Detecting humans from a top-down perspective using an unmanned aerial vehicle

Xeryus Stokkel, s2332795, x.l.x.stokkel@student.rug.nl
Supervisors: Marco Wiering*, Felipe Nascimento Martins*†

July 14, 2015

Abstract

Search and rescue is often time and labour intensive, large groups of people search vast areas for missing people. In this paper we present a system that uses a drone to make search and rescue less resource intensive. The system uses a downward facing camera on the drone to detect people in open areas. The detector uses a sliding window to extract histogram of oriented gradients features that are classified using a Linear Support Vector Machine. Several preprocessing methods and models are compared for their classification and runtime performance. We also introduce a method to dynamically determine whether positive windows are true positives by looking at how they overlap. By doing this we hope to bring down the amount of false positives when the detector is used to count the number of people in an image. Our experiments show that the developed method has good performance on classifying frames as containing persons, and is able to estimate the number of people on a frame with a mean squared error of less than 0.25. Although the detector shows great promise the current implementation is too slow to be of practical use.

1 Introduction

Finding missing persons is a time consuming and labour intensive task. Often the only known information is in which area a missing person is likely to be. The collaboration of a lot of people is needed to search the entire area. Some environments, like mountains and deserts, where people often go missing can be quite hazardous to both the missing person and rescue workers.

When the environment has few visual obstructions it is possible to reduce the amount of labour on a rescue mission by searching from the air. Doing this by plane or helicopter can be expensive. With bad weather it is sometimes impossible to ensure the safety of the rescue workers. Using an autonomous vehicle would alleviate most of the problems surrounding search and rescue.

Using an Unmanned Aerial Vehicle (UAV or drone) would be an appropriate alternative since those can cover large areas and fly over rough and hazardous terrain without the need to risk human life [14]. To reduce the labour needed the drone should be able to detect a missing person by itself and report areas of interest to rescue workers. They can then concentrate their search efforts on areas of where it is likely that they will find the missing person [10].

The UAV that was chosen to test the system that we built is the Parrot AR.Drone 2.0 because it is a cheap platform and it is readily available. This is a quadrotor helicopter that comes with two built-in cameras: a forward facing 720p HD camera and a downward facing 320x240 pixel camera. For this research the latter camera is used, because of this the obtained view has a top-down perspective.

To represent missing persons we got a few volunteers to stand or walk around in a field and recorded them using the AR.Drone 2.0 [2]. The terrain where we recorded the data is a grass field

*University of Groningen, Department of Artificial Intelligence

†Federal Institute of Education, Science and Technology of Espirito Santo (IFES), Department of Control and Automation Engineering, Campus Serra (currently on sabbatical leave)

on the university terrain as this is readily available and we are allowed to fly a drone over it.

The problem is in essence that of pedestrian detection. Previous research has mainly been done from an eye-height perspective [4, 8, 12, 13, 15] while others have used an in-flight perspective [1, 5, 11]. These in-flight perspectives are different than the top-down perspective. The in-flight perspective resembles a bird's-eye-view while the top-down perspective is more akin to satellite images [1, 6].

In this paper we describe the development of a system that is able to autonomously detect humans in images obtained from the drone. We can split this up into two sub-questions: is it possible to distinguish frames containing people from those that don't? This means that an image should be classified positively based on whether there is at least one person in it. When this is the case then we can say that there is a person below the current position of the drone. The second research question is whether this classification system can also be used to count the number of people in a frame.

The method that was used is similar to [4], the histogram of oriented gradients (HOG) algorithm they developed was used and we trained a support vector machine (SVM) [3] using the HOG features.

More details of the method that we used for detection are described in section 2, the dataset that was obtained will be described in section 3. The evaluation is presented in section 4 and the results are discussed in section 5.

2 Method

The method used is in line with Dalal and Triggs [4]. We slide a window over the image and create histogram of oriented gradients (HOG) descriptors for all the windows. These are then classified with a Linear Support Vector Machine (SVM) [3]. The software was written in MATLAB.

Although the method that is used is very similar to Dalal and Triggs, it does differ in a few points. Most of the differences derive from memory or speed constraints, while some of them also rise from the nature of our data. The latter will be explained in further detail in section 3. An important point that is made is that flight characteristics caused the data to be gathered at different flight heights and therefore we need varying sizes of windows to cover the different scales in which subjects can occur in the data.

Algorithm 2.1 Method to generate Colour HOG descriptors

Input: Image I, definition of cells C, definition of blocks B, number of bins s
Output: Feature vector F

1: $V \leftarrow$ vertical gradient of I
2: $H \leftarrow$ horizontal gradient of I
3: $A \leftarrow \text{atan}(V/H) + \frac{\pi}{2}$
4: $M \leftarrow \sqrt{V^2 + H^2}$
5: **if** I is a colour image **then**
6: **for all** $p \in$ pixels of I **do**
7: $c \leftarrow$ channel with largest value M_p
8: $b_p \leftarrow \lfloor A_{p,c} / \frac{pi}{s} \rfloor$
9: $M_p \leftarrow M_{p,c}$
10: **end for**
11: **end if**
12: **for all** $c \in$ C **do**
13: **for all** $p \in$ pixels of c **do**
14: $h_{c,b_p} \leftarrow h_{c,b_p} + M_p$
15: **end for**
16: **end for**
17: **for all** $b \in$ B **do**
18: $F_b \leftarrow \text{concat}([h_c | c \in b])$
19: $F_b \leftarrow \frac{F_b}{\|F_b\|}$
20: **end for**
21: **return** F

2.1 Histogram of oriented gradients

A brief summary of how to make HOG descriptors is given here and in Algorithm 2.1. Full details and effects of parameters of the HOG algorithm are described in [4]. An image can be described by the distribution of orientations and magnitude of gradients within it, these are obtained on line 1-4 in Algorithm 2.1. When the input image is a colour image then flatten the image to one channel by taking, per pixel, the magnitude and angle of the colour channel with the largest magnitude (this is described in lines 5 to 11 of Algorithm 2.1). The relation of the gradients to each other is the most important aspect of the image, not their exact location. Because of this we can split the image up into small spatial regions called cells of $n \times n$ pixels,

C is the set of all cells. Each cell is converted into a 1D histogram h_c of the orientation of the gradients within the cell. Each pixel gets a vote for a single bin, the weight of the vote is the same as the magnitude of the gradient of that pixel. This is done in line 12 to 16.

From these cell histograms we can build a feature vector by concatenating them. However, before we do this there is a step of contrast normalisation. Cells are collected into overlapping blocks of $n \times n$ cells which are defined in B, each element in B is a relation of the cells contained within the block. This means that each cell is part of multiple blocks. The histograms for each cell in a block are concatenated, after this the entire block is contrast normalised by L2-normalization.

2.2 Methodology

Training was done in the same manner as in Dalal and Triggs [4]. Positive labels were created by cropping people out of images and saving them as positive labels. A first round of training was done by generating HOG descriptors for the positive labels. The labelled images were scaled so that each example image is of the same size, for these resized images we create a HOG descriptor. From each negatively labelled image 10 random windows were taken, scaled to the common size and for these HOG descriptors were also made. The SVM was trained on the descriptors obtained during this step.

For a second round of training we picked 10% of the negative images at random. These images were then classified using the SVM obtained in the previous step. The descriptor of each window that has been given a positive label is collected. It is guaranteed that these windows are actually negatives since the image only contains negatives, therefore we can use these descriptors to correct the SVM on the errors that it made. These descriptors were added to the previously obtained descriptors and the combined set is used to retrain the SVM.

Finally we try to find the best c parameter for the SVM by doing five rounds of cross-validation error minimum search. The method used to find the local minima is described in [7]. Because it is not guaranteed that the found minimum is the global minimum we do five rounds of local minima search and select the c associated with the lowest cross-validation error. The final model is then obtained by retraining the SVM with the value of c that has been found.

Algorithm 2.2 Dynamically determine the threshold to distinguish positive areas from negative areas.

Input: 2D histogram of positives H
Output: threshold t

1: $m \leftarrow \max(H)$
2: $M \leftarrow$ vector of length m
3: $A_{\text{prev}} \leftarrow 0$
4: **for** $i := 1$ **to** m **do**
5: $\quad r \leftarrow$ area of $H >= i$
6: $\quad M_i \leftarrow |A_{\text{prev}} - r|$
7: $\quad A_{\text{prev}} \leftarrow r$
8: **end for**
9: $t \leftarrow$ index of $\max(M)$

Unlike [4] only 10% of the negative images are used to obtain corrective examples. This is done for memory and speed considerations. Classifying an entire image is a process that takes several minutes (up to 20 minutes under some parameters). Although this is not a large hurdle, it is still a limiting factor given the time frame of the project. A larger issue is that of memory: the SVM grows larger when it is trained with more data. This, together with the method that MATLAB uses to distribute work amongst processors means that the computer that was used to run the detector would occasionally run out of memory.

The detector has a few measures to speed it up. The stride of the window is equal to the HOG cell size, then the amount of work that needs to be done is reduced because there are fewer windows to create HOG descriptors for. After reducing the number of windows this way there still are 10888 windows per image. Another way the detector was sped up was by using a few pre-set window sizes. All windows were squares and the sizes increased from 64 to 144 pixels in steps of 16 pixels.

2.3 Detector and threshold

To decrease the amount of false positives from the under trained model a threshold function was implemented. The amount of positive windows were tracked on a per pixel basis creating a 2D histogram as shown in Figure 1. This is converted to a binary

Table 1: Parameter settings of different models. Window is the common size that windows get converted to before making a HOG descriptor. Cell, block and bin are the cell size, block size and number of bins parameters for the HOG function.

Model	Window	Cell	Block	Bins	c	Feature size
Default	64×64	8×8	2×2	9	0.7668	1764
Grey scale	64×64	8×8	2×2	9	0.7678	1764
Large window	80×80	8×8	2×2	9	0.3019	2916

image by setting a minimum number of per-pixel positives that should constitute a positive. The positive areas in the binary image are converted into rectangular bounding boxes. If any of the dimensions of a bounding box is below a minimum size then it is discarded as well. This system ensures that the occasional false positives do not influence the classification of an image.

A method of dynamically determining the threshold was implemented, the method that was used is shown in Algorithm 2.2. Every threshold level from 1 to the highest possible level in the 2D histogram is selected and the number of pixels (area) that satisfy the current threshold is calculated (line 5). The difference in areas from one level to another is stored in M as on line 6. In the end the largest change in area is selected and the corresponding threshold level is used (line 9).

This method was created because validation images would occasionally have positive windows throughout the image creating a large area up to a certain threshold. The goal of the dynamic threshold method is mainly to reduce the occurrence of these situations.

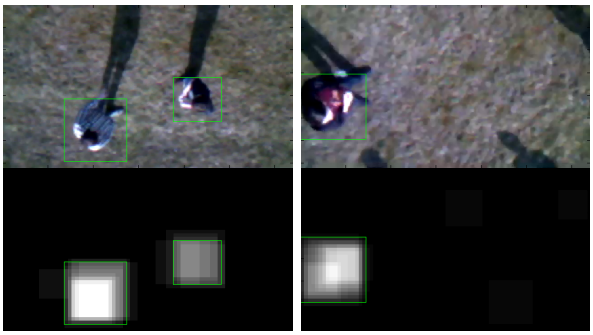


Figure 1: Images as shown in detector mode. The original image is on top while the 2D histogram is on the bottom. The green rectangles represent bounding boxes that indicate where the detector has found positives after applying a threshold to the 2D histogram.

2.4 Models

Several models with different parameters and preprocessing methods were made so that we can compare their speed and classification performance. Because the detector was quite slow it is worth investigating methods to decrease the run time.

The model that was used as a baseline does no preprocessing on images before creating HOG descriptors. It scales windows down to a common size of 64×64 pixels before creating the descriptors. It uses a cell size of 8×8 pixels, blocks are 2×2 cells large, the number of bins in the histograms is 9. This model is referred to as 'default' in the text and figures. An overview of parameter settings is given in Table 1. The second model is based on this, all its parameters are equal. It converts the image to grey scale before classification so it only differs in the preprocessing method.

There is also a third model which uses a common window size of 80×80 pixels. A window size of 64×64 is the same as the smallest examples but limbs are usually 6 to 8 pixels in size meaning that cells are at the upper end of this scale. When a common window of 80×80 pixels is used then limbs are slightly larger and a cell size of 8×8 pixels is about the ideal size to describe appendages [4]. This model is called 'large window' throughout this thesis. The optimal c value for each of the SVMs can also be found in Table 1

3 Data set

The data set was gathered specifically for this research. It consists of images that were recorded using the drone in a single day over several flight hours. During this day the weather was near ideal

to fly a drone. The weather was clear with a moderate breeze. Because of the lighting conditions objects did cast sharp shadows.

There was enough wind to cause the drone to have difficulty trying to maintain a constant flight height while moving, hovering did not pose a problem and it would stay at the same height although it would occasionally drift. This means that the drone was not always the same distance from the volunteers and they appear at different scales in the obtained images. We could correct for this manually using markers like [6], but this is a very labour intensive task and it would require the presence of markers. Instead the detector slides differently sized windows over an image and resized them to a common size. These common sized windows are then classified, this should remove the effects of the drone flying at different heights and the variance in scale of the volunteers.

Although the downward facing camera of the AR.Drone 2.0 has a frame rate of 60 frames per second, images were obtained at 5 frames per second. Instead of storing a video feed these were stored as separate still images. Because of the lower frame rate the data set doesn't include a lot of similar frames so a method like [8] would not work on this data.

3.1 Training set

The training set consists of images from the data set that have been labelled as either negative (no person in the image) or as positive (one or more people in the image). Images and their associated labels were saved, the persons in a positive image were cropped and stored as well. In total there are 148 positive examples while there are 1056 negative examples. During training the positive examples are also flipped either horizontally, vertically or both to obtain 592 positive examples. The images weren't tightly cropped but there are usually 4 pixels surrounding the person since this improves performance of the detector [4].

3.2 Test set

A final set was created to check the results of the trained models. This set contains 247 images of which just 38 include people, all other images are negatives. The distribution of positives is slightly above that in the raw data. It was decided to include about 8 extra images with people in them to be able to gather more meaningful results for counting the amount of people inside an image. Of the positive images 19 contain one person, 14 contain two people, 4 contain 3 people and just one contains 4 people, resulting in 63 people in the test set.

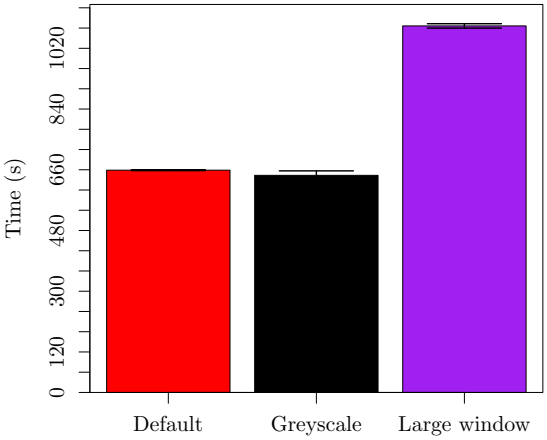
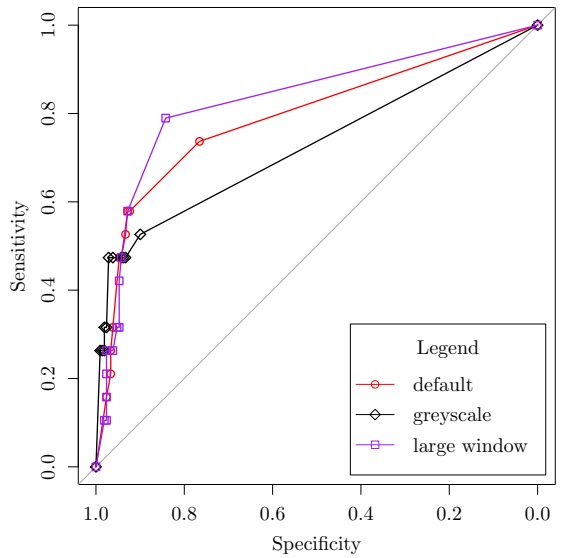


Figure 2: Comparison of run times of different methods to classify all windows in a single image. Lower is better. The error bars show standard deviations of the mean.

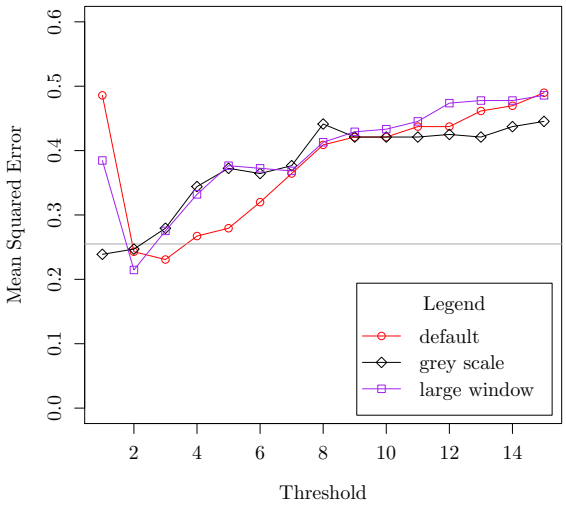
4 Results

The different preprocessing methods have different run times as shown by Figure 2. As the plot indicates there are significant differences between the running times ($F(2, 57) = 1.72 \times 10^4, p < 0.001$). Figure 2 indicates the large intermediate window has a significantly higher run time. There is also a difference between the colour HOG (default) and grey scale HOG ($t(19.39 = 5.12), p < 0.001$) with the latter being slightly faster (659 seconds and 644 seconds respectively).

Receiver Operating Characteristic (ROC) curves that show the classification performance of the different models can be found in Figure 3a, it shows



(a) ROC curves of the various models on the image classification task, AUC for the curves can be found in Table 2.



(b) Influence of threshold on the performance of the person counter. The grey horizontal line shows the naive optimal algorithm.

Figure 3: Performance of the various preprocessing methods on the image classification task (left) and the counting task (right).

Table 2: Comparison of AUC of ROC for the different models described in the text. The default model is used as a baseline and all models are compared against the default model. Therefore there are no statistics available for it.

Model	AUC	Z	p
Default	0.7919		
Grey scale	0.7285	1.3497	0.1771
Large window	0.8317	−0.8884	0.3743

Table 3: The Mean Squared Error of the dynamic threshold method for the various model types. They are compared against the naive optimal MSE of 0.255 to see whether they are lower. All models have 246 degrees of freedom.

Model	MSE	t stat.	p
Default	0.231	−0.55	0.29
Grey scale	0.255	0	0.501
Large window	0.271	0.31	0.621

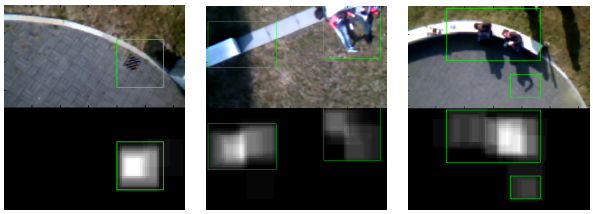
that the grey scale model has the worst performance while the large window performs the best. The default model is used as a baseline. Comparing the Area Under the Curve (AUC) of the other models to the default model yields no significant differences, as is shown in Table 2.

In total there are 63 people in the test. Guessing the number of people in the image would lead to an expected error of $\frac{63}{247} = 0.255$. The models can be used to count the number of people by counting the number of bounding boxes after applying a threshold to the 2D histogram. The mean squared errors (MSE) of this method are shown in Figure 3b. It shows the MSE at varying thresholds. The horizontal line in Figure 3b shows the expected error a naive optimal algorithm that uses the known distribution of people in the test set of 0.255. The MSE for the dynamic threshold are not shown since for most images it is equal to the MSE of a threshold of 3, this was the minimum threshold value that the dynamic threshold method uses.

Only a few points are under the 0.255 line and none of them are significantly under the line. The default model with a threshold of 3 has MSE 0.231

which is not significantly lower than 0.255 ($t(246 = -0.55, p = 0.29)$. The lowest grey scale MSE is at a threshold of 1, its value is 0.239 which is also not significantly lower than 0.255 ($t(246) = -0.41, p = 0.34$). The large window model has the lowest MSE at a threshold level of 2, but this also isn't significant ($t(246) = -0.99, p = 0.162$). The dynamic threshold method does not do any better on the counting task as can be seen in Table 3.

Figure 4: Some examples of common false positives, including drains, large concrete blocks and parts of shadows.



5 Discussion

The run time of the detector is too high to serve a practical purpose: the mean time to classify an image for the fastest detector was 644 seconds, while the slowest detector had a mean run time of 1086 seconds (see Figure 2). This means that the detectors are too slow to process a video feed in real time. It is not uncommon that the computation requirements for search and rescue UAVs are high [14]. The process could be sped up by using parallel processors but to achieve a decent speed up a lot of processors are necessary. This solution might prove to be a too large investment for real world search and rescue applications.

There are two factors that cause the run time to be so high. The first is that the current implementation is written in MATLAB, this is a very slow language when compared to other mainstream languages like C or Java. Using one of these to implement our method is likely to give a decent speed-up. The other factor is that we are doing a lot of calculations multiple times per pixel. The window slides over the image and we calculate the gradients separately for each window. Calculating the gradients is what the HOG algorithm spends the majority of time on. This means that the gradient of a pixel needs to be recalculated for each window that it is part of. A faster implementation would be to first calculate the gradients of the entire image and slide the window over the result. We suggest that future work primarily looks into speeding up the method this way.

When counting the number of people inside images the models perform worse than the naive optimal method for most thresholds. All models do perform better than naive optimal at a threshold level of 2. Both the default and grey scale method perform better than the naive optimal method at 2 different levels. The error of the detectors at these thresholds aren't significantly lower than the naive optimal level. The dynamic threshold method uses a minimum threshold of 3, because of this the error of the dynamic threshold (see Table 3) is always higher than the naive optimal level because most models have poorer performance at these levels. The naive optimal is only valid for this test set as the distribution of people in the images is known. In real world applications or other tests set the naive optimal would not be applicable. The dynamic threshold method is very close to the naive optimal as there are no significant differences between them. The dynamic threshold method thus prevents over fitting to the test set that was used.

The performance of the models on classifying entire images is quite good. The models have decent AUC (see Table 2 and Figure 3a). The grey scale model is slightly faster than the default model but it does not perform as well, although the difference in performance is not significant. The large window model has a higher AUC but is not significantly better than the default model. It is slower than the other models so there is no benefit in using the large window model over the other two models based on its ROC performance. Between the three evaluated models the best run time/performance ratio seems to be with the default model.

The data set was gathered on one location in a single day, because of this it is likely that the SVM has over fitted for the lighting circumstances and location. When the model would be applied to different data then it is likely that the performance would be worse, especially if the ground is different. A more universal and viable system would need more varying data to prevent over fitting.

Another effect of the small amount of data, in combination with the under trained models, is that

the detector doesn't only detect persons. The models detect high contrast areas which are often relatively wide. One of the common false positives is the heads of shadows cast by people, these are false positives which is not too bad since the purpose of the system is to narrow down the locations of persons. Other common false positives include objects like drains and concrete blocks. These objects don't occur often in the data set so the detector's performance isn't influenced too negatively by them. Examples of these false positives as indicated by the detector can be found in Figure 4. These false positives do indicate that the detector does not model persons. Objects like drains and concrete blocks are visually different from people so the model has not learned to distinguish persons from non-persons.

There are some possible other class distinctions that the models make. The most obvious is that it has learnt to distinguish grass/tiles from everything else. With this data set this would mean that a distinction is made on either contrast or structure. The former would mean that the system that has been built is a glorified edge detector. This is unlikely because some high contrast objects and edges are not labelled as positive. Some of these are about the same size as limbs, meaning that they should be more likely to be classified as positive since they do look more like the positive examples. Because of these factors it is unlikely that the models separate the data based on contrast alone.

Another explanation is that windows are being classified based on the global structure since this is the intention of HOG. Grass and pavement in the images are always correctly classified. Grass is highly random while the pavement is highly repetitive. Other objects show larger overall structure so it is possible that the models classify based on the presence of overall structure regardless of what that structure is. One of the reasons for this could be that there are relatively few positive examples and that these examples have a high variability. Figure 1 shows some of the variation between the examples, a person can be viewed straight from the top but also at an angle from the side. Because there are only 148 positive examples (592 with mirrored images) with a high variation it might be that the models are not able to generalise very well and thus it also includes objects that are not humans because they look relatively similar.

The developed detector shows great promise although there are some issues with it. Future work can focus on improving the run time of the detector. Training the system on more data is likely to alleviate most of the errors made by the detector. This is the main direction future work can take since the developed system has reasonable performance even if it is under trained.

6 References

[1] Paul Blondel, Alex Potelle, Claude Pégard, and Rogelio Lozano. Fast and viewpoint robust human detection for SAR operations. In *Safety, Security, and Rescue Robotics (SSRR), 2014 IEEE International Symposium on*, pages 1–6. IEEE, 2014.

[2] Alexandre S. Brandão, Felipe N. Martins, and Higor B. Soneguetti. A Vision-based Line Following Strategy for an Autonomous UAV. In *12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2015. (To appear).

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[5] Helen Flynn and Stephen Cameron. Multimodal people detection from aerial video. In *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013*, pages 815–824. Springer, 2013.

[6] Calvin Hung, Zhe Xu, and Salah Sukkarieh. Feature learning based approach for weed classification using high resolution aerial images from a digital camera mounted on a UAV. *Remote Sensing*, 6(12):12037–12054, 2014.

[7] Jeffrey C Lagarias, James A Reeds, Margaret H Wright, and Paul E Wright. Convergence properties of the Nelder–Mead simplex

method in low dimensions. *SIAM Journal on optimization*, 9(1):112–147, 1998.

[8] Constantine Papageorgiou and Tomaso Poggio. A trainable pedestrian detection system. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, volume 4, pages 35–39. IEEE, 1999.

[9] Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez, and Markus Müller. pROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC bioinformatics*, 12(1):77, 2011.

[10] Piotr Rudol and Patrick Doherty. Human body detection and geolocalization for UAV search and rescue missions using color and thermal imagery. In *Aerospace Conference, 2008 IEEE*, pages 1–8. IEEE, 2008.

[11] Jan C van Gemert, Camiel R Verschoor, Pascal Mettes, Kitso Epema, Lian Pin Koh, and Serge Wich. Nature conservation drones for automatic localization and counting of animals. In *Computer Vision-ECCV 2014 Workshops*, pages 255–270. Springer, 2014.

[12] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[13] Paul Viola, Michael J Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 734–741. IEEE, 2003.

[14] Sonia Waharte and Niki Trigoni. Supporting search and rescue operations with UAVs. In *Emerging Security Technologies (EST), 2010 International Conference on*, pages 142–147. IEEE, 2010.

[15] Qiang Zhu, M-C Yeh, Kwang-Ting Cheng, and Shai Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1491–1498. IEEE, 2006.