AUTOMATED MULTI-MODAL SEARCH AND RESCUE USING BOOSTED HISTOGRAM OF ORIENTED GRADIENTS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

by

Matthew Lienemann

December 2015

COMMITTEE MEMBERSHIP

TITLE:	Automated Multi-Modal Search and Rescue using Boosted Histogram of Oriented Gradients
AUTHOR:	Matthew Lienemann
DATE SUBMITTED:	December 2015
COMMITTEE CHAIR:	Lynne Slivovsky, Ph.D. Professor of Eletrical Engineering
COMMITTEE MEMBER:	John Saghri, Ph.D. Professor of Electrical Engineering
COMMITTEE MEMBER:	Jane Zhang, Ph.D. Professor of Electrical Engineering

ABSTRACT

Automated Multi-Modal Search and Rescue using Boosted Histogram of Oriented Gradients

Matthew Lienemann

Unmanned Aerial Vehicles (UAVs) provides a platform for many automated tasks and with an ever increasing advances in computing, these tasks can be more complex. The use of UAVs is expanded in this thesis with the goal of Search and Rescue (SAR), where a UAV can assist fast responders to search for a lost person and relay possible search areas back to SAR teams. To identify a person from an aerial perspective, low-level Histogram of Oriented Gradients (HOG) feature descriptors are used over a segmented region, provided from thermal data, to increase classification speed. This thesis also introduces a dataset to support a Bird's-Eye-View (BEV) perspective and tests the viability of low level HOG feature descriptors on this dataset. The low-level feature descriptors are known as Boosted Histogram of Oriented Gradients (BHOG) features, which discretizes gradients over varying sized cells and blocks that are trained with a Cascaded Gentle AdaBoost Classifier using our compiled BEV dataset. The classification is supported by multiple sensing modes with color and thermal videos to increase classification speed. The thermal video is segmented to indicate any Region of Interest (ROI) that are mapped to the color video where classification occurs. The ROI decreases classification time needed for the aerial platform by eliminating a per-frame sliding window. Testing reveals that with the use of only color data and a classifier trained for a profile of a person, there is an average recall of 78%, while the thermal detection results with an average recall of 76%. However, there is a speed up of 2 with a video of 240x320 resolution. The BEV testing

reveals that higher resolutions are favored with a recall rate of 71% using BHOG features, and 92% using Haar-Features. In the lower resolution BEV testing, the recall rates are 42% and 55%, for BHOG and Haar-Features, respectively.

Keywords: Unmanned Aerial Vehicles (UAV), Histogram of Oriented Gradients (HOG), Haar, AdaBoost, cascade, thermal, computer vision, search and rescue, multi-modal

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

AOV Angle of View. 24, 25

BEV Bird's-Eye-View. iv, v, 23, 28, 30, 56

BHOG Boosted Histogram of Oriented Gradients. iv, v, 17, 30, 34, 36, 40, 43, 45, 47, 49, 50, 53, 56

FPGA Field Programmable Gate Array. 57

FPPI False Positives Per Image. 24, 41, 43, 45, 49, 53, 56

FPPW False Postives Per Window. 6, 9

GPU Graphics Processing Unit. 57

HOG Histogram of Oriented Gradients. iv, 2, 4–9, 11, 12, 15–17, 23, 49, 50, 52, 53

LBP Local Binary Parts. 7

OSU Oklahoma State Universtity. 40

ROI Region of Interest. iv, 2, 8, 9, 11, 22, 45, 47

SAR Search and Rescue. iv, 1

SVM Support Vector Machine. 6, 30, 50, 52, 53

UAV Unmanned Aerial Vehicle. iv, 1, 2, 8, 9, 47

ViPER Video Performance Evaluation Resource. 23

Chapter 1

Introduction

In SAR, the outcome of a search is resolved within the first few hours of response. SAR teams set-up a theoretical search area based on the last known location and the distance a subject could have traveled in the time elapsed. Introduction to Search and Rescue [18] outlines the various resources used for SAR as described in Table 1.1. These resources provide the SAR effort the tools needed to maximize the probability of success in the minimum amount of time. Unfortunately these methods are prone to the need of skilled searchers, which may not always be available.

Further resources can expand to more technological efforts as described in the Australian National Search and Rescue Manual and the Navy Search and Rescue Manuals by using aircraft [3] [8]. Observers are used on search aircrafts to monitor equipment such as radar and thermal scanners. However, radar may only be used if beacons are available, so the observers must rely on visual signals and thermal scanners. The well-trained observers may be affected by fatigue, and the speed of the aircraft also affects the effectiveness of the observers. We can improve this method by automating the observer's job through color and thermal cameras and computer vision techniques. The larger aerial platforms can also be replaced by UAVs, which are cheaper, easier to operate, and offer longer endurance.

The observer can use the thermal camera to locate a person quicker through a color camera by only observing the heat signature and the outline of the person.

Table 1.1: SAR Common Teams and Resources [18]

Teams	Description
Hasty Search Teams	Highly Skilled in land navigation. Quickly check high probability areas. Locate clues to establish search area, and provide up to date information that maps may not include.
Grid Search Teams	Search in well-defined segments in tight search areas. Can be damaging to environment and clues. Not as well trained as trackers and hasty search teams. Usually a last resort effort.
Trackers	Search for clues and can provide a direction of travel as well as determine areas of high probability. Best utilized in early phases of response.
Search Management	Analyze the resource needs based on incident objectives.
Investigation	Gather information about subject.
Technical Teams	Specific rescue needs and skills such as rope rescue and water rescue,
Canine and Equestrian	Canine: Used for scent tracking. Equestrian: Horses provide high platforms to search.

In a similar fashion, the thermal camera can provide an initial segmentation of a possible person to classify in an automated system. However, the thermal camera only provides sparse spatial information, due to resolution of the frame. By creating this segmentation, or ROI, a mapping to the color camera can be made for further classification. Through the use of these sensing modes, it is possible to correctly identify a person from an aerial perspective using feature descriptors over a segmented region provided from thermal data.

The major problem involved in this work is in classifying a segmented region as a possible human. The features must be robust for variations in illumination and background. The feature set must also be fast to calculate, due to the moving UAV platform and to effectively search an area for any lost people. The two main methods of feature extraction that are researched in this project are HOG [5] and Haar-like features presented by Viola-Jones [25]. We investigate the use of the HOG features combined with the speed of Viola-Jones. Another aspect that is considered is the speed of identifying a region of interest versus a sliding window technique, using the combined feature extraction technique.

We discuss previous works on various human detection techniques and similar projects in Section 2. Section 3 provides the theoretical framework for human detection via computer vision, and thermal segmentation. Section 4 analyzes the problem and special considerations involved using an aerial platform, describes the training of the feature set, and evaluates the human detection and thermal segmentation in specific environments. Section 5 concludes our thesis, noting the successes and current problems with our method, as well as possible future work for expansion on the thesis.

Chapter 2

Related Work

The research for human detection can be divided into two sub-areas; common feature sets for human detection and improvements to these feature sets. In order to determine a proper classifier for aerial human identification, we examine the techniques commonly used in human detection via computer vision and adapt these techniques for aerial identification of a human.

2.1 Feature Sets for Human Detection

In order to determine whether a human is found in an image, a set of features must first be defined. In the early 2000's, a real-time face detection algorithm was proposed by Viola and Jones [25]. Viola and Jones uses integral images to quickly calculate Haar-like features. A Haar feature set is a set of rectangular features that detect change in contrast to neighboring rectangular groups of pixels. The feature set produces 162,336 features for each feature (shown in Figure 2.1) at every position and scale in a 24x24 detection window.

In order to train the large amount of features, a Cascaded-AdaBoost Classifier is used to select the best features. The Cascade Classifier also increases the efficiency by rejecting most false positives early in the detection stage, and spends more processing time on more reliable Haar-features. Viola et. al [26] expand on the Haar features to detect moving humans. This method has proven to be faster than other detection methods, however, HOG based classifiers continuously outperform the Haar features, [28], [5], [16].

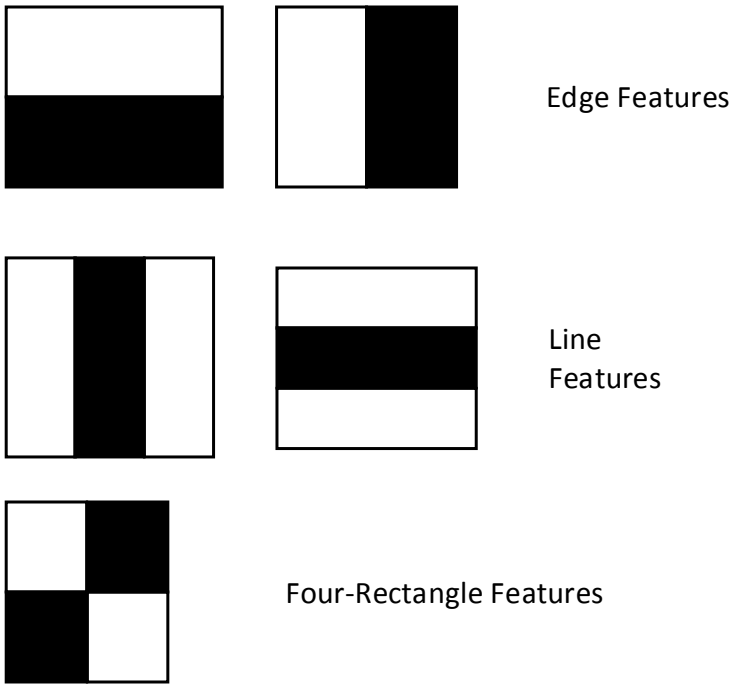


Figure 2.1: Haar Feature Set. [25]

In 2005, Dalal and Triggs [5] presented a framework known as HOG that extracts features based on gradients. It significantly outperforms other feature sets for human detection in terms of accuracy, including Haar features. HOG features use gradients to cast votes into an histogram based on orientation. Each histogram is collected over a block of pixels called cells in a detection window and a set of cells is normalized to provide invariance to illumination. Given a detection window of 64x128, the HOG method creates a rich descriptor of 3780 features. The features are trained on a Support Vector Machine to classify a window as human or non-human. However, calculating the HOG features is costly. Dalal [5] reports that processing a 320x240 scale-space image is approximately 1 frame-per-second.

2.2 Improvements to HOG

Due to the success of the HOG feature set, many researchers have sought to decrease it's computational complexity by reducing the dense feature calculations. Each feature of HOG is calculated over rectangular regions known as cells or blocks, and within each cell a histogram is formed based on gradients. Common speed-ups includes removing the tri-linear interpolation in favor of linear interpolation for binning, and rectangular region calculations are sped up through integral images. The following research papers expand on the fundamentals of HOG by increasing speed, or handling occlusions.

In 2006, Zhu et al. [28] speeds up the HOG algorithm through a Cascade of HOGs. This fast human detection method detects humans from 5 to 30 frames per second from a 320x280 frame, depending on the density of the scan. Zhu et al. improves on the HOG features through the combination of integral images to calculate histograms and by using the cascade rejector approach. The method uses a parts-based detection using a single window approach by using variable sized blocks from 12x12 to 64x128. The variable sized blocks captures both small local gradients and larger global gradients. This method produces 5021 total blocks in a 64x128 window, each of which contains a 36 dimension histogram. To train the 5021 possible blocks, a Cascaded AdaBoost learning algorithm with Support Vector Machines (SVMs) to select the best features from the feature set. From the tests performed, it has been shown that the Cascade of HOGs [28] is not as accurate as Dalal and Triggs original HOGs implementation [5]. The Cascade of HOGs has a detection rate of 95% while HOG has a detection rate of 97% at 0.01 False Postives Per Window (FPPW). However, as the FPPW goes down, it is comparable and faster.

In 2007, Jia and Zhang [16] improves HOG by using each histogram bin as a feature in a cascade classifier, similar to Zhu et al. [28] . Each feature is defined by its cell position, parent block position and the orientation bins, to produce tens-of-thousands of features in an 18x36 detection window. To simplify the calculation process, integral images are used similar to Zhu et al. [28], and each feature is found in constant time with only 8 lookups. To reduce the amount of features, a Cascade AdaBoosting architecture, with classification and regression trees as the weak learners is used. This method does perform faster than HOG [5] but has lower detection rate. It is also slower than Viola and Jones [26] but has a higher detection rate.

Wang et al. [27] develop a HOG detection with partial occlusion handling by combining HOG features with cell structured Local Binary Parts (LBP). To speed up the HOG-LBP, integral histograms are used in the same manner as Zhu et al. [28]. Zhu et al. reports that tri-linear interpolation does not fit well with integral images, however, Wang et al. apply a Convoluted tri-linear interpolation over each integral image. The tri-linear interpolation reduces aliasing effects of strong edge pixels at cell boundaries. The linear interpolation voting scheme assigns the pixel's weight to different histogram bins, whereas tri-linear interpolation distributes the weight to all neighbors. HOG-LBP also provides occlusion handling by constructing an occlusion likelihood map. This map then generates a confidence score which determines if the a parts-based detector is used, or no further processing is needed.

2.3 Aerial Human Identification

The following papers make use of Haar and HOG features for aerial human identification. The method presented by Oreifej, O. et al. [19] attempts to identify a human from an aerial platform by using HOG features, a voter candidate model, blob extraction of the target, and alignment to eliminate variation of camera orientation and human pose. The overall results from this method showed a mean average precision of 89.6%. It should also be noted that Oreifej, O. et al. identified a manually selected person that is then identified in test images. Their presented method does provide a solution to identify people in low quality aerial images, where there is little availability of minor details of a person and only dominant features (head and torso) are available.

The following two research papers expand on aerial human identification through the use of thermal cameras to create a ROI, and then their classifier is applied to this ROI. Helen Flynn and Stephen Cameron [13] propose a part based model presented by Felzenszwalb et at. [12] in order to detect the high variability of a person. However, the parts-based model is slower to process than HOG descriptors. Helen et al. incorporate a multi-modal process to reduce the processing time by using a thermal camera to create a ROI that is mapped to a color frame for further evaluation. The determination of the ROI is based on the assumption that a person is hotter than the surroundings, so a hard-threshold of high intensity pixels is used to extract from the thermal video.

Similar to Helen et a. [13], Piotr Rudol and Patrick Doherty uses a multi-modal approach [22]. Instead of a direct mapping of the thermal ROI to the frame matched colored image, as Helen et al. have done, Piotr et al uses the altitude, pitch and yaw information of the UAV, arguing that direct mapping

will use more memory. The meta-data from the UAV translates the thermal pixels to world coordinates and are used to acquire corresponding pixel locations on the color frame. Each ROI is then subjected to a detector based on extended Haar features.

Schmidt and Hinze [24] proposed a method to detect and track people in high altitude aerial images by taking advantage of global position, time data, and casted shadows from a person. The detection phase uses boosted enhanced Haar-like features for object shape, rectangle features for color information, and a detector designed for a person and a potential shadow.

Gaszczak et. al [15] have presented a way to detect people and vehicles from UAV imagery using thermal data. Every frame detects people or vehicles using Haar classifiers with multivariate Gaussian shape matching as a secondary confirmation. The thermal images are processed with the Haar classifiers, as opposed to using the color images for classification. The results from the classifier are further processed based on the size constraints of the objects, which is based on the altitude and camera filed of view. The detected object will be discarded if it does not meet the size requirement.

Table 2.1 summarizes the improvements to HOG features by comparing tests on the INRIA dataset. Similar FPPW are shown to provide a fair performance metric. The speed of the detector is also included in Table 2.2 if it is provided in the respective method. If a test was not available, it is labeled as N/A.

Table 2.1: Improvements to HOG Performance Comparisons

Method	FPPW	Detection Rate		
		Improved HOG	HOG	Haar
Zhu et. al. [28]	0.001	92%	94%	70%
Jia and Zhang [16]	0.001	88%	94%	75%
Wang et al. [27]	0.0001	98%	93%	N/A

Table 2.2: Improvements to HOG Time Comparisons. Each metric is over 12,800 detection windows per 240x360 image

Method	Time(ms)		
	Improved HOG	HOG	Haar
Zhu et. al. [28]	106	7000	55
Jia and Zhang [16]	55	5000	32
Wang et al. [27]	N/A	N/A	N/A

Chapter 3

Theoretical Framework

The main goal of this project is to correctly identify a person from an aerial perspective using modified HOG descriptors over a segmented region provided from thermal data. The HOG descriptor derivation is addressed, and analyzed where improvements will be made. We will also implement the multi-modal process using a thermal and color camera that are frame matched. It has been shown to greatly improve timing as well as reduce false positives [13] [22], however, we will extract thermal thresholds through Otsu's method [1] to create ROIs. The overall proposed system is outlined in Figure 3.1.

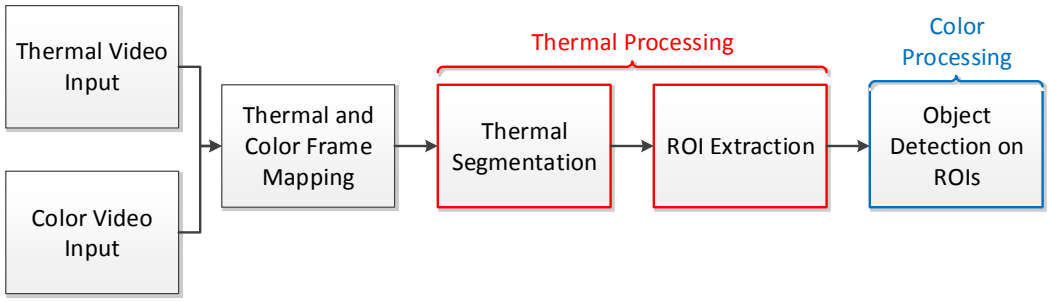


Figure 3.1: System Diagram of the Search and Rescue Computer Vision System.

3.1 Histogram of Oriented Gradients Feature Description

The main idea of HOG is to use a detection window that is divided into NxM cells that accumulate a histogram for edge orientations. The cells are normalized with overlapping blocks to provide better invariance to illumination. After normalization, the cells in the detection window are concatenated into a

1-dimensional feature that can then be used for classification. The overview of the HOG process can be seen in Figure 3.2. The following section will show how the HOG features are calculated.

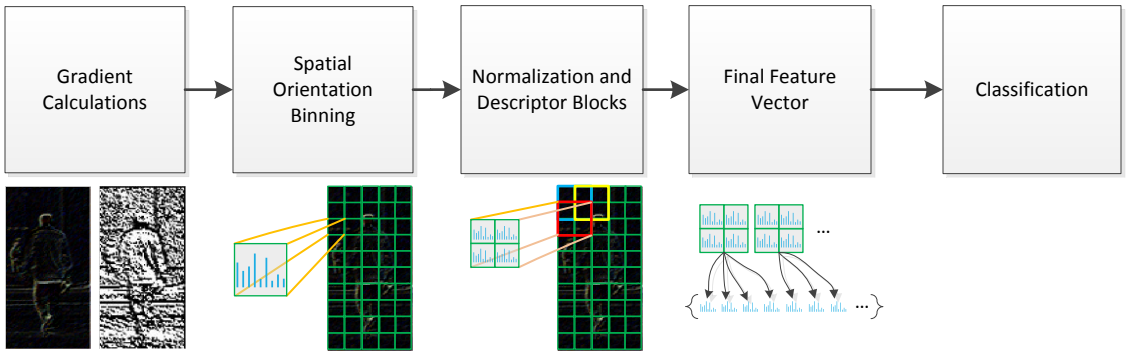


Figure 3.2: Overview of the Histogram of Oriented Gradients. Each image has its gradients and orientations calculated. Histogram bins are calculated over cells, where the bins correspond to the orientation, and each vote is a portion of the gradient. Cells are normalized over blocks, where the blocks overlap one another. The final feature vector are the normalized cells. [5]

The first step, as outlined in Figure 3.2, is to compute the gradient magnitudes and orientations of the input image. To calculate the magnitude, two images are formed; G_V which captures vertical gradients and G_H which captures horizontal gradients. Both are calculated by convolving the original image Img, with the kernel $[1, 0, -1]$ and its transpose, which can be seen in equations 3.1 and 3.2. The kernel $[1, 0, -1]$ is used since Dalal and Triggs [5] have found that when computing the gradients, small kernels exhibit the best performance. Using G_H and G_V, the magnitude is calculated by Equation 3.3, where r and c are row and column pixel values.

$$G_H = Img \otimes \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{3.1}$$

$$G_V = Img \otimes [1, 0, -1] \tag{3.2}$$

$$M[r, c] = \sqrt{G_V^2[r, c] + G_H^2[r, c]} \tag{3.3}$$

Similar to calculating the magnitude, Equation 3.4 shows how the orientations are obtained by using the same vertical and horizontal gradients, G_H and G_V.

$$O[r, c] = \arctan \frac{G_V[r, c]}{G_H[r, c]} \tag{3.4}$$

3.1.1 Spatial/Orientation Binning

After the magnitude and orientation images are found, histograms can be formed over cells. Cells are a rectangular area of pixels, where each pixel in a cell casts a weighted vote based on the gradient magnitude and orientation (see Figure 3.2). The voting scheme uses the orientation as bins, spaced evenly over 0°-180° or 0°-360°, and the magnitude casts a weighted vote. The voting process is as follows:

1. Find the closest bins for a given pixels orientation (the neighbors).

2. Given a pixels magnitude, calculate a linear vote between the neighbors. Equation's 3.5 and 3.6 shows how the vote is calculated, given P_M as the pixel's magnitude, P_O as the pixel's orientation and N_x as neighbor X.

$$Vote_{N1} = P_M * \left| \frac{N_1 - P_O}{N_1 - N_2} \right| \tag{3.5}$$

$$Vote_{N2} = 1 - Vote_{N1} \tag{3.6}$$

3.1.2 Block Normalization

Dalal and Triggs [5] explains that the gradient strengths varies over a wide range due to variations in illumination and foreground-background contrast. To solve this, cells are grouped into spatial blocks that overlap one another. Within each block, the histograms are normalized. Dalal and Triggs uses a 50% overlap of cells on each block. They have also shown that the square root of the L1-Norm has shown the best detection rate, however, it is slower to process.

All histograms from a block are concatenated to form a 1D vector, $\mathbf{b}$. Equation 3.7 describes how $\mathbf{b}$ is normalized using the L1-sqrt method, where $\|\mathbf{b}\|$ is the euclidean norm.

$$L1_{sqrt-b} = \sqrt{\frac{\mathbf{b}}{\|\mathbf{b}\|}} \tag{3.7}$$

3.1.3 Final Feature Descriptor and Classification

The final feature descriptor is a 1D normalized array of histograms. This descriptor is applied to a Support Vector Machine for training or classification. The final descriptor size of a window depends on the cell size, block size and number of bins. If we consider a 64x128 detection window, divided into 7 blocks across and 15 blocks vertically, with each block containing 4 cells with 9 histograms, there are a total of 3780 features.

3.2 Integral Images

Integral Images are presented [26] to quickly detect faces in an image using rectangular features. The integral image is a representation of the original image where each Integral Image pixel at location (r, c) contains the sum of the pixels up and to the left of (r, c). The generation of the integral image can be done in $O(1)$ time, with just one pass over the image. Equation 3.8 describes this process,

$$II(-1, 0) = II(0, -1) = 0 \tag{3.8}$$

$$II(r, c) = i(r, c) + II(c - 1, r) + II(c, r - 1) - II(c - 1, r - 1) \tag{3.9}$$

where r and c are row and column pixel values, $II(r, c)$ is the integral image and $i(r, c)$ is the original image. Equation 3.8 specifies the edge case of the top most left pixel of an image, $i(0, 0)$.

After creating the Integral Image, any rectangular sum can be computed given four reference points. For example, to calculate the area of D in the original image as shown in Figure 3.3 , the following values from the integral image are needed: P_A, P_B, P_C and P_D. The sum of the rectangle area D is thus $P_A + P_D - P_C - P_B$.

3.2.1 Integral Histograms

To improve the speed of HOG, Integral Histograms can be used to calculate a cell at some bin n. To do this, we generate our bins in the same fashion as Zhu et. al. [28]; each pixel's orientation and magnitude is discretized into N bins, or bin images. Figure 3.4 represents how each bin image is constructed. Each bin

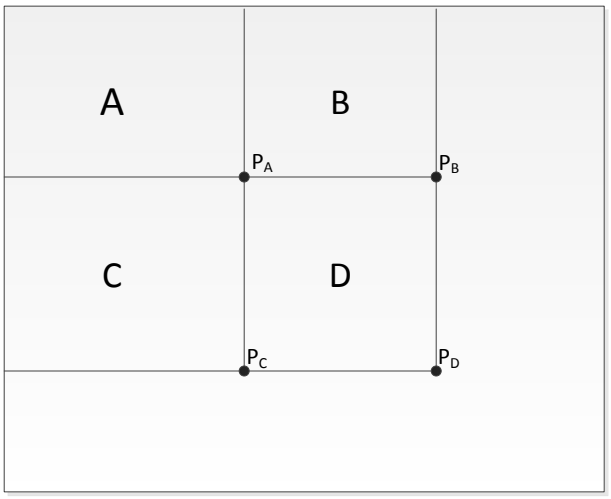


Figure 3.3: Integral Image calculating the area D, given the four reference points P_A, P_B, P_C and P_D.

image is an orientation, and each pixel value in a bin is the magnitude based on the voting scheme as described in Section 3.1.1. Each of these bin images is then transformed into a respective integral image for feature computation. Each bin calculation for a HOG feature is then 4xN operations.

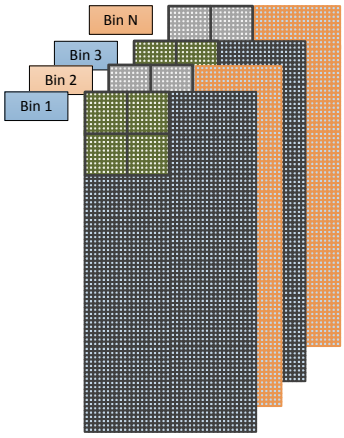


Figure 3.4: Representation of N Bin Images. Each bin image corresponds to some angle, and each pixel location contains a vote of the magnitude from the original image.

3.3 Boosted Histogram of Oriented Gradients

In order for the HOG feature descriptors to be more suitable for aerial human detection, a small detection window is needed. The original HOG descriptors uses a 64x128 detection window, which will translate to a lower maximum altitude (see Figure 4.4). The smaller detection window will allow for higher maximum altitude searches. Also, as mentioned in Section 4.1, the speed of the detector is critical. Given a maximum detection time of approximately two seconds to detect an object before it leaves a frame at a height of 30 meters, Figure 4.4, a detector that can detect the object multiple times before it leaves the frame will be more reliable.

To achieve the time and detection goals, the BHOG detection scheme will be used [16]. BHOG offers the discriminative power of HOG and the detection time of Cascade Haar-Features, while having a rich number of features in a smaller detection window. The framework behind BHOG consists of a Cascaded AdaBoost Classifier detection similar to Viola-Jones, and each feature is based on HOGs.

3.3.1 BHOG Feature Definition

The BHOG features, $f(C, B, k)$, are defined by its cell position $C(c_c, r_c, w_c, h_c)$, parent block position $B(c_b, r_b, w_b, h_b)$ and the orientation bin K, where r and c are the pixel row and column coordinates, respectively, w is the width, and h is the height. To form the features, the gradient magnitude $M[r, c]$ and orientation $O[r, c]$ are first calculated using Equations 3.3 and 3.4. Binning is simpler than the original HOG method presented in Section 3.1.1. The orientation bin range

of $[0, 180]$ is divided into K bins and each bin, β_k, is denoted

$$\beta_k = \begin{cases} M[r,c] & \text{if } O[r,c]\epsilon bin_k \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

where $O[r,c]\epsilon bin_k$ denotes that $O[r,c]$ is assigned to the nearest bin k.

The final feature value, $f(C,B,k)$, is defined as

$$f(C,B,k) = \frac{\sum_{(r_c,c_c)\epsilon C} \beta_k(r,c) + \varepsilon}{\sum_{(r_b,c_b)\epsilon B} G(r,c) + \varepsilon} \tag{3.11}$$

where $\sum_{(r_c,c_c)\epsilon C} \beta_k(r,c) + \varepsilon$ is the sum of the respective values in the k^{th} bin in cell C which belongs in block B, and $\sum_{(r_b,c_b)\epsilon B} G(r,c) + \varepsilon$ is the sum of the gradients G in block B. The ϵ value is some small number in case of any division by zero.

Each feature can be quickly computed by using an integral image of each bin and gradient image as described in Section 3.2. Each bin is constructed into it's own image and a corresponding integral image is created, $II_{\beta,k}$, for a total of k integral images. A gradient image II_G is also formed, and a total of 8 lookups are needed to calculate a feature as described in Equation 3.12,

$$\begin{aligned} \sum_{(r_c,c_c)\epsilon C} \beta_k(r-1,c-1) = {} & II_{\beta,k}(r_c,c_c) + II_{\beta,k}(r_c+h_c-1,c_c+w_c-1) \\ & - II_{\beta,k}(r_c+h_c-1,c_c-1) - II_{\beta,k}(r_c,c_c+w_c-1) \end{aligned} \tag{3.12}$$

$$\begin{aligned} \sum_{(r_b,c_b)\epsilon B} G(r,c) = {} & II_G(r_b-1,c_b-1) + II_G(r_b+h_b-1,c_b+w_b-1) \\ & - II_G(r_b+h_b-1,c_b-1) - II_G(r_b-1,c_b+w_b-1) \end{aligned} \tag{3.13}$$

and the edge cases, $II_G(-1, c_b) = II_G(-r_b, -1) = II_G(-1, -1) = 0$. The same holds for $II_{\beta,k}$.

Each feature block is defined in the same manner as Jia et. al [16]. Figure 3.5 shows the various blocks that are used. Each shaded area within a block represents a cell where bin calculations take place, and the whole rectangle represents a block. While generating a feature pool, a set ratio r, block stride s_b, a scaling factor s_f and minimum block size w_{min} h_{min} are used to constrain the amount of features generated. Given $r = \{1:2, 1:1, 2:1\}$, $s_b = 0.5$, $s_f = 1.2$, $w_{min} = 4$, $w_{min} = 4$ and a maximum training sample size of 18x36, the feature pool contains 124,821 features.

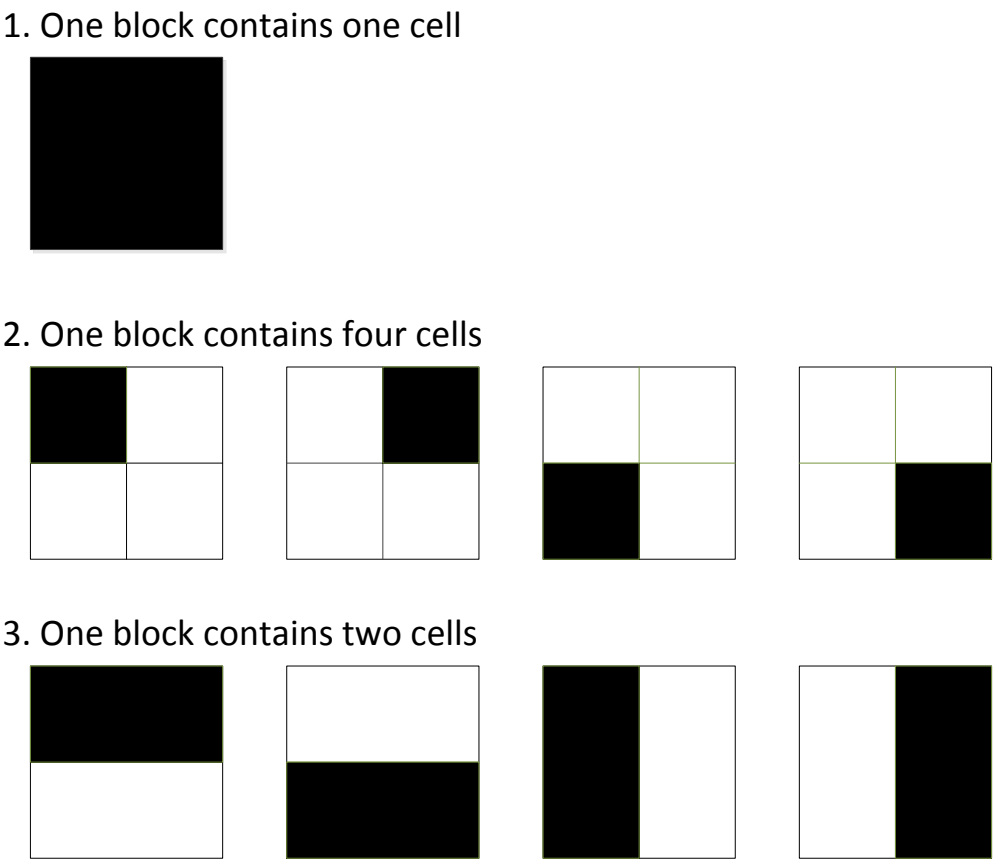


Figure 3.5: Boosted Hog Feature Templates.

3.4 Thermal Segmentation

The assumption that a human will be hotter than the surrounding environment can be taken advantage of in order to only classify a small region of interest. This process acts as a first layer of classification based on thermal data and segmentation.

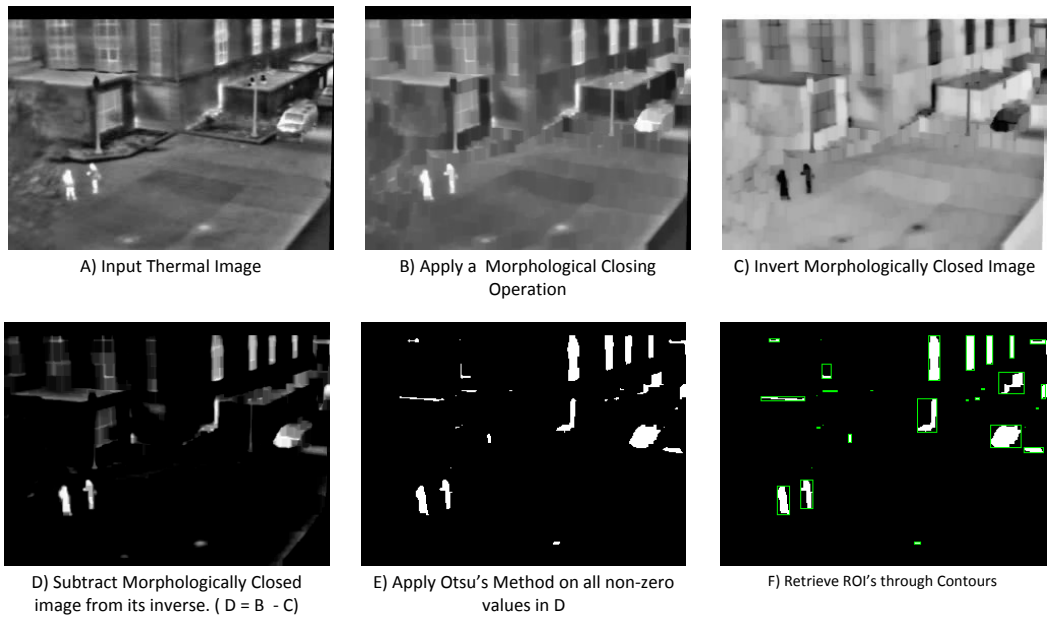


Figure 3.6: Thermal Segmentation Flow.

The flow of the thermal segmentation process can be seen in Figure 3.6. Beginning with the original thermal image (A), a morphological closing operation is performed over the image (B). This is done to reduce the non-uniformity of an objects temperature. In the case of a human, clothes usually block the high temperature of the human body, creating thermal voids and the closing operation will close any openings presented by the differences in temperatures. Next, the result of the morphological close is inverted (C) and is then subtracted from the morphological close image (B) to reveal the hot spots (D). To finalize the

segmentation, Otsu's method [1] is used to create a binary image (E) by creating a global threshold based on all non-zero values from the hot spots image (D). The final binary image (F) defines where the ROIs are located through the use of an OpenCV function, findContours().

After the ROIs are received, a detection window will center itself at the top left pixel of a thermal ROI. Scanning will then continue in a raster scan fashion, and the detection window will shift by Δp_r and Δp_c pixels, in each row and column, respectively. The scanning will end when the detection window's center reaches the bottom right pixel. The process can be more clearly seen in Figure 3.7. This process will continue for every ROI at every scale.

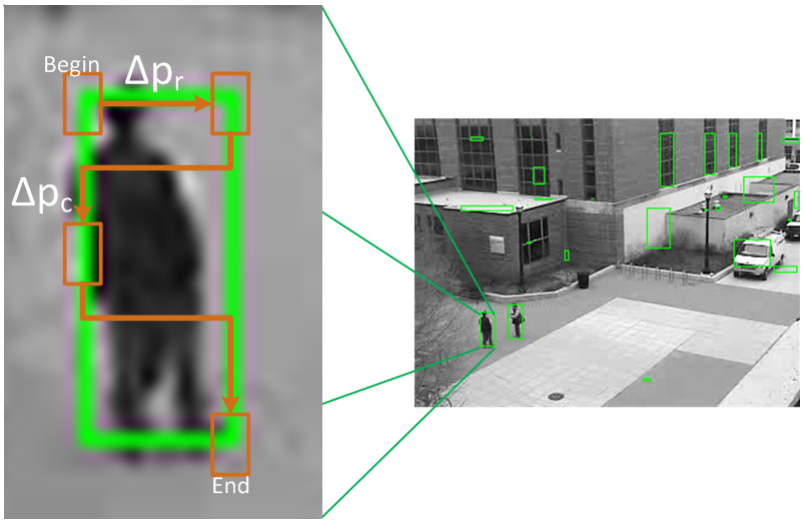


Figure 3.7: Thermal ROI Scanning Example.

Chapter 4

Evaluations

Two datasets tests the speed and accuracy of the feature set; thermal and color matched frames [7] and BEV videos [20]. Each dataset provides an environment to test the feature set in this project and will be compared our implementation of HOG features, and OpenCV's implementation of Haar-Cascade features. The frame-matched thermal and color video evaluates the speed of the detector coupled with thermal data. The thermal data also acts as layer of detection, as previously described. The BEV dataset consists of two videos taken from rooftops, each with different heights and lighting conditions.

In order to evaluate each frame in a video, we adopt the per-image evaluation from Dollar et al. [9]. Each video is manually annotated using The Video Performance Evaluation Resource (ViPER) [6] to create a ground truth bounding box (BB_{gt}) around each possible candidate. Each detection and ground truth bounding box in a frame is evaluated based on the PASCAL [11] evaluation measurement in Equation 4.1, which calculates the area overlap between the ground truth and predicted bounding boxes. A correct detection between BB_{GT} and a predicted bounding box, BB_P, is any overlap greater than or equal to 50%, as defined in the PASCAL Visual Object Classes Challenge [11], anything less is considered a miss. Each BB_{GT} is also labeled for any partial or fully occluded people, which are ignored in the evaluation process since our detector does not take into account any occluded objects. The ignored BB_{GT} does not count as

either a hit or miss.

$$a = \frac{area(BB_{GT} \cap B_P)}{area(BB_{GT} \cup B_P)} \tag{4.1}$$

The videos will be evaluated based on the recall rate and False Positives Per Image (FPPI), where the FPPI is calculated as the average false positives found in a frame over the entire length of the video.

4.1 Minimum Speed of Detector

The speed at which a human needs to be detected is dependent on the height of the aerial platform and the resolution of the camera. It will be assumed that the aerial platform is traveling parallel to the vertical resolution of the camera (see Figure 4.1). Given the horizontal and vertical Angle of View (AOV) β_H and β_V, the altitude of the aerial platform (A) and the vertical and horizontal resolution of the camera, R_V and R_H we can determine the ground resolution and the time for an object to span that resolution. We can also translate the size of an object with these parameters from pixels to meters.

Using the dimensions outlined in Figure 4.2, the horizontal ground resolution R_H and vertical ground resolution R_V are calculated as follows:

$$V = 2 * A * \tan(\frac{\beta_V}{2}) \tag{4.2}$$

$$H = 2 * A * \tan(\frac{\beta_H}{2}) \tag{4.3}$$

The time for an object to pass in a frame (t_O) will be dependent on the aerial platform. We will use the velocity of the two aerial platforms the RMAX

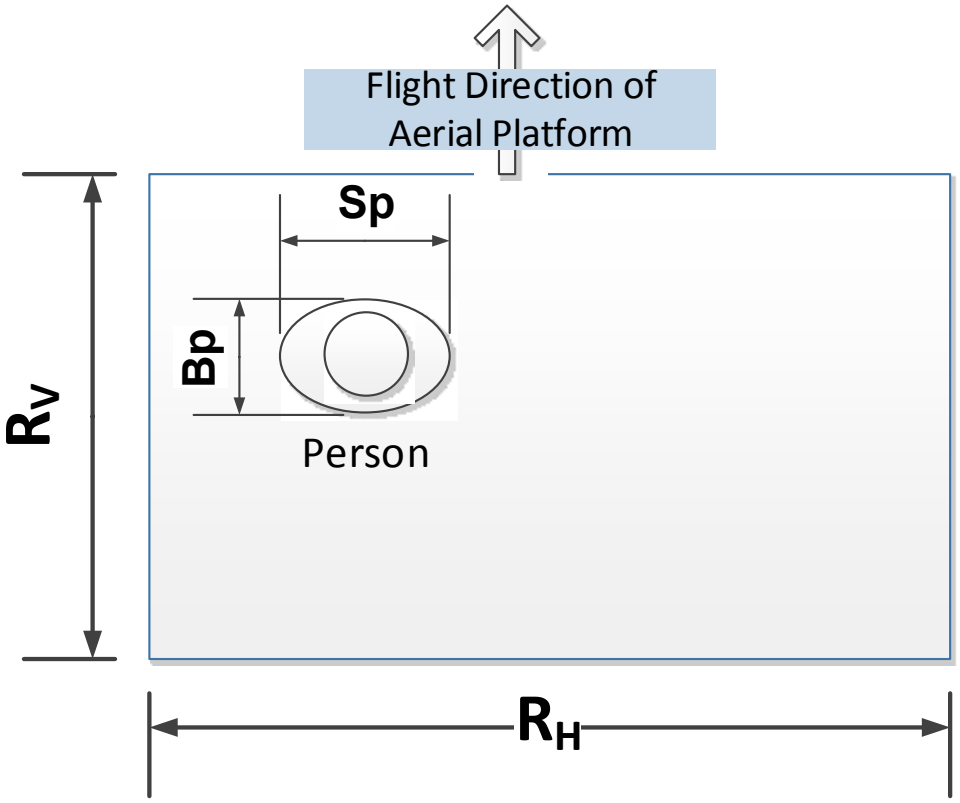


Figure 4.1: Bird's Eye View Geometry. Where R_V corresponds to vertical ground resolution, R_H is horizontal ground resolution, A is altitude of the aerial platform, and β is the Angle Of View.

project currently uses, the RMAX helicopter and a Telemaster Radio Controlled Airplane. The cruising speed of the RMAX helicopter is 5.56 m/s [23] and the Telemaster's cruising speed is 15.4 m/s [4]. Equation 4.4 shows the how to calculate t_O, given aerial platform's speed, P_{speed}.

$$t_O = P_{speed} * V \tag{4.4}$$

Next we will observe how t_O is affected according to the altitude. We will use an AOV based on the wide angle lens, Tamron 23FM65L, with $\beta_H = 71.6$ and

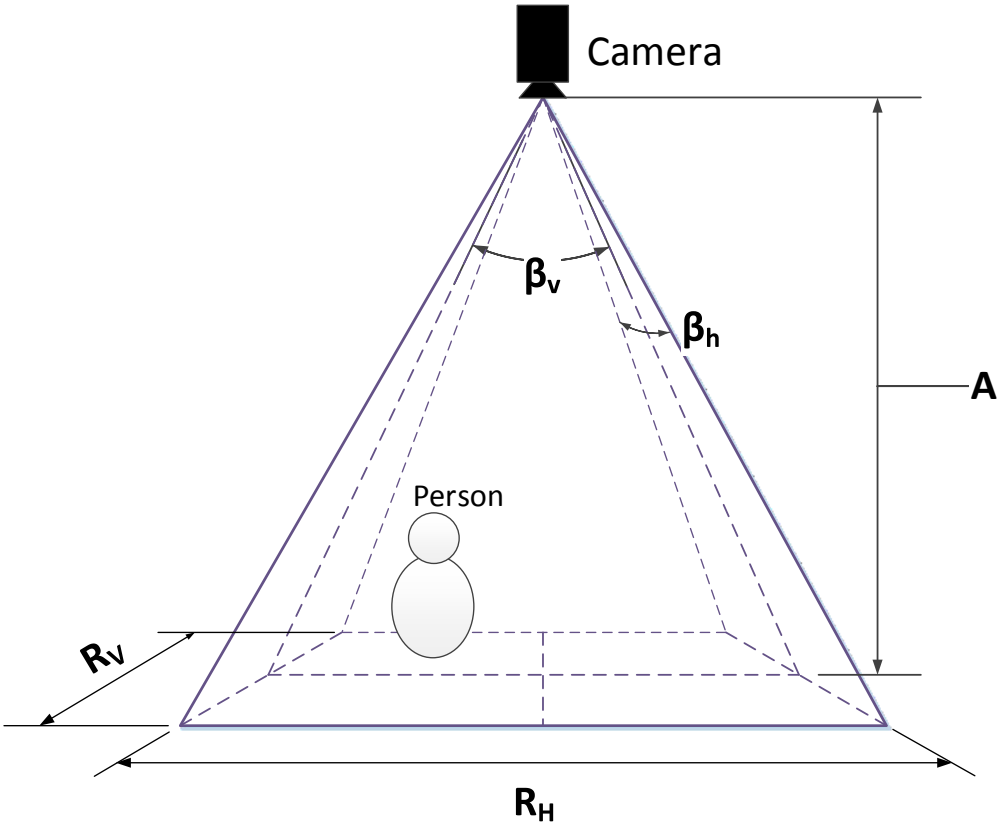


Figure 4.2: Camera Geometry.

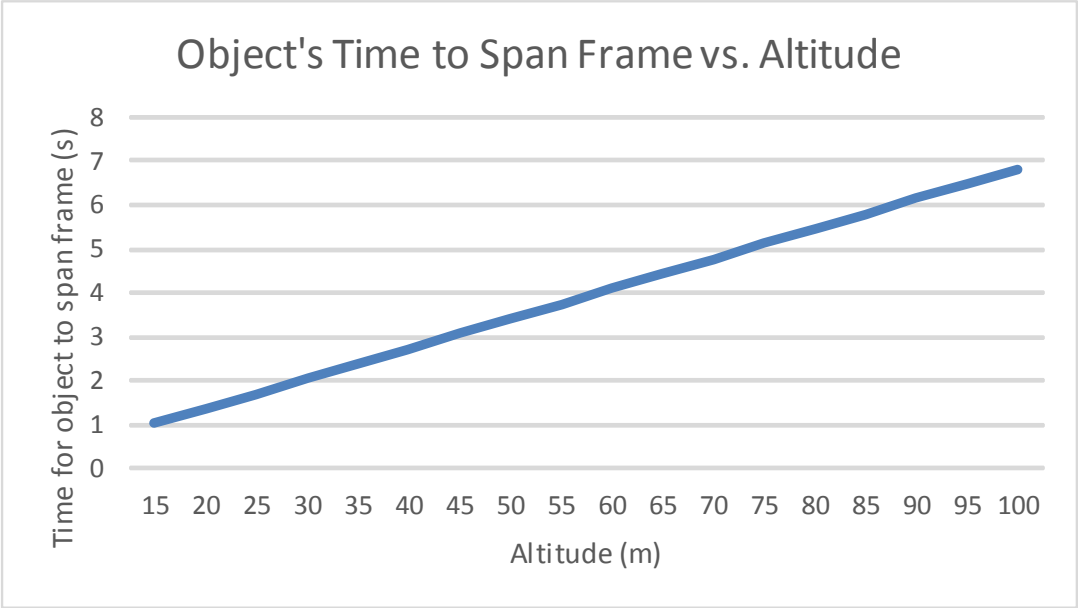


Figure 4.3: The amount of time an object needs to span a frame at various altitudes.

$\beta_V = 55.5$. Figure 4.4 provides us with a benchmark that the feature detector needs to meet, and it is assumed that the object is standing still relative to the aerial platform. Note that the time is not affected by the resolution.

It should also be noted, that the size of a person at various altitudes is important, since higher altitudes will contain lower resolution objects. We will be using the previous assumption that the human is in a bird's-eye view perspective. In the bird's-eye view perspective, the most notable features are the shoulder width (S_p) and bust depth (B_p), which can be seen in Figure 4.1. The size of a human in pixels (P_{Pixels}) is calculated according to the Equation 4.5. We use the values $B_p = 25cm$ and $S_p = 48.9cm$, for shoulder width and bust depth, respectively, and are based on the anthropometric measurements of an American male in the early 2000's [2].

$$P_{Pixels} = V_r * H_r * (S_p * B_p)/(V * H) \tag{4.5}$$

Figure 4.4: Square pixels of a human in a bird's eye view, with varying altitudes and resolutions. The size of the human is based on the bust and shoulder length.

4.2 Training

Training involves the use of a positive (human) and negative (non-human) datasets. Two training sets are used: INRIA dataset for thermal testing, and a dataset for BEV testing. The BEV annotations are chosen based on concentric shoulder alignment, where the shoulders are centered in the sample window. The BEV positive sample windows are mirrored, rotated, and flipped to generate more poses from a single image. The end result generates 2,185 positive training images. Figure 4.5 shows 16 samples of the BEV positive training dataset.

Negative training images are randomly sampled from the 1218 image INRIA dataset used in Dalal and Triggs [5]. Each sample is generated by randomly generating a width and height with the same ratio as the positive images, and then randomly generating the top left corners of the sample. A random negative

sample is approved if it's variance is above 0.001, to ensure that the random sample isn't too uniform. The sample is then resized to the desired training size. Eight negative sample images are shown in Figure 4.6.

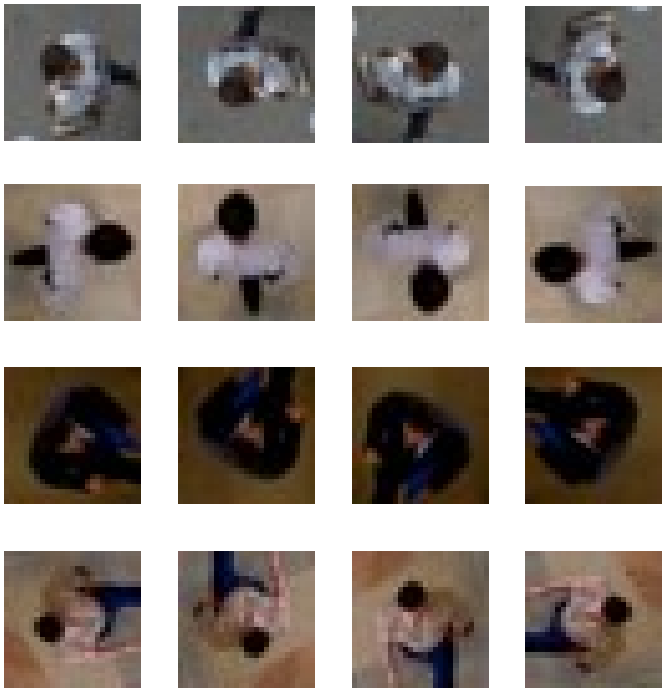


Figure 4.5: Example of Positive Training Images, sized at 25x25 pixels.

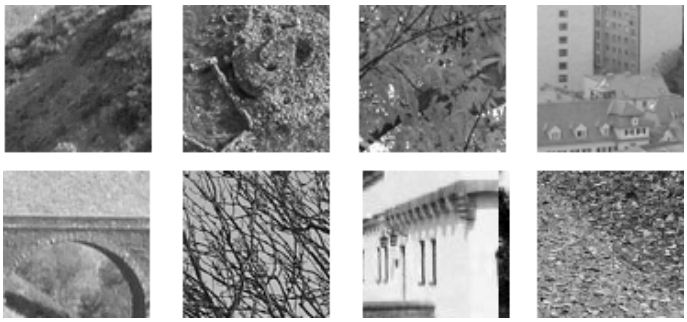


Figure 4.6: Example of Negative Training Images.

4.2.1 Cascaded AdaBoost Training

The BHOG descriptors produces 120,933 features for a 25x25 detection window from the BEV dataset. The use of a SVM would not be applicable for BHOGs due to the time it would take to train and predict these features. Hui et al. [16] use a cascade classifier that consists of Classification and Regression Trees (CART) as a weak classifier to train over BHOG features, and demonstrated that a stump classier performs better than their CART classifier. We will be using a stump classifier method as presented in the Viola-Jones face detection framework [10].

AdaBoosting is a meta-learning algorithm first proposed by Freund and Schapire [14]. The main idea behind boosting is to combine several weak weighted classifiers to produce one strong classifier. The weak classifier is defined as any classification method that can produce an error rate less than 50%. Boosted classification was extended by Viola and Jones [10] by cascading AdaBoost classifiers in order to quickly invalidate false positives.

AdaBoost Algorithm

In this project, we will be using a Gentle AdaBoost classification method, which has been shown to outperform Discrete AdaBoost [17]. The Gentle AdaBoost training algorithm is based on the node training algorithm from Jian-Qing et al. [21], however we use an iterative update of the weighted squared errors to speed up processing, similar to Viola and Jones.

In order to measure the squared error, we first perform a stump classifier over all positive and negative samples, as outlined in Table 4.1. The stump classifier sorts each feature based on the feature value extracted from each training sample.

Every sample is given a weight w_i and the feature values are then sorted in increasing order.

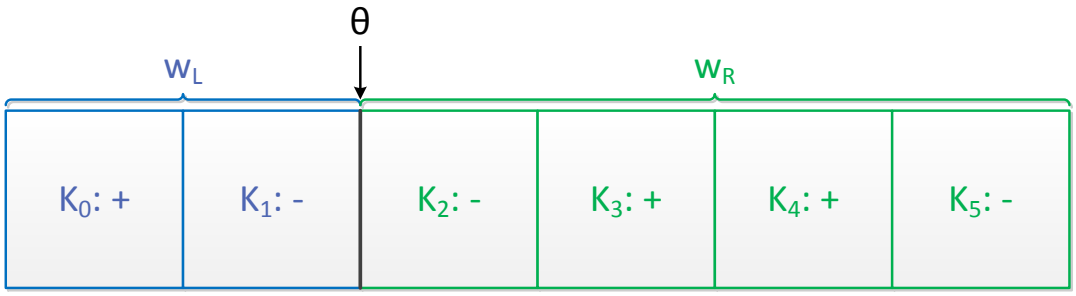


Figure 4.7: Example Diagram of a Stump Classifier Given 6 Samples.

Figure 4.7 provides an example of the structure of a stump classifier given a set of 6 samples. The stump classifier is split by the threshold, θ, giving an error of classification on either side. These errors are denoted as errors on the left e_L and right side e_R of the threshold, and are calculated as shown in Equation 4.6 and Equation 4.7,

$$e_L = \sum_{k=0, f(x_k)<\theta}^{k=K-1} w(x_k) * (y_k - \alpha_l)^2 \tag{4.6}$$

$$e_R = \sum_{k=0, f(x_k)>\theta}^{k=K-1} w(x_k) * (y_k - \alpha_r)^2 \tag{4.7}$$

$$\theta = \frac{f(x_k) + f(x_{k+1})}{2} \tag{4.8}$$

where, $f(x_k)$ is the feature value for some sample x_k, K is the total amount of samples, y_k is the sample label ($y_k = 1$ for positive samples, and $y_k = -1$ for negative samples), and α_l and α_r are sum of positive sample and negative sample weights on the left and right sides of the threshold. The final threshold for the stump classifier is determined by Equation 4.8, where the errors are at a

minimum, $min(e_l + e_r)$. In order to efficiently update the squared errors, four values are iteratively updated; the positive and negative sample values on the left side of the threshold (w_lp and w_ln, respectively) and similarly for the right side (w_rp and w_rn). The left values are updated to the current sample's label (S) as shown in Equation 4.9 and 4.10.

$$w_lp = \sum_{s=0, y_s=-1}^{s=S} w(x_s) \tag{4.9}$$

$$w_ln = \sum_{s=0, y_s=1}^{s=S} w(x_s) \tag{4.10}$$

The right positive and negative sample weights are updated according to w_lp and w_ln, as shown in Equations 4.11 and 4.12, where w_tp and w_tn are the total positive and negative weights.

$$w_rp = w_tp - w_lp \tag{4.11}$$

$$w_rn = w_tn - w_rn \tag{4.12}$$

Next, we can calculate the squared error of each side of the current threshold by using the normalized difference of the sum of positive and negative weights as expressed in Equations 4.13 and 4.14.

$$\alpha_l = \frac{w_lp - w_ln}{w_lp + w_ln} \tag{4.13}$$

$$\alpha_r = \frac{w_rp - w_rn}{w_rp + w_rn} \tag{4.14}$$

Finally, the iteratively updated squared error for each side is calculated by using Equations 4.15 and 4.16.

$$e_l = w_l p * (1 - \alpha_l)^2 + w_l n * (-1 - \alpha_l)^2 \tag{4.15}$$

$$e_r = w_r p * (1 - \alpha_l)^2 + w_r n * (-1 - \alpha_r)^2 \tag{4.16}$$

Once the ideal threshold is identified, the stump classifier returns the threshold, θ, the normalized differences of the weights, α_l and α_r, and the best feature f. Table 4.1 demonstrates the pseudo-code to calculate the best weak threshold for a given feature set and sample set.

Table 4.1: Weak Stump Classifier Pseudo-code

Calculate the best threshold for a given set of BHOG features F over positive and negative samples, P_s and N_s, respectively.

Input: F : Set of all possible BHOG features
P_m : set of (m) positive samples
N_l : set of (l) negative samples
wP_m : Normalized Positive Weights
wN_l : Normalized Negative Weights

1. loop: For every feature $f = 1 \dots F$

   (a) Initialize Values:
   $wtp = \sum_{s=1}^{m}(wP_{t,s})$,
   $wtn = \sum_{s=1}^{m}(wP_{t,s})$,
   $wlp = wln = wrp = wrn = 0$
   $e = LARGE\_VAL$

   (b) Create a sorted list: Calculate the feature value $f(s)$ for each sample $S_{(s=m+l)} = \{P_m, N_l\}$, and sort the feature values in ascending order.

   (c) loop: For every sorted sample $s = 1 \dots S$

      i. Calculate errors for current s.
      - $wlp = wlp + wP_s$
      - $wln = wln + wN_s$
      - $wrp = wtp - wlp$
      - $wrn = wtn - wln$
      - $\alpha_l = \frac{(wlp-wln)}{(wlp+wln)}$
      - $\alpha_r = \frac{(wrp-wrn)}{(wrp+wrn)}$
      - $el = wlp * (1 - \alpha_l)^2 + wln * (-1 - \alpha_l)^2$
      - $er = wrp * (1 - \alpha_r)^2 + wrn * (-1 - \alpha_r)^2$

      ii. Update threshold based on current error
      if $(el + er) < e$
      $e = (el + er)$
      $\theta = \frac{f(s)+f(s+1)}{2}$
      $f_{best} = f$
      $\alpha_{l,best} = \alpha_l$
      $\alpha_{r,best} = \alpha_r$

2. Return θ, f_{best}, $\alpha_{l,best}$, and $\alpha_{r,best}$

In order to build a set of the best features, as determined by the stump classifier, AdaBoost is used, as outlined in Table 4.2. A user defined value T is provided to determine the number of weak features to generate. The user also provides a set of positive and negative training samples. Each weak feature h is determined by the stump learner in Table 4.1, and is defined in Equation 4.17, where s is a sample, f is the feature, and α_l, α_r, and θ are defined in Equations 4.13, 4.14, and 4.8, respectively. Weights for each sample are updated as a function of h and the label of the sample, y, as shown in Equation 4.18. The weights are then normalized so that $\sum_s w_s = 1$. Once T features are generated, the final classifier in Equation 4.19 can be performed on a given sample.

$$h(\alpha_l, \alpha_r, \theta, f, s) = \begin{cases} \alpha_l & f(s) < \theta \\ \alpha_r & \text{otherwise} \end{cases} \tag{4.17}$$

$$w_{x,T+1} = w_{x,T} * e^{-y_x * h(\alpha_l, \alpha_r, \theta, f, s)} \tag{4.18}$$

Table 4.2: AdaBoost Algorithm [10]

Input: T : Number of best features
F : Set of all possible BHOG features
P_m : set of m positive samples
N_l : set of l negative samples

1. Initialize weights:

$$\text{Positive Weights} : wP_{1,s} = \{\tfrac{1}{2*m}\}$$
$$\text{Negative Weights} : wN_{1,s} = \{\tfrac{1}{2*l}\}$$

2. for t = 1 .. T:

   (a) Normalize weights,
   $$wP_{t,s} \leftarrow \frac{wP_{t,s}}{\sum_{j=1}^{m} wP_{s,j}}$$
   $$wN_{t,s} \leftarrow \frac{wN_{t,s}}{\sum_{j=1}^{l} wN_{s,j}}$$

   (b) Select the best weak classifier with respect to the weighted error, e, using the stump classifier, Table 4.1.

   (c) Define the weak classifier: $h(\alpha_l, \alpha_r, \theta, f, x)$, see Equation 4.17.

   (d) Update the weights:
   $$wP_{x,T+1} = wP_{x,T} * e^{-h(\alpha_l,\alpha_r,\theta,f,x)}$$
   $$wN_{x,T+1} = wN_{x,T} * e^{h(\alpha_l,\alpha_r,\theta,f,x)}$$

3. The final strong classifer for the node is:

$$C(x) = sign(\sum_{t=1}^{T} h_t) \tag{4.19}$$

Cascaded AdaBoost Algorithm

Viola and Jones [10] extend the AdaBoost algorithm to a cascaded architecture as shown in Figure 4.8. Each AdaBoost Classifier acts a node with a set of learned features. Every node tests a detection window with said features, and if that node fails, the detection window is determined as a failure. In order for a detection window to return positive, every node must return positive. The advantage of using a cascade over a single AdaBoost Classifier is that the cascade uses the small set of the strongest features first, and only tests on other weaker features if a node has passed. This way any false positives are quickly thrown out.

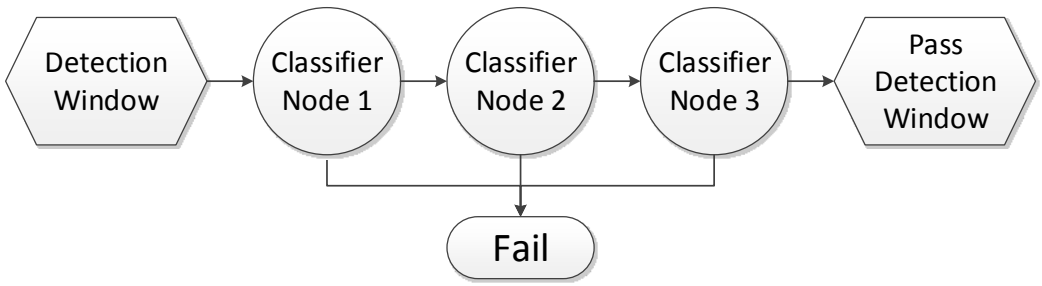


Figure 4.8: Cascade Classifier Architecture.

Selecting the ideal features is accomplished programmatically where a user defines an overall target false positive rate of the cascade, F_{target}, a maximum false positive rate per node, f_i, and minimum detection rate per node, and d_i. Table 4.3 describes the process to build the Cascaded AdaBoost classifier. The positive and negatives samples are split into two sets; the training set and evaluation set. Each positive and negative evaluation set consists of randomly selected images from the respective input samples, up to 30% the input sample size. The training set consists of the remainder input samples. The inner loop starting at 2.a selects

the features for every node. Training begins at step 2.a.i using the classifier in Table 4.2 with $T = 1$ number of features. Step 2.a.ii evaluates the current selected feature(s) to determine the current false positive rate F_i and current detection rate D_i. Each evaluation is based on Equation 4.20, where θ_t is an adjustable threshold for each cascade node. Step 2.a.iii decreases the threshold,θ_t, in order to achieve a detection rate of at least $d * D_{i-1}$, and then evaluates F_i with the updated threshold.

$$C(x) = sign((\sum_{t=1}^{T} h_t) - \theta_t) \tag{4.20}$$

Table 4.3: Building the Cascade AdaBoost Algorithm [10]

Input: F_{target} : Overall Cascade False Positive Rate.
f_{max} : Maximum false positive rate per node.
d_{min} : Minimum detection rate per node.
P_m : Set of m positive samples.
N_l : Set of l negative samples.

1. Initialize values:

   $i = 0$; $D_i = 1.0$; $F_i = 1.0$;

2. while $F_i > F_target$:

   $i = 0$; $F_i = F_{i-1}$; $T = 0$;

   (a) While $F_i > f * F_{i-1}$

      i. Use P_{train} and N_{train} to train the AdaBoost classifier in 4.2 using T features.

      ii. Evaluate the current cascade classifier on a validation set to determine F_i and D_I.

      iii. Decrease the threshold for the i^{th} classifier and re-evaluate the i_{th} cascade classifier until a detection rate of at least $d * D_{i-1}$. is reached. F_i will also be updated until the detection rate is reached.

   (b) If $F_i > F_{target}$, then evaluate the current cascaded classifier on the set of negative examples and put any false detections into the set N_{train}

4.3 Testing

The testing environment consists of a system with a Intel Core2 Quad CPU Q9550, at 2.84GHz, with 8GB of ram on a 64-bit Windows 7 Operating System. All timed tests are ran independently to provide the most accurate times.

4.3.1 Thermal Testing

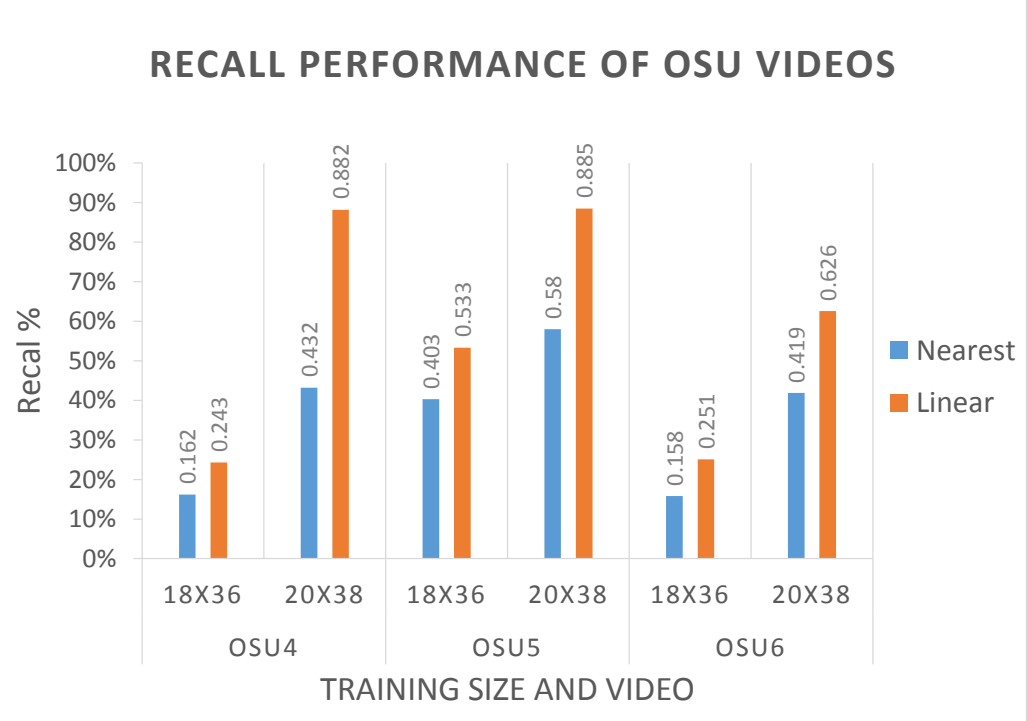
The first evaluations are made using BHOG features trained over the INRIA dataset. We first use three of the color videos from the Oklahoma State Universtity (OSU) Color-Thermal Database [7] to test the viability of the BHOG detector. A monolithic 100 feature classifier is first used to evaluate the BHOG classifier. Four monolithic classifiers are trained using two training sizes of 18x36 and 20x38. The 20x38 training set images are oversized to determine whether or not the edge effects of calculating the magnitude and orientations affects the classification. After the magnitude and orientations are calculated, the resulting images are trimmed by 1-pixel on each side for further processing. Both training sets are also trained with two bin voting schemes; the nearest scheme as shown in Section 3.3.1 and a linear voting scheme as shown in Section 3.1.1.

Table 4.4 outlines the properties of the videos tested. A stride length of two-pixels is used to scan the images, and is scaled according to a user defined scaling factor of 0.90 for a total of 52803 possible detection windows. Overlapping detections are grouped together using the OpenCv function groupRectangles(). Figure 4.9 shows that the oversized trained images continuously achieves a higher recall rate than the non-oversized trained images. This is because each integral image is computed per frame instead of per detection window, leaving any edge effects at the borders of the frame and not at each window. Additionally, a linear

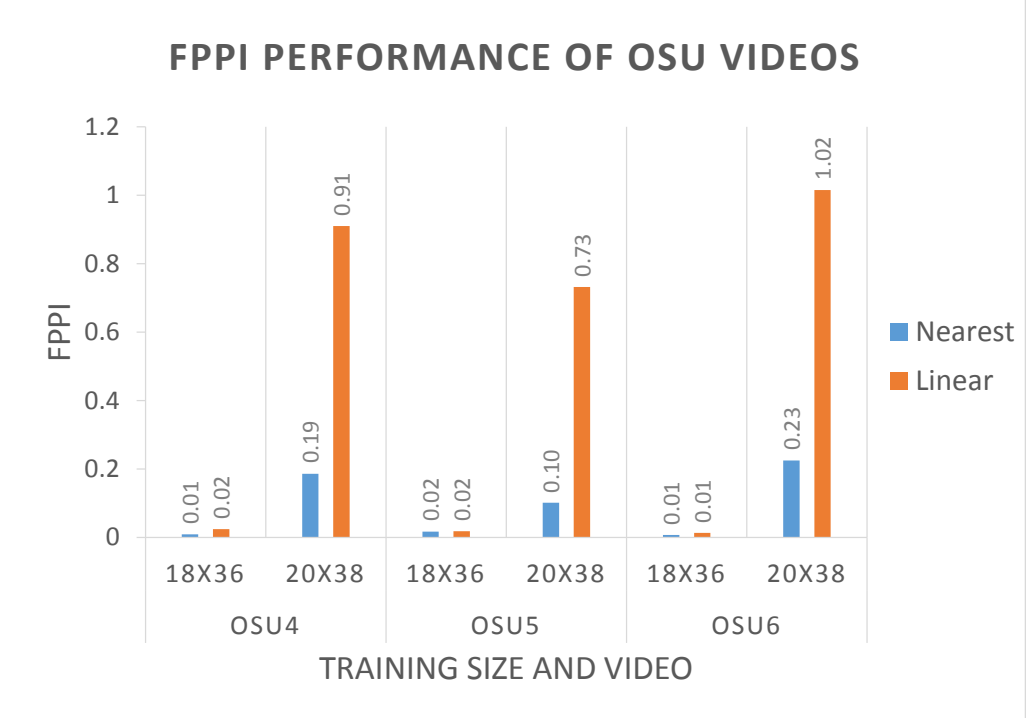
voting scheme also achieves a higher recall rate than the nearest-vote scheme. However, the performance of the oversized linearly trained classifier produces a higher FPPI.

Table 4.4: OSU Video Properties and Detection Settings

Video	Resolution (w x h)	Total Frames	Stride (pixels)	Scaling Factor	Total Detection Windows
OSU4	320 x 240	1506	2	0.9	52803
OSU5	320 x 240	2032	2	0.9	52803
OSU6	320 x 240	1651	2	0.9	52803



(a) Recall Results



(b) FPPI Results

Figure 4.9: Recall (a) and FPPI Results (b) of Different Training and Voting Methods over the OSU Dataset.

Using the data from the monolithic tests, we train a cascade of BHOG features over the INRIA dataset. We opted to train the cascade at a maximum of 14 levels, each with a maximum of 50 features to reduce training time. To demonstrate the performance of the cascade, we test each video over 10 different cascade levels. Figure 4.10 shows that the cascade has a similar detection rate to the monolithic classifier at comparable FPPI rates. Also, the processing time is substantially reduced with the Cascade having speed improvement of approximately 8, over 52803 total detection windows, as can be seen in Table 4.5. Figure 4.12 shows example frames of detections and common false positives in each frame, where the false positives tend to focus around rigid objects. OSU6 results in a 50% recall due to certain detections only occurring where contrast is sharp (clear definition of the person and background), as shown in Figure 4.11.

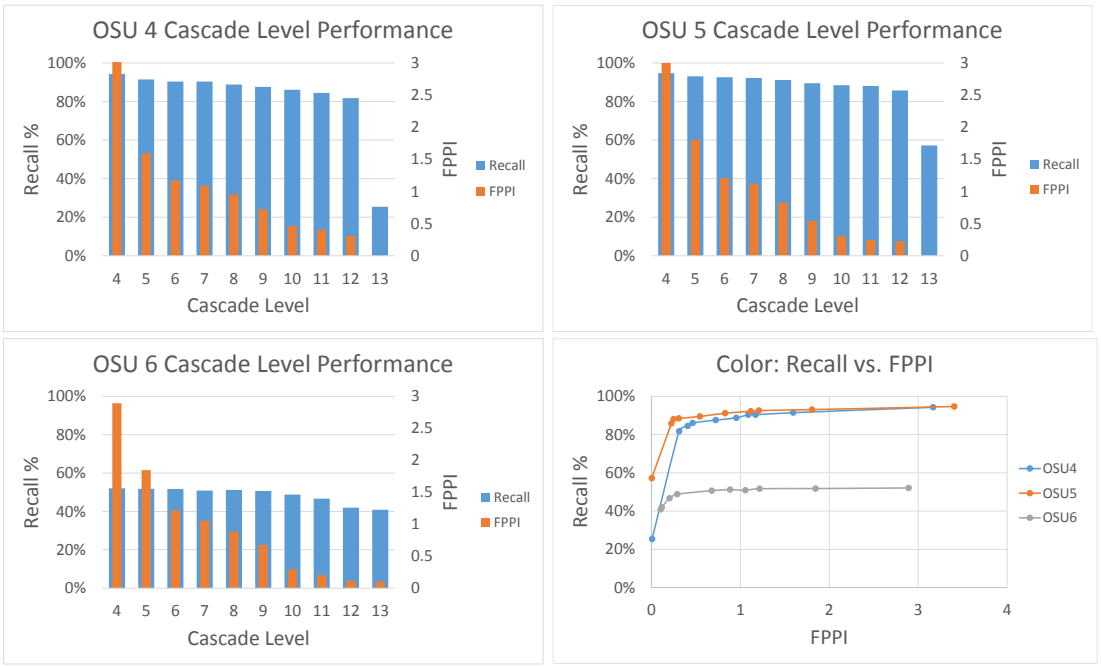


Figure 4.10: Recall and FPPI Results of OSU Videos over 10 Cascade Levels.

To further reduce the processing time, the thermal data is used to scan only portions of a video, as described in Section 3.4. The pixel stride is set to 2 pixels

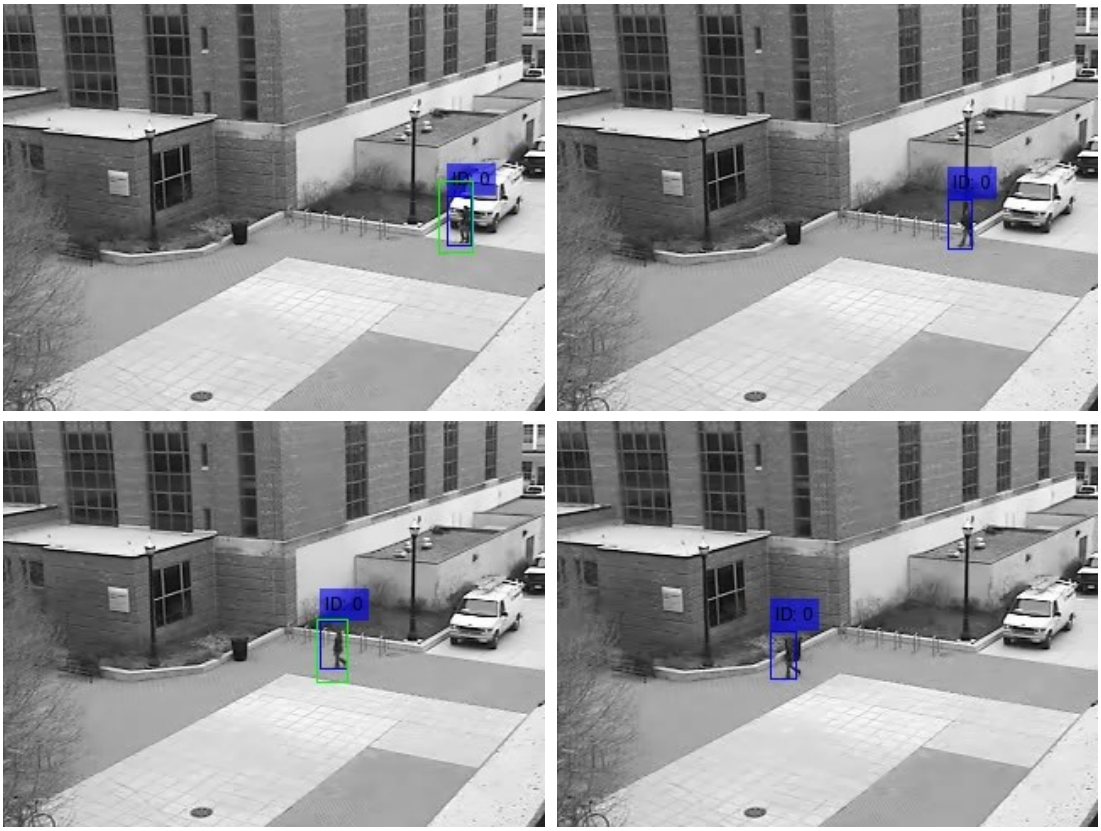


Figure 4.11: OSU6 False Positive Examples.

Table 4.5: Average Detection Times Using AdaBoost and Cascade Classification

	OSU4	OSU5	OSU6	Average
AdaBoost BHOG Time (s)	0.8383	0.846	0.8426	0.8423
Cascade BHOG Time (s)	0.1058	0.1063	0.1047	0.1056
Speed Up	7.923	7.959	8.048	7.977

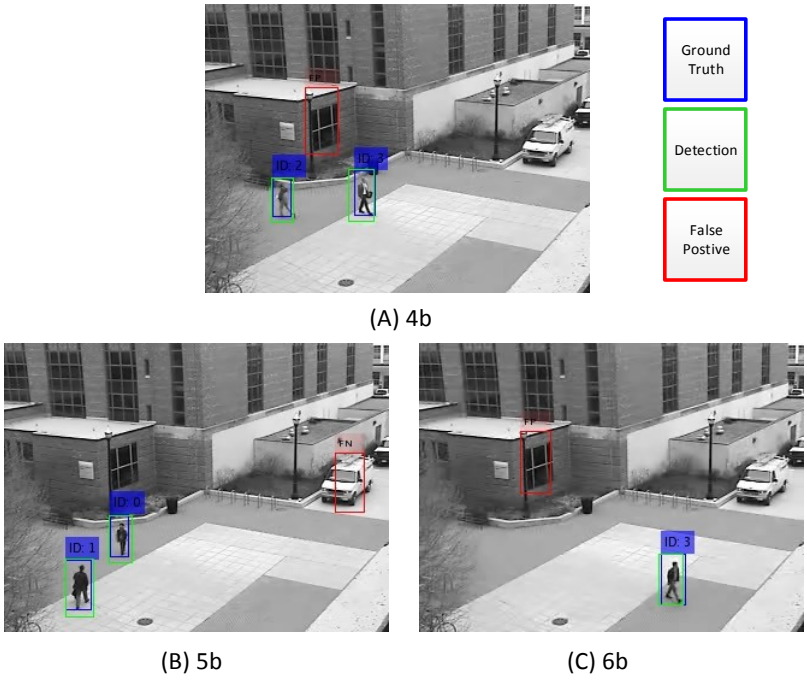


Figure 4.12: Example of Detections in OSU Video Dataset.

in both row and column directions with a scale factor of 0.9. Figure 4.13 shows the thermal Recall and FPPI results of each OSU video over 10 cascade levels, and Figure 4.14 shows a comparison with non-thermal videos at the 5th cascade level. We can see that the color and thermal combination resulted in comparable performance, with a lower recall in OSU4 and similar recall rates in both OSU5 OSU6. However, due to a smaller scan area using the thermal ROI's, the FPPI rate is dramatically lower in all videos.

The final processing time is also dependent on the video's thermal density,

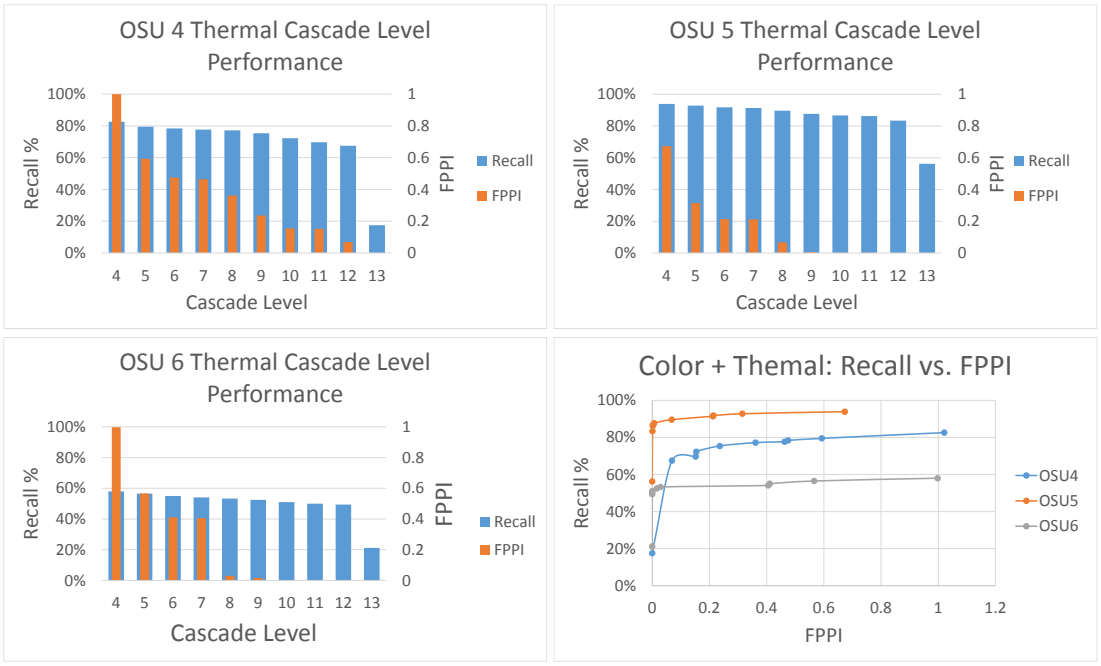


Figure 4.13: Recall and FPPI Results of OSU Thermal Videos over 10 Cascade Levels.

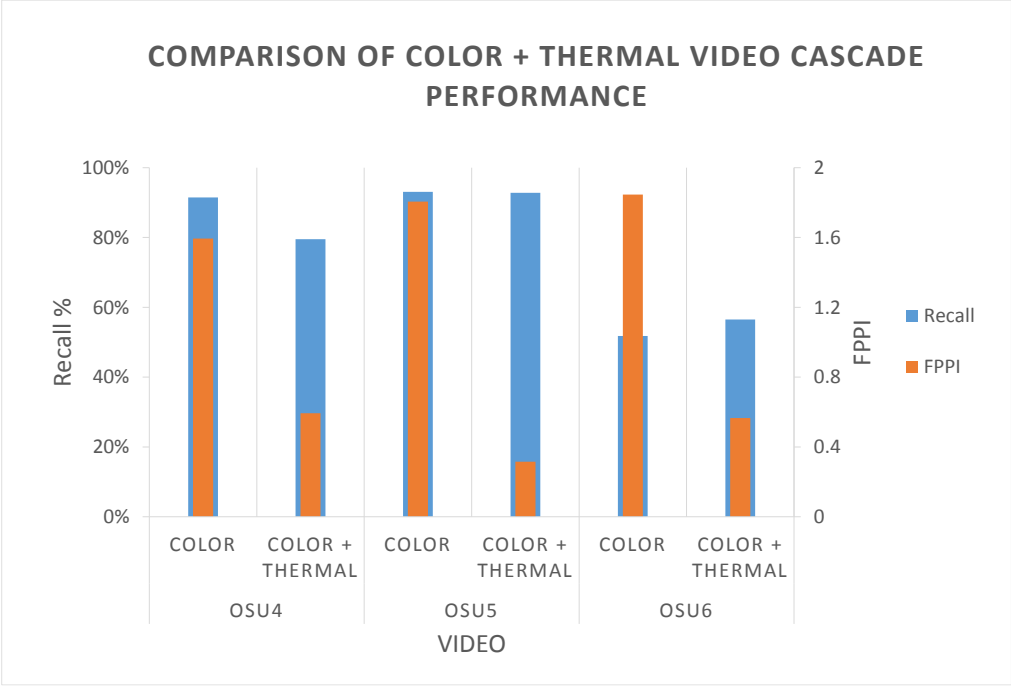


Figure 4.14: Comparison of the Color and Thermal Video Cascade Performance.

where a larger density corresponds to a larger scan area. The density is defined as the average amount of detection windows per frame, as shown in Equation 4.21 . Each video in Table 4.6 has a fairly small density, compared to the original 52803 detection windows and consequently a much lower processing time. Compared to the Cascade times in Table 4.5, we see a speed-up of approximately 2. It is expected that the speed up should be linear with the amount of detection windows, however, during thermal scanning, there are significant overlaps with nearby thermal ROI's, as well as overhead to map from the thermal image to the color image.

$$t_d = detWindows_{total}/frames_{total} \tag{4.21}$$

Table 4.6: Thermal and Color Video Processing Time Comparison

	OSU4	OSU5	OSU6	Average
Cascade BHOG Time (s)	0.1058	0.1063	0.1047	0.1056
Cascade Thermal BHOG Time (s)	0.0560	0.0549	0.0544	0.0551
Density	4210.90	3859.72	3451.64	3840.75
Speed Up	1.890	1.937	1.925	1.918

4.3.2 Bird's Eye View Testing

The next videos tests the viability of using a birds-eye view for camera placement on a UAV using the two videos from the BIWI Walking Pedestrians Dataset [20]. The video frames that are used for testing are outlined in Table 4.7, where EWAP_HOTEL and EWAP_ETH represent the two videos. Table 4.8 provides the total possible detections and the time to scan each video using BHOG features. Each video has distinct characteristics that will test where our detector

succeeds and fails. Example frames of the EWAP_HOTEL and EWAP_ETH videos are shown in Figure 4.15. EWAP_HOTEL provides a variety of pedestrians, in a well lit environment, offering various degrees of contrast with the background. Many occlusions also occur with the surrounding environment. The EWAP_ETH video is dimly lit and recorded at a high altitude, causing pedestrians to be hard to locate, however, most pedestrians are not occluded.

(a) EWAP_ETH Frame Sample (b) EWAP_HOTEL Frame Sample

Figure 4.15: BIWI Walking Pedestrians Dataset.

Table 4.7: EWAP Video Properties and Detection Settings

Videos	Resolution (w x h)	Frame Span	Stride (pixels)	Scaling Factor	Total Detection Windows
EWAP_HOTEL	720 x 576	10000-11000	2	0.9	342454
EWAP_ETH	640 x 480	8250-9250	2	0.9	249517

Table 4.8: EWAP Detection Times

Videos	Total People	Total Positives	Average Time per Frame (s)
EWAP_HOTEL	38	7638	1.75
EWAP_ETH	45	9237	1.672

We test a Cascade in the same manner as the previous tests. The pascal rating is set to 0.2 to account for the wide variety of ground truth box ratios versus the predetermined ratio of the detection boxes. We will examine the total

recall and FPPI rates. We also compare the BHOG cascade method with the OpenCV Haar-Cascade classifier and our implementation of HOG features.

With the EWAP_HOTEL video, we achieve a recall rating of 71.5% and a FPPI of 9.116 at 10 cascade levels. We run tests over 10 cascade levels to provide a better overview of how the classifier performs. Figure 4.16 shows the Recall and FPPI over the 10 cascade levels. Some of the false positives are consistent through each frame focusing on rigid structures. In Figure 4.18 we see that the lamp post, branches, and various body parts are marked as a positive detection. We also see that some occlusions are correctly identified, however, occluded objects are ignored in our analysis. Our detector also tends to focus around the head and shoulder region.

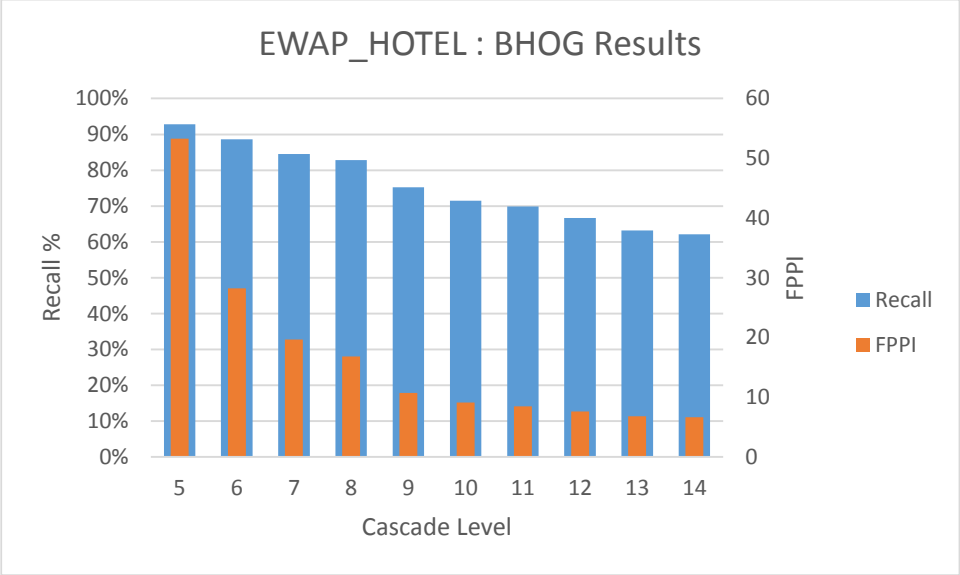


Figure 4.16: BHOG Cascade Results on the EWAP_HOTEL Video.

The EWAP_ETH video produces a recall rate of 42.15 and a FPPI rate of 23.617 at level 10 of the cascade. Figure 4.17 shows the Recall and FPPI over 10 cascade levels. In Figure 4.19 we see false positives tend to occur around rigid structures and shadows similar to the EWAP_HOTEL video. We also see

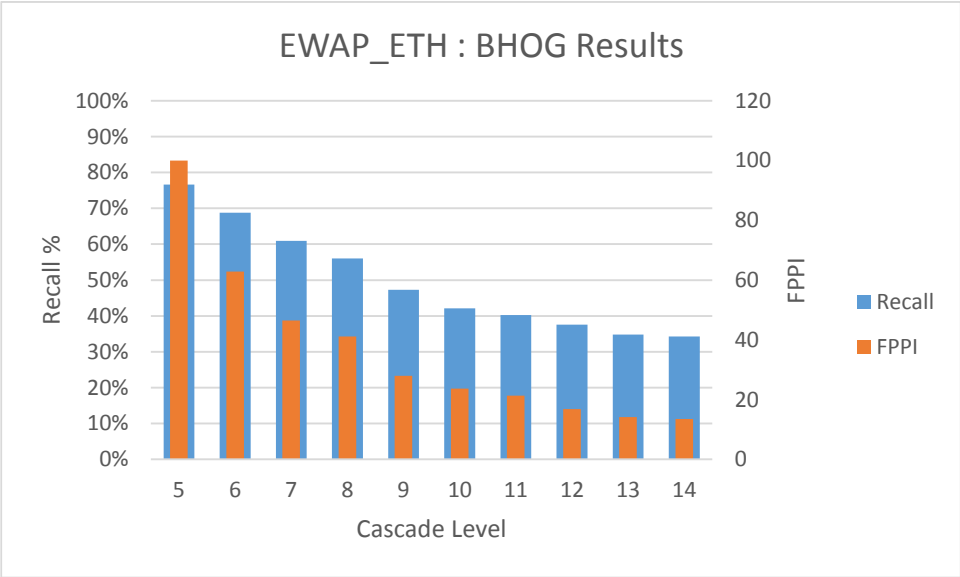


Figure 4.17: BHOG Results on the EWAP_ETH Video.

that our detector fails with the lower resolution video as compared to the higher resolution EWAP_HOTEL video.

Next, we compare the BHOG features to the Haar features using a Haar-Cascade [25] and HOG features trained on a SVM. The HOG-SVM and Haar-Cascade both use 2184 positive samples for training, and sample from a set of 1218 images for negative samples. We use the OpenCV cascade training and testing libraries for the Haar Features using similar training parameters as the BHOG features as shown in Table 4.9, with a minimum hit rate of 99.7%, a maximum false alarm rate of 75%, a maximum of 50 features per cascade node, and a total of 11 stages.

The HOG features are computed as described in Section 3.1, and are trained using OpenCV's SVM training functions, with the parameters shown in Table 4.10, and is bootstrapped to reduce false positives. The SVM does not use validation as described in training the BHOG features (Section 4.2), so 3654 total

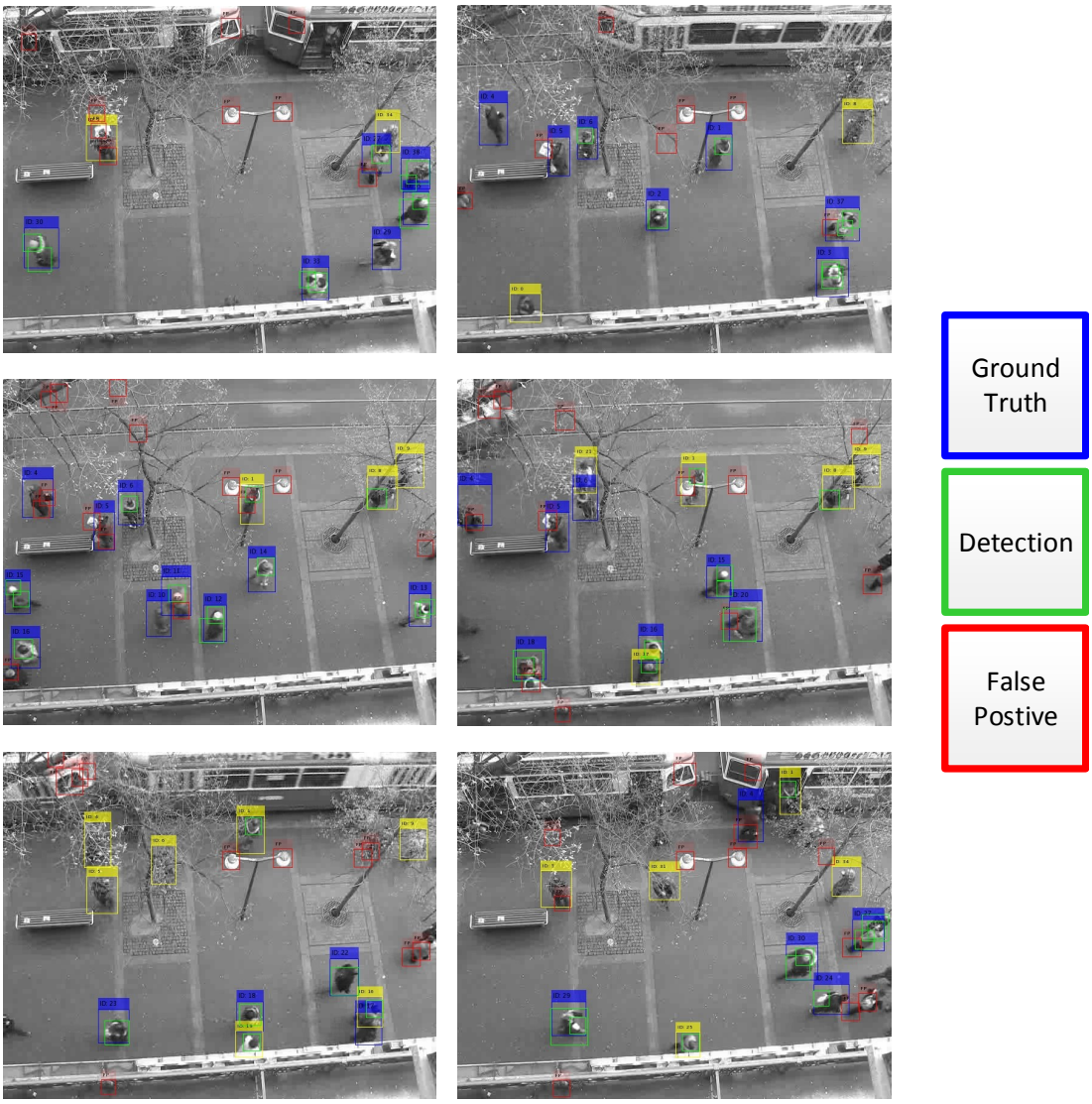


Figure 4.18: EWAP_HOTEL Annotated Frames.

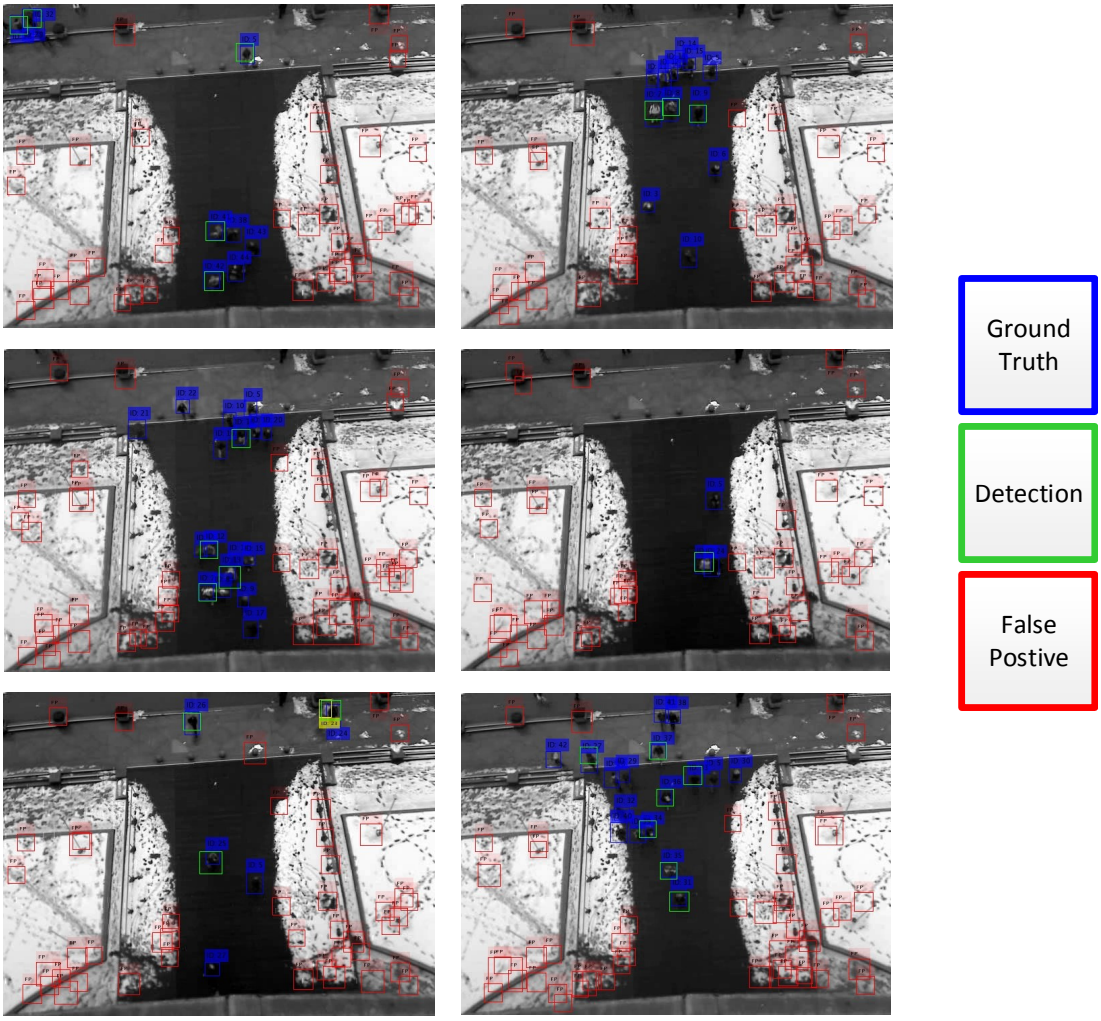


Figure 4.19: EWAP_ETH Annotated Frames.

negative samples are generated. The SVM type is an ϵ-Support Vector Regression, where the distance between feature vectors from the training set and the fitting hyper-plane must be less than p, and outliers use a penalty multiplier of C. The chosen SVM kernel type is linear.

Table 4.9: Training Parameters for OpenCV Haar Cascade Training

	Minimum Hit Rate	Max False Alarm Rate	Max Weak Count	Number of Stages
Haar-Cascade	99.7%	75%	50	40

The results of the HOG-SVM and Haar-Cascade classifiers are shown in Ta-

Table 4.10: OpenCV SVM Training Parameters

	SVM Type	Kernel Type	c Value	p Parameter
HOG-SVM	EPS_SVR	Linear	0.01	0.1

ble 4.12 and Table 4.11, for both the EWAP_HOTEL and EWAP_ETH videos, respectively. The Haar-Cascade out performs the BHOG features at higher levels, and both the BHOG and Haar-Cascade models outperform our HOG-SVM model. The Haar-Cascade also exhibits poor detection in the EWAP_ETH video.

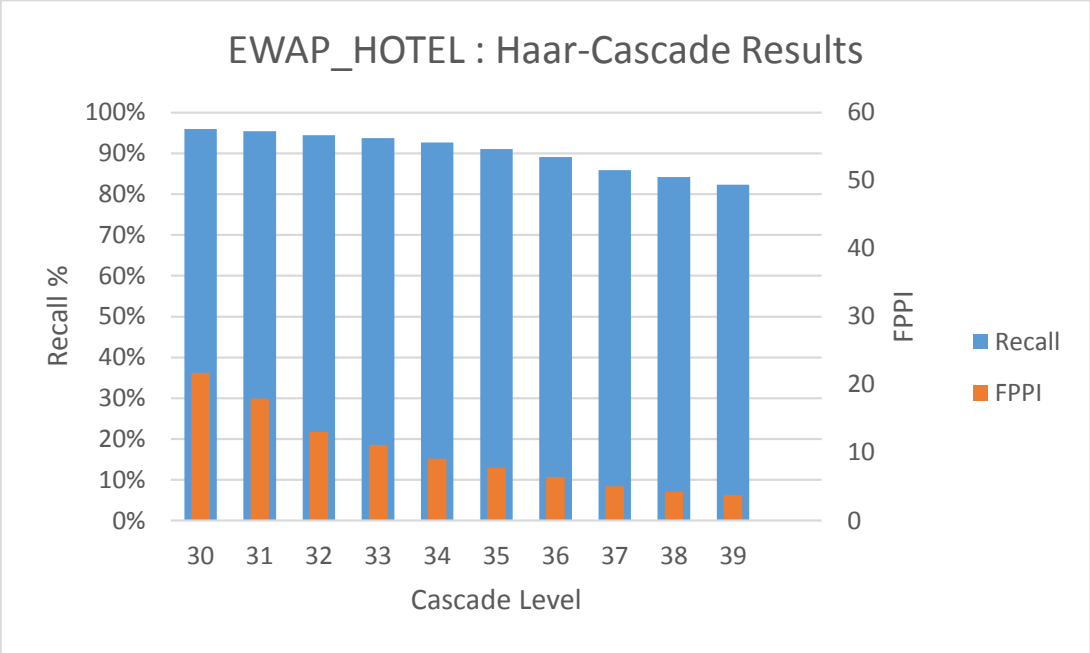
Table 4.11: Results of Comparisons between HOG-SVM, Haar-Cascade, and BHOG classifiers on the EWAP_HOTEL Video

HOTEL	RECALL	FPPI
BHOG 10 Levels	71.5%	9.116
Haar-Cascade 34 Levels	92.7%	9.117
HOG-SVM	37%	8.552

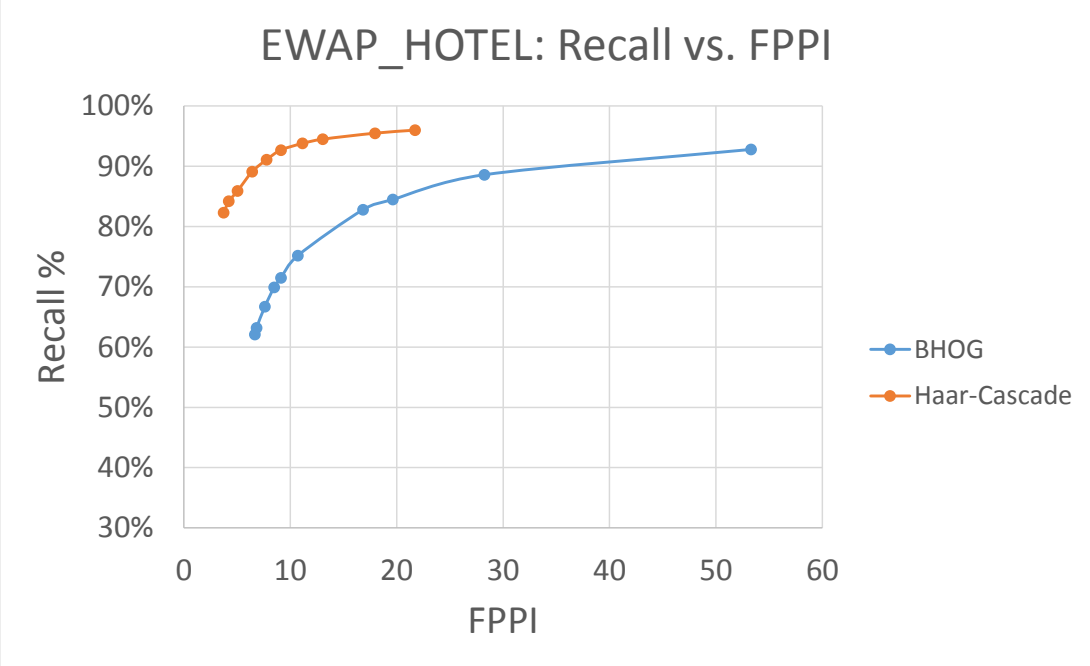
Table 4.12: Results of Comparisons between HOG-SVM, Haar-Cascade, and BHOG classifiers on the EWAP_ETH Video

ETH	RECALL	FPPI
BHOG 10 Levels	42.1 %	23,617
Haar-Cascade 34 levels	55.1%	60.838
HOG-SVM	35.7%	27.668

To provide better comparison with the BHOG and Haar-Cascade, we compare 10 levels of both detectors. Figure 4.20 and Figure 4.21 provide the statistics on the EWAP_HOTEL and EWAP_ETH videos, respectively. We see that Haar-Cascade outperforms the BHOG cascade at similar FPPI rates. For example, at 9.1 FPPI, the Haar-Cascade achieves a recall rate of 92.7%, where as the BHOG cascade achieves a recall rate of 71.5%.

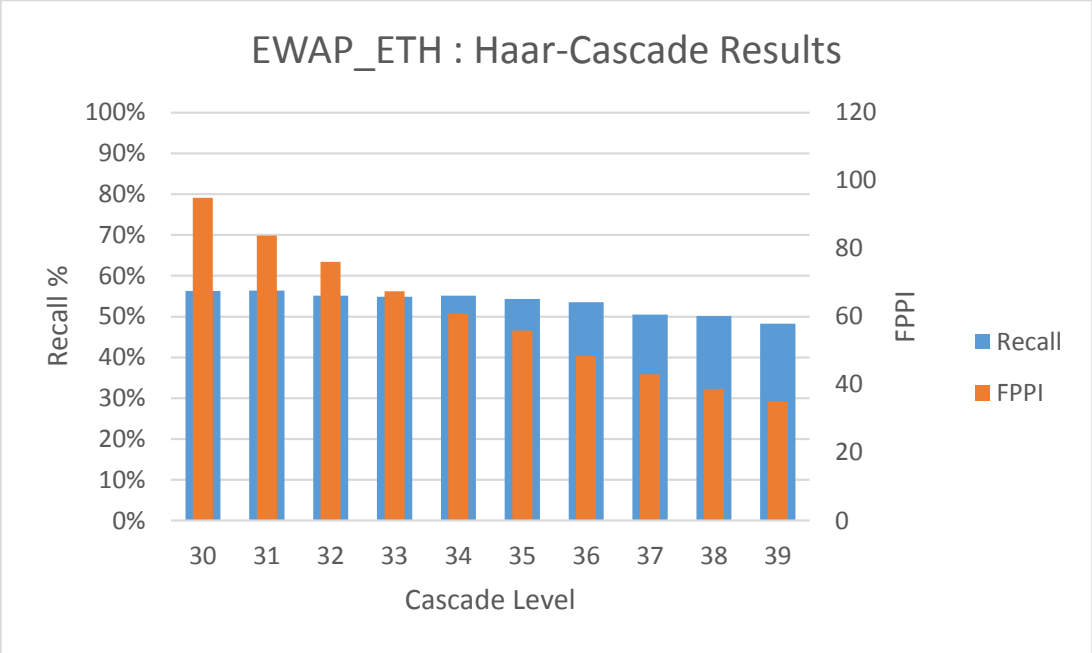


(a) EWAP_ETH Results over 10 Cascade Levels

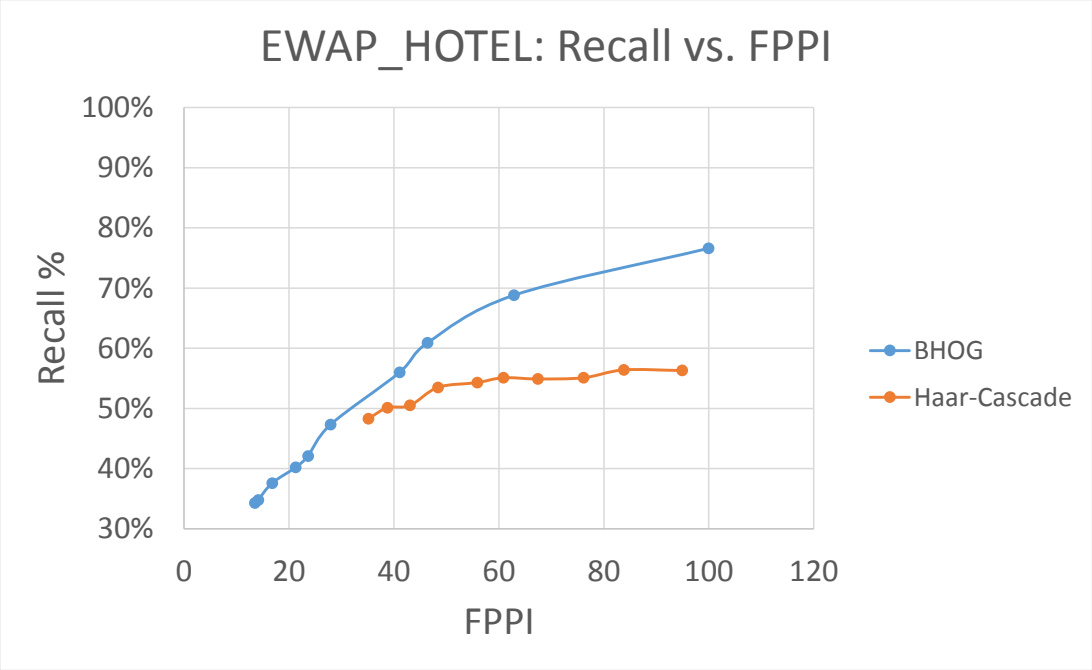


(b) EWAP_HOTEL Recall vs. FPPI Performance

Figure 4.20: Haar-Cascade Results on the EWAP_HOTEL Video.



(a) EWAP_ETH Results over 10 Cascade Levels



(b) EWAP_ETH Recall vs. FPPI Performance

Figure 4.21: Haar-Cascade Results on the EWAP_ETH Video.

Chapter 5

Conclusion and Future Work

In this work we analyzed the use of thermal segmentation with a BHOG feature classifier, using the INRIA dataset for thermal testing and our own compiled dataset for birds-eye view testing. We demonstrated that the thermal segmentation has a speed up of 2 with videos of 320x240 resolution. The speed up however is highly dependent on the thermal density of the environment. Thermal segmentation also decreases the FPPI by only focusing on the hottest parts of the image. The BHOG classifier was also able to classify people in the OSU video dataset with recall rates of 91.5%, 93.1%, and 51.8%, and FPPI of 1.59, 1.81, 1.85, in OSU4, OSU5 and OSU6 videos, respectively.

Our dataset has shown promise with the higher resolution EWAP_HOTEL video. The BHOG feature set has an overall recall of 74.3% and 9.811 FPPI at 10 cascade levels. Most of the false positives tend to be around rigid objects, such as limbs and tree branches. Our dataset unfortunately fails in a lower resolution, higher altitude scenario where people are harder to locate, with a recall rate of 42.1% recall rate and FPPI of 23.617 at 10 cascade levels.

Overall the Haar-Cascade outperforms the BHOG features with our BEV dataset. At 34 cascade levels, the Haar-Cascade exhibits a recall rate of 92.7% and a FPPI of 9.117 in the EWAP_HOTEL video. Similarly, the Haar-Cascade also fails in the EWAP_ETH video with a recall rate of 55.1% and a FPPI of 60.838. The failure of the EWAP_ETH can be attributed to the training of the features over our dataset, since our dataset does not include enough low resolution

training examples.

There are many methods to explore for future work for aerial identification of a human. In order to build upon the work presented in this project, more positive lower resolution training samples can potentially improve the classifier with lower resolution images. Our dataset also is trained using all positions of a human which introduces a lot of noise for training, so in order to improve upon this, future tests can focus on training for one pose. However, training for one pose would also mean exploring rotation invariant detectors or other machine learning techniques to handle multiple poses. The speed of an object detector can also be improved through hardware, such as Graphics Processing Unit (GPU), or Field Programmable Gate Array (FPGA) to compute object features. In addition, other multi-modal processes can be investigated to improve on the color and thermal data. Along with segmenting for regions of interests, the thermal data can also be used to enhance the features through image fusion techniques.

BIBLIOGRAPHY

[1] A threshold selection method from gray-level histograms. *Systems, Man and Cybernetics, IEEE Transactions on*, 9(1):62–66, Jan 1979.

[2] Nasa-std-3000 man systems integration standards. `http://msis.jsc.nasa.gov/volume1.htm`, 2008.

[3] Australian Maritime Safety Authority (AMSA). *National Search and Rescue Manual*, 1 edition, dec 2013. Australian SAR Manual.

[4] S. Bhandari, K. Ortega, D. Stone, and S. Conant. Cal poly pomona's unmanned aerial system design approach and implementation for use in the 2012 auvsi student unmanned aerial systems (suas) competition. 2012.

[5] N. Dalal and B. 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 vol. 1, June 2005.

[6] D. D. David Mihalcik and R. Ahuja. The video performance evaluation resource, http://viper-toolkit.sourceforge.net/, 2006.

[7] J. W. Davis and V. Sharma. Background-subtraction using contour-based fusion of thermal and visible imagery. *Comput. Vis. Image Underst.*, 106(2-3):162–182, May 2007.

[8] Department of the Navy Office of the Chief Operations. *Navy Search and Rescue (SAR) Manual*, apr 2009. US Navy Search and Rescue Manual.

[9] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A

benchmark. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 304–311, June 2009.

[10] T. Ephraim, T. Himmelman, and K. Siddiqi. Real-time viola-jones face detection in a web browser. In *Computer and Robot Vision, 2009. CRV '09. Canadian Conference on*, pages 321–328, May 2009.

[11] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[12] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[13] H. Flynn and S. Cameron. Multi-modal people detection from aerial video. In R. Burduk, K. Jackowski, M. Kurzynski, M. Wozniak, and A. Zolnierek, editors, *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013*, volume 226 of *Advances in Intelligent Systems and Computing*, pages 815–824. Springer International Publishing, 2013.

[14] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[15] A. Gaszczak, T. P. Breckon, and J. Han. Real-time people and vehicle detection from uav imagery. In *Proc. SPIE*, volume 7878, pages 78780B–78780B–13, 2011.

[16] H.-X. Jia and Y.-J. Zhang. Fast human detection by boosting histograms

of oriented gradients. In *Image and Graphics, 2007. ICIG 2007. Fourth International Conference on*, pages 683–688, Aug 2007.

[17] R. Lienhart, E. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *In DAGM 25th Pattern Recognition Symposium*, pages 297–304, 2003.

[18] E. H. Martin. *Introduction to Search and Rescue*. National Association for Search and Rescue, second edition, may 2008.

[19] O. Oreifej, R. Mehran, and M. Shah. Human identity recognition in aerial images. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 709–716, June 2010.

[20] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 261–268, Sept 2009.

[21] J. qing Zhu and C. hui Cai. Real-time face detection using gentle adaboost algorithm and nesting cascade structure. In *Intelligent Signal Processing and Communications Systems (ISPACS), 2012 International Symposium on*, pages 33–37, Nov 2012.

[22] P. Rudol and P. 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, March 2008.

[23] A. Sato. *The RMAX Helicopter UAV*. Yamaha Motor CO., LTD., 2003. RMAX Specifications.

[24] F. Schmidt and S. Hinz. A scheme for the detection and tracking of people tuned for aerial image sequences. In U. Stilla, F. Rottensteiner, H. Mayer,

B. Jutzi, and M. Butenuth, editors, *Photogrammetric Image Analysis*, volume 6952 of *Lecture Notes in Computer Science*, pages 257–270. Springer Berlin Heidelberg, 2011.

[25] P. Viola and M. 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–I–518 vol.1, 2001.

[26] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 734–741 vol.2, Oct 2003.

[27] X. Wang, T. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 32–39, Sept 2009.

[28] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. 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, 2006.