

Pedestrian Detection using Boosted Features over Many Frames

Michael J. Jones and Daniel Snow
Mitsubishi Electric Research Labs
201 Broadway 8th FL, Cambridge, MA 02139
mjones@merl.com and snow@merl.com

Abstract

A scanning window type pedestrian detector is presented that uses both appearance and motion information to find walking people in surveillance video. We extend the work of Viola, Jones and Snow [10] to use many more frames as input to the detector thus allowing a much more detailed analysis of motion. The resulting detector is about an order of magnitude more accurate than the detector of Viola, Jones and Snow. It is also computationally efficient, processing frames at the rate of 5 Hz on a 3 GHz Pentium processor.

1. Introduction

This paper extends the work of Viola, Jones and Snow [10] on detecting pedestrians in video using both appearance and motion information. The goal is to develop an algorithm with enough speed and accuracy to be practical in real outdoor surveillance scenarios. In typical outdoor surveillance scenarios, the camera is static and the pedestrians are often very small (around 20 pixels tall). Although the pedestrian detector described in [10] achieved state-of-the-art accuracy with a detection rate of 90% and a false positive rate of 10^{-5} per image window, this false positive rate is still too high to be practical for real applications. The goal here is to reduce the false positive rate by at least an order of magnitude while maintaining the high detection rate.

The key to achieving this is to use more frames as input to the pedestrian detector. Instead of the minimal two frames used in [10], we use ten frames which allows much more motion information to be analyzed. The other major difference is that we divide the detector into eight separate direction-specific detectors (north, south, east, etc). This is analogous to the state-of-the-art in multi-view face detection in which different face poses are ultimately detected by separate detectors [4].

The basic framework uses AdaBoost [7] to select

a set of features that separates pedestrians from non-pedestrians for each direction class. The input to the classifier is a set of image windows from ten consecutive frames of a video sequence. An image window is simply a rectangular area of an image. The feature set consists of Haar-like features that act either on a single frame (appearance features), or on the difference between two frames (motion features). These features will be described in more detail later. In addition we use a soft cascade [1] to make the resulting detector very efficient to compute. Soft cascades were chosen over standard cascades because of the ease in trading off speed and accuracy.

2. Related Work

Much of the recent work in person or pedestrian detection has focused on detecting people at least 100 pixels tall in static images ([2], [11], [8], [5], [6]). Our approach is one of the few that uses motion information in addition to appearance information but does not rely on tracking. Dalal, Triggs and Schmid [3] also use motion information, but use a very different approach that relies on optical flow to represent motion. The need to compute optical flow makes it more computationally expensive than our approach. Their use of histograms of oriented gradient features breaks down when detecting very small pedestrians as we do here.

The approach taken in this paper is distinguished from previous work by various attributes. It analyzes a moderate number of frames at once (10 frames), it detects very small pedestrians, has a much lower false positive rate than static detectors (about 1 in a million false positives per window) and is fast (about 5 Hz for 360x240 pixel images on a P4 3 GHz machine).

3. Input Representation

The input representation used in this work is one of its unique aspects. Ten consecutive frames from a 15 fps

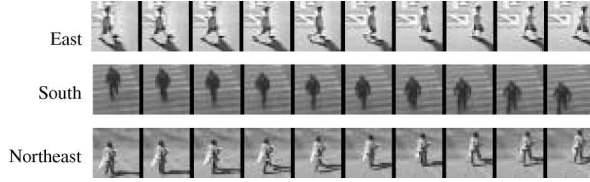


Figure 1. Positive examples for three directions of pedestrian motion. Each row of 10 windows comprises one example.

camera are used as input to the pedestrian detector. This allows a modest amount of motion (about one walking step depending on walking speed) to be analyzed by the classifier. For the low resolution pedestrians we are trying to detect, motion information is more informative than appearance and thus more frames should lead to greater accuracy. We have chosen a base input window size of 24 pixels wide by 26 pixels high. The window has to be large enough to allow a pedestrian to be visible within it both in frame 1 and in frame 10. The typical pedestrian height in our training examples is 20 pixels. An *example* (used for training a classifier) consists of 10 image windows from 10 consecutive frames. For some typical positive examples in the east, south and northeast directions see figure 1.

At this resolution, a pedestrian is roughly a person shaped blob usually moving in an approximately straight line with a walking motion in the leg region and a mostly translational motion in the torso region (arms are usually not very visible). Very few if any non-pedestrian windows in a video meet this description. This representation also implies that it will be advantageous to split the detector into different classes based on direction of movement. We found experimentally that we could cover the full 360° range of directions using 8 different detectors. We combine the 8 detectors in the simplest possible way, by running all detectors on every image window and averaging any overlapping detections. More sophisticated combination techniques could lead to further improvements.

4. Features

Our pedestrian detector classifies gray scale image windows based on the values of simple features. The features are selected from a large pool of possible features using Real AdaBoost [7]. The learning procedure is described in the next section. First we will describe the types of features that make up the feature pool.

A feature in this work is a simple binary classifier.

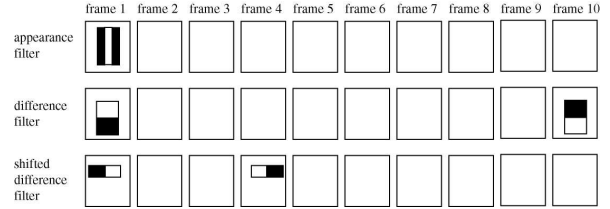


Figure 2. Examples of each of the three different types of filters.

It consists of a *filter* that takes image windows from n consecutive frames as input, a threshold and a positive and negative vote. Thus, a feature, h_i , is defined as

$$h_i(x_1, x_2, \dots, x_n) = \begin{cases} \alpha & \text{if } f_i(x_1, x_2, \dots, x_n) > \theta_i \\ \beta & \text{otherwise} \end{cases} \quad (1)$$

where x_j is an image window from frame j and x_k is the same image window from frame k , $f_i(\cdot)$ is a filter, θ_i is a threshold, and α and β are real valued votes. In the experiments reported here, $n = 10$.

There are three basic types of filters. The first type is an appearance filter. It consists of a Haar-like filter that acts on a window from a single input frame. We use the same Haar-like filters as used in [9]. The second type of filter is a difference filter which computes the absolute value of the difference between a Haar-like filter acting on image windows in two different input frames. This allows various patterns of motion to be detected. The third type of filter is a shifted difference filter which computes the absolute value of the difference between a Haar-like filter acting on a window in one input frame and a shifted version of the same filter acting on the same window in a second input frame. This allows motions in a particular direction (the direction of the shift) to be detected. Although a filter takes windows from n frames as input, any particular filter only uses 1 or 2 of those windows to compute its value. The three basic types of filters are illustrated in figure 2.

The main reason for using these types of simple features is that they are very fast to compute. The integral image representation [9] can be used to make computation of such filters extremely efficient.

5. Learning Framework

Instead of building a cascade of classifiers as in [10], we use a soft cascade [1]. A soft cascade is a single strong classifier (linear combination of weak classifiers) in which evaluation of the classifier can terminate early

if the cumulative sum of weak classifier outputs falls below one of the rejection thresholds associated with each weak classifier. The advantage of this approach over a cascade is that it is easier to trade off between speed and accuracy by simply changing the rejection thresholds. The rejection thresholds are set after AdaBoost training and can be adjusted later depending on speed and accuracy requirements. See Bourdev and Brandt [1] for an algorithm to compute them.

The AdaBoost learning algorithm [7] is used to both select a series of features (weak classifiers) from the pool of possible features and to set the parameters for each feature. The parameters of a feature are θ , α and β as described in section 4. The result of AdaBoost training is a strong classifier of the form

$$C(x_1, \dots, x_n) = \begin{cases} 1 & \sum_{i=1}^N h_i(x_1, \dots, x_n) \geq T \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

6. Experiments

Using the framework described above, we trained eight different pedestrian detectors for the following directions of motion: north, south, east, west, north-east, southeast, northwest, southwest. The training data consisted of frames from 25 video sequences of various lengths with a total of 78,858 frames. The training sequences came from shooting short (a few minutes long) video sequences using a consumer mini-DV camcorder in various outdoor locations. Each sequence was taken from a height of between 1 and 8 stories. Bounding boxes around each pedestrian for every frame were marked by hand.

From this raw data, an initial set of positive and negative training examples (consisting of 10 image windows each) were extracted for each direction. Each pedestrian bounding box was scaled to be 24x26 pixels (width \times height). This yielded examples such as in figure 1. For negative examples, image windows from 10 consecutive frames that did not significantly overlap with pedestrian boxes were selected (and scaled to 24x26 pixels). The initial training set consisted of 2500 positive examples and 2500 negative examples. Bootstrapping was used to replace the negative examples with new more difficult negatives during training.

For training, the filter set from which AdaBoost selected filters consisted of about 40,000 filters randomly selected from the much greater set of possible filters of the three basic types shown in figure 2. Real AdaBoost learning was run for 400 iterations to yield strong classifiers with 400 features for each of the 8 directions trained.

To evaluate a detector, it must be scanned across all positions, scales and frames of a video sequence. The details of the scanning process are as follows. First, an image pyramid is created for each frame. We used a scale factor of 0.75 between frames of the pyramid. For a set of 10 consecutive frames, the detector is evaluated in each position of each level of the image pyramid while shifting the detection window with a 1 pixel step size. For a particular position, the ten 24x26 pixel image windows are given as input to the soft cascade detector which outputs a 0 or 1 to indicate whether a pedestrian was detected in that position. If two or more detections overlap significantly then their bounding boxes are merged by averaging their top left x and y coordinates and their heights and widths. For a 360 x 240 pixel image, our scanning process evaluates 176,920 windows.

7. Results

A test set consisting of 21 video sequences was collected in similar but different scenarios to the training sequences. The test set consisted of 83,152 frames, 450 unique pedestrians and 23,469 ground truth pedestrian bounding boxes. The detection rate for a detector is computed over all pedestrians in all frames independently. The same pedestrian in frame 1 and frame 2 is counted as two instances to be detected. The pedestrian in frame 1 will be detected while looking at the appropriate window in frames 1 through 10 while the pedestrian in frame 2 will be detected while looking at the appropriate window in frames 2 through 11.

A receiver operating characteristic (ROC) curve was computed for the combined detector (consisting of the merged results of all 8 detectors). The combined detector achieves 93% detection rate with about a 10^{-6} false positive rate. Since a single 360 x 240 pixel frame has 176,920 windows in it, the false positive rate is equivalent to .177 false positives per frame. We also tested the pedestrian detector described in Viola-Jones-Snow [10] on our test set. For detection rates above 70%, the results show an order of magnitude improvement in the false positive rate. ROC curves for both detectors are shown in figure 3.

It is interesting to analyze the filters chosen. Across all 8 detectors, 25% of the filters selected by AdaBoost training were appearance type filters, 56% were difference type filters and 19% were shifted difference type filters. This clearly shows the importance of motion over appearance for detecting low resolution pedestrians.

The running time to scan all 8 detectors over all scales of a 360 \times 240 image is about 0.2 seconds or 5 frames per second on a Pentium 4 2.8 GHz computer. Some example detections are shown in figure 4 (only

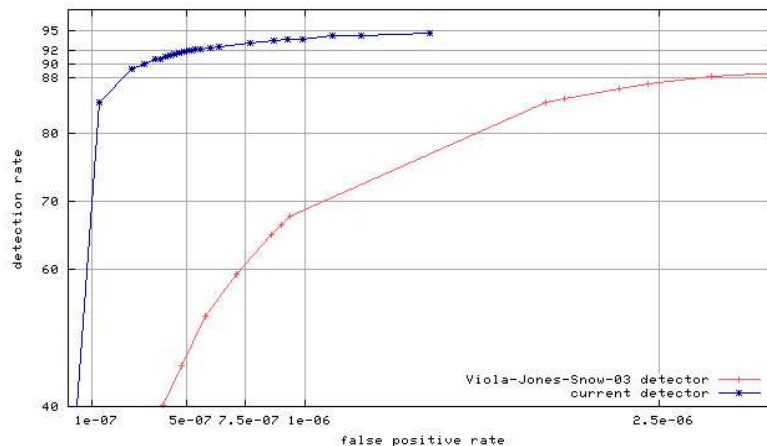


Figure 3. ROC curves comparing the combined detector described in this paper to the detector described in Viola-Jones-Snow 2003.

the first frame of the 10 frame sequence is shown).

8. Conclusions

We have presented an extension to the pedestrian detector of Viola, Jones and Snow [10] that uses many frames in a scanning window style detector. This extension allows much more sophisticated motion analysis than was possible in [10]. The resulting detector is able to detect pedestrians in typical surveillance scenarios with a detection rate of about 93% with a false positive rate per window of 1 in a million which is an order of magnitude improvement over [10]. Most other work in human/pedestrian detection either looks only at a single image or builds on top of a general tracker for video. Our approach does neither of these. It builds a scanning window type detector that acts directly on the pixels from a window in 10 consecutive frames of video. The improvement in accuracy we achieved shows the importance of motion information for detecting low resolution pedestrians.

References

- [1] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *CVPR*, pages II:236–243, 2005.
- [2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
- [3] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, pages 428–441, 2006.
- [4] C. Huang, H. Ai, Y. Li, and S. Lao. High-performance rotation invariant multiview face detection. *IEEE Patt. Anal. Mach. Intell.*, 29(4):671–686, 2007.



Figure 4. Scene with many moving cars

- [5] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *CVPR*, 2005.
- [6] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *ICCV*, 1998.
- [7] R. Schapire and Y. Singer. Improving boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3), 1999.
- [8] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on riemannian manifolds. In *CVPR*, 2007.
- [9] P. Viola and M. Jones. Robust real-time face detection. *Int. J. Computer Vision*, 57:137–154, 2004.
- [10] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *Int. J. Computer Vision*, 63(2):153–161, 2005.
- [11] Q. Zhu, M. Yeh, K. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *CVPR*, pages 1491–1498, 2006.