PEDESTRIAN DETECTION IN LOW-RESOLUTION VIDEOS

by

Hisham Sager

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Electrical Engineering).

Golden, Colorado

Date _______________

Signed: _____________________________

Hisham Sager

Signed: _____________________________

Dr. William Hoff

Thesis Advisor

Golden, Colorado

Date _______________

Signed: _____________________________

Dr. Randy Haupt

Professor and Head

Department of Electrical Engineering and Computer Science

ABSTRACT

In this dissertation, we address the problem of detecting pedestrians in low-resolution videos taken by stationary cameras and cameras mounted on aerial platforms. The target pedestrians are as small as 20 pixels in height. Pedestrian detection is a key problem in computer vision, and has several applications such as search and rescue, law enforcement and general surveillance, and military applications against enemies. The most common approach for pedestrian detection is to use high resolution single frame, and there is relatively little work focused on pedestrian detection in low resolution scenarios. Under more realistic and challenging conditions, performance of the available approaches degrades rapidly, while approaches that are based on appearance cues mostly fail since the pedestrians are only a few pixels in height. Detecting pedestrians is a challenging task, since pedestrians are non-rigid objects, ranging over a high variability of poses, appearances, and scales. The detection task typically takes place at scenes with a lot of background clutter which require high classification precision. The task becomes more challenging in aerial videos, due to the rapid change in viewpoints and scales, which makes video stabilization more difficult. Furthermore, the much larger search space and the small number of pixels on the target make it difficult to distinguish pedestrians from background clutter. In this problem domain, the existing state-of-the-art pedestrian detection methods have poor performance. Therefore, there is a clear need for a method that can detect pedestrians in low resolution imagery.

To overcome these challenges, we propose a novel detection method that recognizes pedestrians in short sequences of frames (about ½ second in duration). We extend the single-frame HOG-based detector to a multiple frames detector. Our approach uses video stabilization to detect potential moving objects. A classifier is then applied to volumetric normalized features extracted from spatio-temporal volumes surrounding the potential moving objects. On several challenging stationary and aerial video datasets (PETS 2001, Stationary VIRAT, UCF-2009, UCF-2007, and aerial VIRAT), our detection accuracy is significantly better than the standard single-frame HOG-based detection method, and outperforms the previously published detector that uses Haar-like features at the same low false positive rates.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

To my parents, my wife, and my children.

ACKNOWLEDGEMENTS

I would like to thank **Dr. William Hoff** for his guidance and support. His constant motivation and faith in my ability was a very important factor in the completion of this work.

I would also like to thank **my wife** for her continuous support for more than five years, remaining away from her family that was 10,000 kilometers distant.

I am also grateful to **Dr. Hua Wang** for his advice, and I am so thankful to all of my thesis committee members for serving on the committee:

Dr. Kevin Moore, Dean of the College of Engineering.

Dr. Christian Debrunner, Lockheed Martin Company.

Dr. Paul Martin, Professor in the Math Department.

Dr. Hua Wang, Professor in the Department of Electrical Engineering and Computer Science.

I would like to express my thanks to the Ministry of Higher Education and Scientific Research of my country Libya who sponsored my graduate program.

I would also like to thank all of the professors who taught me in the Department of Electrical Engineering and Computer Science and thank my colleagues for their support.

CHAPTER 1: INTRODUCTION

The task of detecting people in videos has attracted growing interest from both academia and industry, and is an important and active area of research. In this thesis, we address the special problem of detecting pedestrians in videos. By "*pedestrians*", we mean upright people who are walking. Much current work on pedestrian recognition focuses on detection by cameras mounted on moving cars, motivated by the need for safe operation of self-driving cars. Another scenario is detecting pedestrians in surveillance videos. In this case, the videos are taken by stationary outdoor cameras or by moving cameras mounted on an aerial platform such as a helicopter or unmanned aerial vehicle (UAV). Unlike the automotive scenario, in surveillance videos the camera is typically looking down on the scene at an oblique angle, and the pedestrians can be quite small in the images. In this thesis, we focus on the latter scenario; *i.e.*, that of surveillance video.

Pedestrian detection in surveillance videos has many applications, including commercial, search and rescue, law enforcement, border monitoring, and military applications. For example, pedestrian detection offers ways to measure visitor traffic that flows into and around retail stores, malls and shopping centers. Pedestrian detection can help maximize the efficiency and effectiveness of businesses and provides insight into the movement of individuals. It is not limited to the entrance point of a company's business, but has a wide range of applications that provide information to management on the volume and flow of people throughout a location [1]. Figure 1.1 shows some examples of different applications.

Figure 1.1: Examples of surveillance scenarios: (a) Aerial video of open plaza. (b) A street surveillance stationary camera. (c) Military application, aerial VIRAT dataset.

The problem of detecting pedestrians in surveillance video can be challenging for many reasons: There is a wide range of pedestrian poses and appearance, including variety of clothing. The lighting can vary and shadows can be present. Background clutter can have a similar appearance to pedestrians. Pedestrians can be partially occluded by other objects, or by other pedestrians.

As discussed in Chapter 2, the most successful approaches to this problem use histogram of oriented gradient (HOG) features, with a support vector machine (SVM) classifier [2]. The approach requires many labeled training example images of people. Extensions to this method detect parts of the person (*e.g.*, head, arms, and torso) instead of just looking for the overall shape *e.g.* [3]. However, these methods require fairly high resolution images of people.

In the case of surveillance video, the size of pedestrians can vary widely in the image, due to the distance from the camera. When the size of a pedestrian becomes very small, many shape details are lost, and it is difficult to distinguish a pedestrian from a non-pedestrian. For example, Figure 1.2 shows an example pedestrian at four different resolution levels. As described in Chapter 2, existing algorithms for pedestrian detection do fairly well for high resolution images, but performance degrades dramatically when the height of pedestrians is 30 pixels or less.

(a) (b)

Figure 1.2: An example pedestrian at four different resolution levels: The height of the pedestrian is 140, 50, 20, and 10 pixels tall. (a) Images at actual size. (b) The same images as in (a), except that the images are stretched for easier visualization. The example is from VIRAT dataset [7].

The task of pedestrian detection in aerial videos is even more difficult than in videos captured by stationary cameras. The camera is often higher, and the size of pedestrians is therefore smaller. The effective resolution of the video is often degraded due to motion blur and haze, further reducing the available visual information on shape and appearance.

One approach to detecting pedestrians is to first detect moving objects in the scene. In this approach, an estimate of the background image is computed (often called a *background model*). Then moving objects in the scene are detected by finding the difference between the current frame and the background model (*e.g.*, [4]). These foreground regions could be due to moving objects such as pedestrians. However, other moving objects (such as cars, blowing trees) can also cause foreground regions to be detected. Sensor noise can also cause false detection of moving objects. In the case of aerial video, where the camera is continuously moving, the current image must first be registered to the background image. Misregistration can cause false foreground objects to be detected. At areas of high contrast such as edges of buildings, a slight misregistration can cause false foreground objects. Thus, simply detecting foreground regions is not sufficient for pedestrian detection, although it does help reduce the search space (*i.e.*, one need only search for pedestrians in the vicinity of detected foreground image regions).

Another approach is to do "detection by tracking". As discussed in Chapter 2, detections over multiple images can be associated and formed into "tracks". The tracker must perform object correspondence from frame to frame to generate the tracks. Correspondence can be done using shape and appearance features, as well as a motion model of the moving object (*i.e.*, velocity). With this method, detections of objects as small as a few pixels in size can be formed into tracks. Discrimination of "pedestrian" *vs.* "non-pedestrian" for very small objects can be done using the motion model – *i.e.*, if the object moves at about the expected speed of a pedestrian, then it must be a pedestrian. One problem with this method is that it cannot distinguish between a pedestrian and a non-pedestrian moving at about the same speed.

Another problem with the "detection by tracking" approach is that it requires a fairly long sequence of images of the scene, in order to build up tracks. For example, the approach of [5] can detect and track vehicles and people only a few pixels in size, but uses sequences ranging from tens of seconds to minutes long. In the case of aerial video, sequences of this length may not be available. In the case of a camera mounted on a rapidly moving helicopter or UAV, the camera is continuously translating and rotating, and occasionally undergoes large amplitude rotations. The camera does not dwell for long on a particular portion of the scene. Thus, algorithms that rely on a long sequence of observations to build up a motion track model may not be applicable.

As an example, Figure 1.3 shows snapshots from a video taken from a small quadrotor UAV flying rapidly over a field [6]. The camera moves erratically, undergoing large amplitude rotations and translations. As a result, people are usually only within the field of view for a short time (as briefly as several seconds). Although the size of people varies due to the changing altitude of the camera, the height of people is often as small as 20 pixels tall.

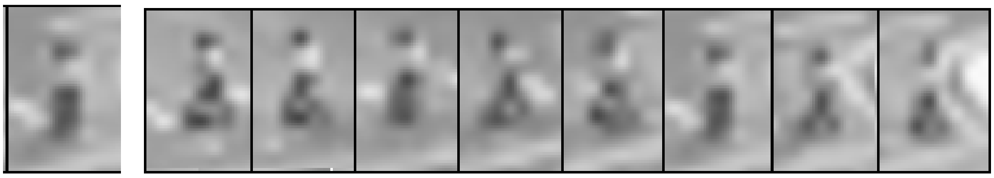
Figure 1.3: Example images from aerial video (taken about 4 seconds apart). Size of images is 380×640 pixels.

As another example, Figure 1.4 shows snapshots from an aerial video sequence from VIRAT dataset [7]. The images are roughly 10 seconds apart in time. This example shows some of the characteristics of aerial videos such as changing viewpoints and scales, which presents crucial challenges to the task of video stabilization and dealing with any imperfect pixel-to-pixel alignment. The example also shows the variety of objects in the scene; this includes various types of facilities, such as buildings, storage, and parking lots, as well as vehicles and people. The images are 640×480 pixels, taken at a 30Hz frame rate. Although the zoom changes occasionally, the typical height of people is about 20 pixels tall. Detecting people in images such as these is extremely difficult, due to the low resolution.

Figure1.4: Sample image shots changing over time due to flight motion.

1.1 Approach

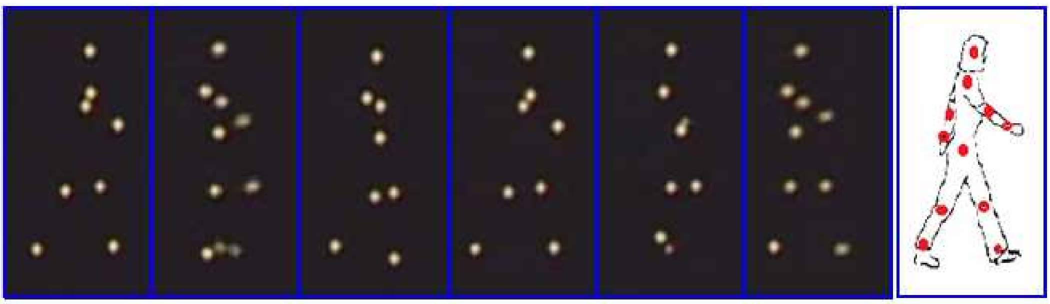
Although it is very difficult to recognize a person in a single low-resolution image, the task is much easier when a short sequence of images is used. For example, Figure 1.5 (a) shows a single low-resolution frame in which it is difficult to recognize the object. The right portion of the figure is a sequence of frames in which a subject is performing a recognizable movement; in this case, walking. Despite the deficiency of recognizable features in the static image, the movement can be easily recognized when the sequence is put in motion on a screen.



(a) (b)

Figure 1.5: (a) Single frame. (b) Sequence of frames from video of walking person.

This phenomenon is well known from the pioneering work of Johansson [8], whose moving light display (MLD) experiments showed convincingly that the human visual system can easily recognize people in image sequences, even if the images contain only a few bright spots (attached to their joints and represented with the red dots in Figure 1.6). A static image of spots is meaningless to observers, while a sequence of images creates a vivid perception of a person walking, running, dancing, *etc.* The media upon which these words are printed precludes the reader from experiencing the impact of viewing the video.



(a) (b)

Figure 1.6: (a) Dot configuration of a walking subject. (b) Bright spots attached to subject joints.

Inspired by these examples, we can use the additional information provided by motion to compensate for the lack of visual details in a single image. The approach of our work (described in Chapter 3) uses features extracted from a spatiotemporal volume of images. A volume composed of up to *N* "slices" is used, where each slice is a small subimage window of size of 32×32 pixels. This volume represents a duration of about one second or less (Figure 1.7). As in the situation with the human visual system, the information contained in the additional images should help the system recognize the pedestrian.

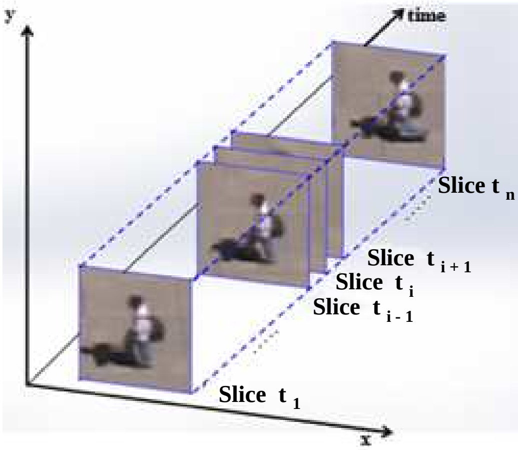
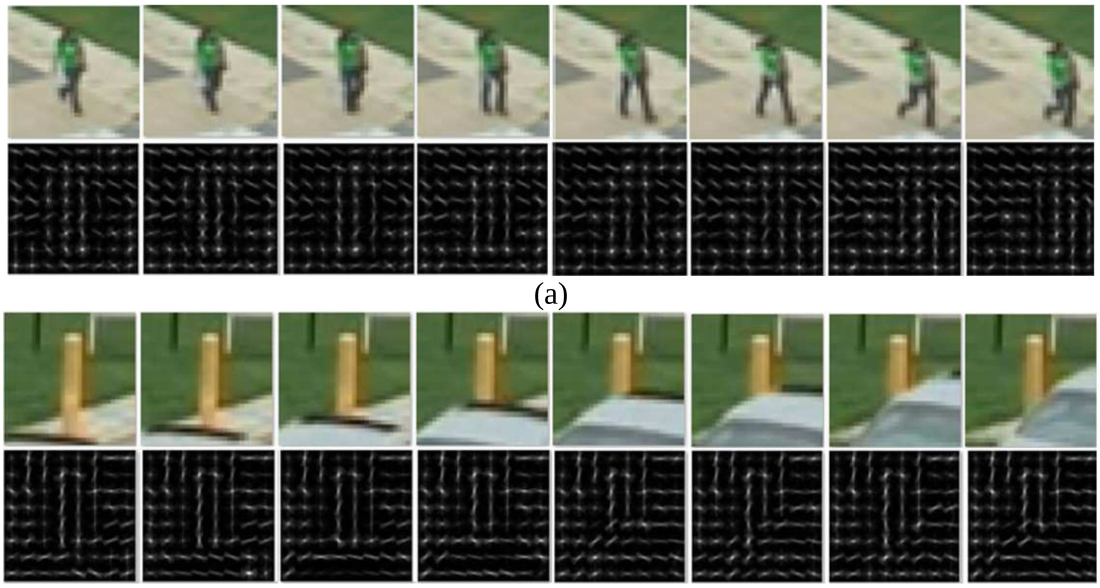


Figure 1.7: Spatiotemporal volume of images (slices) is used to gather more information about a moving object. The collection of these slices is called a volume; which is shown with the blue lines in the figure.

In this thesis, we take the single-frame HOG-based detector commonly used in pedestrian detection and extend it to multiple frames. Our approach (described in Chapter 3) uses HOG features extracted from a spatiotemporal volume of images. The idea is that the motion of a person walking is distinctive, and we can train a classifier to recognize the temporal sequence of feature vectors within the volume.

As an example, consider the images of a walking person shown in the top row of Figure 1.8. The corresponding HOG features are shown in the second row. The third row shows images of a region containing a moving car. The corresponding HOG features (bottom row) of the negative example are visually quite different from those of the positive example. This example indicates that it may be possible to train a classifier to recognize the pattern of features of a pedestrian from that of non-pedestrian.



(a)

(b)

Figure 1.8: (a) Positive example (pedestrian). Images are 32x32 pixels. (b) Negative example (part of a car passing by a post). In these two examples: the top row is a sequence of slices subsampled from a longer sequence of 32 slices (1 second duration), and the bottom row shows the HOG features of the sequence.

A possible difficulty in using short image sequences for detection is that the space of possible motions of a pedestrian is potentially large. A person can walk with different speeds, amplitudes, and gait patterns. Their arms can be swinging naturally or be carrying something. Although generally periodic, the pattern of motion can vary over time. For example, a pedestrian can turn part way through the sequence, change from a walk to a run, or change from a walk to a stop.

We address these difficulties in two ways: First we keep the sequence length fairly short (*i.e.*, one second or less). This limits the amount that motion can vary over that time duration. However, it is still possible to have examples where the motion variability is large enough to cause detection failure, as shown in Chapter 4. The second way we address the difficulties is to use a large number of training examples to train the classifier. We need to have sufficient examples that cover the range of possible motions that a pedestrian can have.

An overview of the detector algorithm is shown in Figure 1.9. The algorithm starts with stabilizing the video by registering each frame to a reference frame. Secondly, a background subtraction algorithm is used to identify foreground "*regions of interest*" (ROIs), which correspond to potential moving objects. Then a novel multi-frame HOG-based detector is used to search for pedestrians in the final ROIs.

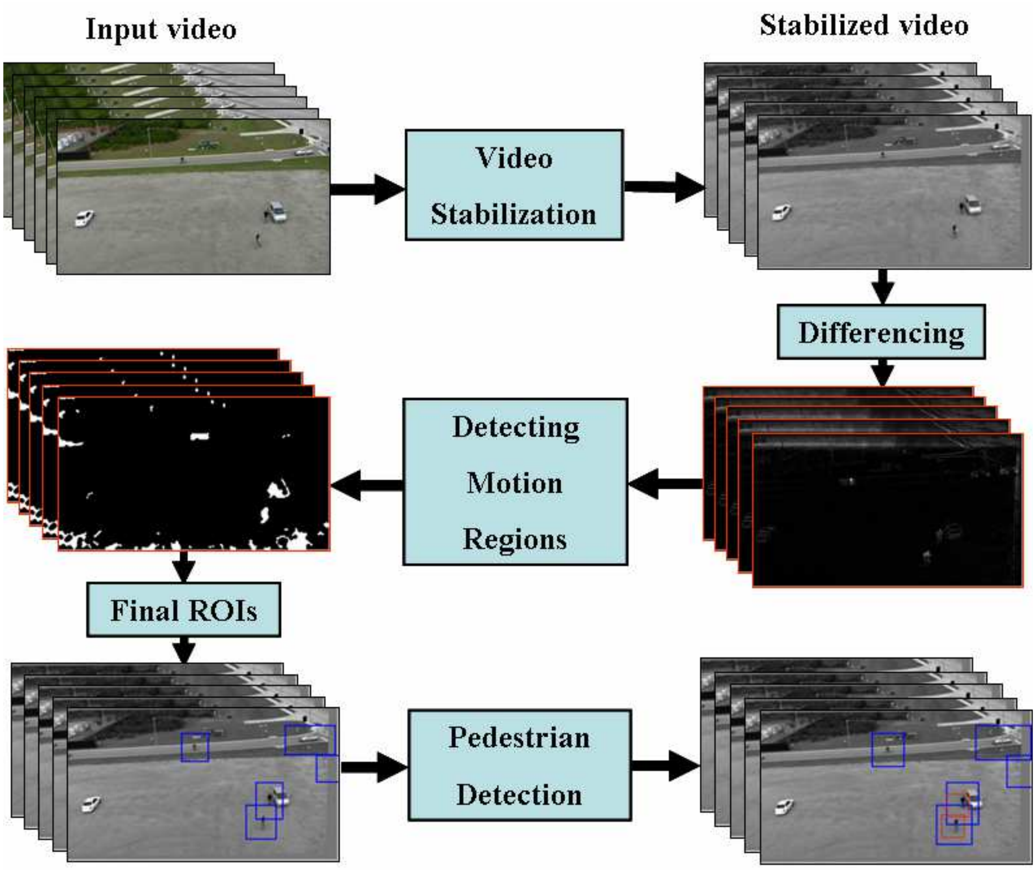


Figure 1.9: After stabilizing the video, we find the motion regions using a background subtraction algorithm. A trained multi-frame HOG-based detector is used to recognize pedestrians around these motion regions.

1.2 Objective and Contributions

The objective of this thesis is to show that pedestrians can be successfully detected in low-resolution video using short-term image sequences. The time duration used should be long enough to capture the characteristic walking motion of a pedestrian. However, it need not be long enough to perform tracking. Thus, the method is applicable to situation such as video taken from a rapidly flying UAV as shown in Figure 1.4.

Although our system uses multiple images to perform detection; it is a detector, not a tracker. As a detector, it simply reports the presence of a pedestrian at a specific location in the image sequence and does not perform association between detections. On the other hand, the aim of a tracker is to generate the trajectory of a moving object over time by establishing correspondences between the object detections across frames. However, our detector can be incorporated into any standard tracking algorithm, such as a multiple hypothesis tracker or a particle filter-based tracker, if a longer sequence of images of the same scene is available. The improved accuracy of our detector should make it possible to estimate more reliable tracks using a shorter sequence of images. More discussion and comparison between detection and tracking is presented in Section 2.5 of Chapter 2.

The main contribution of this thesis is the development of a new pedestrian detector that is able to detect pedestrians in surveillance videos at lower resolutions than has been reported in previous work. The algorithm performance has been evaluated through experiments on several standard public datasets; including some challenging aerial datasets. The new pedestrian detection system significantly outperforms several existing state-of-the-art pedestrian detection systems (Chapter 4). The system is suitable for outdoor stationary surveillance cameras as well as aerial imagery. A preliminary version of this work was published in the IEEE WACV 2014 conference [9] and final version has been submitted to the IEEE CVPR 2015 conference.

1.3 Assumptions and Limitations

Some assumptions of this work include:

- We use grayscale optical imagery only, even though thermal (infrared) imagery has been used in some of the previous work and could be helpful for detecting people. The method described in this thesis could certainly be applied to infrared imagery.
- Fully (or mostly) visible (not occluded) upright walking people are targeted for detection. The camera is assumed to be looking down at the people at an oblique angle. The learned model is constrained to such poses.
- Pedestrians are assumed to be walking with a moderate speed. This allows for a reasonable amount of information to be included as input to the classifier. It is about one second of walking (or less), which is about two walking steps. Depending on camera frame rate, this corresponds to about 16 to 32 frames. A base slice size of 32×32 pixels has been chosen so that with a pedestrian 20 pixels tall who has a moderate walking speed (not very fast walking and not running), the pedestrian will stay within the detection volumetric space.

1.4 Outline

This chapter has given a brief introduction and defined the problem of detecting low-resolution pedestrians in surveillance videos, and some of the applications. It has described the work's motivation and inspiration and provided an overview of the proposed approach, assumptions, and contributions of this thesis. The remainder of this thesis is broken into four chapters as follows: **Chapter 2** describes the state-of-art in the field of pedestrian detection and presents previous work related to the topic. The chapter also presents a comparison between the new detection approach and tracking methods and summarizes a critique to previous work. **Chapter 3** describes the new proposed approach in detail; starting from video stabilization until a classification decision is made. **Chapter 4** describes in detail the experiments, used datasets, adjusted parameters, and algorithm performance at different settings. The chapter also discusses the experimental results, and comparison to other methods' results. **Chapter 5** summarizes the key findings of the proposed approach, discusses conclusions, presents approach advantages and limitations, suggests solutions, and finally, provides some pointers to future research directions and approach improvements.

CHAPTER 2: LITERATURE REVIEW

There is an extensive body of literature on people detection, although there is relatively little work on pedestrian detection in low-resolution videos, and very little work on pedestrian detection in aerial videos. Most work focuses on pedestrian detection in single high-resolution images.

Some work on pedestrian detection in automotive applications uses contextual information to improve detection performance [10]. For example, pedestrians are often around vehicles in traffic scenes, and algorithms can make use of that fact. Our work does not use contextual information since we wanted to make our approach more general and not be limited to traffic scenes.

Following the convention of [11], we use the following terms: a *near scale* or *high resolution pedestrian* is defined as a pedestrian over 80 pixels tall; *medium scale* is 30-80 pixels; and *far scale* or *low-resolution* is under 30 pixels. Comprehensive reviews can be found in [1], [11], [12], [13], and [14]. The cited works on pedestrian (and people in general) detection can be classified according to the number of frames used by the detection system. This includes two categories: (1) Pedestrian detection in single images, and (2) pedestrian detection in sequences of images. The previous work is reviewed according to these two categories. In addition, work on the special problem of people detection in aerial videos is reviewed.

Section 2.1 reviews pedestrian detection in single images. Section 2.2 covers the work of pedestrian detection that is based on image sequences. Work on people detection in aerial videos and images is reviewed in Section 2.3. Section 2.4 presents a critique of previous work. A comparison with tracking is presented in Section 2.5.

2.1 Pedestrian Detection in Single Images

There have been many approaches to detecting people in images. One class of approaches uses explicit knowledge of human kinematics as the basis of implementation. These approaches use articulated object models and attempt to match image features to body parts (see for example the survey of [15]). We do not consider this approach because in low-resolution images, body parts may not be clearly visible.

Approaches that are more applicable to lower resolution images are those that attempt to detect the whole person. Instead of an explicit model, an implicit representation is learned from examples, using machine-learning techniques. These approaches typically extract features from the image and then apply a classifier to decide if the image contains a person. Typically, the detection system is applied to sub-images over the entire image, using a sliding window approach. A multi-scale approach can be used to handle different sizes of the person in the window. Alternatively, the detection system can be preceded by a region-of-interest selector, which generates initial object hypotheses, using some simple and fast tests. Then the full person detection system is applied to the candidate windows.

A recent survey found that the most successful approaches for single frame pedestrian detection use some form of gradient histograms [13]. The "Histogram of Oriented Gradient" (HOG) features were introduced by Dalal and Triggs [2]. In their approach, the detector is applied to a subimage (typically 64×128 pixels). The subimage is divided into small regions called cells. In each cell, the gradient magnitude and orientation is computed at each pixel. To compute the gradient histogram for the cell, each pixel casts a vote, weighted by its gradient magnitude, for the bin corresponding to its gradient orientation. The local histograms are normalized across regions of the image, called blocks. The combined histograms from all cells then represents the descriptor, or feature vector, for the subimage. An example of HOG features is shown in Figure 2.1 (a).

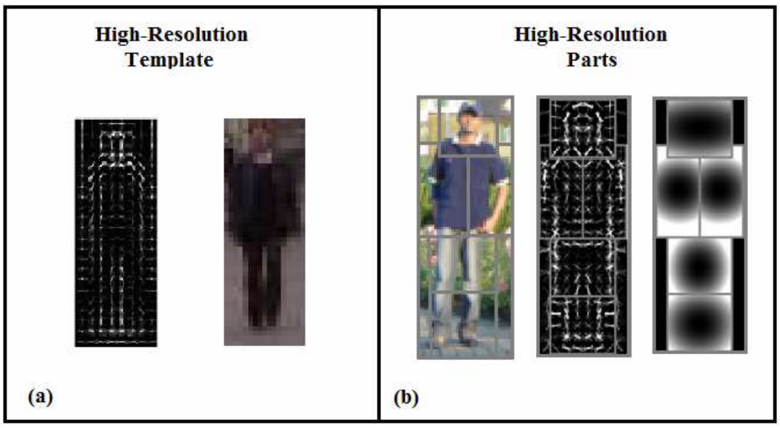


Figure 2.1: Detections obtained with (a) single component person model [2], (b) model defined by several high resolution part filters. The parts filters specify weights for HOGs features. Their visualization shows the positive weights at different orientations, [3].

The approach of Dalal and Triggs has been extended by others. Felzenszwalb *et al.* trained deformable part models for object detection [3]. Instead of using a single model for a person, they use a set of models for each part (such as head, arms, and torso). The part models are related to a root model using a set of deformation models representing the expected location of body parts (Figure 2.1 (b)). Park *et al.* developed a multiresolution model that automatically switches to parts only at sufficiently high resolutions [16]. The work integrated a rigid HOG-based template for low-resolution, with a deformable parts model for high resolution. They found that part-based models are not useful for pedestrian heights less than 90 pixels. The work of [17] uses orientation and magnitude of gradient features that are similar to HOG. According to the authors, their algorithm gives accuracy similar to that of the Dalal algorithm.

Once features are extracted from an image, they are then sent to a classification algorithm. To train a classifier, feature vectors from a large number of training images are used. The training images are labeled as positive ("contains a person") or negative ("does not contain a person"). One common type of classifier that is used is a Support Vector Machine (SVM), [18]. The SVM looks for an optimal hyper-plane as a decision function. The classification task is to take an input vector x and assign it to one of the two classes. More details about how SVM classifier works are discussed in Section 3.5.

An alternative classifier that is commonly used in pedestrian detection is adaptive boosting, or "AdaBoost" [19]. AdaBoost combines the output of many simple classification algorithms into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are adjusted in favor of those instances misclassified by previous classifiers.

The Dalal algorithm is of interest to our work, since it is so widely used to detect pedestrians. In Chapter 4, we compare the performance of our algorithm to theirs. According to evaluations of pedestrian detectors [13], no single feature has been shown to outperform the HOG feature for near scale pedestrians, although other features can provide complementary information, which can increase performance. Evaluations found that the performance of HOG-based pedestrian detectors degrades rapidly at medium and far scales *(i.e.*, less than 80 pixels in height). A low-resolution image simply does not provide enough information to accurately detect pedestrians based on shape.

2.2 Pedestrian Detection in Image Sequences

Instead of using a single frame pedestrian detector, a different approach can be used. Namely, a detector can be developed which directly analyzes a short sequence of frames. Since a sequence contains more information than a single frame, the classification result can potentially be more accurate. The remainder of this section reviews past work using this approach. The research on detecting people in videos is not as voluminous as that of single frame methods.

2.2.1 Optical Flow and Frequency Based Features

One method to make use of temporal information is to use optical flow as a feature. The approaches of [20] and [21] employ optical flow-based motion descriptors. The algorithm in [20] combines HOG descriptors with optical flow-based motion descriptors. The detector combines appearance descriptors extracted from a single frame of a video sequence with motion descriptors extracted from optical flow. It scans a window across the image at multiple scales, and runs a linear SVM classifier. In [21] the authors attempted to recognize an individual's action by matching the sequence of optical flow descriptors with examples in a database. They looked for the best matches for classification (*e.g.*, ballet, tennis). According to the results reported in [21] the approach was successful at medium scale and required clear backgrounds. The optical flow-based features appear to help in high resolution [13], but in low-resolution scenarios, detection results are poor due to noise, camera jitter, and the limited number of pixels available.

Another class of approaches [22] and [23] perform foreground segmentation and frequency analysis. The approach of [22] segments a moving object from the background and applies time-frequency analysis to characterize the periodic motion. Periodic motion shows up as peaks in the spectrum. The algorithm classifies different types of periodicity due to different types of motion symmetry such as walking/running humans, running dogs, and flying birds. The approach of [23] tries to discriminate and track pedestrians in low-resolution aerial video sequences. In this algorithm, foreground objects are segmented and tracked. Frequency-based measures are used to detect the existence of human motion (*e.g.*, the motion of a leg).

2.2.2 Haar-Like Features

An alternative to HOG features is the so-called Haar-like features. Haar-like features owe their name to their similarity to Haar wavelets. In mathematics, a Haar wavelet is a sequence of square shaped functions that form a wavelet basis used in analysis [24]. Haar-like features are differences of rectangular regions in the images. These features are simple and very fast to compute. Although each feature is not very discriminatory, a large number of features can be chained together to achieve good performance. The method of AdaBoost can be used to train a classifier and select features [25, 26, 27].

One of the first sliding window detectors was proposed in 2000 by Papgeorgiou and Poggio [28]. This algorithm used Haar-like wavelets as input descriptors. They extended the idea of using intensity differences between local adjacent regions to a wavelet-based representation for people, car, and face detection. For classification, they combined the Haar wavelets features with a polynomial SVM classifier.

Haar-like features were also used by Viola and Jones [29]. Although their detector can be trained to detect a variety of classes of objects, it was motivated primarily by the problem of detecting faces. In their detection framework, they employed AdaBoost to both select the best features and to train the classifier. Approaches of [30], [31], [32], and [33] also used Haar-like features.

Viola and Jones [31] extended their work of face detection to detecting pedestrians. In this method, Haar-like wavelets are used to compute features in pairs of successive images to detect motion. A rectangle of pixels in the current image is compared to the corresponding rectangle in the previous image, as well as a shifted version of the rectangle in four directions: up, down, left, and right. These represent four directions of walking. AdaBoost is used to build a cascade classifier based on a set of manually labeled training images.

Jones and Snow [33] extended the above algorithm to make use of ten images in a sequence. Three types of features were used: Haar-like features applied directly at each frame (appearance filter), absolute difference of Haar-like features in different frames, and a shifted difference filter to capture the motion of pedestrians. Figure 2.2 shows the three different types of filters. The first row illustrates one particular appearance filter. Such filters act on a single window, which could be any of the 10 windows. The second row shows a particular difference filter. Such filters consist of one filter acting on one window and the negative of that filter acting on another window. The third row shows a particular shifted difference filter. Different classifiers were trained to detect pedestrians for each of eight walking directions (north, northeast, east, southeast, etc). Each detector was trained at the detection window scale of 24×26 pixels. The AdaBoost learning algorithm was used to select a set of features to separate pedestrians from non-pedestrians for each direction class of the eight classes. Because of the large number of features that need to be examined at each stage, the training time can be quite slow (in the order of weeks) [3].

The Jones and Snow algorithm is of interest to our work, since it was specifically designed to detect pedestrians in low resolution videos. In Chapter 4, we compare the performance of our algorithm to theirs.

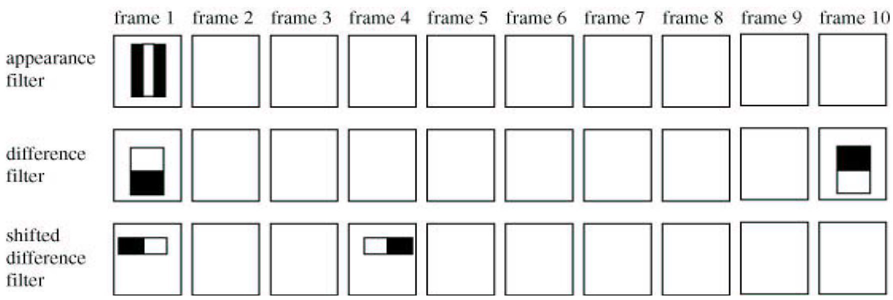


Figure 2.2: Three different types of filters 0.

2.2.3 Spatio-Temporal Features

Features can also be extracted using gradients in both the spatial and temporal directions. The work of [35] extracts spatiotemporal interest points to recognize behaviors (*e.g.*, of mouse) by analyzing the motion using cuboid descriptors. Their interest point detector is composed of a *2D* Gaussian smoothing kernel, applied along the spatial dimensions, and a pair of *1D* Gabor filters applied temporally. Gabor filters are used to obtain localized frequency information since they are sensitive to change in phase and give sinusoid response. The detector is tuned to fire whenever variations in local image intensities contain periodic frequency components. The strongest responses will be evoked whenever periodic motion is detected (*e.g.*, bird flapping its wings). Local histograms of gradients were then extracted in the neighborhood of each detected interest point. This approach could be used to detect pedestrians instead of recognizing actions. However, in low-resolution image sequences, it would be difficult to extract interest points since the volume is so small and the pedestrian's size is only a few pixels.

On a related topic, the work of [32] recognizes sign language "words" (*i.e.*, for deaf people) from video sequences, using volumetric spatiotemporal features based on Haar-like features. The method relies on a learning system to combine these basic features in such a way as to depict the motion of hands. In this work, the Haar-like features are 3D volume summations, rather than 2D rectangle summations. Similar to the 2D case, the 3D features can be computed very quickly. In this case, integral volumes are used rather than the integral images of the 2D case. AdaBoost is used for classification.

2.3 Pedestrian Detection in Aerial Images

As previously mentioned, there is very little work on detecting pedestrians in low resolution aerial videos. Aerial videos are challenging due to the potentially rapid translation and rotation of the camera, which can cause motion blur. Also, since the camera is continuously moving, it is more difficult to create a background model for the purpose of segmenting foreground objects. However, some motion compensation can be done by registering each image to a reference image. In this way a short-term background image can be computed, which can be used to detect foreground (moving) objects using image subtraction [23].

The few papers that address pedestrian detection in aerial images often use the same methods that were discussed above, namely HOG-like features with an SVM classifier (*e.g.*, [36]) or Haar-like features with an AdaBoost classifier (*e.g.* [37]). The work of [36] presents an algorithm for people detection and identification in a set of aerial images. First, a subject manually selects a person in one aerial image; then the algorithm automatically finds that person in another image. To search for the candidate person's image, the algorithm starts by detecting blobs for candidate locations of people, and then extracts histograms of color values (HSV channels) and HOG features from the detected blobs. An SVM classifier is used to classify these feature vectors. Although reported results are promising, the scenarios are presented in high resolution and need colors for matching. When pedestrians are only a few pixels tall, appearance cues-based approaches such as these can fail. This limits the general application of this technique, especially for low-resolution scenarios.

The approach presented in [37] combines thermal (infrared) and optical (visible) imagery to detect cars in aerial images and uses only thermal imagery for people detection. To detect people, they use metadata such as the field of view of the camera and the UAV altitude, since the algorithm constrains the size of candidate detected (rectangular) regions with a maximum width and height. These regions are detected using color clustering and edge detection to get candidate locations of humans. Then Haar features are extracted from these regions with an AdaBoost classifier employed for classification. Quantitative results were not presented.

The only research found that uses image sequences in aerial scenarios was [23]. After video stabilization, foregrounds are detected using frame differencing. For each candidate moving object, a sequence of detection window is used to compute frequency measures to detect the presence of periodic motion of some body parts, such as legs and arms. Searching for moving object kinematic features might be processed over a period of up to four seconds. The algorithm was not described in detail, and there are some doubts as to whether this approach would be feasible in low-resolution sequences.

Another interesting approach is that of [38], which uses single images. This work uses metadata (latitude, longitude, altitude, as well as pitch, yaw and roll) to predict the orientation of the ground plane normal, the orientation of shadows, and the size of people's shadows. They detect blobs to get candidate locations of humans and then Daubechies wavelet [39] features are extracted from these areas by scanning a small image window around them. Feature vectors are then fed to an SVM classifier for classification: human/non-human. While the algorithm achieves good results in low resolution (24×26 pixels image windows), it is not clear that shadows will always be present in the image. Another problem with this method is that it needs prior information that may not be available.

2.4 Critique of Previous Work

In summary, state of the art methods in pedestrian detection use HOG features in conjunction with a classifier such as SVM. However, approaches that use single images perform poorly when pedestrians are less than 80 pixels in height. The inclusion of motion features increases the detector performance. The Jones and Snow algorithm [33] achieved good results on low resolution videos. However, it is possible that more challenging datasets (*e.g.*, with cluttered background, and more variety in viewing angles) would result in lower performance. Indeed, a subsequent evaluation [13] found that this approach fails to detect pedestrians with heights of 30 pixels or less.

Our algorithm, described in Chapter 3, differs from existing approaches since it compensates for the lack of information in low resolution images by gathering information over a relatively longer sequence of images.

There has been very little work on detecting pedestrians in low resolution aerial images. The few papers that were found use similar approaches as in stationary surveillance videos. One of the goals of this thesis is to develop a method that can successfully detect pedestrians in aerial videos.

2.5 Detection versus Tracking

As previously mentioned, although our detector uses multiple images, it is a detector, not a tracker. To avoid any potential reader confusion between our proposed detection approach and tracking, we give a short review of tracking methods here, along with a discussion of the distinction between detection and tracking, and where our method lies.

Tracking is a topic that is well studied with wide applications. Since 2000, tracking algorithms have focused primarily on surveillance applications [27]. Tracking can be defined as the problem of estimating the trajectory of an object in the image plane as it moves in different frames of a video.

Of course, detections based on short sequences (called "*tracklets*") can be fused into long tracks. However, in aerial image sequences taken from a rapidly moving camera, a person may not be visible for very long, so a long track cannot always be assembled. The problem of assembling detections into long tracks will not be addressed. The idea of fusing detections into tracks could be one of the paths for future work.

When image sequences are available, the additional information may be used to improve performance. One approach is to apply the single frame detectors to each image and fuse the results into tracks. The association of detections into tracks can be a difficult problem. It has been extensively studied and a number of statistical techniques have been developed. Joint Probability Data Association Filtering (JPDAF) and Multiple Hypothesis Tracking (MHT) are two widely used techniques for data association, see *e.g.* [40].

Tracking methods can be classified into three categories [42]: (1) *Point* tracking where the detected objects are represented by points and the association of these points is based on the previous object state, which might include object position, motion, *etc.* (see Figure 2.3 (a)). (2)

Kernel tracking which is based on object shape and appearance (*e.g.*, rectangular patch with an associated histogram, as shown in Figure 2.3 (b)). (3) *Silhouette* tracking where tracking is performed by estimating the object region in each frame (*e.g.*, shape matching or contour evolution, as shown in Figure 2.3 (c)).

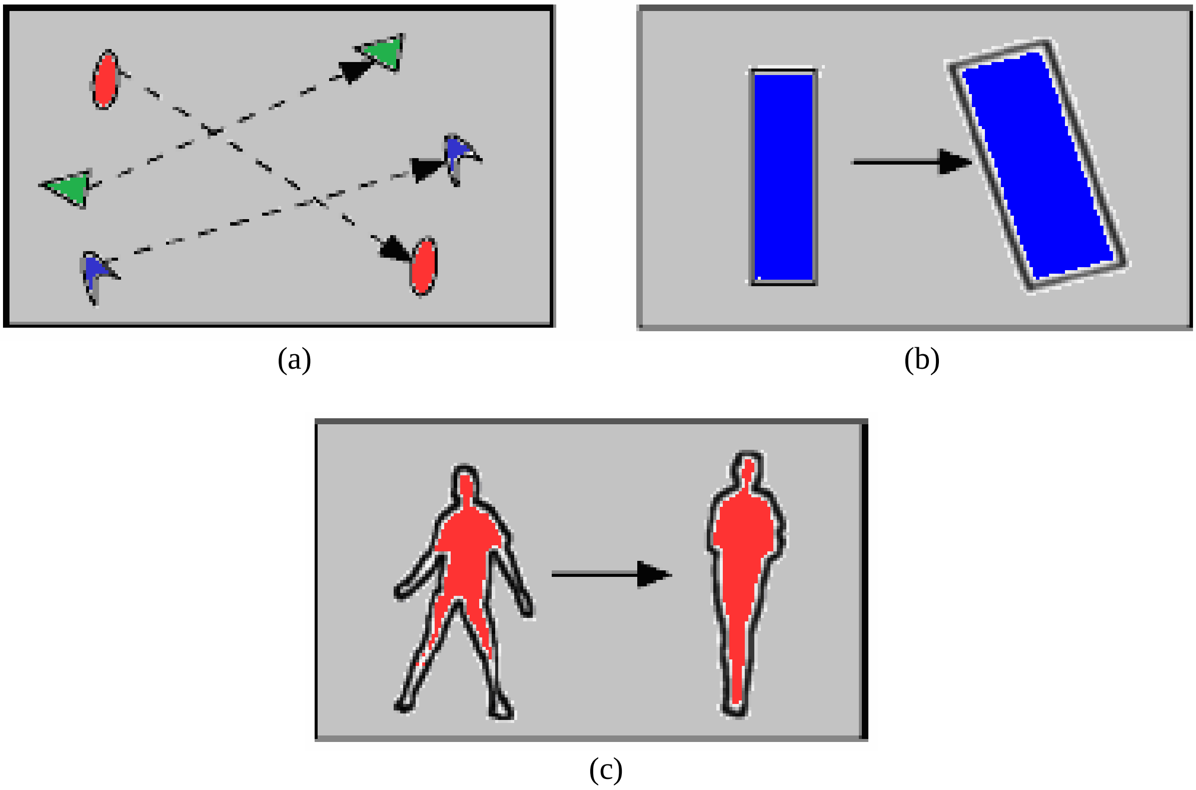


Figure 2.3: The three main tracking categories. (a) Point tracking. (b) Kernel tracking. (c) Silhouette tracking.

Usually there is a relationship between the tracking algorithm and object representation (point, points, silhouette, geometric shape, *etc.*), and the application domain. For example, to represent a human body, a contour or a silhouette is a good representation. For object association, there are many appearance features used; for example, color features, textures, covariance features, histogram of gradient orientation, *etc.* ([42], [43], and [44]).

Pedestrian tracking requires gathering information about a tracked subject over a relatively longer time period than what is required for detection. For example, detection requires as low as a single frame which could correspond to 1/30 of a second, whereas the literature on tracking suggests up to 4 seconds interval length required to identify kinematic features of the tracked subject. In [45] in order to obtain meaningful tracks that capture the motion characteristics over longer durations of time, sequences of several seconds are used. In [5] tracking is done based on motion and appearance with mean track duration of 21 seconds, and it might be as long as 4 minutes of tracking with association between targets and computed tracks based on spatial proximity for at least 3.2 seconds.

The pedestrian tracker of [46] uses the output of the part-based detector of [47] to create what are called tracklets and residuals. A tracklet is a short track segment of high confidence detections, and residuals are those with low confidence detections according to a certain threshold. Tracklet descriptors are based on color, motion, and height. The tracker tracks pedestrians by associating detections across consecutive frames; however, if no more matches are found for the current tracked object, the track is terminated. A new tracklet is created whenever detection cannot be associated with other tracked objects. All the part detection responses that do not get associated with a track are used as residuals during the tracklet association stage. The algorithm then associates tracklets using the MHT tracker to connect tracklets into longer tracks. Figure 2.4 illustrates the residuals, tracklets, and associating tracklets into final tracks.

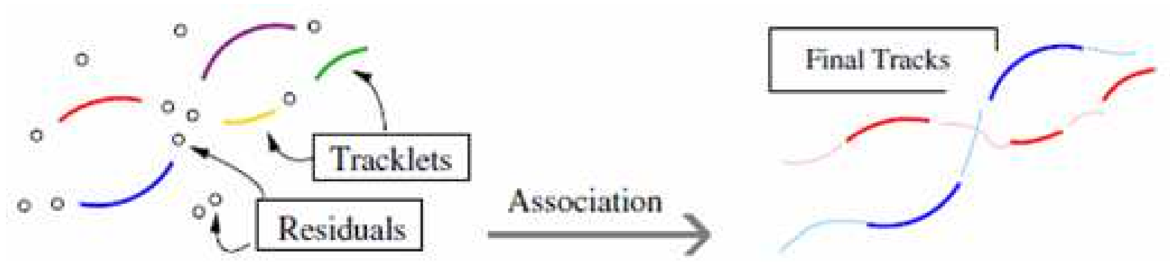


Figure 2.4: The tracklets having same color belong to the same object trajectory. Association paths of a tracklet are shown by using the lighter shade of the color used for the tracklet; *e.g.* light blue segments are associations path generated using tracklet in deep blue.

The main challenge for the tracker occurs when detector output is unreliable and sparse because detectors usually deliver a discrete set of responses, which yields false positives and false negatives [44]. A tracker can compensate for an unreliable detector by fusing the noisy detections into an estimated track state. False detections can be rejected (since they do not belong to a track) and missing detections can be tolerated by predicting the position of the object from the estimated state. The work of [48] and [49] track vehicles, and if there is missing detection, the object location is estimated to compensate for the unavailable detections.

One advantage of the proposed detector is that it can be integrated into a standard tracker, which is much easier than integrating other detection approaches which are based on single (or few) frames. For example, if we use ½ second of time duration for detection (Figure 2.5), and the resulting detections (tracklets) are used as input to a tracker system; tracking information can be used to estimate locations of missed detections. This reduces the number of false negatives and improves the overall performance of the system.

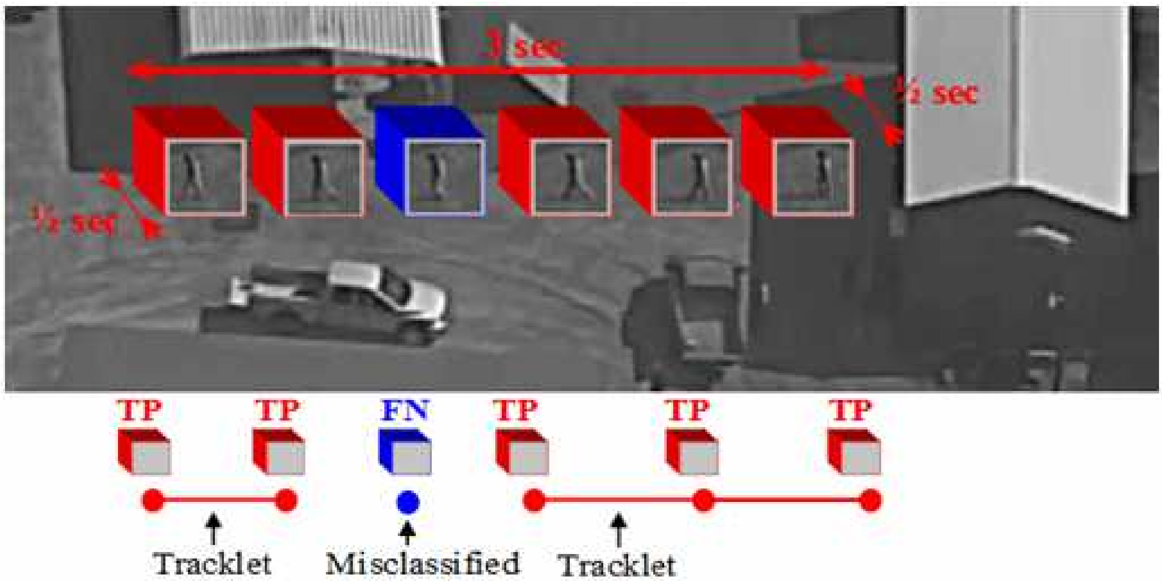


Figure 2.5: Short- term detections (in red) for 0.5 second each can be integrated into long duration tracking (*e.g.* 3 seconds). Tracking information can be used to recover from miss-detections. Volumes in red color represent True Positives (*TP*), and volumes in blue color represent False Negatives (*FN*).

CHAPTER 3: PEDESTRIAN DETECTION METHOD

This chapter describes the new pedestrian detection method in detail. Experimental results and analysis will be presented in Chapter 4. Section 3.1 gives an overview of the approach. Sections 3.2 and 3.3 discuss the methods for video registration and detecting ROIs, respectively. A detailed description of volume formation and feature extraction is presented in Section 3.4. The classification technique is discussed in Section 3.5, and Section 3.6 summarizes training details. Section 3.7 describes the process of non-maximum suppression.

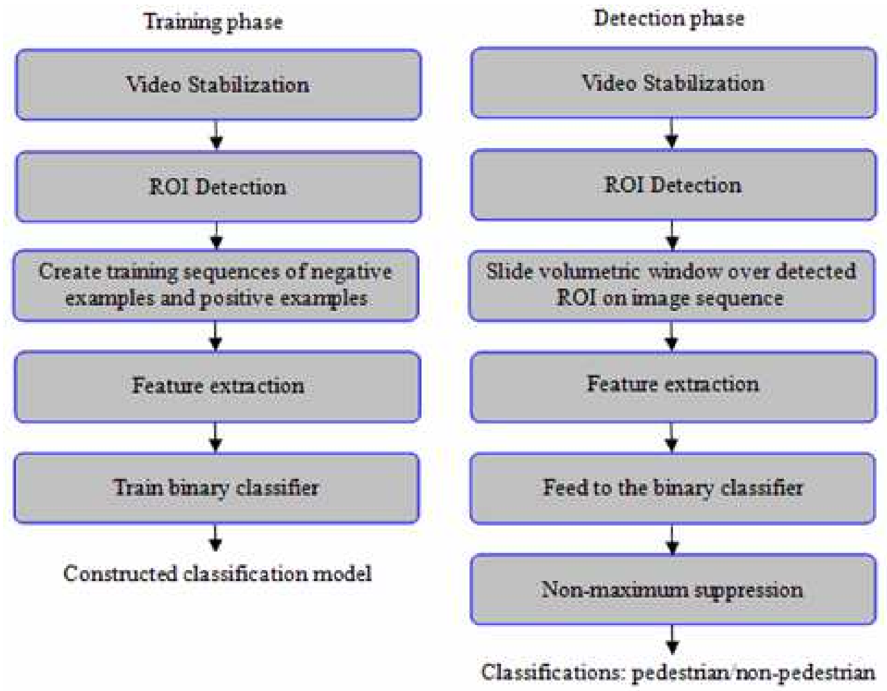
3.1 Overview of the Approach

Figure 3.1 shows the architecture of the entire pedestrian detection system, which contains two phases: a training phase and a detection phase. In the training phase, positive training examples (*i.e.*, volumes containing pedestrians) and negative training examples (*i.e.*, volumes not containing pedestrians) are created. Features are then extracted from these volumes. These examples are used to train a binary classifier, which is used to provide classification decisions. Figure 3.1 (a) shows this training phase. The detection phase is shown in Figure 3.1 (b). In the detection phase, the binary classifier constructed during the training phase is used to scan over detected ROIs in sequences of unseen testing images to search for pedestrians. This phase has several components: video stabilization (for aerial videos), ROI detection, formation of spatio-temporal volumes, feature extraction, classification, and non-maximum suppression.

In this work, color images are not used since we assumed that color does not contribute significantly to detection. It may be useful for tracking, but color (in the form of the hue and saturation component) is not a useful indicator of the presence of a pedestrian. Shape (as estimated by the gradient of image magnitude) is more useful. Therefore, we first convert all color images to grayscale prior to processing.

3.2 Video Stabilization

Video stabilization is applied to short overlapping sequences of 32 frames. We start with the first frame of each sequence and use it as the reference frame for the sequence. The remaining frames are registered to the first frame. The results are overlapping groups of 32 frames, which are co-registered. This aids the next step, which is to detect regions of interest containing potential moving objects. In the case of videos taken from stationary cameras, the stabilization step can be skipped, since the images are already co-registered.



(a) (b)

Figure 3.1: The architecture of our overall pedestrian detection system which contains two phases: training phase and detection phase

Image registration (or image alignment) is the process of transforming one image to minimize the difference between it and another (reference) image. Let **T**(**p**; **b**) represent a transformation, where **p** is a pixel location and **b** is a vector of parameters. The transformation **q** = **T**(**p**; **b**) takes the pixel location **p** from the first image and maps it to a pixel location **q** in the second image. The goal of image registration is to find the parameters that minimize the error between two images I_1 and I_2; *i.e.*, minimize

$$E = \sum_x [I_2(\mathbf{T}(\mathbf{p};\mathbf{b})) - I_1(\mathbf{p})]^2 .$$

One method to estimate the transformation parameters is to first find a small number of corresponding points between the two images. These points, called "tie points" or "control points", can be selected manually or discovered automatically. In our work, we find the corresponding points automatically, as described below. Once we have a sufficient number of corresponding points, we can solve for the transformation parameters. Figure 3.2 shows a simple example of a spatial transform that maps point **p** from one image to point **q** in another image.

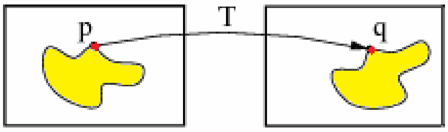


Figure 3.2: Example of a transformation between two images.

In our work, we use a "homography", or "projective transform". This has the form

$$\tilde{\mathbf{q}} = \mathbf{T}(\mathbf{p};\mathbf{b}) = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix}$$

where $\tilde{\mathbf{q}}$ is the transformed point in homogeneous coordinates. To recover the coordinates of the point **q** from the homogeneous coordinates, the resulting point $\tilde{\mathbf{q}}$ must be scaled so that the third element equals 1 (*i.e.*, by dividing $\tilde{\mathbf{q}}$ by its third element). A homography has 8 unknown parameters (although the matrix has 9 elements, there are only 8 degrees of freedom due to the scaling), so at least four corresponding points are required to solve for the transformation.

A homography can accurately model the projection of a plane from one viewpoint to another viewpoint in a projective camera such as a pinhole camera. In our problem domain, we assume that the camera is looking down at the ground, which is approximately a planar surface. Thus, the homography transform models the relationship between any image in the sequence and the reference image.

The assumption of a planar surface is only an approximation, although it is usually good if the camera is high above the ground. However, any objects (such as buildings and trees) above the plane will be miss-registered, and may result in ROIs that do not correspond to actual moving objects. Our classifier will subsequently filter these out, since motion patterns within these ROIs do not match the patterns of a walking person.

To automatically determine corresponding points, Harris corner interest points [50] are matched between the reference image and each subsequent image. A large number of points are used, to achieve better accuracy. However, some of the point correspondences may be incorrect. These outlier correspondences will corrupt the estimated transformation, and it is important to filter them out. We use the RANdom SAmple Consensus (RANSAC) algorithm ([3], [51]) which is a widely used approach to this problem. The RANSAC algorithm starts by selecting a minimal set of points randomly and solving for the transformation. It counts the number of points that agree with this transformation. If this transformation has the highest number of inliers so far, it is saved, and then the algorithm repeats for a number of trials and returns the best transformation.

Figure 3.3 shows the idea of overlapped groups of 32 frames. We perform registration for each sequence of 32 frames, and then shift the reference frame by four frames. Algorithm 1 presents the overall process of video stabilization.

Algorithm 1: Video Stabilization

Input: Frames of aerial video: $\{f_1, f_2, f_3, \ldots, f_n\}$.
Output: Registered frame groups: $\{R_1, R_2, R_3, \ldots, R_m\}$.
$i = 1$;
while $i \leq n$ **do**
    $sequence\_count = 1$
    $f_{Reference} \leftarrow f_i$
    $R_j \leftarrow \{f_i\}$
    $IP \leftarrow$ Find Interest Points($f_{Reference}$)
    $k = i + 1$
    while $sequence\_count \leq 32$ **do**
        $MIP \leftarrow$ Match interest points between $f_{Reference}$ and f_k
        H = Fit a homography between IP and MIP
        Transform f_k using H, and append to the set R_j
        $sequence\_count = sequence\_count + 1$
        $k = k + 1$
    end
    $i \leftarrow i + 4$
end

3.3 ROI Detection

After stabilization, background subtraction is used to identify foreground objects in the scene for subsequent analysis. The literature on background subtraction and motion segmentation is vast, and there are many surveys for the topic of background subtraction; (*e.g.*, [52], [53], and [54]). Background subtraction approaches were developed as early as the 1970s; for example, the work of [55] which used frame differencing to detect moving objects. Subsequent approaches proposed the use of statistical methods to model the background.

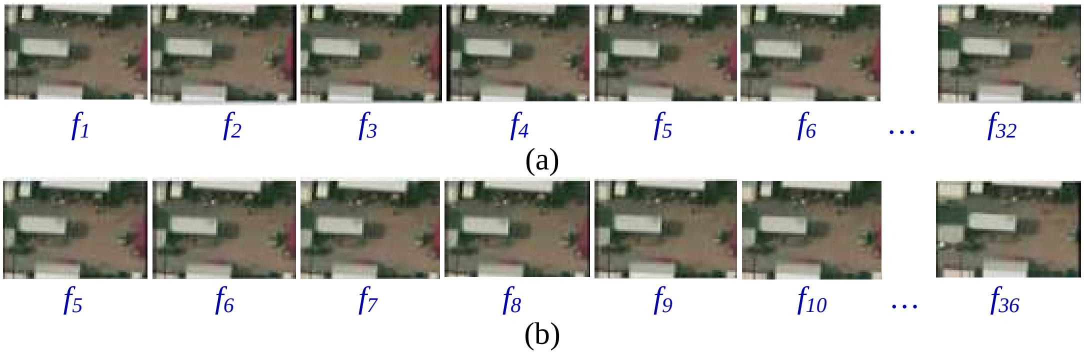


Figure 3.3: Examples of overlapping sequences of 32 frames each (to be registered together). (a) Sequence1: frame1 to frame32. (b) Sequence2: frame5 to frame36.

The work of [4] proposed the use of the median over a window of images to model the background. [56] presented the idea of modeling the background by representing each pixel with a mixture of Gaussians and updating the model over time. The updating was done recursively, which can model slow changes in a scene, but not rapid changes. Most methods assume that the scene is observed over a (relatively) long period of time, and that changes in the scene occur gradually.

In our application, we may not be able to observe the scene for a long period of time. Thus, we have to compute a “local” background model for only a short sequence of images. The method we use is very simple. We compute a background model for each group of 32 registered frames by computing the mean of the frames. Although it is a simple model, the experiments show that it is reliable in detecting initial foregrounds.

We then take the difference between the middle frame in each sequence and the background model for that sequence. Initial foreground pixels are identified by thresholding the difference image, D. These foreground pixels could correspond to real moving objects or possibly static objects due to non-perfect stabilization. The difference between current frame at pixel (i,j) and the background is:

$$D(i,j) = | I\ (i,j) - B(i,j) |$$

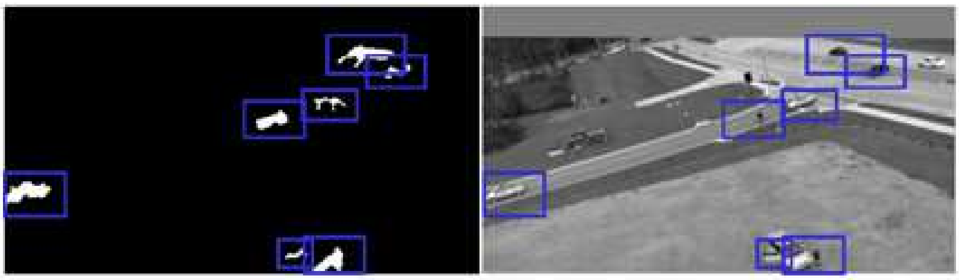
A pixel in the difference image D is classified according to an empirical threshold T_L as follows:

$$D(i,j) \in \begin{cases} Foreground, \text{ if } D(i,j) > T_L \\ Background, \text{ if } D(i,j) \le T_L \end{cases}$$

In our work, the threshold was empirically set to 15−25 (depending on the image sequence). Morphological opening and closing operations are applied to eliminate small regions and join broken regions. Then connected components are found to obtain initial ROIs. The final ROIs are the ones that pass the final area thresholds: A_L and A_H. For example, a ROI is included as a member in the final ROI set if its area in pixels satisfies the following condition:

$$A_L < Area_{ROI} < A_H$$

In our work, these area thresholds were empirically set to 20 and 500 pixels, respectively. An example of an image containing the final ROIs is shown in Figure 3.4.



(a) (b)

Figure 3.4: An example of detected ROIs shown on: (a) The binary image. (b) The registered Image.

3.4 Formation of Spatiotemporal Volumes and Feature Extraction

A sliding window of size 32×32 pixels is scanned within the bounding box of each ROI detected above (see Figure 3.5). At each position, a spatiotemporal volume is created by extracting a sequence of subimages (slices), at a fixed position in the registered images, for N frames (we used up to 32 frames). At a typical camera frame rate of 30 frames per second, this corresponds to a duration of up to one second (see Figure 3.6).

The slice window size was chosen to be 32×32 pixels. This size is large enough so a pedestrian remains within the window throughout the sequence at normal walking speeds, which usually corresponds to about ½ pixel per frame. Since our detector is trained to detect pedestrians with a height of approximately 20 pixels, this allows a border of about 6 pixels above and below the person.

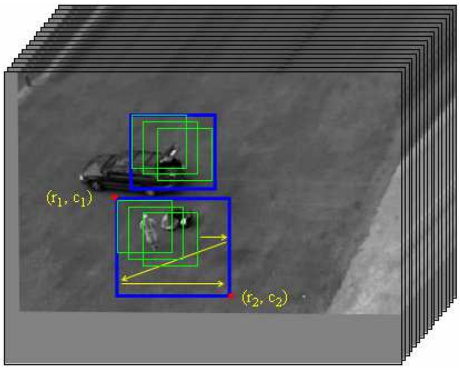
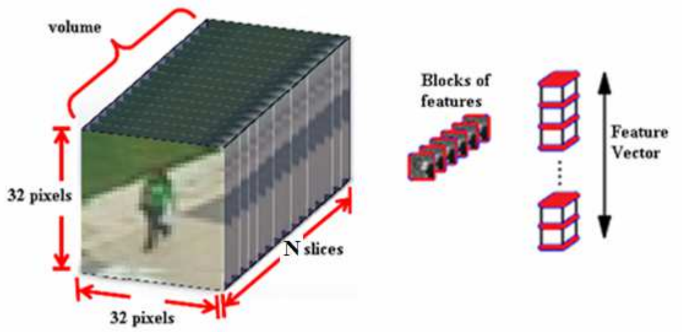


Figure 3.5: The sliding window is shown in green. The bounding boxes of detected ROIs are shown in blue. For example, the green window slides from ROI corner point (r_1, c_1) to (r_2, c_2) searching for pedestrians. Note: the grey area (on left and bottom borders) results from the image registration process.

To handle possible variations in scale, we extract volumes at multiple scales in the image sequence by creating a pyramid of images of different sizes. A scale factor of 0.75 is used between levels of the pyramid (for a total of 6 pyramid levels). This allows us to detect people that are taller than 20 pixels – the detector will detect people at the level of the image pyramid where the height is about 20 pixels.



(a) (b)

Figure 3.6: (a) Spatiotemporal Volume of size of 32 pixels × 32 pixels × *N* slices. (b) Features from all blocks the volume are concatenated into a single feature vector.

3.4.1 Feature Extraction

The next step is to extract features from the series of slices that make up the volume (Figure 3.7). We first compute the image gradient at each point in each slice. This is done by convolving a gradient mask M with the gray scale image I in x and y directions. A simple scheme of centered 1-D mask is used as follows.

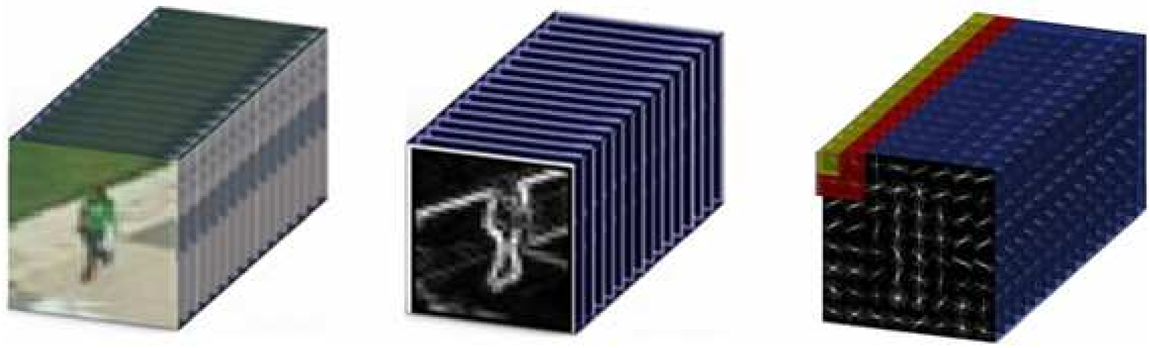
$$G_x = M_x * I, \text{where } M_x = [\text{-}1 \ 0 \ 1]$$

$$G_y = M_y * I, \text{where } M_y = [\text{-}1 \ 0 \ 1]^{\mathrm{T}}$$

The resulting G_x and G_y are used to compute gradient magnitude and orientation as follows.

$$|G(x,y)| = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$

$$\tan(\phi(x,y)) = \frac{G_y(x,y)}{G_x(x,y)}$$



(a) (b) (c)

Figure 3.7: Spatiotemporal Volume. (a) Positive example. (b) Gradient. (c) Computed HOG, with block (shown in red color), and cell (shown in yellow color)

HOG features are then extracted from each of the slices that make up the volume. We divide each 32×32 pixel slice into square cells (typically 4×4 pixels each), and compute a histogram of gradient directions in each cell. Gradient directions are quantized to 9 bins, which represent unsigned directions from 0°-180°. Each pixel within the cell casts a vote for the bin in the histogram corresponding to its gradient direction, weighted by its gradient magnitude. The result is a histogram for each cell, consisting of 9 values. Figure 3.8 shows an example of the steps of extracting HOG descriptor features from an image.

Following the method of [2], cells are grouped into (possibly overlapping) blocks, where each block consists of 2×2 cells. The histograms from each cell within the block are concatenated to make a combined histogram, with 36 elements.

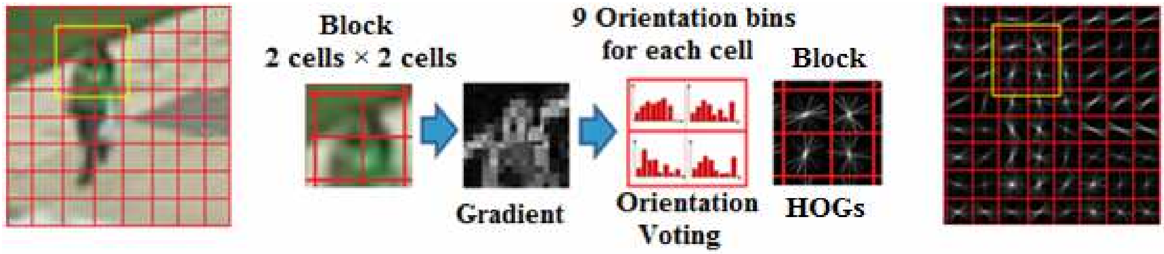


Figure 3.8: Interpreting gradient orientations into 9 bins histogram for each cell, and computing HOG features. Slice size is 32×32 pixels, cell size is 4×4 pixels, block size is 2×2 cells, and number of bins is 9. (Note: assuming block overlapping).

For example, assume we have the image of Figure 3.9(a). The block (of size 2×2 cells) in red color has the following feature vector, where each cell (in yellow color) has a size of 4×4 pixels. Each pixel within this cell casts a gradient magnitude weighted vote for an orientation based histogram channel. The HOG feature vector of this block is

$$f = \left(h_1^1,\ldots,h_9^1,\ h_1^2,\ldots,h_9^2,\ h_1^3,\ldots,h_9^3,\ h_1^4,\ldots,h_9^4\right).$$

For example, for the second cell, the 9 features are:

$$\left(h_1^2,\ldots,h_9^2\right).$$

Assume the orientations given in Figure 3.9(b) and the magnitudes given in Figure 3.9(c); the magnitude voting for 9 bins can be as shown in Figure 3.9(d).

Finally, the histograms from each block and each slice are concatenated into a single large vector. This represents the feature vector for the volume. More about the best choices of key parameters (*e.g.*, cell size, block size, number of bins) will be discussed in Chapter 4.

3.4.2 Normalization

The gradient magnitude can vary widely in images, due to changes in illumination. To avoid undesirable effects on detection performance, we normalize the gradients. Normalization can be performed in the input space (*i.e.*, the image values) or in the feature space (*i.e.*, the histogram values). Normalization in the input space amounts to rescaling the intensity of the pixels of the images. This normalization strategy has little or no effect on performance and sometimes decreases the performance. Normalization in the feature space outperforms the one in the input space ([2], [57], and [58]).

In the work of [59], they normalized gradients within each block. This was done as follows. Let v be the descriptor vector for the block and ε be a small constant. They normalize each vector using the equation

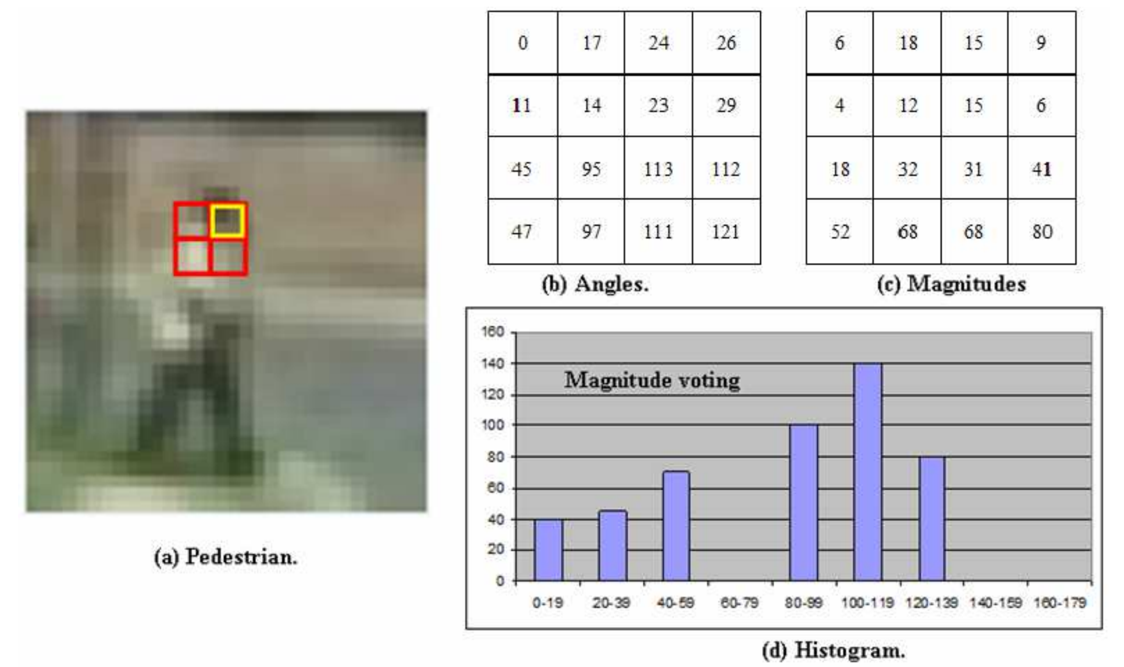


Figure 3.9: Example HOG feature vector for one cell of size 4×4 Pixels. (a) An example pedestrian. (b) The computed orientations for one cell. (c) The computed

$$v \leftarrow \frac{v}{\sqrt{\|v\|_2^2 + \varepsilon}}$$

where $\|v\|_2^2$ is the squared L_2-norm. The magnitude of the vector was then clipped to an empirically defined threshold, and renormalized again.

In our algorithm, we normalize gradients within each "volumetric block". A volumetric block is the set of blocks at the same place in the image, across all slices, as shown in Figure 3.10. We use empirically derived values in range of 0.001 to 0.0001 for the constant ε, and clip the feature vector magnitude to an empirically defined threshold, and renormalized again. The result of the normalization step is a set of feature vectors that are better invariant to changes in illumination or shadowing.

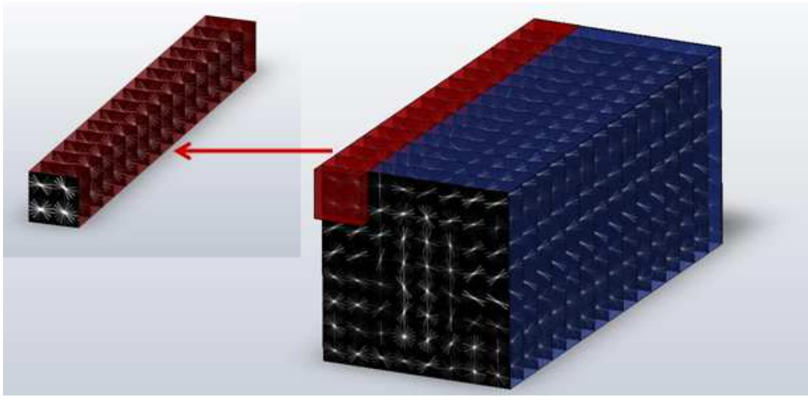


Figure 3.10: (a) Volumetric Block. (b) Spatiotemporal volume of 16 slices.

3.4.3 Dimensionality Reduction

The feature vector size is determined by the number of orientation bins in each cell (n_{bins}), the number of cells in each block (n_{cells}), the number of blocks in each slice (n_{blocks}), the number of slices per volume *(n_{slices})*. For example, using a volume of 16 slices, each slice of size 32×32 pixels, cells of size 4×4 pixels with 9 bins for histogram (*i.e.*, the angular histogram bins are evenly spaced over 0°–180°), with block of size 2×2 cells with no overlap between blocks, the feature vector size (FV_s) can be computed as:

$$FV_s = n_{slices} \times n_{blocks} \times n_{cells} \times n_{bins}$$
$$FV_s = 16 \times 16 \times 4 \times 9 = 9216 \text{ features}$$

If block overlap is allowed, there are more blocks in each slice and the feature vector size is correspondingly larger.

It is desirable to reduce the size of the feature vectors if possible. Using lower dimensional features produces models with fewer parameters, which speeds up the training and detection algorithms, while keeping a reasonable detection performance. The Principal Components Analysis (PCA) technique has been used for dimensionality reduction in many well known computer vision algorithms (*e.g.*, SIFT descriptor [59] and HOG descriptor [3]). Many pedestrian algorithms apply PCA to their feature vectors, (*e.g.* [60]).

Following this reduction method, PCA is used to reduce the dimensionality of the features. Feature vectors are transformed to principal component space, and only those principal components that account for the most variance in the data are kept. In the learning stage, a large number of 36-dimensional HOG features (*i.e.*, for each block) are collected and PCA is performed on them. The number of principal components is determined using the rate of cumulative contribution. Eigenvalues are used to calculate the cumulative contribution rate to determine the number of dimensions as following. Assume we initially have n eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$$

The contribution rate e_k of the k^{th} principal component corresponding to the eigenvalue λ_k is defined as

$$e_k = \frac{\lambda_k}{\sum_{i=1}^{n} \lambda_i}$$

The cumulative contribution rate of the first m principal components ($m < n$) is defined as

$$E = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{n} \lambda_i}$$

To reduce the dimensionality, we use half of the principal components. Experimentally, we find that the top 50% of the principal components account for 80% to 90% of the variance in the data.

Let's assume that v_i is the feature vector of training example i, and we have N training vectors, each of dimension M. The mean vector can be defined as

$$\bar{v} = \frac{1}{N} \sum_{i=1}^{N} v_i$$

The full orthonormal matrix U can be obtained by solving the eigen equation of the covariance matrix Σ.

$$\Sigma U = U\Lambda, \ (UU^T = I)$$

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}(v_i - \bar{v})(v_i - \bar{v})^T$$

where Λ is the diagonal matrix of which the diagonal elements are the eigenvalues. The principal scores (the representation of features in the principal space) are obtained using the projection matrix U as

$$y_i = U^T (v_i - \bar{v})$$

These are the new features projected in the new space (*i.e.,* the output of the PCA algorithm). After training on reduced dimensionality features (the chosen subspace using truncated U), the PCA scores for any HOG features are computed using this equation according to the selected number of principal components. The results of PCA experiments are discussed in Section 4.6.5.

3.5 SVM Classifier

SVMs were proposed by Vapnik ([17], [18]) and have yielded excellent results in various data classification tasks. The SVM classifier has been adopted in human detection in many computer vision algorithms. SVM is effective in high dimensional spaces, even in the cases where the number of dimensions is greater than the number of samples. This section gives a short explanation of SVM and defines some terms related to SVM classifier used throughout this thesis.

During classification, the SVM uses a subset of training points in the decision function. These points are called support vectors, *SVs*. The use of a subset of training points (*i.e.,* the *SVs*) saves the memory cost. In addition, there are many kernel functions that can be used for the decision function. As a common example, kernel function can be defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space (kernel examples will be shown later). There are some common kernels used, and it is possible to specify custom kernels [61].

To define some terms, assume we have the two dimensional data of Figure 3.11. The data include some positive and negative examples. We look for a linear function (hyperplane) *f(x)* that separates the two classes. The best discrimination function is the one that maximizes the margin between positive and negative examples. Using a linear model, the two-class classification problem can be put in the form $f(x) = w^T x + b$, where w denotes the learned weights (weight vector). w is normal to the separation line and b is a bias parameter. Assume we have training data X, such that $(x_1, y_1), ..., (x_l, y_l) \in X \times Y, Y = \{-1, +1\}$. For positive and negative examples

$$\mathbf{x}_i \text{ positive } (y_i = 1): \qquad w.x_i + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad w.x_i + b \leq -1$$

For support vectors: $w.x_i + b = \pm 1$, as shown in the figure below. The distance between point and hyperplane is $\frac{|w.x_i + b|}{\|\mathbf{w}\|}$. Therefore, the margin is given by $\frac{2}{\|\mathbf{w}\|}$, and we need to maximize this margin to have good separation. This is equivalent to minimizing $\|\mathbf{w}\|^2$, see [62] and [63] for more details. The purpose of training is to find w and b such that:

$$\arg\min_{w,b} \frac{1}{2} w^T w$$

$$\text{Subject to } y_i(w.x_i + b) \geq 1$$

This is an example of a quadratic programming problem in which we are trying to minimize a quadratic function subject to a set of linear inequality constraints.

To solve this problem, Lagrange multipliers are introduced and the solution is $w = \sum_i \alpha_i y_i x_i$, where α_i is the Lagrange multiplier, and x_i is support vector. $\alpha_i \neq 0$ only for SVs. The final SVM predictor (the decision function) can be expressed for test point x as

$$f(x) = w^T \cdot x + b = \sum_{i \in SV} \alpha_i y_i x_i^T .x + b$$

Given *f(x)*, the classification of a new point x is obtained as

$$y = sign(f(x)) = \begin{cases} +1 & f(x) > 0 \\ -1 & f(x) < 0 \end{cases},$$

$$\text{i.e., } y = sign\left(w^T x + b\right)$$

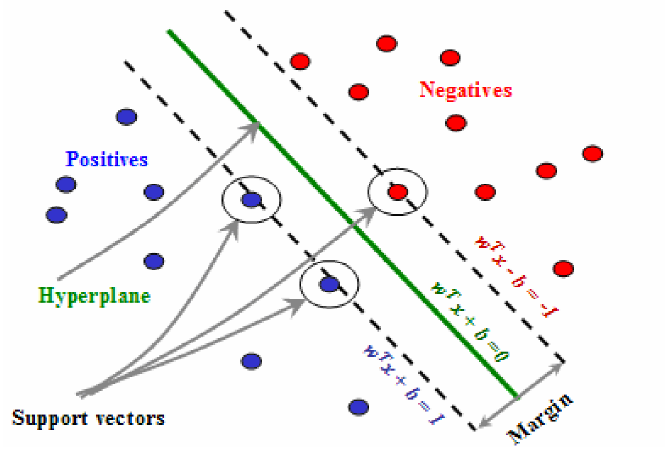


Figure 3.11: Example of two dimensional data includes positives and negatives. The hyperplane separates the examples into two classes.

In the case where data is not linearly separable, the original input space can be mapped to some higher-dimensional feature space where the training set is separable:

$$\Phi : x \rightarrow \Phi(x).$$

A simple example is shown in Figure 3.12. Assume that we need to separate the red circles (positive examples) from the blue circles (negative examples) on a plane as shown below on the left figure. Transforming this data into a higher dimensional space (*e.g.*, 3 dimensions in this case) through the mapping shown in the figure would make the problem much easier since the points are now separated by a simple plane. This embedding on a higher dimension is called the kernel trick.

If we define a kernel function K such that $\mathrm{K}(\mathrm{x}_i, \mathrm{x}_j) = \Phi(\mathrm{x}_i) \cdot \Phi(\mathrm{x}_j)$, where $\Phi(\mathrm{x})$ is a feature space transformation, then in order to classify new data x_j, the decision boundary becomes $\sum_{i \in SV} \alpha_i y_i \mathrm{K}(\mathrm{x}_i, \mathrm{x}_j) + b$. The following are examples of commonly used kernel functions:

- Linear kernel: $K(x_i, x_j) = x_i^T x_j$
- Polynomial kernel: $K(x_i, x_j) = (1 + x_i^T x_j)^P$
- Radial-Basis Function (RBF) kernel $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$

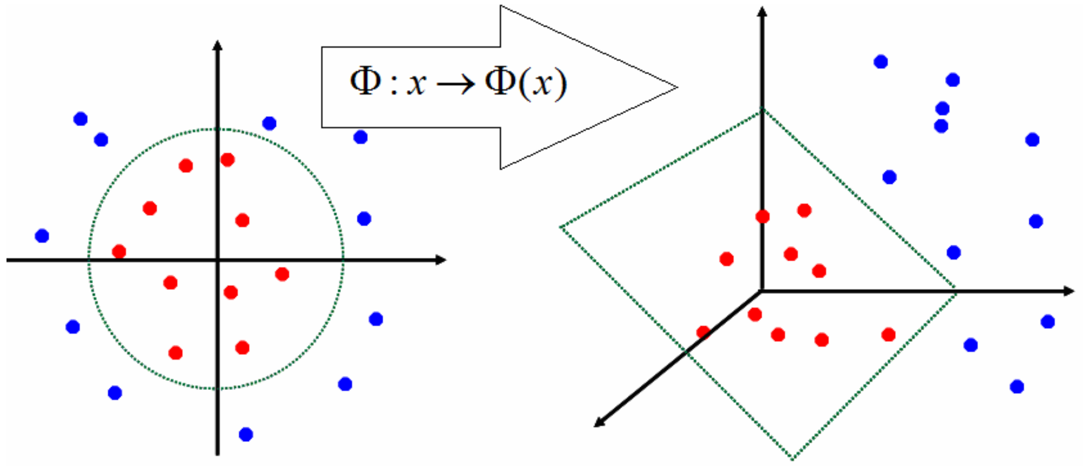


Figure 3.12: A simple example when the original input space is mapped into some higher-dimensional feature space where the training set is separable.

In this thesis, two SVM kernels – a linear kernel and a radial basis function kernel were used. Although the nonlinear kernel gives slightly more accurate results, for simplicity and speed, the linear kernel is chosen as the baseline classifier throughout this study.

In practice, the class-conditional distributions may overlap, so the case of exact separation of the training data could lead to poor generalization. Therefore the SVM is modified to allow some of the training points to be misclassified (soft margin). To do this, slack variables $\xi_n \geq 0$ are defined; where $n = 1,..., N$, and $\xi_n = 0$ for data points that are on or inside the correct margin boundary ([64], [17]).

Therefore, adding the slack variables to handle non-separable cases makes the optimization problem become $\arg\min_{w,b,\xi_n \geq 0} C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|w\|^2$ such that $y_i(w.x_i + b) \geq 1 - \xi_i$ and $\xi_n \geq 0$, where the regularization parameter $C > 0$ controls the trade-off between the slack variable penalty and the margin. In the limit $C \rightarrow \infty$, the SVM for separable data is recovered.

In the experiments shown in Chapter 4, a freely available SVM-based classifier is used: the OSU-SVM toolbox version 3.0 [65]. This is a MATLAB SVM toolbox based on the C++ package SVMLIB. It retains the high efficiency of SVMLIB but at the same time has the convenience brought from MATLAB.

3.6 Training

The data is separated into two sets: a training set and a testing set. The training set is used to train the classifier. Each instance in the training set X contains one target value and several attributes. The target value is called class label and for our binary case its value is +1 for pedestrian example and -1 for non-pedestrian example. The attributes are the features or observed variables that represent each example. The goal of the learning process is to produce an optimal decision function f (based on the training data) that can recognize pedestrian examples.

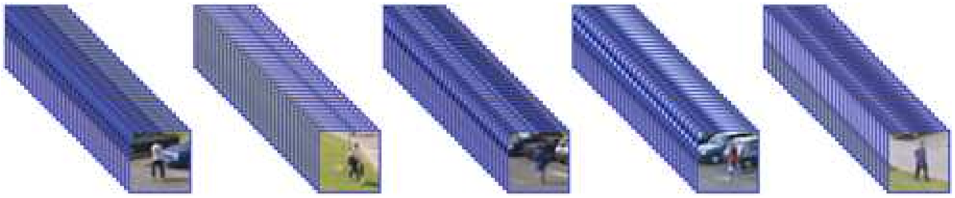
$f : R^m \rightarrow \{-1,+1\}$, that is based on data $(x_1, y_1),...,(x_l, y_l) \in X \times Y, Y = \{-1,+1\}$,

where l is the number of training instances that are vectors belonging to the space $X \subseteq R^m$. This function predicts the target values of unseen (test) data correctly given only the test data attributes.

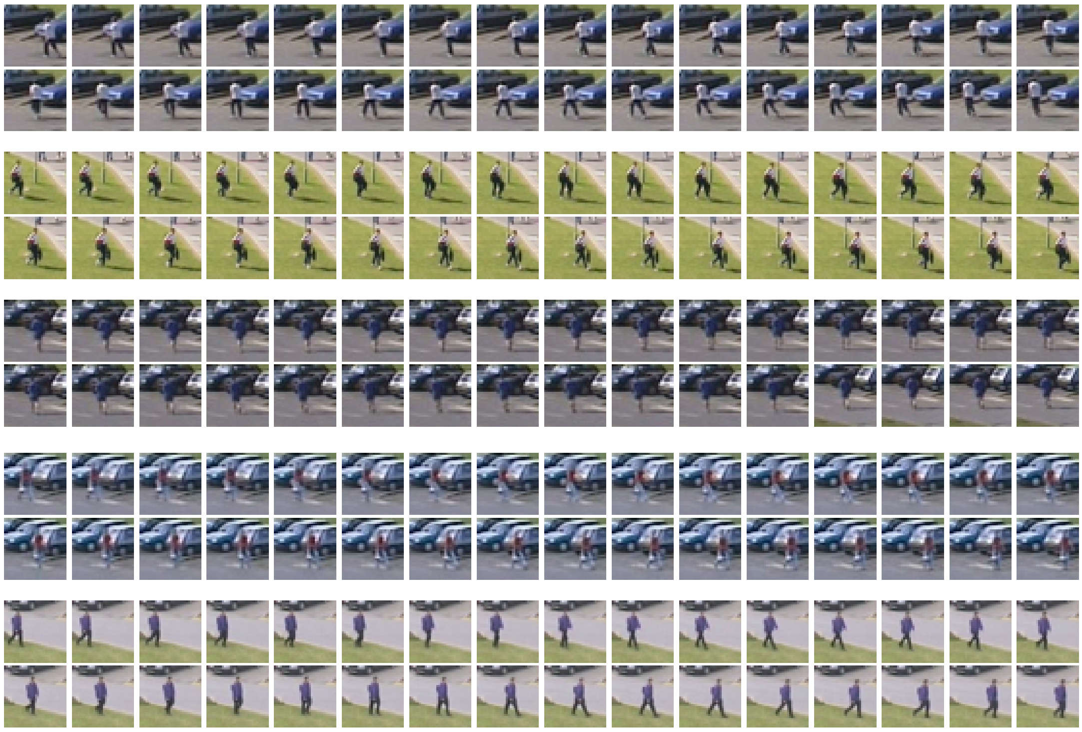
The first stage of training is to create the training data which contains positive examples (subwindows containing a pedestrian) and negative examples (completely person-free subwindows). To extract positive examples (Figures 3.13) from the training videos, the following procedure was followed. A pedestrian was manually selected in one of the images (in one of the detected ROIs) and a square subwindow was extracted from the image surrounding the pedestrian. This subwindow was scaled such that the person was 20 pixels tall, and the subwindow size was 32 × 32 pixels.

This size is large enough so the pedestrian remains within the window throughout the whole sequence, at normal (moderate) walking speeds, which usually corresponds to ½ pixels per frame. Training pedestrians were chosen un-occluded, centered in the middle slice (central slice) of the sequence, with no interference with other pedestrians, and away from image borders.

Next, a sequence of subwindows was extracted from the registered images; half of them preceding and half of them following the central image, at the same fixed place in all (registered) images, and the subwindows were similarly scaled. A total of 32 such slices were assembled into a spatiotemporal volume, representing a single positive example. We placed the starting position of the central window to ensure that the person remained in the 32 × 32 window throughout the duration of the 32 slice sequence.



(a)



(b)

Figure 3.13: (a) Positive examples. (b) Each 2 rows of 32 windows (slices) comprise one positive example. In each example, slice at time t_1 is shown on top left, and slice at time t_{32} is shown on bottom right

Negative examples (Figures 3.14) were also extracted from the training images. These were spatiotemporal volumes of the same size as the positive examples, but sampled randomly from completely person-free areas of detected ROIs. The binary classifier is trained using these examples.

3.6.1 Cross-Validation

The performance of SVM classifier depends on the choice of the regularization parameter C and the kernel parameters. For example, for RBF kernel, the bandwidth parameter γ is the only kernel parameter to be selected. Adapting the hyper parameters is referred to as SVM model selection. One of the simplest and most widely used methods for estimating prediction error is cross-validation. Cross-validation was used to select the value of C to be used with linear kernel, and also to select C and γ for the RBF kernel.

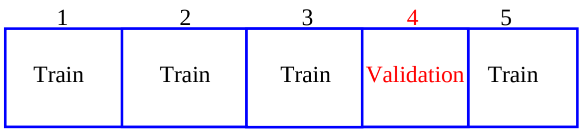
Figure 3.14: Negative examples. Each 2 rows of 32 windows (slices) comprise one example. In each example, slice at time t_1 is shown on top left, and slice at time t_{32} is shown on bottom right.

Let $L(TrueClass, PredictedClass)$ be the price paid for classifying an observation belonging to class C_1 as C_2. So, if we have input example X, with predicted class $\hat{f}(X)$, and actual class Y, the expected prediction error (or expected test error) is

$$Err = E[L(Y, \hat{f}(X))]$$

This method directly estimates the expected error *Err*; the error when method $\hat{f}(X)$ is applied to an independent test sample from the joint distribution of X and Y ([66], [61]).

In cross-validation, data is partitioned into n segments. One sample is chosen as a validation set and the accuracy of the model derived from the remaining *(n-1)* segments is scored. This is repeated for all the n samples so that every sample acts as a validation set. The predictive error obtained is used as a measure of internal validation of the predictive power of the classifier developed using the full data set. All aspects of the classifier development process should be repeated. For example, if we use 5-fold cross-validation, part of the available data is used to fit the model, and a different part to test it. The data is split into 5 roughly equal-sized parts. The scenario looks like the following:



For the n^{th} part (fourth above), we fit the model to the other $(n-1)$ parts of the data, and calculate the prediction error of the fitted model when predicting the n^{th} part of the data. We do this for $n = 1, 2, \ldots, 5$ and combine the 5 estimates of prediction error.

In this work, 5-fold cross-validation was used for parameter selection, by partitioning the training data into 5 equally sized segments and then iterations of training and validation were performed to pick the best parameters for the SVM kernels.

3.6.2 Baseline Classifier

The baseline classifier throughout this thesis uses the following parameters (except if specified otherwise): blocks are of size 2×2 cells, with no overlap, and each cell consists of 4×4 pixels. The gradient filter [-1, 0, 1] is used, 9 bins are used for gradient orientations, and we normalize volumetric blocks using the L2-Hys norm. The size of image slices is 32×32 pixels and a soft linear SVM is used with control parameter C = 0.01. The parameters were set using cross validation or empirically. Section 4.4 of the next chapter shows how the best choices of key parameters are made through experiments.

3.7 Non-Maximum Suppression

Non-maximum suppression (NMS) is applied to all detections in the image with confidence above a certain threshold as a postprocessing step to remove redundant detections. NMS is the task of finding all local maxima in an area of an image. The first appearance of the term of ‘non-maximum suppression’ was in an edge detection context [67], and then it was adopted for other applications. The work of [68] shows different aspects of implementing NMS tasks.

As described in the algorithm above, during testing a volumetric detection window is scanned across detected ROIs in the image sequence at all positions within the ROI and all scales within the image pyramid. The detector typically generates many multiple responses around the target object (*i.e.*, the pedestrian). A standard convention to deal with this is to remove any detector responses in the neighborhood of detections with locally maximal confidence scores.

Each detection is defined by a score and a bounding box *(BB)*. The score is the decision value the classifier produces as a classification output. The *BB* is of the same size and at the same (*x*, *y*) coordinates of the detection window that detected the pedestrian. For each instance of a pedestrian, we may get multiple overlapping detections; that is, overlapping *BBs*.

Detections are sorted according to their detection decision value. Then a greedy algorithm is used to select the detection with the highest score, and nearby *BB* detections are merged to a single final detection. The *BB* that is covered with at least 50% of previous *BB* is considered a repeated detection and eliminated. This method is commonly used by other researchers in the field of pedestrian detection, (*e.g.*, [3]).

CHAPTER 4: EXPERIMENTS AND RESULTS

This chapter presents the experimental work, the various datasets used for evaluation, and the results and findings. In addition to evaluating our algorithm, we compare our results to two other algorithms: the Dalal-Triggs algorithm [2], which is among the most popular approaches for single frame pedestrian detection, and the Jones and Snow algorithm ([1], [33]), which was the best performing algorithm on low resolution pedestrians that we found. If we limit our algorithm to use only a single image, it is essentially the same as the Dalal-Triggs algorithm. Therefore, we can directly compare the performance of our algorithm to that of the Dalal-Triggs algorithm on each of the datasets. In the case of the Jones and Snow algorithm, we did not have an implementation of that to work with. However, Jones and Snow give performance results on one of the datasets that we used, so we can compare our algorithm to theirs on that dataset.

Section 4.1 gives an overview of metrics used to evaluate algorithm performance. Datasets used are presented in Section 4.2. Section 4.3 gives an overview of the results. Section 4.4 analyzes the effect of the number of slices on performance. Detection examples and discussion are presented in Section 4.5. Performance and the effect of different parameters are discussed in Section 4.6. In Section 4.7 we discuss the effect of frame randomization.

4.1 Evaluation Metrics

Different measures can be used to evaluate different characteristics of classification algorithms. One well known approach used for evaluating a detector's performance is the *Receiver Operating Characteristics* (ROC) curve. The ROC is based on another commonly used evaluation tool called the *confusion matrix*, [69]. These two tools are used because of their ability to visualize characteristics of binary classifiers.

4.1.1 Confusion Matrix

In the machine learning field, a table known as a confusion matrix is used to visualize the performance of an algorithm. Its name reflects the fact that the matrix makes it easy to see if the system is confusing two classes. The confusion matrix contains the information about actual and predicted classifications made by a classification system [70]. Table 4.1 shows the confusion matrix for a two-class classifier.

Table 4.1: Confusion matrix for two-class classifier

		Predicted Class	
		Positive (pedestrian)	Negative (non-pedestrian)
True Class	Positive (pedestrian)	TP	FN
	Negative (non-pedestrian)	FP	TN

This matrix forms the basis for many commonly used metrics in the object detection field. The numbers along the major diagonal (*TP*, *TN*) represent the correct decisions made, and the numbers off this diagonal (*FP*, *FN*) represent the errors (the confusion) between the classes. In the context of this study, the entries in the confusion matrix can be defined as follows:

- **False Positives (*FP*):** examples predicted as positive (pedestrian), which are from the negative class (non-pedestrian).
- **False Negatives (*FN*):** examples predicted as negative (non-pedestrian), whose true class is positive (pedestrian).
- **True Positives (*TP*):** examples correctly predicted as pertaining to the positive class (pedestrian).
- **True Negatives (*TN*):** examples correctly predicted as belonging to the negative class (non-pedestrian).

In this thesis, "examples" are the volumes within ROIs that were tested and classified by the detector.

In this work, we use standard metrics that are defined based on the entries of the confusion matrix. The "Detection Rate" (*DR*) measures how accurate the classifier is in sensing targets of interest. It is the proportion of positive examples (pedestrians) that were correctly identified.

DR is also called "Recall", "True Positive Rate (*TPR*)", or "Sensitivity". *DR* is calculated using the equation:

$$DR = \frac{\text{Positives Correctly Classified}}{\text{Total Positives}}$$

$$DR = \frac{TP}{TP + FN}$$

The "False Positive Rate" (*FPR*) is the proportion of negative examples (non-pedestrians) that were incorrectly classified as positives (pedestrians), as calculated using the equation:

$$FPR = \frac{\text{Negatives Incorrectly Classified}}{\text{Total Negatives}}$$

$$FPR = \frac{FP}{TN + FP}$$

Another commonly used metric is "Accuracy", which is the proportion of the total number of predictions that were correct. It is determined using the equation:

$$Acc = \frac{\text{Correctly Classied Examples}}{\text{Total Number of Trials}}$$

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

We define the ratio β as the proportion of positive class in the dataset such that

$$\beta = \frac{P}{P + N}$$

where *P* is the number of positive examples, and *N* is the number of negative examples. We can obtain the following equation:

$$Acc = \beta \times sensitivity + (1 - \beta) \times specificity$$

If a dataset is balanced, then $\beta \approx 0.5$; in this case, maximizing the overall accuracy is equal to maximizing both sensitivity and specificity with the same weight. However, in the case with an imbalanced dataset with $\beta \approx 0$ (positive class minority), maximizing the overall accuracy will bias toward maximizing specificity more than sensitivity, and vice versa as β approaches 1.

4.1.2 Receiver Operating Characteristic (ROC) Curve

The ROC curve is a useful tool for visualizing and evaluating binary classifier performance. It depicts the tradeoff between hit rates (*DR*) and false alarm rates of a classifier. The ROC curve illustrates the performance of a binary classifier system as some of its parameters are tuned (varied). It is created by plotting *DR* (y-axis) versus *FPR* (x-axis); that is, the fraction of true positives out of the total actual positives versus the fraction of false positives out of the total actual negatives ([71], [72], and [73]).

To show some details of how to read the ROC curve, Figure 4.1 shows examples of three ROC curves. Point (0, 0) represents a non-positive classification; this means the classifier's output was negative all the time. Point (1, 1) on the other hand, represents all-positive classification all the time. That is, the output of the classifier was always positive. Point (0, 1) represents perfect classification, whereas point (1, 0) represents perfect misclassification. Thus, the ROC curve helps in choosing a threshold which defines a point towards the top left of the curve that has the effect of maximizing *TPR* while minimizing *FPR* (in the direction of the red arrow in Figure 4.1).

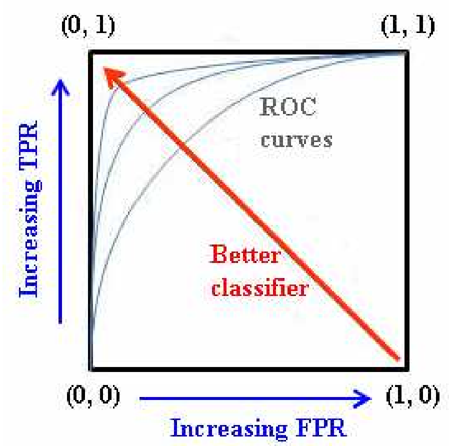


Figure 4.1: General example of ROC curve.

The operating point for a classifier can be chosen so that the classifier gives the best trade-off between the costs of decreasing true positives against the costs of increasing detected false positives. For example, if the target application is a surveillance system and all positive responses must be shown to a human operator who can only process a certain number of events per some time period, it is important to limit the false positive rate to some acceptable value. The performance of the classifier is then determined by the detection rate at that false positive rate. This can be used to compare two classifiers – for example, in Section 4.3, when we compare our method to other detection methods; we evaluate our system's *DR* at the same *FPR* that the other methods use.

4.2 Experimental Procedure and Datasets

Following the method of [3], a detection is considered to be correct if there is at least 50% overlap between the detection window and the ground truth; *i.e.* they share 50% of the area. To save time, we apply the detector to every fourth frame in a sequence, instead of every frame. We still construct the volumes using up to 32 consecutive frames, but the volumes in time direction are shifted every four frames. We treat detections at different times as if they were independent. This means that a pedestrian detected in frames 1-32 and also in frames 5-36 is counted as two detected instances. A tracker (perhaps developed in future work) could assemble these detections into a single track.

There are a number of datasets that have been used for pedestrian detection. Many of these are specifically designed to evaluate pedestrian detection from moving cars. Since our target application is surveillance video from stationary or aerial images, these datasets are not appropriate for evaluating our system. Other datasets consist of video taken by indoor surveillance cameras. In these datasets the people are relatively large in the images. Since our target problem is detecting people in low resolution images, these are also not applicable.

We found five commonly used datasets that are representative for our application. These are two stationary camera datasets (PETS2001, VIRAT public 1.0), and three aerial datasets (VIRAT Fort AP Hill, UCF-2009, UCF-2007). In the stationary datasets, videos were collected from a stationary surveillance camera. In such scenarios, no video stabilization is needed. All the images were converted from color to grayscale since color information was not used during feature extraction. In addition, grayscale images were used during image registration.

All these datasets are low resolution; however, the height of people in some images is greater than 20 pixels. Although our detector was designed to detect people with heights of 20 pixels, it can still detect these larger pedestrians. Since an image pyramid was used, the detector can detect people at the image level where the height was about 20 pixels. This guarantees that at some level of the pyramid the people will be close to 20 pixels height and can be detected by the algorithm.

4.2.1 Stationary Datasets

The PETS 2001 dataset was released in 2001 [74]. PETS 2001 is probably the most popular of the PETS series in automated surveillance research [67]. The PETS 2001 dataset was also used by Jones and Snow to evaluate their algorithm [33]. The PETS 2001 dataset contains 16 video sequences of about two to four minutes length, with a frame rate of 25 frames/second, and frame size of 768 pixels in width and 576 pixels in height. Half the videos are designated as training, and half for testing. The sequences were taken by two stationary cameras mounted on high vantage points looking down upon a street and parking lot in front of a building. Cars and pedestrians periodically move through the scene at different times of the day with different light conditions (Figure 4.2).

The second stationary camera dataset is the stationary VIRAT (Video Image Retrieval and Analysis Tool) dataset. This dataset was designed to be realistic, natural and challenging for video surveillance domains in terms of activities and pixel resolution on pedestrians [7]. The video sequences were taken by stationary cameras mostly at the top of high buildings to record large numbers of event instances across a very wide area while avoiding occlusion as much as possible. The cameras look down upon a scene containing streets with buildings, trees, and parking lots. Cars and pedestrians periodically move through the scene. Pedestrians in this dataset appear in cluttered backgrounds and have a wide range of appearances, due to different poses, body sizes, and outdoor lighting conditions. The dataset consists of twenty video sequences; each approximately 0.5 to 5 minutes length with a frame rate of 30 frames/second, and frame size of 1280 pixels in width and 720 pixels in height. The heights of pedestrians within the videos range from 25 to 200 pixels (Figure 4.3).

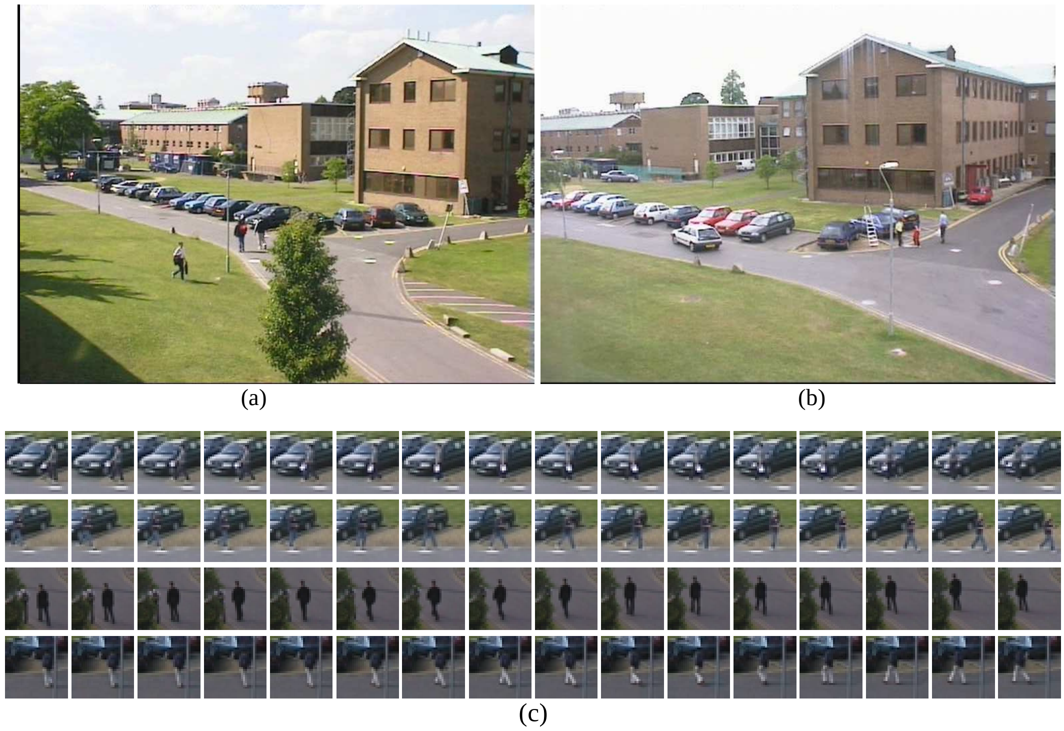


Figure 4.2: (a) and (b) Two frames from PETS 2001. (c) Four different example sequences. Note: video sequences were taken at different time of the day, with different illumination conditions.

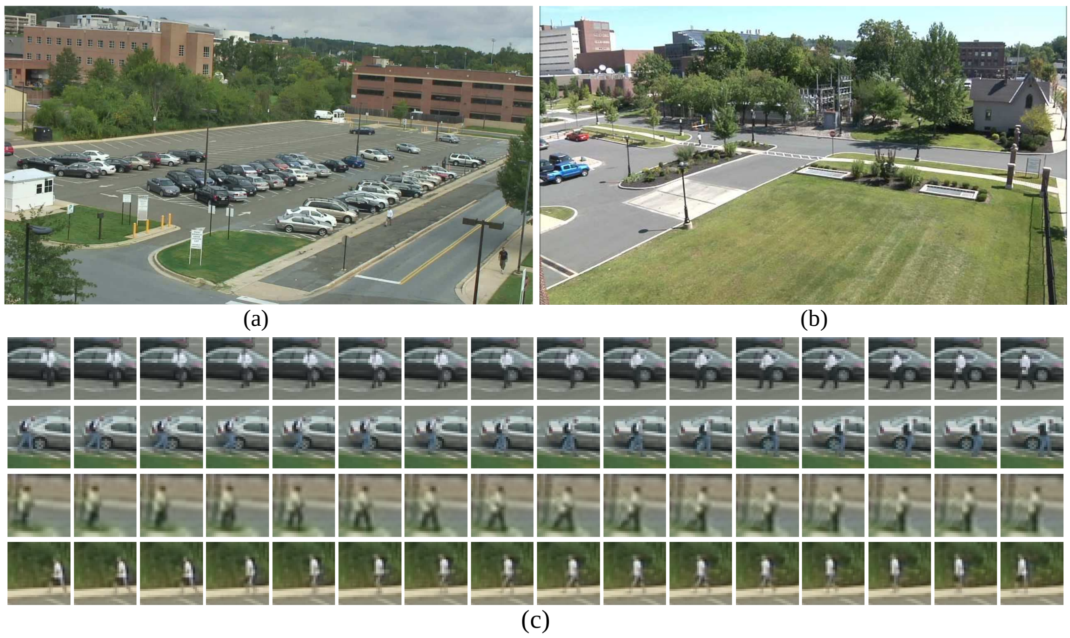


Figure 4.3: (a) and (b) Two frames from stationary VIRAT. (c) Positive example sequences.

4.2.2 Aerial Datasets

The VIRAT Fort AP Hill aerial dataset was recorded using an electro-optical sensor from a military aircraft flying at a height up to 1000 meters. The resolution of these aerial videos is 640×480 with 30Hz frame rate, and the typical pixel height of people in the collection is about 20 pixels. The data was collected in natural scenes to be realistic, showing people in standard contexts. Directed actors were minimized; most were general population. The videos include buildings and parking lots where people and vehicles are engaged in different activities. The data is challenging in terms of low resolution, uncontrolled background clutter, diversity in scenes, rapidly changing viewpoints, changing illumination, and low pedestrian image sharpness [7]. Figure 4.4 shows examples of frames and sequences from the aerial VIRAT dataset.

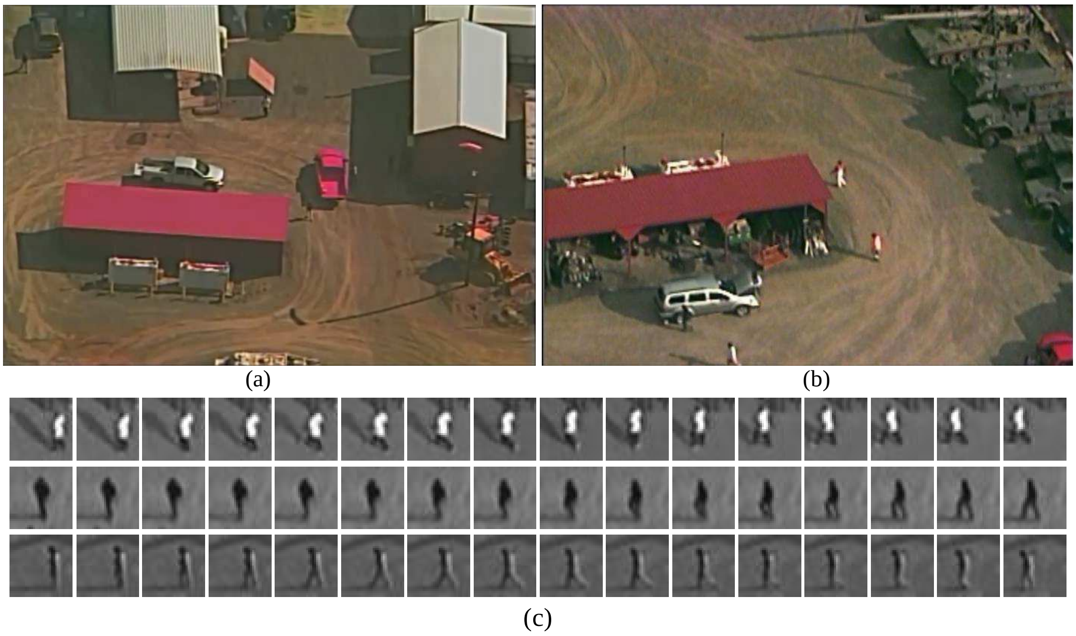


Figure 4.4: (a) and (b) Two example frames from aerial VIRAT. (c) Different example sequences created from registered frames.

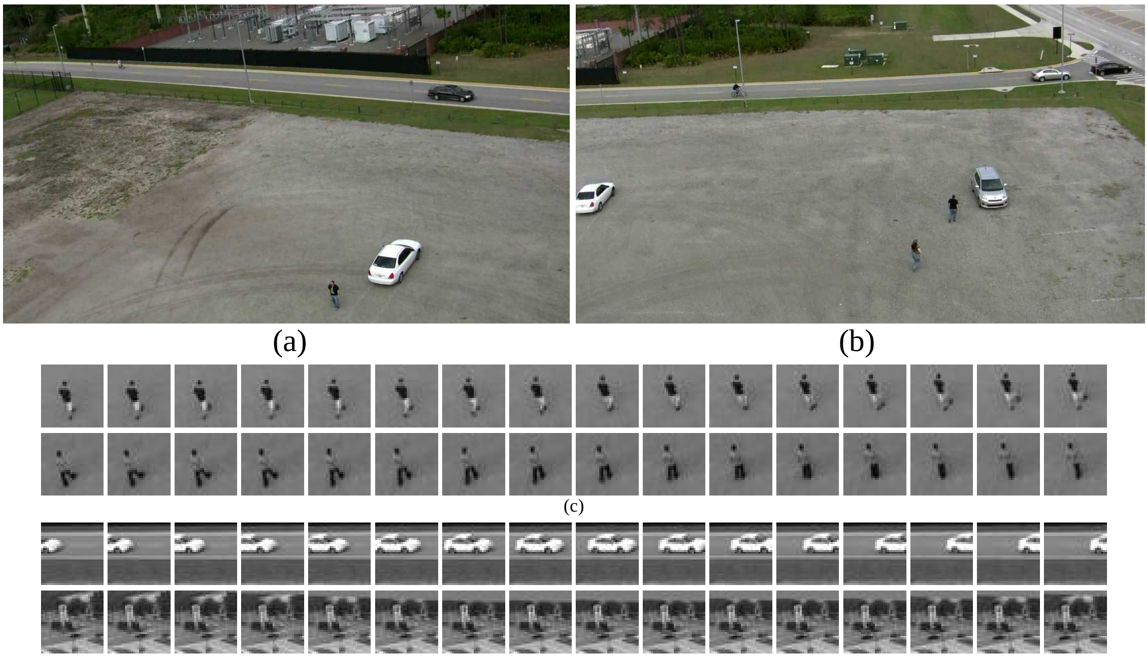
This dataset also contained sequences with some very low-resolution pedestrians, in which pedestrians are of less than 20 pixels in height. Figure 4.5 shows some very low-resolution images from the aerial VIRAT dataset, and some example sequences. Although our algorithm is designed to detect pedestrians of height 20 pixels or more, we also evaluated the algorithm on some of this data as well, to understand how the performance degrades as the resolution is lowered.

Figure 4.5: (a) and (b) Two example frames from aerial VIRAT. (c) Two very low-resolution positive example sequences created from registered frames. (d) Two negative example sequences.

The UCF-2009 dataset is from the University of Central Florida. It is also known as the UCF-Lockheed Martin Dataset (we call it UCF-2009 to distinguish it from an earlier UCF dataset of 2007). Video sequences were obtained using an R/C-controlled blimp equipped with a camera mounted on a gimbal in a dirt parking lot near the football stadium in Florida. The flying altitudes ranged from 400–450 feet. Actions were performed by different actors. The UCF-2009 dataset has a resolution of 540×960 pixels with a 23Hz frame rate. Figure 4.6 shows two example frames and four example sequences.

The UCF-2007 dataset is an earlier dataset from UCF, and is more challenging (compared to the UCF-2009 dataset) due to large variations in camera motion, rapidly changing viewpoints, changes in object appearance, pose, object scale, cluttered background, and illumination conditions. In this dataset, people walk in different directions in a park and get close to trees and bushes. Figure 4.7 shows two example frames and some example sequences.

The UCF-2007 data suffers from interlacing, motion blur, and poor focus. Interlaced video is a technology that was developed in the early days of television. The interlaced signal contains two fields of a video frame captured at two different times, which exhibits motion artifacts called interlacing effects. The effect becomes more obvious particularly in areas with objects in motion (Figure 4.8 (top), page 54).



(a) (b) (c) (d)

Figure 4.6: (a) and (b) Two example frames from aerial UCF-2009. (c) Positive example sequences (pedestrians) created from registered frames. (d) Negative example sequences.

(a) (b) (c) (d)

Figure 4.7: (a) and (b) Two example frames from aerial UCF-2007 dataset. (c) and (d) Positive examples sequences created from registered frames.

To rectify the interlace artifacts we remove every other row and column, therefore halving the frame resolution (Figure 4.8 (bottom)). The UCF-2007 dataset has a resolution of 854×480 pixels with a 30 Hz frame rate, and after deinterlacing the resolution becomes 427×240 pixels.

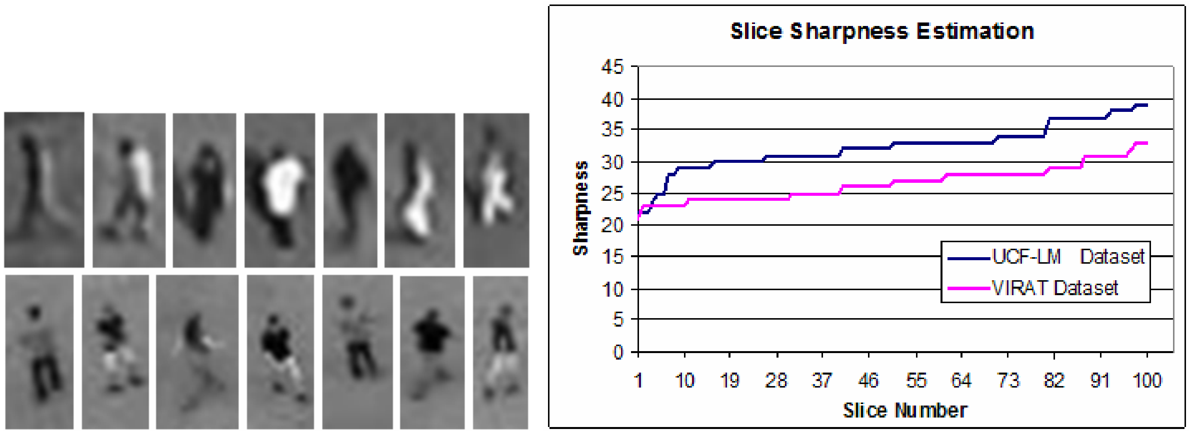
Figure 4.8: (Top) Example frame from UCF-2007 dataset with interlacing artifacts. (b) The same frame of part (Bottom) after deinterlacing.

4.2.3 Image Sharpness Estimation

We noticed that the datasets differ in image quality. In some datasets, such as aerial VIRAT, the images are not as sharp. This is important because our algorithm is based on image gradients. Large gradients occur at sharp discontinuities in the image, which occur at boundaries of objects and therefore represent information on shape. If images are blurred, the image gradient decreases in magnitude and the orientation becomes less reliable.

We can quantify the sharpness of the images. Many algorithms have been proposed to estimate image sharpness or blurriness. Most methods use edge-appearance models [75] since edge-appearance is the most affected component by image blurriness. Some of these algorithms compute the average gradient magnitude in the image. Kumar *et al.* [76] proposed to use the average absolute value of the discrete second derivative.

The method we used to estimate sharpness was to compute the average gradient magnitude in targeted sub-images. Since the sub-images of interest to our application are the ones containing pedestrians, we measured sharpness in those. Figure 4.9 (a) shows some examples from aerial VIRAT and UCF-2009. Figure 4.9 (b) shows a sharpness comparison between a sample of 100 slices from those two datasets. The results confirm that the VIRAT dataset is not as sharp as the UCF-2009 dataset, so we hypothesize the detector performance will be lower.



(a) (b)

Figure 4.9: (a) Examples from VIRAT aerial dataset (top raw), and UCF 2009 dataset (bottom). (b) Sharpness estimation for a sample of 100 slices from VIRAT and 100 slices from UCF datasets (sorted by sharpness value).

4.3 Detector Performance Overall Results

The detector was applied to the test videos of the five datasets described above. The results in this section use the following default parameters: Blocks are of size of 2×2 cells, with no overlap, and each cell consists of 4×4 pixels. Gradients are computed using the [-1, 0, 1] filter. We use 9 bins for gradient orientations, and normalize volumetric blocks using the clipped L2-norm as described in Section 3.4.2. The size of image slices is 32×32 pixels and the volume consists of 16 slices. A soft linear SVM is used with a cost parameter C (with a value of 0.01 determined through cross-validation) that controls the trade-off between slack variable penalty and the margin. In Section 4.4, we present an analysis of the effect of changing these parameters.

To show the main concept, Figure 4.10 presents an example of the results of applying the algorithm to one of the aerial datasets. As described in Chapter 3, the process involves video stabilization (except for stationary camera videos), determining the temporary background frame, computing the difference image, defining initial foreground, applying some morphological operations, defining final ROIs, and applying pedestrian detector around these ROIs to search for pedestrians.

The moving object detector that detects ROIs in image sequences gives a good detection rate in detecting potential moving objects. For example, in 520 sequences of frames from PETS 2001 dataset (each sequence contains 16 frames) and 524 sequences from Aerial VIRAT dataset (each sequence contains 16 frames), the moving object detection rate was more than 95%. In these sequences, most of the ROIs containing pedestrians are detected. For example, in Aerial VIRAT, in 1607 ROIs only 49 are not detected, and in PETS 2001, in 1929 ROIs only 88 are not detected.

4.3.1 Stationary Datasets

For the PETS 2001 dataset, we extracted 2,560 training examples from the training videos. 960 of them were positive examples and 1600 were negative examples. After training, the detector was applied to ROIs in the test videos. The total number of tested examples was 1,235 positive examples (16 slices each) and 8,730 negative examples (16 slices each). The subjects in the examples are always upright walking, with a wide range of variations in pose, appearance, clothing, illumination and background. Pedestrians are walking in various directions at different distances from the camera, and examples have a variety of blurriness.

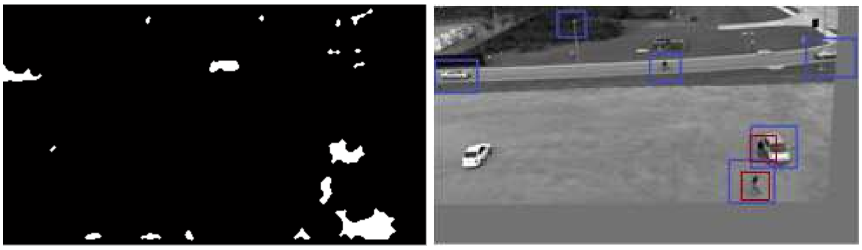
Using the detector with the default parameters, a detection rate of 94.7% was achieved with a false positive rate (*FPR*) of 10^{-6}. This means that 94.7% of the actual pedestrians were detected correctly as pedestrians, and for every 1,000,000 non-pedestrian volumes tested, only one example would be classified as a pedestrian.

At the same *FPR*, the Dalal algorithm [2], which uses single images, achieved a detection rate of 73%. On the same dataset, the Jones and Snow algorithm [33] achieved a detection rate ranges from 84% to 93%, and when they combine 8 detectors (each one was trained separately) they achieve a detection rate of 93%. Figure 4.11 (top) shows the ROC curves of the three detectors. To generate these curves, we varied the detection threshold of the detector.

(a) Frame number 1. (b) Frame number 16.

(c) Frame number 1 (gray scale). (d) Frame number 16 registered to frame number 1.

(e) Temporary background. (f) Difference = Frame number 16 - background.



(g) Detected motion regions. (h) Detected ROIs (blue), and pedestrians (red).

Figure 10: Results of the steps of applying the proposed approach on a sequence of images from UCF 2009 dataset.

For the stationary VIRAT dataset, the training set consisted of 1760 examples: 780 of them were positive examples and 980 were negative examples. The trained detector was applied to video sequences different from the training set. After training, the detector was scanned on detected ROIs in sequences of images containing positives and negatives. The total number of tested examples was 720 positive examples (16 slices each) and 3860 negative examples (16 slices each).

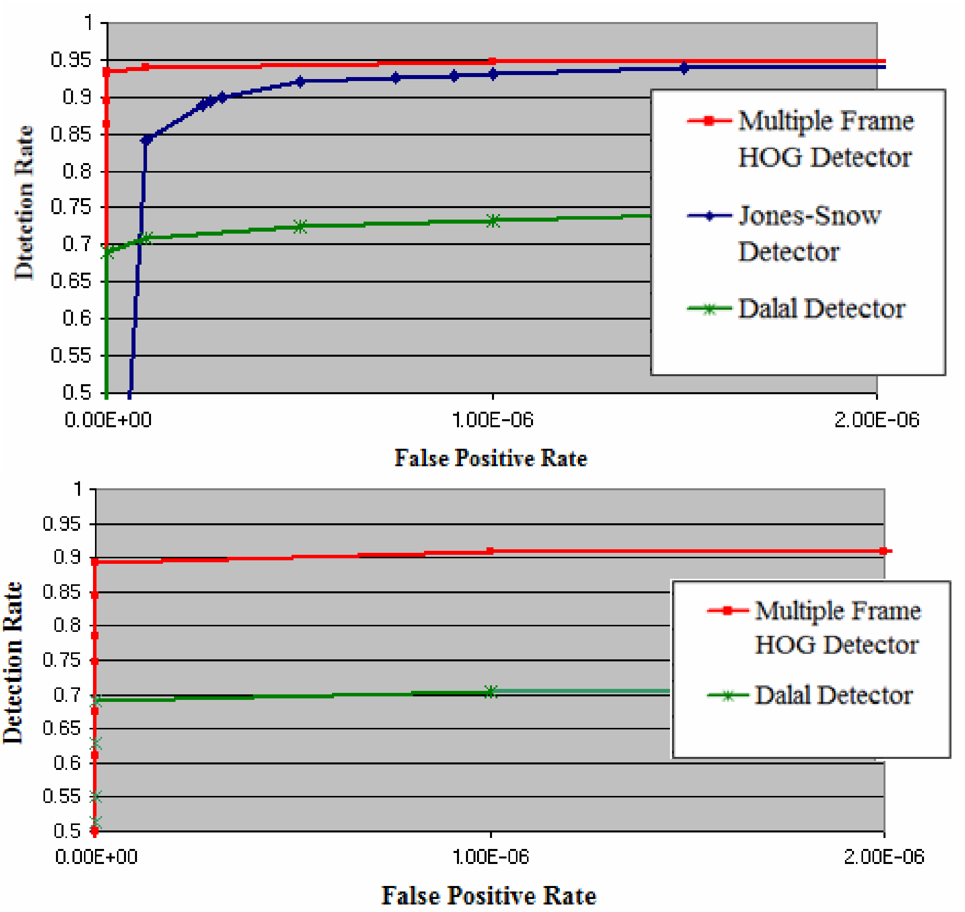


Figure 4.11: (Top) ROC curves of the three detectors on PETS 2001 dataset. (Bottom) ROC curves of the two detectors on Stationary VIRAT dataset.

Using the detector with the default parameters, we achieved a detection rate of 91% with a false positive rate of 10^{-6}. This means that 91% of actual pedestrians were detected correctly as pedestrians, and 1 in 1,000,000 tested non-pedestrian volumes were incorrectly classified as pedestrian. At the same *FPR* rate, the static detector of Dalal and Trigs [2] achieved a *DR* of 70%, on the same dataset.

4.3.2 Aerial Datasets

For the aerial VIRAT dataset, the training set consisted of 1,280 positive examples (16 slices each) and 1,280 negative examples (16 slices each). After training, the detector was scanned on detected ROIs on test set sequences. 30,000 frames were used as a test set. A total of 12,600 volumes were classified during the scanning over the detected ROIs, of which 5,016 were positive examples and 7,584 were negative. Using the detector with the default parameters, it achieved a *DR* of 78% at *FPR* of $4{\times}10^{-3}$. This value of FPR means that only 4 in 1000 tested non-pedestrian volumes were classified as pedestrians. At the same *FPR* the single-frame Dalal detector achieved a detection rate of 39%.

We also applied the detector to portions of this dataset where pedestrians are less than 20 pixels in height. In this experiment, the training set consisted of 160 positive examples (16 slices each) and 320 negative examples (16 slices each). After training, the detector is scanned within each detected ROI in test set sequences. A total of 460 volumes were classified during the scanning over the detected ROIs, of which 160 were positive examples and 300 were negative. Using our dynamic detector on this dataset achieved a *DR* of 55% at *FPR* of $4{\times}10^{-3}$.

For the UCF-2009 dataset, the training set consisted of 1,000 positive volumes (16 slices each) and 1,000 negative volumes (16 slices each). During testing, the trained detector was scanned within each detected ROI in test set sequences. More than 24,000 frames were used as a test set. A total of 5,880 volumes were classified; 2,184 of them were positives, and 3,696 were negatives. Using the detector with parameters tuned for the best performance on this dataset, *DR* is 92% at *FPR* of $4{\times}10^{-3}$. At the same *FPR* the Dalal detector achieved a *DR* of 50%.

For the UCF-2007 dataset, the training set consisted of 250 positive volumes and 250 negative volumes. During testing, a total of 500 volumes were classified; half of them positives and half of them were negatives. Using the detector with the default parameters, *DR* is 73% at *FPR* of $4{\times}10^{-3}$. At the same *FPR* the Dalal detector achieved a *DR* of 41%.

Figures 4.12 (a), (b), and (c) show the ROC curves for our multi-frame HOG detector and the single frame Dalal detector on these two datasets. The new detector outperforms the single-frame Dalal detector at all *FPR* rates.

4.3.3 Summary and Discussion

Using the default algorithm parameters, our new multi-frame HOG based detector consistently outperforms the Dalal single-frame detector. It also outperforms the Jones and Snow multi-frame detector on the PETS 2001 dataset (we did not have an implementation of their code, and so could not compare their results on the other datasets).

Table 4.2 shows detection rates for the Dalal algorithm and the new multi-frame HOG-based algorithm. *DR*s are evaluated at a constant *FPR* of 1×10^{-6} (for the first two) and 4×10^{-3} (for the last three).

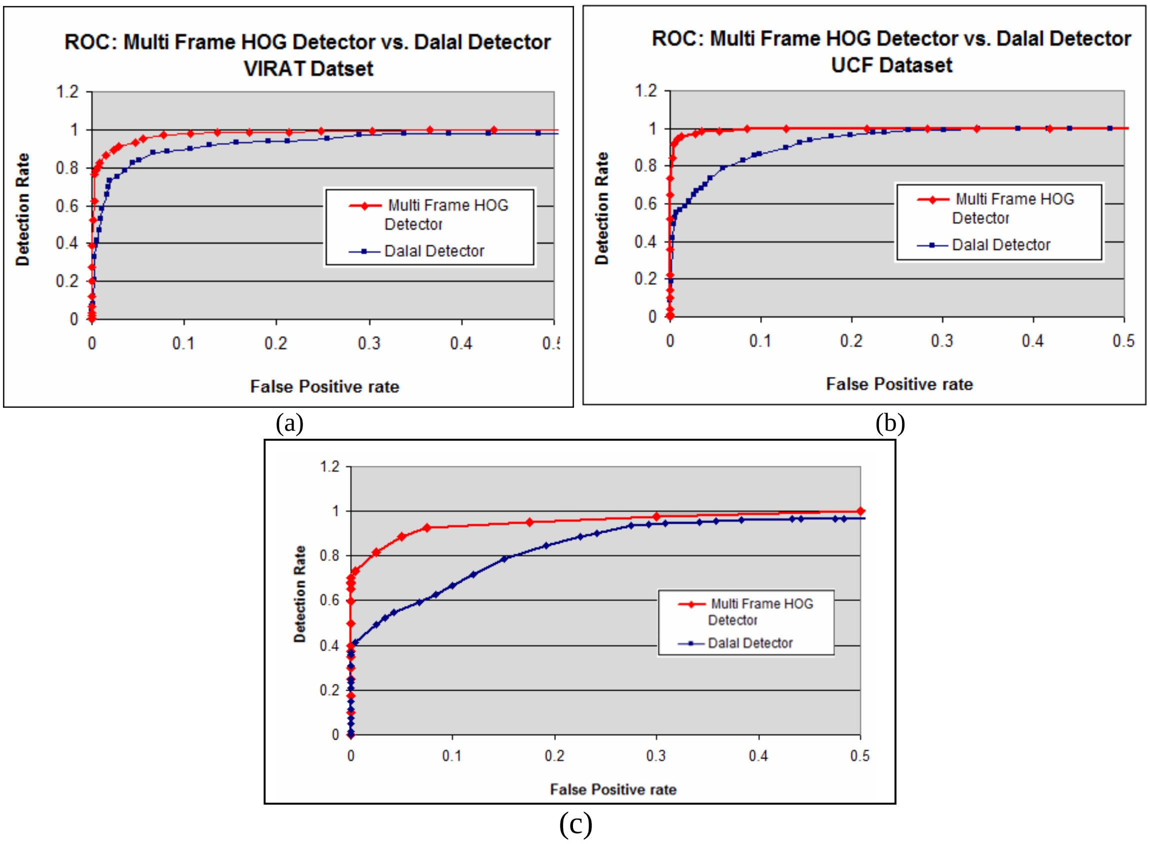


Figure 4.12: ROC curve for Multi-Frame HOG detector and the Dalal detector on (a) Aerial VIRAT dataset and (b) Aerial UCF-2009 dataset. (c) Aerial UCF-2007 dataset.

Table 4.2: *DRs* for the Dalal detector and the new detector

Dataset	Dalal Algorithm *DR*	Multi-frame HOG based *DR*
PETS 2001	73%	94.7%
Stationary VIRAT	70%	91%
Aerial VIRAT	39%	78%
UCF 2009	50%	92%
UCF 2007	41%	73%

Figure 4.13 shows the ROC curves for the multi frame HOG detector on all five datasets. Several observations can be made from these curves. First, the detector gives better detection rates and lower false positive rates when it is applied to stationary videos than when it is applied to aerial videos. One reason is that the aerial videos tend to be more noisy and blurry than stationary videos since they are taken from higher altitudes. The effective resolution of pedestrians in aerial videos is lower than that of stationary videos. Another reason is that in aerial videos, the videos need to be stabilized, and the stabilization is not perfect. The diversity in scenes in aerial scenarios requires the detectors to learn a greater variety of cluttered backgrounds.

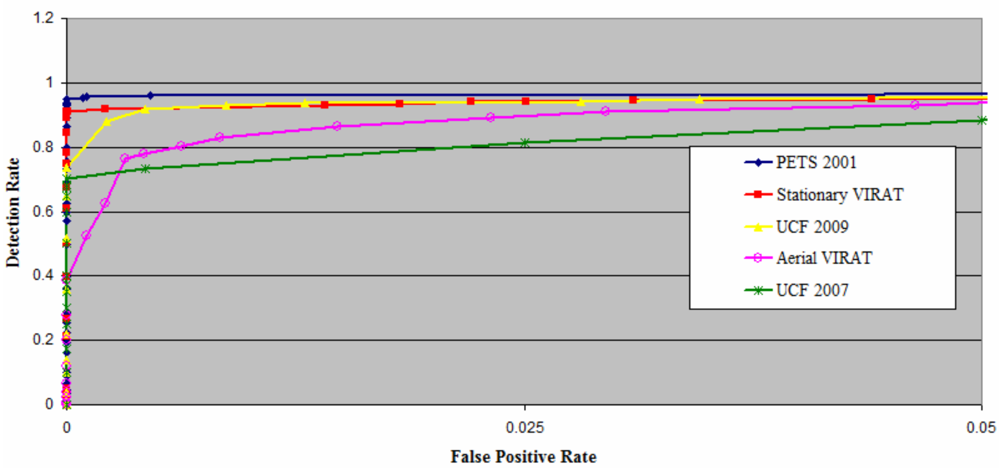
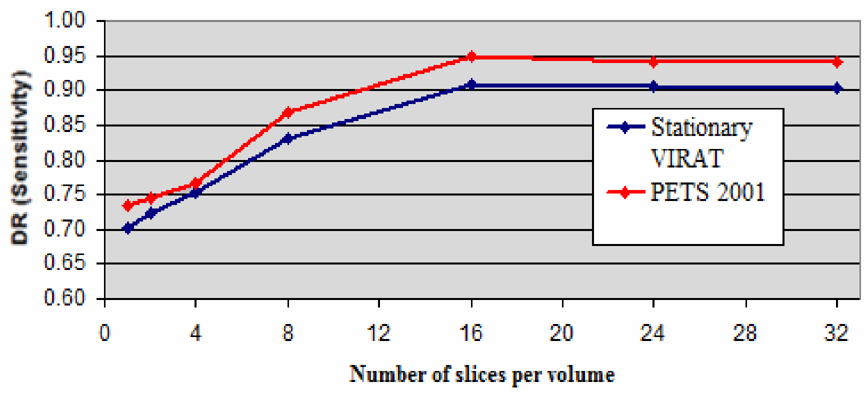


Figure 4.13: ROC curves for the Multi-Frame HOG Detector on different datasets.

For the aerial datasets; aerial VIRAT has detection rates consistently lower than those of the UCF-2009 dataset. As shown in Section 4.2.3, the images in aerial VIRAT are not as sharp as in UCF-2009. Also, the background clutter and diversity in scenes in aerial VIRAT makes it more challenging for the detector to be trained than the case of UCF-2009.

4.4 Effect of Number of Slices on Performance

One of the key contributions of this work is the use of multiple frames to compensate for the lack of information in low-resolution scenarios. We tested the effect of the number of slices N on the performance of the detector. For values of N ranging from 1 to 32, we trained the classifier on volumes with N slices, and then tested the trained classifier on volumes with the same number of slices. The curves of Figure 4.14 show detection rates (for constant *FPR*) as *N* is varied for stationary and aerial datasets.



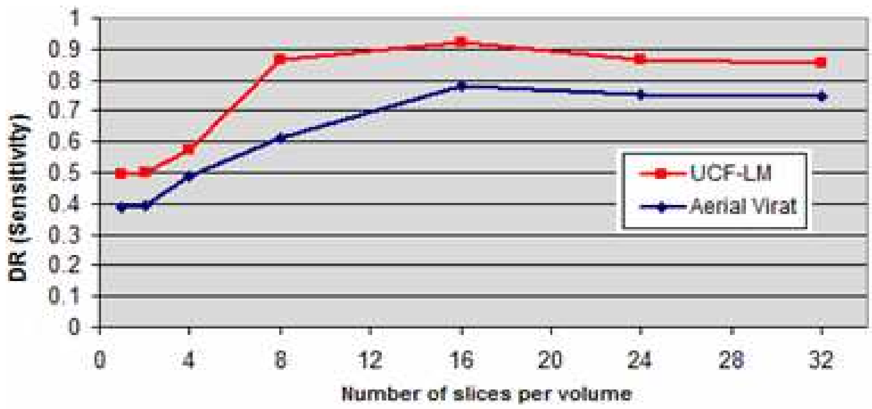


Figure 4.14: The effect of the number of slices per volume on *DR*. (Top) stationary datasets. (Bottom) Aerial datasets.

The curves (Figures 4.14) show that improvement increases with the number of slices until a total of 16 slices is reached. After that, adding more slices does not improve *DR*s. One possible reason is that there is enough motion in 16 frames for the classifier to tell whether the tested example is a pedestrian or not. Note that the case where the number of slices is equal to one represents the standard single-frame HOG-based pedestrian detector method (Dalal detector [2]), and it always gives the lowest detection rates.

In the aerial VIRAT dataset, the use of a single frame (*i.e.*, one slice per volume classifier) gives a *DR* of 40% at *FPR* of 4×10^{-3}. At the same *FPR*, the use of 16 slices per volume raises the *DR* to 78%. For the UCF-2009 dataset, the use of a single frame gives a *DR* of about 50% at *FPR* of 4×10^{-3}, and as the number of frames used reaches 16 frames, the *DR* goes up to 92% at the same *FPR*. ROC curves are shown in Figure 4.15, for classifiers using different numbers of slices.

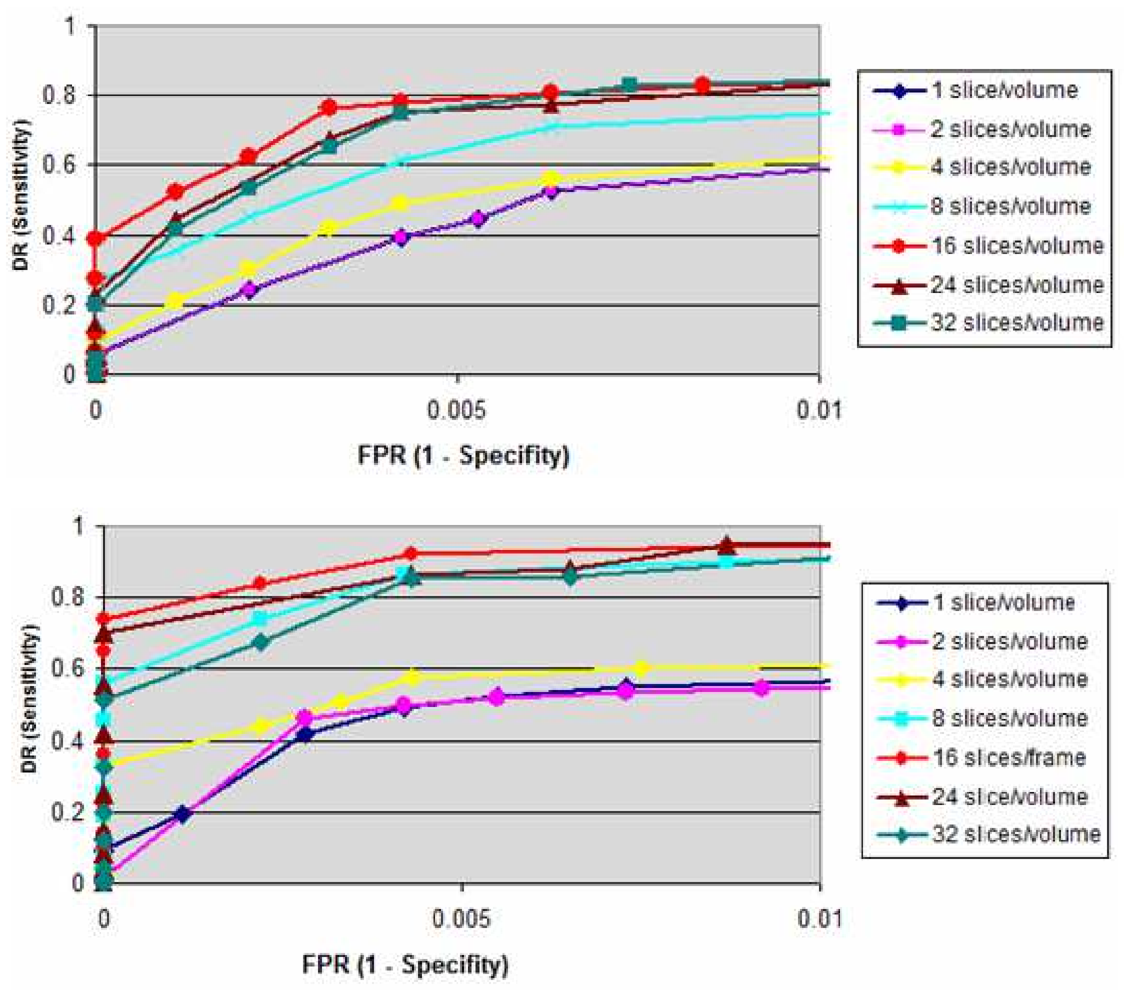


Figure 4.15: ROC of the classifiers with different number of slices/volume: (Top) Aerial VIRAT Dataset. (Bottom): Aerial UCF-2009 Dataset.

4.5 Detection Examples and Discussion

Figures 4.16–4.19 show example frames from the different datasets used in this work. Detection results are shown as boxes, where *TP* is “true positive”, *FP* is “false positive”, *TN* is “true negative”, and *FN* is “false negative”.

Figure 4.16 (a) and (b) shows two frames from the aerial VIRAT dataset. Examples of detection results are shown as boxes (we are not showing all the *TN*s in these figures). Figure 4.16 (c) shows the sequence of slices for one of the *TP* detections. Figure 4.16 (d) shows an example of a sequence of slices for *TN* detections. *TNs* result from scanning the classifier around false ROIs that correspond to motion regions resulting from non-pedestrian motion (*e.g.*, vehicles) or from static objects due to non-perfect stabilization. In this example, the ROI corresponds to a static object (building edges). Since the motion pattern for this object does not match that of a pedestrian, the classifier labels this as a non-pedestrian.

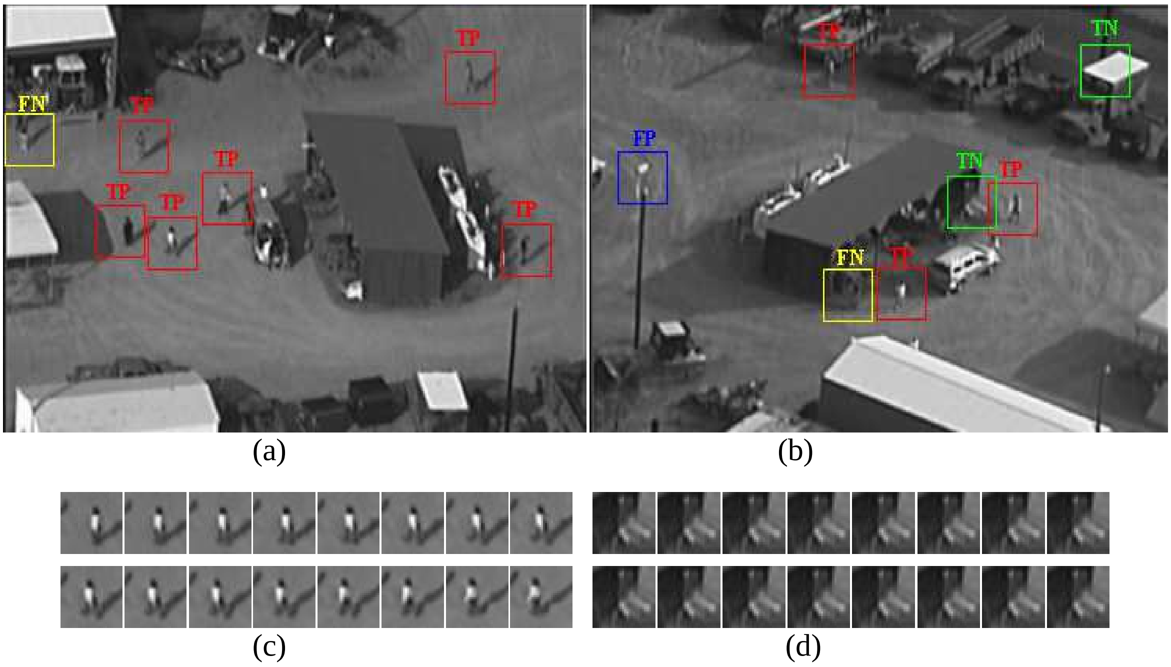
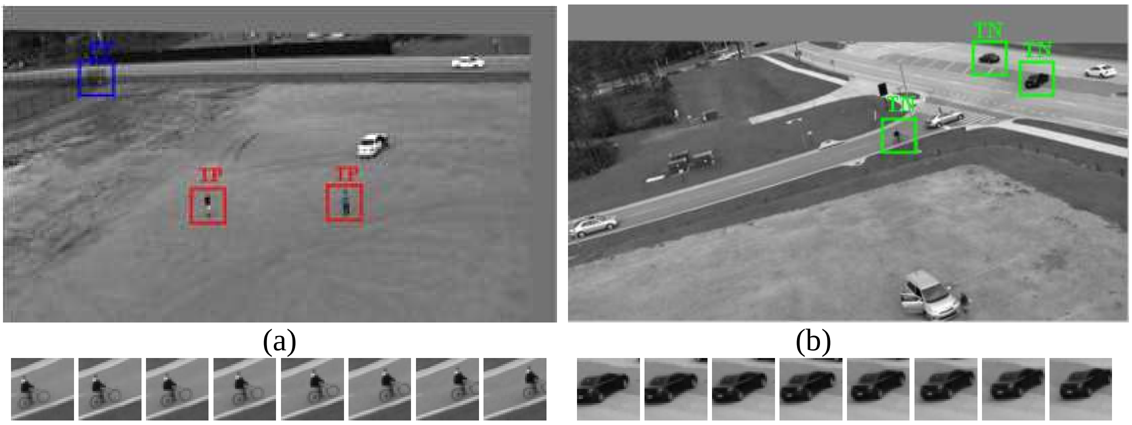


Figure 4.16: (a,b) Two frames from VIRAT dataset with detections. (c) Slices for one TP in (a). (d) Slices for the TN shown in (b).

Figure 4.17 (a) and (b) shows two frames from the UCF-2009 dataset. Examples of sequences of slices for *TN* detections are shown in Figure 4.17 (c) and (d). The *TNs* shown here resulted from scanning the classifier around false ROIs corresponding to non-pedestrian motion; (*e.g.*, a car and a bicycle). The motion patterns here do not match that of pedestrians. Therefore, the classifier labels them as non-pedestrians.

Figure 4.18 shows frames from UCF-2007 dataset with detections, and Figure 4.19 shows examples from the stationary VIRAT dataset and PETS 2001 dataset with detections.



(a) (b)

(c) (d)

Figure 4.17: Detections on two frames from UCF-2009 Dataset. (a) *FP* and *TPs*. (b) *TNs*. (c) and (d) Sequences for *TNs* shown in (b).

(a) (b)

Figure 4.18: (a) and (b) Two frames from UCF-2007 dataset with example detections.

(a) (b)

Figure 4.19: Example detections, (a) PETS2001 dataset. (b) Stationary VIRAT dataset.

We now examine in detail the behavior of the classifier on two examples from the aerial VIRAT dataset. Example 1 is a case where a pedestrian was detected as true positive, and Example 2 is a case where a pedestrian was a false negative (*i.e.*, the classifier failed to detect the pedestrian). In Figures 4.20 and 4.21, the slices are shown on a larger scale.

As discussed in Chapter 3, during the training phase, the learning process produces the optimal decision function

$$f(x) = (w^T x + b)$$

that classifies unseen data correctly. The classification (pedestrian → +1 and non-pedestrian → −1) is obtained as

$$y = sign(f(x)) = \begin{cases} +1 & f(x) > 0 \\ -1 & f(x) < 0 \end{cases}, \ \ i.e. \ \ y = sign\left(w^T x + b\right)$$

In this section, examples of classifying pedestrians using different classifiers are discussed. Each classifier is trained and tested on sequences containing a different number of slices. The examples describe the effect of adding more slices on the output of the decision function *f*.

Example 1: True Positive

Figure 4.18 shows the first example, and Table 4.3 shows the computed decision values, which is the output of the decision function when it is fed with the feature vector of this example. Each value in the *DV* column represents the decision value computed using a classifier trained and tested on sequences containing a different number of slices. For example, the decision value produced by the classifier that is trained and tested on single slice examples is -0.42756.

As shown in the *DV* column, the positivity of the decision value increases as more slices per volume are used to represent the example. The *DV* reaches its peak value when the number of slices used is between 16 and 24 slices per volume. Obviously, the pedestrian in the example is hard to recognize, but over a longer sequence, the classifier becomes able to analyze motion patterns that were learned during the training stage.

Table 4.3: Decision values vs. number of slices per volume of the example of Figure 1

Number of Slices/Volume	DV	Number of Slices/Volume	DV	Number of Slices/Volume	DV
1	-0.42756	7	-0.175	13	0.3136
2	-0.50974	8	-0.06097	14	0.3485
3	-0.4413	9	0.011	15	0.4078
4	-0.44541	10	0.1095	16	0.475923
5	-0.4178	11	0.164	24	0.491471
6	-0.2274	12	0.2413	32	0.413252

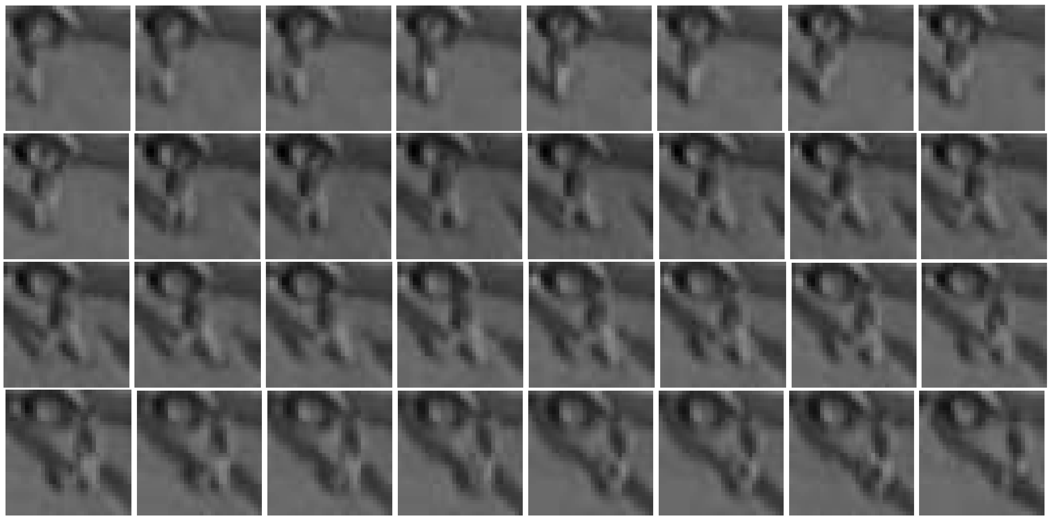


Figure 4.20 (a): Sequence of slices of Example 1. In this sequence: slice # 1 is shown at top left, and slice # 32 is shown at bottom right.

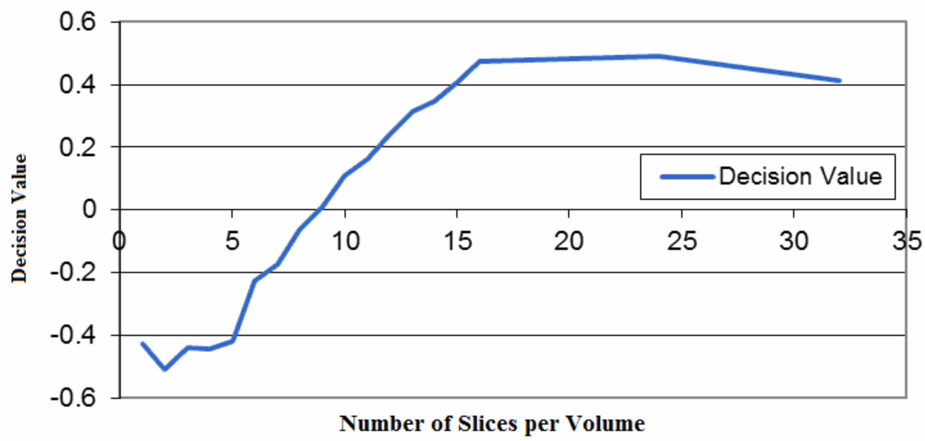


Figure 4.20 (b): The curve of computed DV of Example 1.

Example 2: False Negative

In this example (Figure 4.21), the detection was missed by all classifiers (each classifier was trained using different numbers of slices). Many factors could contribute to making this example a difficult example. In addition to the blurriness of the slices, the amount of motion appears not to be enough, so the example might not include the key frames of the walking step. Also, in slice 13, the pedestrian's shadow starts to be cast on an object in the background, which makes the background object appear to have motion. In slices 16–24, the pedestrian starts changing the direction of walking, and in slices 25–32, the pedestrian's torso is occluded by an object.

Table 4.4: Decision Values of Example 4

Number of Slices/Volume	DV	Number of Slices/Volume	DV	Number of Slices/Volume	DV
1	-1.34077	7	-0.6566	13	-0.4788
2	-0.94728	8	-0.6127	14	-0.4823
3	-0.8411	9	-0.5939	15	-0.4273
4	-0.7721	10	-0.5574	16	-0.48011
5	-0.8167	11	-0.5109	24	-0.76351
6	-0.7892	12	-0.5045	32	-0.81881

Even though the pedestrian is misclassified with both single and multiple slices, the highest negativity occurs in the single slice case (Table 4.4). The lowest negativity occurs when the number of used slices is in the range of 13–16. The cases of 24 and 32 slices have *DV*s with high negativity; this is probably due to the fact that the person changed his walking direction and started walking directly toward the camera, so the motion was not consistent over the entire sequence. Figure 4.21 (b) shows the plot of decision value with respect to the number of slices per volume.

4.6 Analysis of the Effect of Various Parameters

This section gives details about the effect of various parameter choices on detector performance. In the performance evaluation, metrics described in Section 4.1 are used. Throughout this section, results are compared to the baseline detector that has the properties described in Section 4.3, except when specified otherwise.

Figure 4.21 (a): Sequence of slices of Example 2.

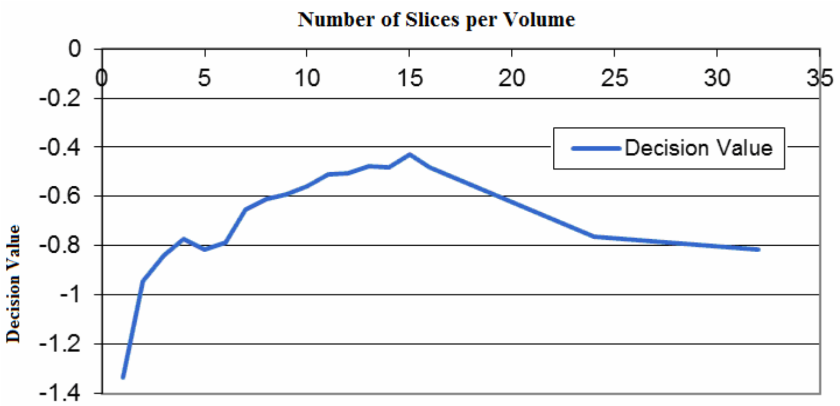


Figure 4.21 (b): The curve of computed DV of Example 3.

4.6.1 Detector Window Size

In early experimental work conducted as a proof of concept and to study the effect of different parameters on performance, training and testing data were created from a real webcam surveillance camera that is publicly available on the Internet [77]. A total of seven video sequences were recorded at different days and times with different light conditions. As training data, 400 volumes were created; half were positive examples and half were negative examples. For testing, 800 frames were used. Figure 4.22 shows a sample image shot from the webcam surveillance video, and some example sequences.

(a) (b) (c)

Figure 4.22: (a) Sample image shot from a surveillance video taken by a webcam camera (WebcamMania) mounted on a building in Eger, Hungary. (b) Two sequences of negative examples (truck and cyclist). (c) Two sequences of positive examples (pedestrians).

In this experiment, the effect of including a margin around the pedestrian in the detection window was studied. The results showed the importance of keeping a moderate margin region around pedestrians. The work began with a slice of 48×48 pixels, which kept a margin of about 14 pixels around the pedestrians (pedestrian height is 20 pixels). Then margin was reduced to only 6 pixels around the pedestrians with a slice size of 32×32 pixels (and still the same pedestrian height). This improved performance by 4%. The reason could be that the classifier might have started learning the background as part of a pedestrian slice when a wider margin around pedestrians was used. Also, narrowing the margin around pedestrians causes loss of performance since this does not allow enough area around the pedestrian's outer edge to be learned by the classifier. Figure 4.23 shows example slices of two different sizes.

4.6.2 Normalization

As discussed in Section 3.4.4, the influence of large gradient magnitudes can be reduced using the clipped L_2-norm. Following the methods of [59], normalized feature vectors were applied. The difference is that the features were normalized within each volumetric block, meaning that the sequence of blocks across the N slices is at the same place in each slice. The volumetric blocks were normalized using L_2-norm followed by clipping to limit the maximum values. Next, the features from all the blocks in all slices were concatenated into a single feature vector.

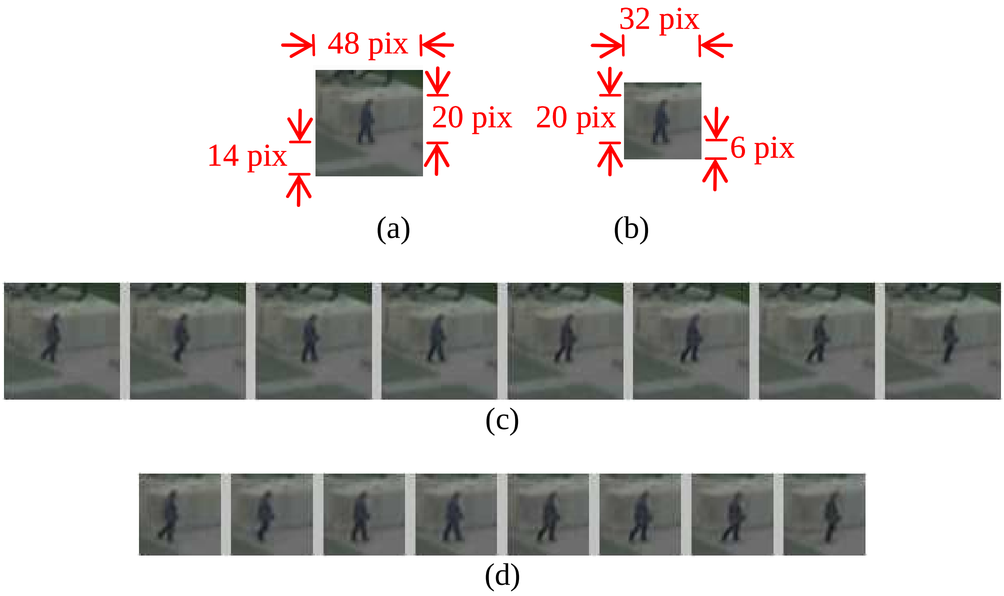


Figure 4.23: (a) 48×48 pixels slice. (b) 32×32 pixels slice. (c) Sequence of slices of size 48×48 pixels each. (d) Sequence of slices of size 32×32 pixels each.

The conducted experiments confirm that normalization improves accuracy over non-normalization and makes the features more invariant to changes in illumination and shadowing. Normalization of the volumetric blocks improved the accuracy detection by a rate of 4% over non-normalization (*e.g.*, from 74% to 78% for the aerial VIRAT database). The experiments show that normalization becomes more critical with the aerial datasets where the changes in illumination and shadowing become more rapid.

4.6.3 Block Overlapping

Figure 4.24 shows the idea of block overlapping. In a block overlapping scheme, each non-border cell becomes part of more than one block. Each non-border contributes more than one time in the final feature vector with different normalization. Overlapping allows encoding the same information multiple times in different ways. Block overlapping may increase performance but at the same time increases the size of the feature vector.

Detector performance was evaluated with block overlapping and non-overlapping. The use of overlapping blocks in the descriptor improves performance by around 2%. Overlapping the blocks allows a feature to contribute to the decision more than one time, whereas if there are no overlapping blocks, a cell is coded only once in the final descriptor. The disadvantage of overlapping the blocks is that if blocks are overlapped by (for example) half of the block size, the feature vector dimension is almost doubled.

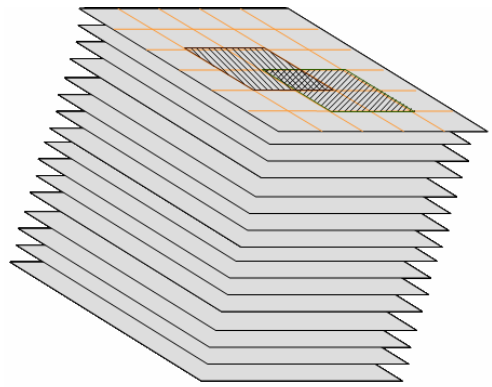


Figure 4.24: Block overlapping: Sequence of 16 slices. Dashed areas are two blocks. Each non-border cell contributes more than one time as it becomes part of more than one block.

4.6.4 The Effect of Cell Size

Figure 4.25 plots *DR* with respect to cell size, for the UCF-2009 dataset. In this dataset, as well as all the others, we find that a 4×4 pixel cell is the optimal cell size and performs best with a block size of 2×2 cells.

Figure 4.26 (b) shows an example pedestrian height of 20 pixels. The size of a person's head, forearm, upper leg (thigh), and lower leg (shin and foot) appear to be approximately 4×4 pixels, which may allow cells of 4×4 pixels to capture the shape and motion of these parts.

Figure 4.26 (c) and (d) gives some insight into what cues the detector uses to make its discrimination decision. Here the coefficients of the trained SVM classifier are used as a way of measuring how much weight each cell has in the final classification decision. Figure 4.26 (c) shows the weights corresponding to each element of the feature vector; that is, the value of the elements of *w's* in the classifier decision function equation $f(x) = w^T x + b$. In each cell, there are 3×3 weights corresponding to the 9 gradient directions. The weights are shown for the central slice in the volume and the brighter pixels represent the higher weights. The figure suggests that the contours of a pedestrian's head and shoulders for the upper body part and lower legs and the region where the feet touch the ground have the highest weights, which represent the main cues for detection.

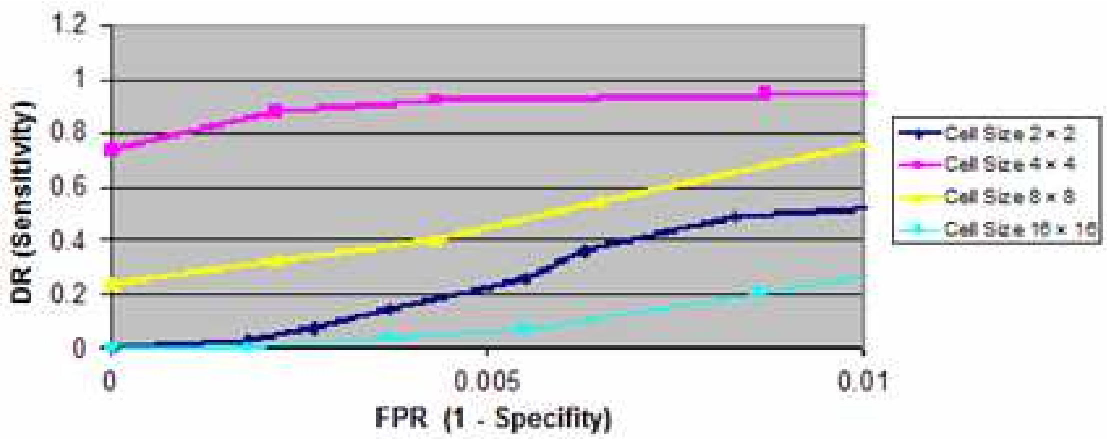
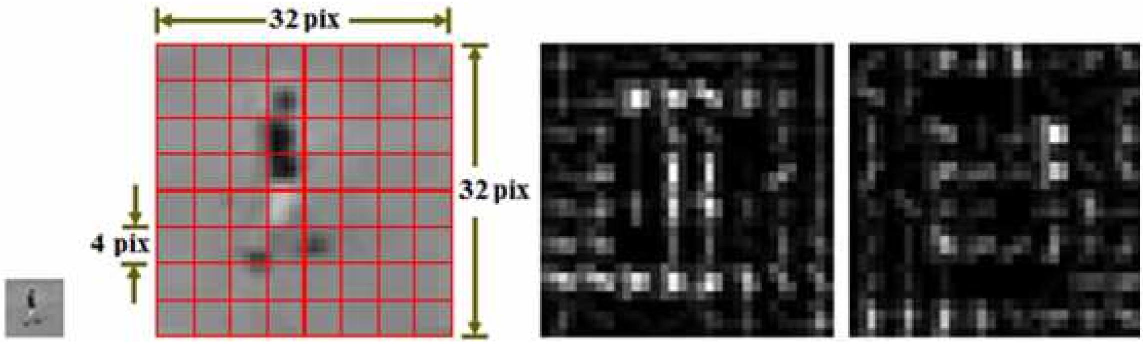


Figure 4.25: Effect of cell size on detection rate



(a) (b) (c) (d)

Figure 4.26: (a) 32×32 pixels slice in its actual size. (b) Same slice enlarged 5 times. Cells of size 4×4 pixels appear to match key parts of the pedestrian's body. (c) Central part of SVM positive weights. Each pixel shows one SVM weight (w's in *DF* equation $f(x) = w^T \cdot x + b$). The brighter pixels represent higher weights. (d) Same as in part (c) but for negative weights.

4.6.5 Dimensionality Reduction

In section 3.4.3 we discussed the use of PCA to reduce feature vector dimensionality. The eigenvalues indicated that about half of the principal components can capture most of the essential information. In particular, using the top 50% of the principal components yields a cumulative contribution of 88% to the variance, on the stationary VIRAT dataset.

Here, we present experimental results on the effect of using different numbers of principal components on pedestrian detection rate. Figure 4.27 shows 6 ROC curves of 6 classifiers each trained with different number of principal components on the stationary VIRAT dataset. The classifier using half the principal components (the pink curve) gives a detection rate about the same as using all of the principal components (*i.e.*, using cumulative contribution of 100%, the dark blue curve). As a result, we can reduce the number of features to half and keep a good performance.

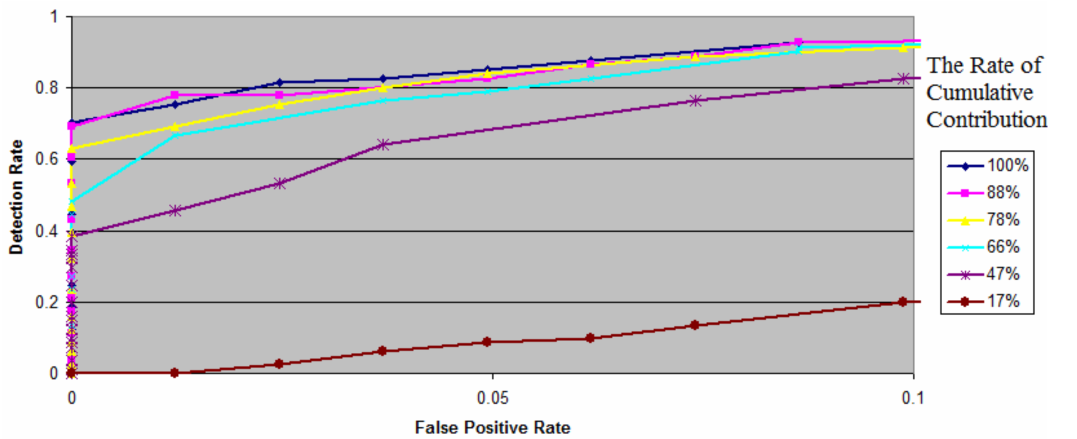


Figure 4.27: ROC of 6 different classifiers trained with different number of principle components.

4.7 Frame Randomization

To confirm our hypothesis that the classifier learns the characteristic motion of walking pedestrians, we randomized the order of frames. The expectation is that giving the classifier temporally incoherent data should reduce the performance.

We followed the same procedure as before to extract ROIs and form spatiotemporal volumes. However, this time we randomized the slices in both the training and testing volumes.

Figure 4.28 shows detection rates obtained from multiple tests on the two datasets. The results show that the use of randomized frame sequences degraded detection rates by an average of 8% in the VIRAT dataset experiments and by an average of 12% in the UCF-2009 dataset experiments at FPR of 4×10^{-3}. For example, for the UCF dataset, when the sequences were used in their normal coherent order, we obtained a detection rate of 92%. In one of the tests in which randomized sequences were used, at the same FPR, the detection rate degraded to 80%. These experiments indicate that the classifier is indeed learning the characteristic motion of walking pedestrians, and that temporal sequencing of frames is important for pedestrian detection.

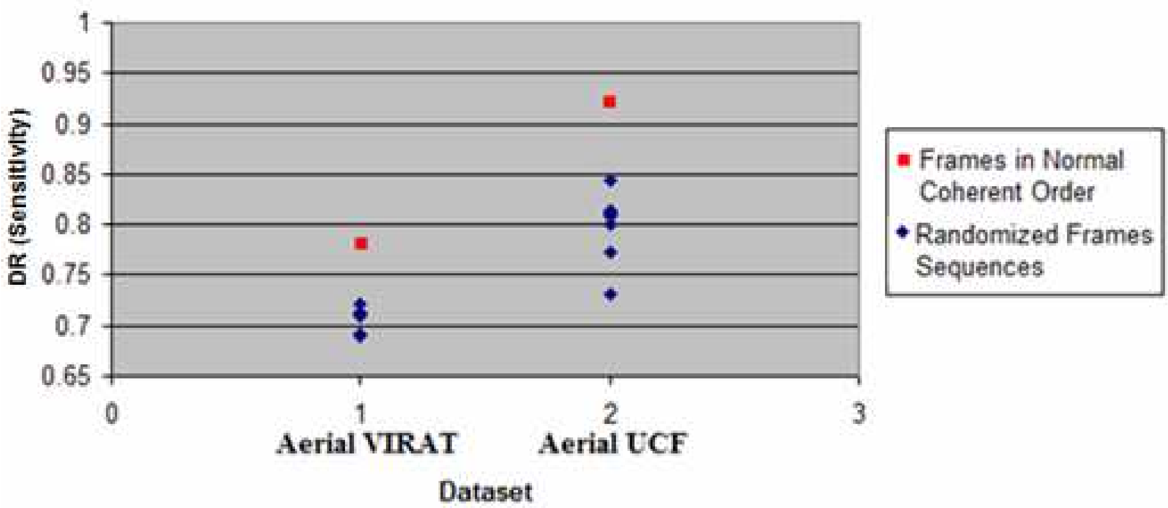


Figure 4.28: The effect of randomizing the order of frames on classification performance. All *DRs* are measured at FPR of 4×10^{-3}.

Chapter 5: Conclusions and Future Work

This chapter summarizes the thesis findings and suggests directions for further research. Section 5.1 provides an overall summary, which is detailed further in Section 5.2 along with key contributions. Section 5.3 discusses limitations, and Section 5.4 presents some of the possible improvements, extensions, and directions that can be explored in a future work.

5.1 Summary

This thesis has addressed the problem of detecting pedestrians in low-resolution videos taken by stationary and aerial cameras. The targeted goal of detecting pedestrians as small as 20 pixels in height was achieved and confirmed using standard datasets. The key novel idea was to take the single-frame HOG-based detector and extend it to multiple frames. Namely, we extract HOG features from a sequence of subimages centered at the point in the image where the detector is being applied. Volumes of features are normalized and then a trained classifier is used to classify the volumetric sequence as a pedestrian or non-pedestrian. We compensate for the loss of information due to low resolution by using a sequence of images for detection.

We restrict the application of the detector to regions surrounding potential moving objects. These regions of interest (ROIs) are found using background subtraction, followed by morphological operations to eliminate small regions, and then connected component labeling. In the case of moving cameras, we first register frames to a local reference frame by matching interest points and then transforming the images using a homography transform.

Good detection accuracy was obtained on several standard datasets that contained low resolution pedestrians (PETS 2001, Stationary VIRAT, UCF-2009, UCF-2007, and aerial VIRAT). Our detector achieved an improvement of 20% or more over the Dalal algorithm, which is a standard pedestrian detection algorithm using single frames [2]. We also obtained a better detection rate than the Jones and Snow algorithm [33], which was the best published detector on low resolution images that we could find.

We analyzed the effect of several key algorithm parameters. One of the most important parameters is the number of slices used by the detector. We found that using multiple images for detection dramatically improves the results. The improvement in detection rate increases with the number of slices until a total of 16 slices is reached, which corresponds to about half a second of walking. Beyond 16 slices, performance levels off. The use of multiple frames has more influence on improving detection rates in aerial videos than in stationary videos.

The effect of other parameters on the system performance was also studied. We found that:

1. A smaller cell size outperforms the use of a large cell size. In low-resolution images, a small cell size is better able to capture the shape and motion of pedestrian's body parts (*e.g.*, person's head, forearm, upper leg, and lower leg). We found that 4×4 pixel cells were optimal.

2. Normalization improves accuracy and makes features more invariant to changes in illumination and shadowing. The experiments show that normalization becomes more critical with aerial datasets where illumination and shadows change rapidly.

3. Detector performance was also evaluated with different block overlapping schemes. The experiments show that the use of overlapping blocks in the descriptor improves performance slightly. However, the block overlap at half of the block size almost doubles the feature vector dimension.

4. The experimental work also shows that the use of nonlinear kernel (a radial basis function kernel) gives slightly more accurate results than using linear kernel. In our baseline detector we used linear kernel for simplicity and speed.

5.2 Contributions

The main contribution of this thesis is the development of a new pedestrian detector that is able to detect pedestrians in surveillance videos at lower resolutions than has been reported in previous work. The detector is novel in that it operates on a short sequence of frames. Although some previous algorithms have used image sequences, our algorithm is unique in that it uses longer image sequences than that used in previous work.

Even though we use longer image sequences than previous work, the sequences are still only about 0.5 to 1.0 seconds long. Our algorithm is a detector, not a tracker. A tracker would use measurements over much longer sequences, and assemble multiple detections into a single track file. The new pedestrian detection system significantly outperforms several existing state-of-the-art pedestrian detection systems (as detailed in Chapter 4).

We conjectured that the motion of a person walking is distinctive, and that the classifier learns to recognize the temporal sequence of feature vectors within the volume. We confirmed this in experiments where we trained and tested the classifier on randomized sequences. In these experiments the algorithm, training procedures, and testing procedures were identical, except that we randomized the order of slices within volumes. The results showed that giving the classifier temporally incoherent data significantly degraded performance.

Another contribution is that our detector is applicable to low resolution aerial videos. Because the detector needs only a short sequence of frames to perform detection, it is applicable to situations where the camera is moving rapidly and does not dwell on the same portion of the scene for very long. Ours is one of the first published methods to show results on several challenging low resolution aerial datasets.

5.3 Limitations

Although the proposed approach gives good results and outperforms state-of-the-art approaches, especially over traditional single-frame based algorithms, there are still some limitations.

1. The algorithm detects only upright walking pedestrians. Subjects have to be walking, not stationary or running. One approach to detecting different motion speeds would be to train a classifier with multiple labels (*e.g.* classifier of [78]); such as slow walking, fast walking, stationary, and running people.

2. The algorithm cannot detect pedestrians that are partially occluded.

3. In the datasets used in this work, videos were captured at an oblique angle. The view angle needs to be known, so that we train on images at approximately the same angle as the test images. We did not test other views such as a straight down view, but a straight down view would probably not work. In a straight down view the characteristic walking motion of the pedestrian’s body may not be visible.

4. The algorithm gives a good discrimination for pedestrians of 20 pixels height or more (the targeted goal), but lowering the resolution to very low-resolution situations (Figure 5.1) degrades the algorithm's performance. In addition to blurriness and an extreme lack of

information, one other possible reason is that several detector parameters need to be tuned for this scenario. In general, a very low-resolution scenario is still challenging and unexplored area in the field. Nevertheless, this algorithm is still useful and gives results better than the single fame method in these scenarios.

Figure 5.1: Very low-resolution examples of pedestrians of height < 20 pixels.

5.4 Future Work

Future work should examine the algorithm robustness on more challenging real world videos. We have begun testing the algorithm on videos taken from a rapidly moving quadrotor UAV [6], as shown in Figure 1.3. These videos have lower quality than the standard datasets that we tested, including greater lens distortion and motion blur. Some preliminary results are shown in Figure 5.2. The results of these experiments show the robustness of our proposed algorithm even on these challenging videos.

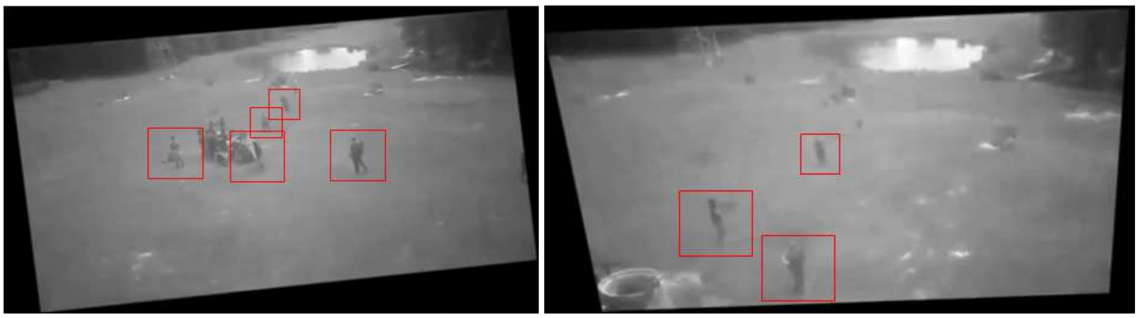


Figure 5.2: Example from the results of detecting pedestrians in challenging UAV videos. Detections are shown at different pyramid levels.

Some of the possible improvements, extensions, and directions that can be explored in the future include the following:

1. To improve detection performance, a multiclass SVM classifier can be used for different walking directions. The idea of using different detectors for different directions could make the classifier more precise in making the classification decision. For example,

eight detectors (or more) could be trained with a specific detector for each direction to cover the full range of 360^o of directions: north, south, east, west, northeast, southeast, northwest, and southwest (Figure 5.3). In this case, for each detector, the positive examples would only include pedestrians walking in a specific direction. The use of a muticlass classifier could also improve the classification of pedestrians with different speeds, ranging from stationary pedestrians to fast-walking or running pedestrians.

Figure 5.3: Eight detectors could be trained with a specific detector for each walking direction to cover the full range of 360o of directions.

2. To reduce the effect of changing illumination or background texture of visual videos, thermal (IR) videos could be used. Figure 5.4 shows an example of visual versus thermal image shots of the same scene [79]. Thermal videos also allow detection to take place at night.

Figure 5.4: shows an example of visual versus thermal image (OSU Color-Thermal Database).

3. The detector could be incorporated into any standard tracking algorithm if a longer sequence of images of the same scene is available. The improved accuracy of the detector should make it possible to estimate more reliable tracks using a shorter sequence of images. The plan for future work is to integrate the detector into a standard tracker. Even in aerial

video from rapidly moving cameras, a person is often in the field of view for multiple seconds. Therefore, multiple detections could be associated into a single track.

4. Although the algorithm does not require metadata, if it is available, it can be used to improve the system performance. Metadata could include camera altitude, angle, pitch, zoom, *etc.* and can be used to constrain the expected size of people.

5. For the scenarios where a wide range of pedestrian scales is expected, typically both high and low resolution candidate windows are resampled to a common size. One way to use all the available information (without downsampling the high resolution pedestrians) is to train different models for different resolutions to improve the performance since the detector will have access to the full information available at each window size. This technique has the disadvantage of increasing the computational cost at both the detection and training phases.

6. The prototype system was written in Matlab, which is good for a scientific environment to perform analysis. However, it is not very fast when compared to C or C++. In principle, the proposed approach should be able to run in close to real time since the operations it uses are similar to other real-time systems based on HOG-like features and SVM classifiers.

References

[1] Hu, Weiming, Tieniu Tan, Liang Wang, and Steve Maybank. 2004. "A survey on visual surveillance of object motion and behaviors." *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 34, no. 3: 334-352.

[2] Dalal, Navneet, and Bill Triggs. 2005 "Histograms of oriented gradients for human detection. 2005." In *Computer Vision and Pattern Recognition. IEEE Computer Society Conference, vol. 1*, pp. 886-893.

[3] Felzenszwalb, Pedro, David McAllester, and Deva Ramanan. 2008. "A discriminatively trained, multiscale, deformable part model." In *Computer Vision and Pattern Recognition. IEEE Conference*, pp. 1-8.

[4] Cucchiara, Rita, Costantino Grana, Massimo Piccardi, and Andrea Prati. 2003. "Detecting moving objects, ghosts, and shadows in video streams." Pattern Analysis and Machine Intelligence, IEEE Transactions on 25, no. 10: 1337-1342.

[5] Basharat, Arslan, Matt Turek, Yiliang Xu, Chuck Atkins, David Stoup, Keith Fieldhouse, Paul Tunison, and Anthony Hoogs. 2014."Real-time multi-target tracking at 210 megapixels/second in wide area motion imagery." In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pp. 839-846.

[6] “Airsoft UAV footage”, downloaded from https://www.youtube.com/watch?v=rppbsvUSpxY, November 2014.

[7] Oh, Sangmin, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee et al. 2011."A large-scale benchmark dataset for event recognition in surveillance video." In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 3153-3160.

[8] Johansson, Gunnar. 1973. "Visual perception of biological motion and a model for its analysis." *Attention, Perception, & Psychophysics 14,* no. 2: 201-211.

[9] Sager, Hisham, and William Hoff. "Pedestrian detection in low resolution videos. 2014." In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pp. 668-673.

[10] Yan, Junjie, Xucong Zhang, Zhen Lei, Shengcai Liao, and Stan Z. Li. 2013. "Robust multi-resolution pedestrian detection in traffic scenes." In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 3033-3040.

[11] Dollár, Piotr, Christian Wojek, Bernt Schiele, and Pietro Perona. 2009. "Pedestrian detection: A benchmark." In *Computer Vision and Pattern Recognition, 2009. IEEE Conference*, pp. 304-311.

[12] Enzweiler, Markus, and Dariu M. Gavrila. 2009. "Monocular pedestrian detection: Survey and experiments." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 31, no. 12: 2179-2195.

[13] Dollár, Piotr, Christian Wojek, Bernt Schiele, and Pietro Perona. 2012. "Pedestrian detection: An evaluation of the state of the art." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34, no. 4: 743-761.

[14] R. Benenson, M. Omran, J. Hosang, B. Schiele. 2014. "Ten years of pedestrian detection, what have we learned?", *ECCV 2014, CVRSUAD workshop*.

[15] Gavrila, Dariu M. 1999. "The visual analysis of human movement: A survey." *Computer vision and image understanding* 73, no. 1: 82-98.

[16] D. Park, D. Ramanan, and C. Fowlkes.2010. "Multiresolution Models for Object Detection." *Proc. European Conf. Computer Vision*.

[17] Liu, Yun-Fu, Jing-Ming Guo, and Che-Hao Chang. 2014. "Low resolution pedestrian detection using light robust features and hierarchical system." *Pattern Recognition 47*, no. 4: 1616-1625.

[18] Cortes, Corinna, and Vladimir Vapnik. 1995. "Support-vector networks." *Machine learning 20*, no. 3: 273-297.

[19] Freund, Yoav, Robert Schapire, and N. Abe. 1999. "A short introduction to boosting." *Journal-Japanese Society For Artificial Intelligence* 14, no. 771-780: 1612.

[20] Dalal, Navneet, Bill Triggs, and Cordelia Schmid. 2006. "Human detection using oriented histograms of flow and appearance." *Computer Vision–ECCV*: 428-441.

[21] Efros, Alexei A., Alexander C. Berg, Greg Mori, and Jitendra Malik. 2003. "Recognizing action at a distance." In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 726-733. IEEE.

[22] Cutler, Ross, and Larry S. Davis. 2000. "Robust real-time periodic motion detection, analysis, and applications." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22, no. 8: 781-796.

[23] Narayanaswami, Ranga, Anastasia Tyurina, David Diel, Raman K. Mehra, and Janice M. Chinn. 2011. "Discrimination and tracking of dismounts using low-resolution aerial video sequences." In *SPIE Optical Engineering+ Applications*, pp. 81370H-81370H. International Society for Optics and Photonics.

[24] Bhamidipati, Sai Lakshmi, Sai Sudha Mindagudla, Harsha Vardhan Devalla, Hima Sagar Goodi, and Hemanth Nag. 2014. "Analysis of Different Discrete Wavelet Transform Basis

Functions in Speech Signal Compression." *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*. Volume 4, Issue 1, Ver. PP 34-38.

[25] Sun, Jie, James M. Rehg, and Aaron Bobick. 2004. "Automatic cascade training with perturbation bias." In *Computer Vision and Pattern Recognition, 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2, pp. II-276. IEEE.

[26] Schapire, Robert E. "The boosting approach to machine learning: An overview. 2003." *LECTURE NOTES IN STATISTICS-NEW YORK-SPRINGER VERLAG*: 149-172.

[27] Moeslund, Thomas B., Adrian Hilton, and Volker Krüger. 2006. "A survey of advances in vision-based human motion capture and analysis." *Computer vision and image understanding* 104, no. 2: 90-126.

[28] Papageorgiou, Constantine, and Tomaso Poggio. 2000. "A trainable system for object detection." *International Journal of Computer Vision* 38, no. 1: 15-33.

[29] Viola, Paul, and Michael Jones. 2001. "Rapid object detection using a boosted cascade of simple features." *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1.

[30] Viola, Paul, and Michael J. Jones. 2004. "Robust real-time face detection." *International journal of computer vision* 57, no. 2: 137-154.

[31] Viola, Paul, Michael J. Jones, and Daniel Snow. 2005. "Detecting pedestrians using patterns of motion and appearance." *International Journal of Computer Vision* 63, no. 2: 153-161.

[32] Cooper, Helen, and Richard Bowden. 2009. "Sign Language Recognition: Working with Limited Corpora." *Universal Access in Human-Computer Interaction. Applications and Services*: 472-481.

[33] Jones, Michael J., and Daniel Snow.2008. "Pedestrian detection using boosted features over many frames." *In Pattern Recognition.* 19th International Conference, pp. 1-4. IEEE.

[34] Szeliski, Richard. 2011. *Computer vision: algorithms and applications*. Springer.

[35] Dollár, Piotr, V. Rabaud, G. Cottrell, and Serge Belongie. 2005. "Behavior recognition via sparse spatio-temporal features." In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pp. 65-72.

[36] Oreifej, Omar, *et al.*. 2010. "Human identity recognition in aerial images." *In Computer Vision and Pattern Recognition, 2010 IEEE Conference on*, pp. 709-716.

[37] Gaszczak, Anna, Toby P. Breckon, and Jiwan Han. "Real-time people and vehicle detection from UAV imagery. 2011." *In Proc. SPIE Conference Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques*, volume 7878, doi: 10.1117/12.876663.

[38] Reilly, Vladimir, Berkan Solmaz, and Mubarak Shah. 2010. "Geometric constraints for human detection in aerial imagery." *Computer Vision–ECCV 2010* : 252-265.

[39] Daubechies, Ingrid. 1998. "Orthonormal bases of compactly supported wavelets." *Communications on pure and applied mathematics* 41, no. 7: 909-996.

[40] Bar-Shalom, Yaakov, and Xiao-Rong Li. 1993. "Estimation and tracking- Principles, techniques, and software." *Norwood, MA: Artech House, Inc*.

[41] Blackman, Samuel S. 2004. "Multiple hypothesis tracking for multiple target tracking." *Aerospace and Electronic Systems Magazine*, IEEE 19, no. 1: 5-18.

[42] Yilmaz, Alper, Omar Javed, and Mubarak Shah. 2006. "Object tracking: A survey." *Acm computing surveys (CSUR) 38*, no. 4: 13.

[43] Wu, Yi, Jongwoo Lim, and Ming-Hsuan Yang. 2013. "Online object tracking: A benchmark." *Computer Vision and Pattern Recognition (CVPR)*, 2013 IEEE Conference on.

[44] Challa, Sudha. 2011. *Fundamentals of object tracking*. Cambridge University Press.

[45] Ali, Saad, and Mubarak Shah. 2006."COCOA: tracking in aerial imagery." *In Defense and Security Symposium*, pp. 62090D-62090D. International Society for Optics and Photonics.

[46] Singh, Vivek Kumar, Bo Wu, and Ramakant Nevatia. 2008. "Pedestrian tracking by associating tracklets using detection residuals." In *Motion and video Computing, 2008. WMVC 2008. IEEE Workshop on*, pp. 1-8.

[47] Wu, Bo, and Ram Nevatia. 2007. "Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors." *International Journal of Computer Vision 75*, no. 2: 247-266.

[48] O'malley, R., M. Glavin, and E. Jones. 2011. "Vision-based detection and tracking of vehicles to the rear with perspective correction in low-light conditions." *IET Intelligent Transport Systems 5*, no. 1: 1-10.

[49] O'Malley, Ronan, Edward Jones, and Martin Glavin. 2010. "Rear-lamp vehicle detection and tracking in low-exposure color video for night conditions." *Intelligent Transportation Systems, IEEE Transactions on 11*, no. 2: 453-462.

[50] Harris, Chris, and Mike Stephens. 1988. "A combined corner and edge detector." In *Alvey vision conference*, vol. 15, p. 50.

[51] Fischler, Martin A., and Robert C. Bolles. 1981. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM 24*, no. 6: 381-395.

[52] Piccardi, Massimo. "Background subtraction techniques: a review." *In Systems, man and cybernetics, 2004 IEEE international conference on*, vol. 4, pp. 3099-3104. IEEE, 2004.

[53] McIvor, Alan M. "Background subtraction techniques." *Proc. of Image and Vision Computing 1*, no. 3 (2000): 155-163.

[54] Brutzer, Sebastian, Benjamin Hoferlin, and Gunther Heidemann. "Evaluation of background subtraction techniques for video surveillance." *In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 1937-1944. IEEE, 2011.

[55] Jain, Ramesh, and H-H. Nagel. 1979. "On the analysis of accumulative difference pictures from image sequences of real world scenes." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 2: 206-214.

[56] Stauffer, Chris, and W. Eric L. Grimson. 1999. "Adaptive background mixture models for real-time tracking." *In Computer Vision and Pattern Recognition. IEEE Computer Society Conference on*, vol. 2.

[57] Graf, Arnulf BA, and Silvio Borer. 2001. "Normalization in support vector machines." In *Pattern Recognition*, pp. 277-282. Springer Berlin Heidelberg.

[58] Tax, David MJ, and Robert PW Duin. 2000. *Feature Scaling in Support Vector Data Descriptions*. Technical report, American Association for Artificial Intelligence.

[59] Lowe, David G. 2004. "Distinctive image features from scale-invariant key points." *International journal of computer vision 60*, no. 2: 91-110.

[60] Ke, Yan, and Rahul Sukthankar. 2004. "PCA-SIFT: A more distinctive representation for local image descriptors." In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2, pp. II-506.

[61] Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel *et al*. 2011. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12: 2825-2830.

[62] Bishop, Christopher M. 2006. *Pattern recognition and machine learning*. Vol. 4, no. 4. New York.

[63] Burges, Christopher JC. 1998. "A tutorial on support vector machines for pattern recognition." *Data mining and knowledge discovery 2*, no. 2: 121-167.

[64] Bennett, K. P. 1992. Robust linear programming discrimination of two linearly separable sets. *Optimization Methods and Software 1*, no. 1:.

[65] OSU-SVM Toolbox for MATLAB. Last Update: 2009-07-17. http://sourceforge.net/projects/svm.

[66] Hastie, Trevor, Robert Tibshirani, Jerome Friedman. 2009. *The elements of statistical learning*. Vol. 2, no. 1. New York: Springer.

[67] Dee, Hannah M., and Sergio A. Velastin. 2008. "How close are we to solving the problem of automated visual surveillance?" Machine Vision and Applications19, no. 5-6: 329-343.

[68] Neubeck, Alexander, and Luc Van Gool. 2006. "Efficient non-maximum suppression." In *Pattern Recognition. ICPR 2006. 18th International Conference on*, vol. 3, pp. 850-855.

[69] Sokolova, Marina, Nathalie Japkowicz, and Stan Szpakowicz. 2006. "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation." In *AI 2006: Advances in Artificial Intelligence*, pp. 1015-1021. Springer Berlin Heidelberg.

[70] Provost, Foster J., Tom Fawcett, and Ron Kohavi. 1998. "The case against accuracy estimation for comparing induction algorithms." In *ICML*, vol. 98, pp. 445-453.

[71] Green, David Marvin, and John A. Swets. 1996. *Signal detection theory and psychophysics*. Vol. 1. New York: Wiley.

[72] Egan, J.P., 1975. "Signal Detection Theory and ROC Analysis." *Series in Cognition and Perception. Academic Press*, New York.

[73] Swets, John A., Robyn M. Dawes, and John Monahan. 2000. "Better DECISIONS through." *Scientific American*: 283, 82–87.

[74] PET dataset. http://www.cvg.cs.rdg.ac.uk/pets2001/ pets2001-dataset.html.

[75] Vu, Phong V., and Damon M. Chandler. 2012. "A fast wavelet-based algorithm for global and local image sharpness estimation." *Signal Processing Letters, IEEE19*, no. 7: 423-426.

[76] Kumar, Jayant, Francine Chen, and David Doermann. 2012. "Sharpness estimation for document and scene images." In *Pattern Recognition (ICPR), 2012 21st International Conference on,* pp. 3292-3295. IEEE.

[77] Webcams Mania. http://www.webcamsmania.com/webcam/egerdoboacuteteacuter.

[78] Harirchi, Farshad, *et al*. 2010. "Two-level algorithm for MCs detection in mammograms using Diverse-Adaboost-SVM." In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 269-272. IEEE.

[79] Davis, James W., and Vinay Sharma. 2007. "Background-subtraction using contour-based fusion of thermal and visible imagery." *Computer Vision and Image Understanding* 106, no. 2: 162-182.