Detecting Pedestrians by Learning Shapelet Features

Payam Sabzmeydani and Greg Mori
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
{psabzmey,mori}@cs.sfu.ca

Abstract

In this paper, we address the problem of detecting pedestrians in still images. We introduce an algorithm for learning shapelet features, a set of mid–level features. These features are focused on local regions of the image and are built from low–level gradient information that discriminates between pedestrian and non–pedestrian classes. Using AdaBoost, these shapelet features are created as a combination of oriented gradient responses. To train the final classifier, we use AdaBoost for a second time to select a subset of our learned shapelets. By first focusing locally on smaller feature sets, our algorithm attempts to harvest more useful information than by examining all the low–level features together. We present quantitative results demonstrating the effectiveness of our algorithm. In particular, we obtain an error rate 14 percentage points lower (at 10^{-6} *FPPW) than the previous state of the art detector of Dalal and Triggs [1] on the INRIA dataset.*

Figure 1. First row shows examples of pedestrian images. The next three rows show three local regions within the pedestrian windows. The last row shows the shapelet features learned by our algorithm for each of the three regions.

1. Introduction

In this paper we consider the problem of detecting pedestrians in still images. A robust solution to this problem would have numerous applications, including automated surveillance, image retrieval and search, and vehicle driver assistance systems. We will employ a window-scanning approach, where rectangular windows such as those shown in Figure 1 are classified as pedestrian or non-pedestrian.

The most important cue for detecting pedestrians in this still image setting is shape. In particular, pieces of shape such as the stereotypical omega pattern formed by the head and shoulders are the important cues for detecting pedestrians. We will refer to these pieces of shape as *shapelet features*[1]. Consider the cropped portions of windows centered on pedestrians shown in Figure 1. Salient shapelet features appear on the head and shoulders, along the legs, and on the arms of the pedestrians in these images.

The contribution of this paper is the development of an algorithm for learning these shapelet features which can discriminate between pedestrians and non-pedestrians. We show that by learning these shapelet features we obtain a pedestrian detector which significantly outperforms previous approaches which either use low-level edge cues, or hand-crafted mid-level features which try to capture the information present in our shapelet features.

The structure of this paper is as follows. In Section 2 we

[1]The term *shapelet* was first used by Refreiger [14], naming a series of localized basis functions of different shapes. Kovesi [7] used the term to refer to a basis of finite support for describing shape. In the same spirit, we use the term shapelet features to refer to features describing local pieces of shape, which we learn rather than specify as a basis.

1-4244-1180-7/07/$25.00 ©2007 IEEE

review the previous work on the pedestrian detection problem. In Section 3 we give the details of our shapelet features and the algorithm. The experimental details, performance results and comparisons are presented in Section 4. We conclude in Section 5.

2. Related Work

The problem of object detection and particularly pedestrian detection has received much attention from the computer vision community. In particular, Mohan et al. [10] detect pedestrians as a combination of parts –legs, torso, arms. Individual SVM detectors are trained for each of the parts, and their outputs are combined into a final classifier, after applying geometric constraints. Gavrila and Philomin [5] and Felzenszwalb [3] compare edge maps of pedestrian templates to edges found in images. Gavrila and Philomin use the Chamfer distance, and develop an efficient hierarchical system. Felzenszwalb uses a generalization of the Hausdorff distance, and learns pedestrian templates from training images. Viola et al. [18] compute spatial and temporal rectangle filters efficiently using the integral image technique, and learn a pedestrian classifier using a variant of AdaBoost. Mikolajczyk, et al. [9] use a part based detector to detect humans. They model humans as assemblies of parts that are represented by SIFT-like orientation base features. Feature selection and the part detectors are learned using AdaBoost. Fink and Perona [4] describe the MutualBoost algorithm for building a combined detector from a set of pre-defined and labeled object parts.

Leibe et al. [8] start with a local feature detection to generate a set of pedestrian hypotheses. By using a training set which contains foreground masks for pedestrians, segmentation masks are computed for these hypotheses. A top-down verification step, using these segmentation masks and Chamfer matching is then applied. Impressive results on images with substantial overlap of pedestrians are presented. Wu and Nevatia [20] also address the problem of overlapping pedestrians, and formulate a joint likelihood that is optimized using a greedy algorithm. Dalal and Triggs [1] use Histogram of Oriented Gradient (HOG) descriptors and Support Vector Machines (SVMs) for building a pedestrian detector. By tuning all the parameters of their HOG features, they compare the use of a variety of feature configurations to find the best configuration for pedestrian detection, on a challenging dataset of human figures. Munder and Gavrila [11] studied the problem of pedestrian classification with different features and classifiers. They find that local receptive fields can do a better job in representing pedestrians and also SVMs and AdaBoost classifiers outperform the other classifiers tested.

Other approaches use video sequences, and apply background subtraction to reduce clutter. The Pfinder system of Wren et al. [19] is an early example of such an approach. Zhao and Nevatia [21] work on difficult scenes involving large crowds of people, and segment foreground blobs into individual people using a Markov chain Monte Carlo technique. Elgammel and Davis [2] handle overlapping people by segmenting foreground regions according to a colour model that is initialized when the people are unoccluded.

3. Method

A major drawback in many object detection algorithms is the fixed set of feature descriptors that they use. The problem with defining features before training the classifier is that there could be some discriminative information that is missed by those features. By learning the feature set, we attempt to use information about the object classes in building our features.

Viola et al. [17, 18], use AdaBoost for face and pedestrian classification, using Haar-like wavelet features. The features they use are low-level, and AdaBoost selects a subset of these features to form the final classifier. In work closer to ours, Wu and Nevatia [20] use AdaBoost with a set of hard coded mid-level features, called "edgelets", as its weak classifiers. These edgelets are a set of pre-defined patterns of edges in different locations. Unlike the Haar-like features, they can harvest more information, but since they are fixed a priori they may not capture all the available information that is useful to discriminate between our object classes.

In our approach, we automatically learn a set of informative mid-level features, rather than hand-coding them. These mid-level features, called *shapelet features*, are constructed from low-level features. We build shapelet features which best discriminate between pedestrians and non-pedestrians. We use the AdaBoost algorithm as our core computational routine, using it once to build the shapelet features, and again to build a final classifier from these shapelets. For brevity, we omit some of the details of AdaBoost; our implementation follows the variant developed by Viola and Jones [17].

The training phase of our algorithm consists of three steps:

1. **Low-level Features:** The input to this step is raw training images. We extract the gradient responses of each image in different directions, and compute local average of these responses around each pixel. These low-level gradients will be used to build more sophisticated mid-level features (shapelets).

2. **Shapelet Features:** For each of a number of small sub-windows inside the detection window, we run Adaboost to select a subset of its low-level features to construct a mid-level shapelet feature. By only using the features inside each sub-window, we force Ad-

aBoost to extract as much information as possible at local neighborhoods of the image. This process will provide us with a shapelet feature for each sub-window. Each of these is intended to be more descriptive than the low-level features and discriminative regarding our object classes. Each shapelet feature consists of a combination of gradients with different orientations and strengths at different locations within the sub-window.

3. **Final Classifier:** The shapelet features only describe local neighborhoods of the image and therefore their individual classification power is still much below an acceptable level. By merging these features together we can combine the information from different parts of the image. In order to archive this goal, we use AdaBoost for the second time to train our final classifier, using shapelet features as its input.

3.1. Low-level features

Many pedestrian detection approaches capture local information as their lowest level features. Approaches include computing image gradients [1], computing wavelet coefficients [16], and applying simple rectangular filters [17] or more sophisticated features such as edgelets [20]. In our work, we use gradient responses as our lowest level features. We use the absolute value of gradient responses, computed in four different directions, as our low-level features. The absolute value is used because the sign of the gradient is uninformative due to varying clothing and background colors. To reduce the influence of small spatial shifts in the detection window, we locally average the gradient information in each direction by convolving the gradient responses with a box filter:

$$S_d(x) = |I(x) * G_d| * B \tag{1}$$

where $*$ denotes convolution, $I(x)$ is the intensity image, G_d is the gradient kernel (e.g. $[-1, 0, 1]$ or $[-1, 0, 1]^T$) that we use to get derivatives in direction $d \in \mathcal{D}$, B is a 2-D averaging box filter (e.g. a 5×5 matrix with all the elements $\frac{1}{25}$) used for averaging, and $S_d(x)$ is our final low-level feature, which captures the amount of gradient at every pixel in direction d. $\mathcal{D}$ is the set of possible directions that we are computing the gradients in. In our experiments we use four directions; $\mathcal{D} = \{0°, 45°, 90°, 135°\}$. Figure 2 shows two pedestrian examples and their $S_d(x)$ in two of the four computed directions.

At this point, for every pixel x in the image, we have the local average of gradient responses in different directions d. These locally smoothed gradient responses are used as our low level features.

The information captured about the classes by each of the low-level features is very little. If used as a classifier, each of these low-level features $S_d(x)$, can only separate our two classes (pedestrian and background) slightly better than random classification. To make our features more informative, we will use AdaBoost to combine them together to create more informative mid-level features, the shapelet features.

Figure 2. Low-level features of two sample images in two of the directions.

3.2. Shapelet Features

We define a *shapelet feature* as a weighted combination of low-level features. Each low-level feature consists of a location, a direction and a strength. Each shapelet feature covers a small sub-window of the detection window, and its low-level features are chosen from that sub-window.

We will consider k sub-windows $w_i \in \mathcal{W}$, $i = 1, \ldots, k$ inside our detection window. Selecting the set of sub-windows $\mathcal{W}$ will be explained in detail in Section 4.2. We will build a separate mid-level shapelet feature feature for each sub-window w_i. To do this, we collect all the low-level features that are inside that sub-window $\{f_d^p = S_d(p) : p \in w_i, d \in \mathcal{D}\}$ and consider them as potential weak classifiers of an AdaBoost run.

In each iteration t of the AdaBoost [17] training algorithm, one of the features $f_t \in \{f_d^p\}$ is chosen as the feature of the weak classifier $h_t(x)$ to be added to the final classifier. This weak classifier is of the form:

$$h_t(x) = \begin{cases} 1 & \text{if} \quad p_t f_t(x) < p_t \theta_t \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

for an image detection window x, where $\theta_t \in (-\infty, \infty)$ is the classification threshold of the classifier and $p_t = \pm 1$ is a parity for inequality sign.

After all T iterations of the algorithm, we get the final classifier $H_i(x)$ for sub-window w_i. This classifier is of the form:

$$H_i(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t^i h_t^i(x) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

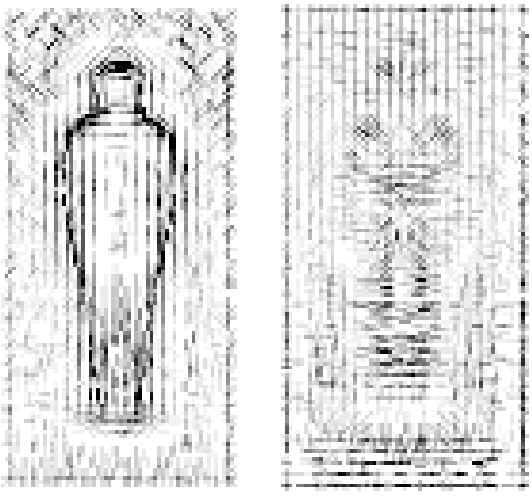
where α_t^i is the selected weight for classifier $h_t^i(x)$, as chosen by the AdaBoost algorithm. We train such a classifier for every sub-window w_i.

Each $H_i(x)$ is a local classifier, containing some of the low-level features inside the sub-window w_i. If we take a second look at the classifier form in equation 3, it can be

seen that the weighted sum of weak classifiers is a continuous value. Let us call this sum $s_i(x) = \sum_{t=1}^{T} \alpha_t^i h_t^i(x)$. A useful characteristic about these classifiers is that this $s_i(x)$ contains more information than only specifying the class by its sign. The further away the value of $s_i(x)$ from the zero, the more certain we are about its estimated class. Therefore this value can be used as a confidence measure of the classification. This is similar to the confidence prediction AdaBoost that has been developed by Schapire and Singer [15].

We define our shapelet features as these $\{s_i(x) : i \in \{1, 2, \ldots, k\}\}$. The index i corresponds to one of the sub-windows $w_i \in \mathcal{W}$, and $h_t^i(x)$ and α_t^i are the parameters associated with the classifier $H_i(x)$. Note that $s_i(x)$ is a shapelet feature that is trained specifically to distinguish between the two classes, based on gradients from its sub-window.

We train these shapelet features for a set of sub-windows inside the detection window. We visualize the results of the shapelet learning algorithm in Figure 3, which shows the sum of all the low-level features selected inside all the shapelet features over the entire the detection window. The selected low-level features are separated in two groups according to their classification parity p_t. This parity shows whether the selected feature is part of a positive (pedestrian) or negative (non-pedestrian) discriminating shapelet feature.



(a) (b)

Figure 3. An illustration of low-level features selected and weighted in shapelet features across the detection window. (a) pedestrian class features, (b) non-pedestrian class features.

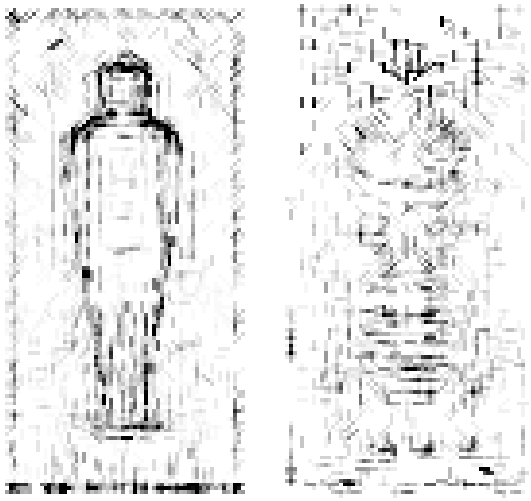
3.3. Final Classifier

Now that we have defined our shapelet features $s_i(x)$, we use AdaBoost to create a final classifier from them. The details of creating weak classifiers $g_t(s)$ are the same as previous step. Each $g_t(s)$ consists of one of the shapelet features $s_t(x)$, a threshold θ_t, and a parity p_t.

The final classifier is in the form of:

$$C(s) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t g_t(s) \geq \lambda \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $s = (s_1(x), s_2(x), \ldots, s_k(x))$ denotes all the shapelet features for input image detection window x. λ is the final classifier's threshold; originally zero but can be adjusted to get different detection and false positive rates. Note that this time the weak classifiers $g_t(s)$ (equivalent of $h_t(x)$ in the previous step) are applied in the new feature domain s instead of the low level features x, and therefore the final classifier will be a combination of weighted thresholded shapelet features. Again, the selection of the shapelet feature, threshold, and parity for use in classifier $g_t(s)$, along with its weight α_t, are chosen using the variant of AdaBoost used by Viola and Jones [17].



(a) (b)

Figure 4. Illustration of low-level features belonging to only the shapelet features inside the final classifier. (a) pedestrian class features, (b) non-pedestrian class features.

When the final classifier is trained, one can illustrate all the low-level features that are inside the selected shapelet features. Depending on the parity p_t assigned to each of the features (classifiers), these features are indicative of either pedestrians or non-pedestrians. An illustration of both sets of features are shown in Figure 4. Note how the feature sum that belongs to the pedestrian class contains the gradients of a person's silhouette.

3.4. Discussion

One might argue that using all the low-level features together in one AdaBoost run, and training a classifier in one

step could result in equivalent performance. There are two arguments against such an approach. First is that using all of the low-level features in one AdaBoost run is very expensive computationally, if not intractable. (We attempted such an experiment, and were unable to complete it.)

The second, and the more important reason is the "laziness" of AdaBoost. AdaBoost always extracts the minimum amount of information, from the training data, needed to predict the classes. This way many semi-redundant features that could be useful on the test set are discarded. In practice, it is often observed that iterating AdaBoost further to select more features, even after correctness on the training set has reached a plateau, proves advantageous. For our shapelet features, we extract more information at local areas of the image, before focusing on the final classification. This over-harvested information could contain noisy data, but overall, the extra information can lead to improved generalization, as our performance on our testing data indicates.

Our approach is related to the FeatureBoost algorithm of O'Sullivan et. al [12], which deemphasizes (or removes) individual features in successive AdaBoost-like iterations. While the motives are related, our work is different in the execution. We explicitly learn features in sub-windows of the detection window, and then combine them in a subsequent classification stage.

Also of note is the work by Huang et al. [6], who build a cascade detector similar to Viola and Jones [17], but use the output of the single detector trained for one layer of the cascade as an additional input to the next, in conjunction with other low-level features. In comparison, our approach instead builds multiple detectors (which are our shapelet features) from different areas of the image, and solely relies on their outputs for training the final classifier.

4. Experimental Results

We evaluate our algorithm by experimenting on two different datasets. One is the MIT pedestrian database [13], a largely used database for evaluation of pedestrian detection systems. Because of the simplicity of this dataset, we will perform most of our experiments on a more challenging dataset, the INRIA dataset [1]. In this dataset, pedestrians are mostly in standing position, but they cover more diverse body poses and a much varying background in comparison to the MIT set. Pedestrian image size in both sets is 64×128.

INRIA dataset consists of 1239 pedestrian images (2478 with their left-right reflections) and 1218 person-free images for training. In the test set there are 566 pedestrian examples ($566 \times 2 = 1132$) and 453 person-free images. MIT pedestrian database contains 923 pedestrian examples, together with their reflections (1846 images). We use 1020 of them for training and 826 for testing. For negative training and test sets, we use the negative sets of the INRIA dataset. We randomly sample 12180 negative samples from the training negative images to train both the shapelets and the classifier. Then we run the trained classifier on the whole negative training set and re-collect some harder samples that are classified as pedestrian. We re-train both the shapelet features and the final classifier using the initial negative samples and the selected harder examples as the negative training set. This bootstrapping process has a positive effect on the performance results and is done on all of our tests mentioned throughout this section.

The running time of our algorithm is reasonably efficient. Our unoptimized implementation in Matlab, exhaustively searches a 320×240 scale-space image (4000 detection windows) in less than 10 seconds.

For the quantification of the results we plot miss rate versus False Positive Per Window tested (FPPW) curves on a log-log scale.

4.1. Normalization

Because of the color, illumination, and background differences from image to image, gradient responses can vary drastically. Feature normalization is an effective way to overcome these sorts of problems.

Instead of normalizing the image intensity or all the gradients at the beginning, we apply our normalization at the shapelet level. This way we can handle local illumination changes much better. We use $L2$–norm normalization. We normalize the feature vector of every shapelet feature, by normalizing all the low-level features that fall into the sub-window associated with the shapelet feature:

$$f_d = \frac{f_d}{\sqrt{(\sum_{i=1}^{k} f_d(i)^2) + \epsilon}} \tag{5}$$

Where f_d is the vector of low-level features inside one shapelet, k is the number of those low-level features, and ϵ is a small number (1 in our experiments). Note that we do not normalize each of the four gradient directions separately. By normalizing the features of different gradient directions together, we prevent introducing noise in the directions that less gradient exists and instead have more emphasis on the dominant direction.

Normalizing every shapelet separately can make the computations very slow. We use the *Integral Image* method to overcome this problem. The Integral Image is a cumulative sum of an image in 2D. We compute the Integral Image of the square of low-level features summed up in the 4 directions, and use that image to find the normalization factor of each shapelet sub-window.

4.2. Shapelet Parameters

The most important parameter in shapelet features is their effective local area defined by their sub-window w_i. For simplicity we will call this parameter as shapelet's *size*. We investigate the influence of shapelet size on the detector performance.

Our shapelet features are learned using the AdaBoost algorithm. There are different ways to limit the number of AdaBoost iterations, such as defining a fixed number of iterations or setting a performance threshold as the ending condition. We fixed the number of weak classifiers relative to the size of each shapelet feature that the AdaBoost is training. We choose $m_i = \sqrt{n_i}$ weak classifiers, where n_i is the number of features inside the sub-window w_i.

We define three sets of shapelet features with different sizes:

- *Shapelet-S:* Small size shapelet feature set with sub-window size of 5×5 pixels and $10 (= \sqrt{5 \times 5 \times 4})$ extracted weak classifiers (low-level features) from each sub-window.
- *Shapelet-M:* Medium size shapelet feature set with sub-window size of 10×10 pixels and 20 extracted weak classifiers from each window.
- *Shapelet-L:* Large size shapelet feature set with a sub-window size of 15×15 pixels and 30 extracted weak classifiers.

We also create four new sets (overall seven sets with the three originals) as different combinations of these three sets. For example, *Shapelet-SM* set is the union of *Shapelet-S* and *Shapelet-M* feature sets.

For each shapelet set, we scan the detection window (of size 64×128) with that shapelet's sub-window size (e.g. 5×5), with strides of 4 pixels between sub-windows. This dense scan, will provide us many local sub-windows inside the detection window. Each sub-window is considered as the effective region of one of the shapelets in our shapelet feature set. That shapelet is learned by using only the low-level features inside its sub-window.

We trained and tested our detector for all the seven shapelet sets described. Performance results of these tests are shown in figure 5. Note that the fine-scale shapelets (Shapelet-S) contain essential information, the best detectors all use these shapelet features. However, adding medium and large scale shapelets (Shapelet-{SM,SL,SML}) significantly improves performance.

4.3. Evaluation and Comparison

To evaluate our detector on the INRIA dataset, we compare our results with HOG-SVM detector of Dalal and Triggs [1], the current state of the art pedestrian detector.

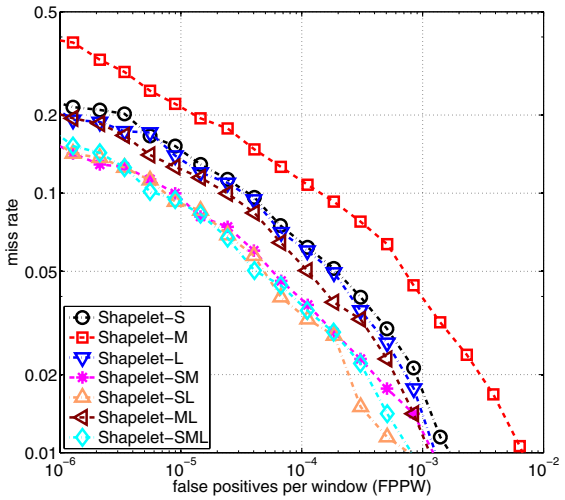


Figure 5. Performance of our detector using different shapelet features sets.

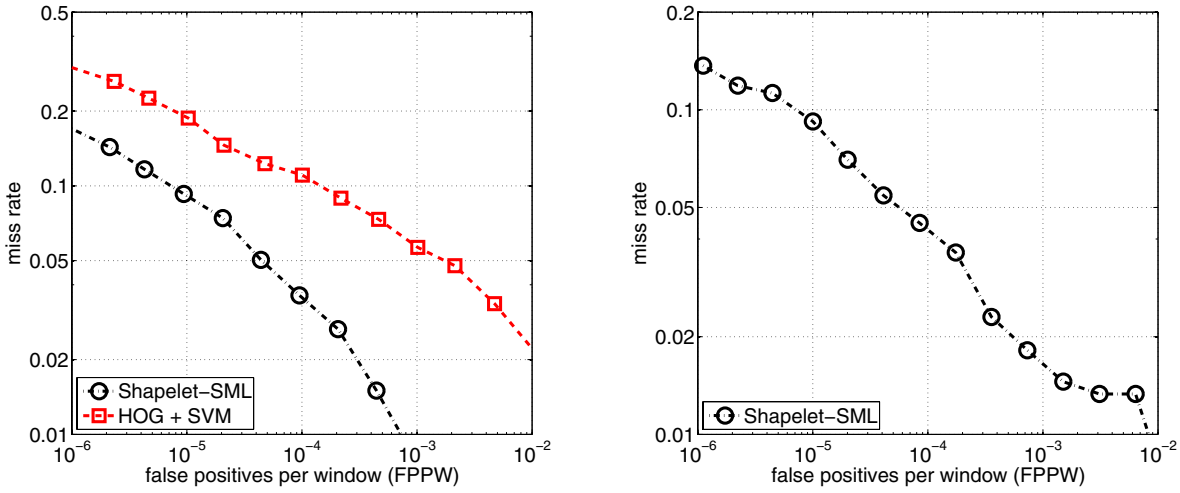
We use the performance results of their detector on the INRIA set that they have provided with the binaries of their detector. The performance of our detector on the INRIA dataset is shown in Figure 6. Our detector outperforms the HOG detector across all FPPW values, for example obtaining an error rate 14 percentage points lower at 10^{-6} FPPW.

Figure 7 gives a qualitative assessment of the mistakes made by our detector, showing the worst false negative (most non-pedestrian-like pedestrians) and the worst false positive (most pedestrian-like non-pedestrian) examples from our detectors point of view. Note that most of false negatives are due to the subject's unusual pose (e.g. riding a bicycle), semi occlusions, or the extremely light or dark environments. False positives usually contain vertical gradients mimicking the torso and leg boundaries.

MIT dataset is a widely used dataset for pedestrian detection algorithms but unfortunately it is not a well-defined set. It does not contain a negative set and the positive examples are not separated into training and testing sets. Therefore, it is not possible to directly compare our method with the previous ones that have used this dataset [10, 20, 1]. In Figure 6(b) we show results of applying our algorithm to our own version of the MIT dataset.

5. Conclusion

In this paper we presented a novel algorithm for pedestrian detection, using learned shapelet features. The power of our detector lies in the algorithm for learning discriminative mid–level shapelet features. The approach of learning discriminative shapelet features over separate regions of the image is able to capture informative cues from all over the



(a) (b)

Figure 6. (a) Performance of our detector on the INRIA dataset compared to the HOG detector of Dalal and Triggs [1]. (b) Performance of our detector on the MIT set.

image. This information is then integrated into a final classifier to detect pedestrians.

We showed quantitatively that these shapelet features can capture more information than fixed feature sets by improving the classification results. In particular, we obtained a lower error rate than the previous state of the art detector of Dalal and Triggs [1] on the INRIA dataset.

References

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, 2005.

[2] A. Elgammal and L. Davis. Probabilistic framework for segmenting people under occlusion. In *Proc. 8th Int. Conf. Computer Vision*, 2001.

[3] P. Felzenszwalb. Learning Models for Object Recognition. *Proc. Conf. Computer Vision and Pattern Recognition*, pages 56–62, 2001.

[4] M. Fink and P. Perona. Mutual boosting for contextual inference. *Neural Information Processing Systems*, 2004.

[5] D. Gavrila and V. Philomin. Real-time object detection for smart vehicles. In *Proc. 7th ICCV*, pages 87–93, 1999.

[6] C. Huang, H. Ai, B. Wu, and S. Lao. Boosting nested cascade detector for multi-view face detection. In *Proceedings of the 17th International Conference on Pattern Recognition*, 2004.

[7] P. Kovesi. Shapelets correlated with surface normals produce surfaces. In *Proc. 10th Int. Conf. Computer Vision*, 2005.

[8] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, 2005.

[9] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. *Proc. ECCV*, 1:69–81, 2004.

[10] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Trans. PAMI*, 23(4):349–361, 2001.

[11] S. Munder and D. Gavrila. An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1863–1868, 2006.

[12] J. O'Sullivan, J. Langford, R. Caruana, and A. Blum. FeatureBoost: A Meta-Learning Algorithm that Improves Model Robustness. *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 703–710, 2000.

[13] C. Papageorgiou and T. Poggio. A Trainable System for Object Detection. *International Journal of Computer Vision*, 38(1):15–33, 2000.

[14] A. Refregier. Shapelets: I. a method for image analysis. *Mon. Not. Roy. Astron. Soc.*, 338(35), 2003.

[15] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 80–91. ACM Press, 1998.

[16] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, volume 1, pages 746–751, 2000.

[17] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, 2001.

[18] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Proc. 9th Int. Conf. Computer Vision*, pages 734–741, 2003.

[19] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. PAMI*, 19(7):780–785, July 1997.

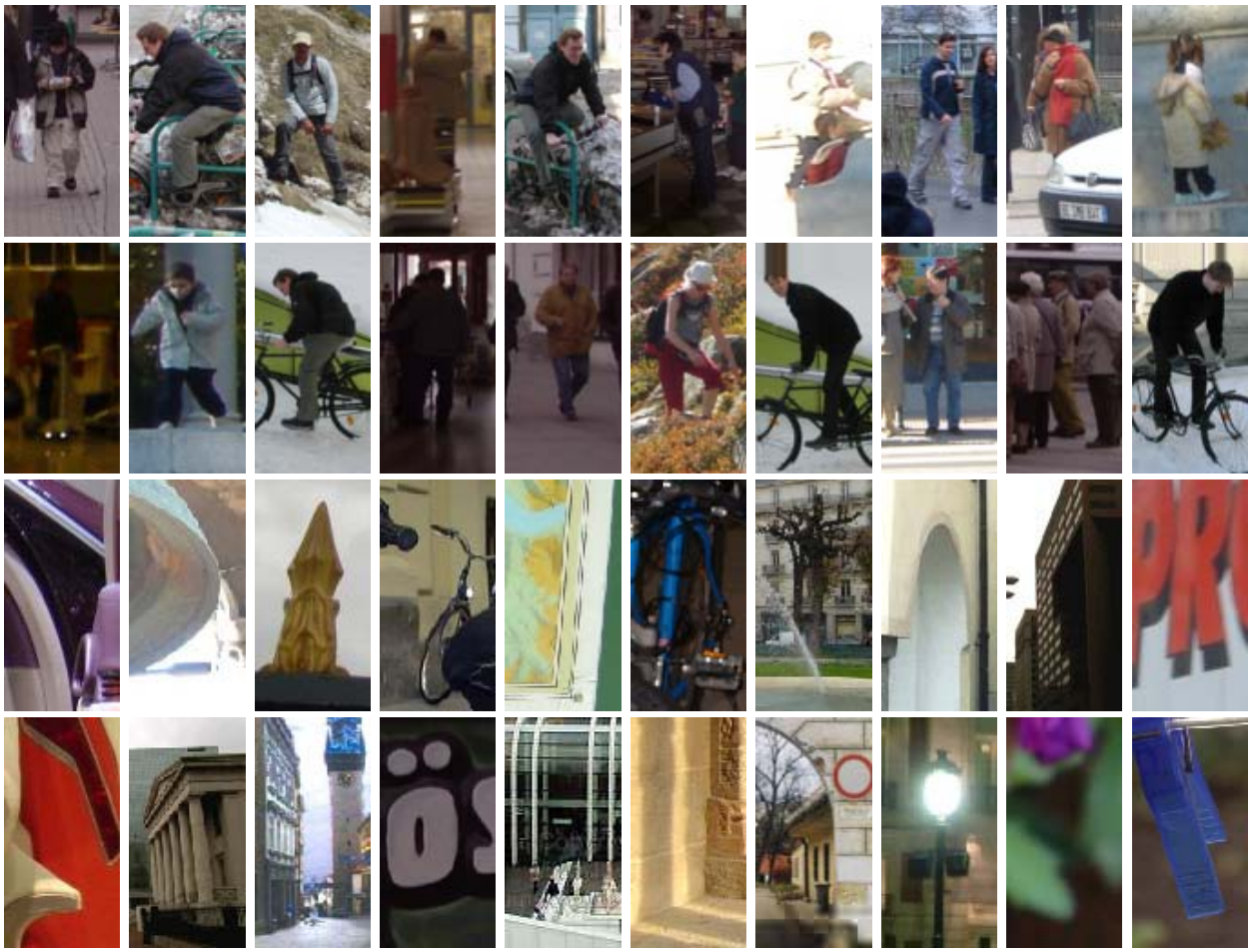


Figure 7. Examples of errors made by our detector. First two rows show false negative examples and the last two rows contain false positive examples.

[20] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Proc. 10th Int. Conf. Computer Vision*, 2005.

[21] T. Zhao and R. Nevatia. Bayesian human segmentation in crowded situations. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, volume 2, pages 459–466, 2003.