Informed Haar-like features for Pedestrian Detection

Sigberto Alarcon Viesca
Stanford University
Stanford, CA
salarcon@stanford.edu

Brandon Garcia
Stanford University
Stanford, CA
bgarcia7@stanford.edu

Abstract

We implemented the Informed Haar-like Features algorithm for pedestrian detection. This was composed of three primary endeavors: (1) feature extraction from images using Integral Channel Features and Haar-like templates generated from a pedestrian model (2) training of an AdaBoosted decision tree classifier (3) development of a sliding window detection algorithm. Combining these components, we present an early iteration of a pedestrian detection system.

1. Introduction

Pedestrian detection has received considerable attention over the last few years [1]. In the simplest terms, it consists of identifying any pedestrians in an image with bounding boxes. It is a well-defined problem and instance of object detection in still images and video that spans several applications, including security and space usage analytics and self-driving cars. In spite of a substantial effort to solve the problem, even the most avant-garde methods still underperform human ability [2].

In this paper, we describe and evaluate an implementation of the successful Informed Haar-like features method using a single static camera [3]. While this method's miss rate of 34.4% at 0.1 false positives per image (FPPI) is below the current best of 14.72% [2] for the Caltech pedestrian dataset, it has the advantage of requiring comparatively little computing power. It is thus better suited for real-time applications and processing of long video files.

With the exception of the image filter computations and machine learning, we implemented the entirety of the Informed Haar-like Features classification and detection. It is freely available on GitHub at github.com/sigberto/informed-haar.

While newer convolutional neural network-based like Fast R-CNN [10] and Faster R-CNN [11] have provided quicker, more accurate platforms for object detection, they require a GPU for practical applications. In contrast, Informed Haar-like Features can run on any modern computer at relatively good speeds.

1.1. Related Work

Viola and Jones provided one of the most influential works in detection with their application of Haar-like templates for feature extraction on face detection [6]. They introduced the notion of integral images, (a way to quickly compute image features as a sum of pixel regions), as well as the use of a boosting algorithm to identify the most relevant features. Other methods emphasized histogram of oriented gradients (HOG) as the main way to extract features, as well as the identification of parts of an object to create models, as in the work of Felzenszwalb et al [7]. Both of these approaches had a significant influence on feature extraction. An additional approach was developed by Dollar et al. [8]. For every image, they applied a series of filters, such as LUV and HSV color spaces, gradient magnitudes, HOG, difference of gradients (DoG), edges and thresholding and extracted features as a weighted sum of these resulting "channels". Variations of this approach have also seen relatively wide adoption for pedestrian detection. The Informed Haar-like Features algorithm combines elements from the above approaches to produce a robust detector—it combines the templates from Viola and Jones, feature types from Felzenszwalb et al. and the integral channel features from Dollar et al. Its feature extraction involves three key steps:

a. Generation of "templates" that correspond to different sections of the body of a pedestrian (head, torso, legs and background)
b. Extraction of image features through the application of image filters like gradient and color spaces
c. Use of the templates to localize and weigh the image features

2. Problem statement

Pedestrian detection consists of identifying pedestrians in an image by constructing a bounding box around each instance (fig. 1). This is a popular problem in computer vision not only because of its applications, but also because of the challenges it presents. Unlike detecting a well-defined and invariant object like a vegetable or a national flag, pedestrians present a number of challenges,

including high intra-class variability (different sizes, skin colors, hair color, cloths and postures), scale, lighting conditions, occlusions, and the sheer number of objects in an average street scene. In addition, many applications like self-driving cars require real-time detection for immediate reaction.

The performance of a classifier is measured by two key figures: miss rate and false positives per image (FPPI). The former is the percentage of pedestrians in a training or test set that were not detected; the latter is the average number of bounding boxes generated that do not correspond to a pedestrian. These are often plotted against each other since they are negatively correlated. (fig. 2).

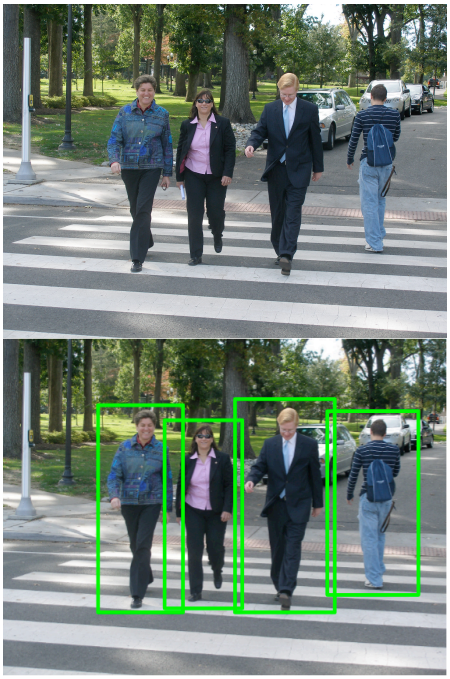

Fig. 1. Example pedestrian detector input (above) and objective output (below)

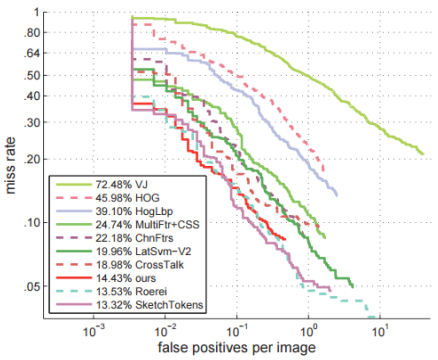

Fig. 2. Results of different detectors on the INRIA dataset using standard evaluation settings [3]

3. Technical Content

Our implementation is composed of three primary endeavors: Haar-like feature generation, classifier training, and detection refinement. The Informed Haar-like Features algorithm approach incorporates prior information about the problem into its approach for generating feature vectors for an image—it models upright humans as three-part objects: head, torso and legs. While there is a high degree of variability in terms of color, pose and even texture, the regularity of the overall shape of a pedestrian—especially the head-shoulder area—exhibits high intraclass similarity and a geometry seldom found elsewhere. We made use of feature values derived from a variety of binary (two regions) and ternary (three regions) Haar-like feature templates and color/gradient information from the image pixels. We trained our classifier using the AdaBoost learning algorithm on shallow decision trees. Using our classifier, we then built a pedestrian detector, using sliding windows and our classifier scores to propose candidate bounding boxes. We further refined our bounding box proposals via non-maximal suppression (NMS). In an attempt to optimize for speed and performance, we tuned the number of features that we trained our classifier on, the number of estimators in our ensemble, and the maximum depth of each individual decision tree. For detection, we adjusted our threshold for NMS, but primarily focused on refining our scaling process.

We used the INRIA Person dataset [5] to train and test our model; this is one of the two of the most popular datasets for the task at hand (the other being the Caltech dataset [4]) and makes comparison with other methods easy and direct. The overall training and testing pipeline is summarized as follows:

Training:

- Obtain average edge map of 120x60 pixels cropped pedestrian dataset with Canny edge detector
- Subdivide the resulting 120x60 into n x n cells. Label each cell as part of head, torso, legs or background
- Generate bimodal and trimodal Haar feature templates based on the above labeling
- Pass each image in the training set through various filters—gradient in the X and Y directions, LUV color scheme, and histogram of oriented gradients (HOG). These result in eleven two-dimensional "channels" as described by Dollar et al., per image
- Combine templates and channels to produce a feature vector for each image
- Train an AdaBoost Decision Tree classifier on the image features

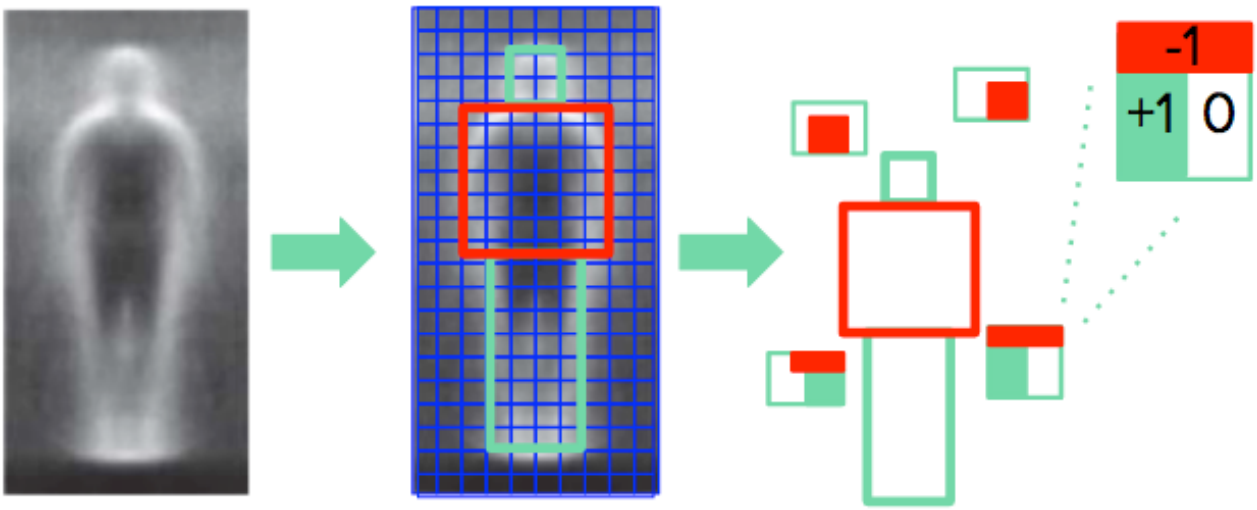

Fig. 3a. Average edge map produced by Canny edge detection

Fig 3b. Segmentation of pedestrian average edge map into body sections and background

Fig 3c. Template generation from segmented average edge map

- Take the most discriminative features and retrain a new classifier on just those top features (in this case, a feature is composed of a template and 1 of our 11 channels)

Testing:

- For each test image, extract channels in the same way as during the training, and slide a pedestrian-shaped 120x60 pixel window over the entire image at different scales.
- For each window, combine the top feature templates and channel feature information to create a feature vector
- Classify each feature vector and keep the bounding boxes corresponding to the feature vectors that are labeled as pedestrians.
- Apply NMS on the resulting bounding boxes to obtain the final predictions on pedestrian locations

3.1. Feature extraction

3.1.1 Template pool

We first compute an average edge map (using Canny edge detection) on all the cropped positive images and divide the resulting average image into cells of n x n pixels (fig. 3a). The authors found that 6 x 6 pixel cells maximize performance. This enables us to create a label $L(i, j)$ for each cell $c(i, j)$ in the image as belonging to each of the parts of the body (head, torso, and legs) that the model considers, as well as the background (fig 3b). This labeling defines a model for the shape of a pedestrian. We can then generate a template pool used to extract features for classification and, similarly, detection. These templates are generated as collections of n x n cells from the segmented average edge map, with sizes ranging from 2 x 1 and 1 x 2 cells to 4 x 3 cells. We take every possible template within this range and slide it over our pedestrian model. At every point that we slide our windows over, we generate either a binary or ternary template, depending on how many components exist in our restricted window view (Note that there can never be four since the head and the legs separated by seven cells along the y-axis). Each section of the cell is assigned either +1 or 0 in the bimodal case and one of +1, 0 and -1 in the trimodal case (fig. 3c) to distinguish which cells belong to which regions. We call this weight matrix for each template W at each position in our pedestrian model.

3.1.2 Feature generation with templates, gradient and color information

For each cell $c(i, j)$, we create an n-dimensional descriptor that contains information from a variety of different channels. Zhang et al originally use 10 channels in their journal publication [3]: 3 color channels, 1 channel for gradient magnitude, and 6 channels for histograms of oriented gradients (HOG). We use the same channels, with the exception of using 2 gradient channels, one in the X and one in the Y directions, as per Zhang's subsequently published dissertation [12]. Each cell $c(i, j)$ is therefore described by a 1 x 1 x 11 feature vector; the 120 x 60 sliding window is represented by a 20 x 10 x 11 volume, since cells are 6 x 6 pixels.

The information from the templates is taken into account by computing an average weight for each of them from the +1, 0, and -1 assignments (eq. 1), i.e. the matrix W. Using W_{avg} we can find the feature value for every template-channel combination (eq. 2). Hence, the length of the feature vector for an image is given by the product of number of templates and the number of channels.

$$W_{avg} = \frac{sgn(W)}{\Sigma 1[W(i,j)=+1]} + \frac{sgn(-W)}{\Sigma 1[W(i,j)=-1]}$$

Eq. 1. Template average weight formula

$$f(t,k) = \sum_{i=0}^{h} \sum_{j=0}^{w} o(x+i, y+j, k)\, W(i,j)$$

Eq. 2. Feature value for every template-channel pair

3.2. Classification and Feature Selection

Like Zhang et al. we train an AdaBoost classifier on our data set in order to determine the most discriminative features for classification and detection. Our features are derived from pairs of templates and 1 of our 11 channels. We used shallow decision trees in our ensemble because they are simple, fast, and robust to outliers. They split the data without being pulled and skewed by noise.

We trained our model on feature vectors of length 24,926. This results from pairing each of our 2266 generated templates with each of our 11 channels. We utilized 2000 estimators in our AdaBoost classifier with a max-depth of 2 for each decision tree. After a 36 hour training period, we learned the most discriminative of the 24,926 features and were able to subsequently train much faster, simpler classifiers. We experimented with varying the number of estimators (200 - 2000) and features (100 - 1000). Figure 4 displays the most discriminative templates found during training; figure 5 shows a weighted map of the cells that most successfully classify pedestrians.

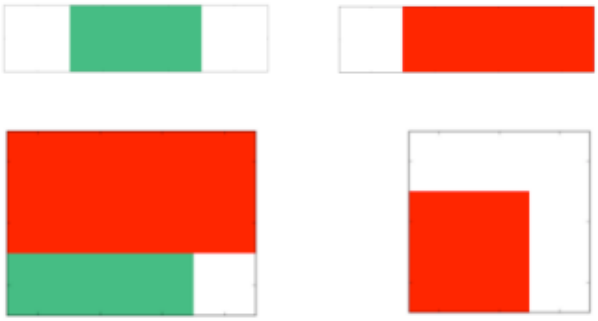

Figure 4. Templates corresponding to our top 4 most discriminative features. Top right (1) is background (white) and head (green). Top left (2) is background (white) and torso (red). Bottom left (3) is torso (red), legs (green) and background (white). Bottom right (4) is head (red) and background (white).

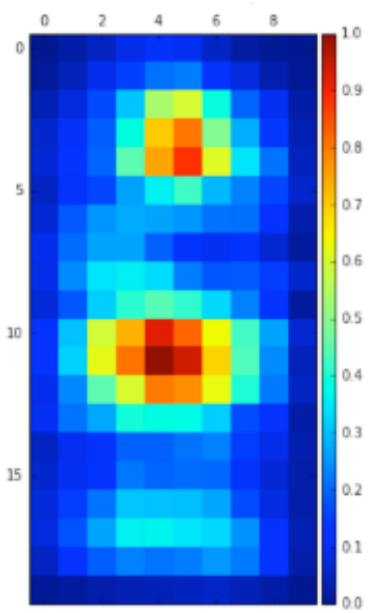

Figure 5. Visualization of high activation cells in a 120 x 60 pixel sliding window used for classifying an image region. After training our classifier, the cells in the area of the head and torso clearly appear to be discriminators for the presence of the pedestrians.

3.3. Detection

Our detection step consists of a sliding window in the shape and size of the pedestrian template model we trained on and consider multiple scales, starting by resizing the image such that its height or width is the same as that of the pedestrian's. We then resize the image to a larger size by a chosen scaling factor and slide the window again so that we can detect pedestrians of different scales. We experimented with the scaling factor (1.09-1.6), as well as the scaling method between consecutive resizings (linear, exponential). We consolidate the resulting bounding boxes by applying NMS to arrive at a final set of candidate detections which are compared against the ground truth bounding boxes provided by the INRIA Person dataset [5].

4. Experimental Setup and Results

4.1. Dataset description

We trained and tested our implementation on the popular INRIA Person dataset [5]. It is based on a training set of 614 positive and 1218 negative images, and a test set of 288 positive and 453 negative as a base. Multiple images contain more than one pedestrian. However, in order to facilitate training, the authors provide "normalized images", which are crops of the original set—for the positive images, they contain the pedestrians in the center of the image at a fixed image size. This format totals 2416 images in the training set and 1132 in the test set. For the negative images, the authors suggest taking random crops of the original negative images, which total 4872 in our case. When we tested our detector, we decided to do so on a random subset of 200 of the test images. This

is fully in response to the high compute and time requirements of our system.

4.2. Evaluation metrics

For classification, we present the accuracy and the F1 score on the test set.

For detection, the dataset provides annotations on the original images: ground-truth bounding boxes on the pedestrians in an image. Evaluation of detection is based on two metrics: false positives per image (FPPI) and miss rate. For a given image, a set of detections and a set of ground truths, these metrics are determined as follows [9]:

1. Sort detections by score for greedy matching
2. If the area of overlap between a detection and a ground truth is greater than 0.5, assign a match (eq. 3) and remove both from the matching pool.
3. After all matches are made, the number of false positives is the difference between proposed detections and number of matches (eq. 4)
4. The number of misses is the difference between ground truths and number of matches (eq. 5)

The above steps are computed for all images and the overall miss rate and FPPI is reported

$$a_o = \frac{area(BB_{dt} \cap BB_{gt})}{area(BB_{dt} \cup BB_{gt})} > 0.5$$

Eq. 3. Area of overlap

$$FP = \#detections - \#matches$$

Eq. 4. False positives per image

$$MR = \#ground\ truths - \#matches$$

Eq. 5. number of misses per image

4.3. Results

We experimented with both classification and detection parameters to obtain the best-performing repressor on our implementation of the Informed Haar-like features algorithm. Our results for both classification and detection are presented below. Tables 1, 2 and fig. 6 present our quantitative results. Figs. 4, 5 and 7 display visualizations of the most informative templates and pedestrian window regions.

4.4. Results analysis

Overall, we observe that our classifier achieved relatively high accuracy and F1 score (table 1) on the INRIA Person test set, even when we significantly reduced the number of features and estimators used. On the other hand, the performance of the detection component of our implementation faired less favorably than expected. While Zhang et al. report a miss rate of 33.4% at 0.1 FPPI, our best-performing detector (closest detector to the origin in the miss rate vs. FPPI curve, fig 6.) only registered 0.7129% miss rate and 11.64 FPPI. We suspect that a couple of reasons could have had a significant influence on this discrepancy.

A factor that could have made a significant contribution is the fact that we had approximately twice as many negative images as positive images during training, whereas Zhang et al. used a factor of ten. Even though our implementation performed well on the test set, it only has less than twice as many negative example per positive example. When we slide a window over an image, we examine as many as 50,000 subimages. Even a false positive rate of 1% would generate 500 false positives pre-NMS. This explains why we observe comparably high FPPI. Qualitatively, we observed that a significant amount of the false positives generated were on sections of the image that had a complete lack of pedestrian components; some were drawn around sand, water, phone booths, and floor tiles. A larger, more carefully selected set of negative images would likely have significantly reduced FPPI. A way to improve the classifier in future work and minimize this problem would be to train it twice: once using ten times more negative images than positives ones and another using hard negative images. The latter would be collected by running the half-trained classifier on random crops of negative images and assembling a new negative training set from those that were incorrectly classified as pedestrians with high confidence and retraining the classifier on these.

Moreover, we observe that our miss rate is significantly below the reported 33.4% at 0.1 FPPI. We believe a significant contributing factor to this is our limited rescaling. Unfortunately, Zhang et al. do not describe their detection process in much detail. While they mention using a sliding window and using NMS, the extent to which they describe scaling is entirely contained by "[t]he spatial step size is set identical to the cell size for speed and the scale step is set to be 1.09 so that there are 8 scales in each octave.". While this gives some guidance, there is no mention of what scale the image starts at relative to the pedestrian window of 120 x 60 pixels, or a mention of a stopping condition. We used two scaling methods:

1. Scale image height or width to minimum of 120 or 60 pixels, whichever is larger, while keeping the aspect ratio and subsequently increasing the size by a scaling factor a fixed n times in exponential form.
2. Scale image to four times the height or width of the sliding window, whichever is larger, while keeping the aspect ratio and subsequently decreasing the size of the image a variable n times in exponential form until the image is small enough to have approximately the same height as the pedestrian window.

Row	Number of Features (most important)	Number of Estimators	Max Tree Depth	Accuracy	F1 Score
1	24,926	2000	2	0.9782	0.9713
2	1000	2000	2	0.9771	0.9701
3	250	500	2	0.9737	0.9655
4	100	2000	2	0.9642	0.9532
5	100	1000	2	0.9635	0.9523
6	100	500	2	0.9649	0.9540
7	100	200	2	0.9611	0.9494
8	100	200	3	0.9494	0.9611

Table 1. Classification accuracy and F1 score
Classification parameters
Classification performance

Unfortunately, we had to keep the scaling factor relatively large and the number of resizes relatively small due to the large amount of time our implementation required to test and evaluate each image. For example, at a scaling factor of 1.33 and 6 resizings, each image takes approximately 5 minutes to process; testing on all 741 images in the test set entails 62 hours. Reducing the scaling factor to 1.09 and increasing the number of resizes would result in an even longer time. We attribute this inefficiency at least partly to the fact that our implementation is in Python, which can be 700 times slower than C++ and 50 times slower than Matlab [13]. Either of these languages or even a Cython base would have made experimentation easier and faster.

Comparing rows 7 & 8 in table 1 and rows 3 & 8, and 4 & 9 in table 2 it is clear that that tree depth had minimal impact on classification and virtually none on detection. While accuracy decreased slightly and F1 score increase slightly, the difference is too small to make any statistically significant conclusions—a different test set could potentially reverse those results. We attribute the lack of change in detection performance to the fact that because the classifier changed very slightly, so would the resulting detection bounding boxes. This change would not be substantial enough to change the matches between the detections and the ground truth, thus leaving miss rate and FPPI mostly intact.

We observe that the scaling factor and number of resizes had a significant effect on both miss rate and FPPI. This is most likely because the pedestrian window detector is trained on pedestrians of a very regular size—small translations or scalings can significantly change the predicted label and misclassify seemingly simple images.

At an NMS threshold of 0.5, we observe that the detector with the smallest miss rate spanned a relatively large number of resizes at a relatively small scaling factor, allowing it to sample a given image for pedestrians at a more comprehensive set of scales overall. The power of the scaling parameters becomes obvious when comparing rows 3 & 4 of table 2. Increasing the number of rescalings by one decreased miss rate by 0.08, but at the cost of more than doubling FPPI: we scan a larger number of potential pedestrians but also misclassify more of them as a consequence.

The number of estimators and the number of features chosen also had a minimal impact on classification and detection (table 2, rows 1-3). We attribute this to two observations. First, our method selects the most discriminative features from the original pool; there should be a logarithmic relationship between the number chosen and their performance. Second, the classifier distinguishes between only two classes, which substantially lowers the threshold required to successfully classify any given image.

Unsurprisingly, the NMS threshold had a very direct impact on both miss rate and FPPI (rows 15-17 in table 2). Decreasing the threshold discards fewer candidate bounding boxes, resulting in more matches, but also more false positives, and vice versa.

Row	Number of Features (most important)	Number of Estimators	Max Tree Depth	Scaling Factor	Number of Resizes	NMS Threshold	Miss Rate	FPPI
1	1000	2000	2	1.2	3	0.5	0.7794	7.01
2	100	2000	2	1.2	3	0.5	0.7826	9.165
3	100	200	2	1.2	3	0.5	0.7587	9.445
4	100	200	2	1.2	4	0.5	0.6755	22.67
5	100	200	2	1.09	4	0.5	0.6878	20.92
6	100	200	2	1.15	3	0.5	0.7812	9.365
7	100	500	2	1.2	3	0.5	0.7526	9.670
8	100	200	3	1.2	3	0.5	0.7587	9.500
9	100	200	3	1.2	4	0.5	0.6755	22.67
10	100	200	3	1.2	4	0.3	0.7776	14.71
11	100	200	2	1.2	4	0.3	0.7776	14.71
12	250*	500	2	1.4	4	0.5	0.7129	11.64
13	100*	200	2	1.5	3	0.5	0.7280	19.72
14	250*	500	2	1.6	3	0.5	0.7266	12.95
15	100**	200	2	1.33	6	0.5	0.7036	15.95
16	100**	200	2	1.33	6	0.3	0.8288	10.59
17	100**	200	2	1.33	6	0.7	0.6326	24.65

Table 2. Detection miss rate and false positives per image.
Classification parameters
Detection parameters
Detector performance

5. Conclusions and Future Work

We have shown that despite unfavorable detection results, our ground-up implementation of the Informed Haar-like features algorithm has the potential to deliver respectable performance on pedestrian detection for the INRIA Person dataset. Our experiments with number of classifiers and features, NMS threshold, scaling factor and number of resizes revealed that the first two variables have little effect on classifier and detector performance, while the latter three have significant power on miss rate and FPPI.

In spite of the relatively poor performance of this iteration of our implementation, we believe we now fully understand why it behaved in the way we observed. After noticing the high accuracy and F1 score of the classifier and the contrasting high miss rate and high FPPI of the detector, we wrongfully concluded that the poor detection performance was a product of only the detection part of

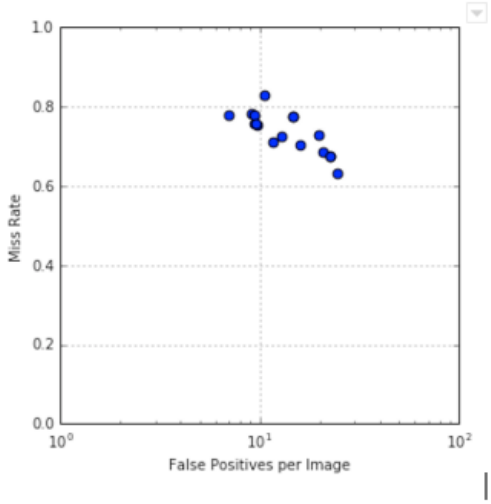

Fig. 6. Miss rate vs FPPI for all detectors

Fig. 7. Sample output from detector system

the pipeline, without taking into account the fact that the test sets for classification were starkly different and much more regularly structured than that of detection. This led us to perform a series of experiments with no significant increase in the detection metrics. Subpar classification explains why no change in rescalings or NMS produced any more fruitful results. As previously described, the first step of future work is to retrain our classifier with many more negative data points. Secondly, and equally importantly, is to train once more with hard negative images—a set compiled from using the classifier on negative images and selecting false positives produced with high confidence and retraining with those images to boost the robustness of the classifier. We could then use our current rescaling pipeline, and greatly reduce the number of false positives. We could also scan images over a greater scale range to minimize miss rate.

The next implementation of the algorithm would be in a more low-level language like MATLAB or C++ to minimize training and testing times and enhance the rate at which we perform experiments.

Possible extensions to the implementation include introducing new channels with SIFT-like features to increase robustness to rotation and reduce dependability of an upright pedestrian. We would also train and test on alternative datasets, including the Caltech pedestrian video dataset.

References

[1] R. Benenson, M. Omran, J. Hosang, B. Schiele. Ten Years of Pedestrian Detection, What Have We Learned?. ECCV, 2014

[2] S. Zhang, R. Benenson, M. Omran, J. Hosang, B. Schiele. How far are We From Solving Pedestrian Detection? Unpublished, 2016.

[3] S. Zhang, C. Bauckhage, A. Cremers. Informed Haar-like Feature Improve Pedestrian Detection. CVPR, 2014

[4] P. Dollár, C. Wojek, B. Schiele, P. Perona. Pedestrian Detection: A Benchmark. CVPR, 2009

[5] N. Dalal. INRIA Person Dataset. French Institute for Research in Computer Science and Automation, 2005

[6] P. Viola, M. Jones. Robust Real-Time Face Detection. IJCV 2004.

[7] P. Felzenszwalb, D. McAllester, D. McAllester. A Discriminatively Trained, Multiscale, Deformable Part Model. CVPR 2008

[8] P. Dollar, Z. Tu, P. Perona, and S. Belongie. Integral Channel Features. BMVC, 2009.

[9] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian Detection: An Evaluation of the State of the Art

[10] R. Girshick Fast R-CNN. ICCV, 2015.

[11] S. Ren, K. He, R. Girshick, J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. ICCV 2015

[12] S. Zhang. Efficient Pedestrian Detection in Urban Traffic Scenes. Doctoral dissertation, University of Bonn, 2014

[13] A beginners guide to using Python for performance computing. SciPy.org. http://scipy.github.io/old-wiki/pages/PerformancePython