working with some success, and it may be possible to merge the two detectors together to have face recognition and MiRo recognition. This is left open for further consideration.

References

Baltrusaitis, Tadas, Peter Robinson, and Louis-Philippe Morency (2016). "OpenFace: an open source facial behavior analysis toolkit". In: *IEEE Winter Conference on Applications of Computer Vision*.

Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool (2006). "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 404–417.

Dalal, N. and B. Triggs (2005). "Histograms of Oriented Gradients for Human Detection". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 1, pp. 886–893.

Lowe, David G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2, pp. 91–110.

miroDetection (2017). "miroDetection Repository". In: URL: `https://github.com/mattdoubleu/miroDetection.git`.

Sauvola, J. and M. Pietikäinen (2000). "Adaptive document image binarization". In: *Pattern Recognition* 33.2, pp. 225 –236.

Vapnik, Vladimir Naumovich (1995). *The Nature of Statistical Learning Theory*. London: Springer.

Whelan, Matthew (2017). "A Vision-Based Study on Collective Behaviour in Mammal-Like Robots". In: URL: `https://www.academia.edu/35067347/MSc_A_Vision-Based_Study_on_Collective_Behaviour_in_Mammal-like_Robots`.