Towards Practical Evaluation of Pedestrian Detectors

Mohamed Hussein
Department of Computer Science
University of Maryland
College Park, Maryland 20742
Email: mhussein@cs.umd.edu

Fatih Porikli
Mitsubishi Electric Research Labs
Cambridge, MA 02139
Email: fatih@merl.com

Larry Davis
Department of Computer Science
University of Maryland
College Park, Maryland 20742
Email: lsd@cs.umd.edu

***Abstract*—Despite recent significant advancement in the area of pedestrian detection in images, little effort has been devoted to algorithm evaluation for practical purposes. Typically, detectors are evaluated only on color images. It is not clear how the performance would be affected if other modalities are used, e.g. thermal or near infrared. Also, detectors are evaluated on cropped images that have the same size as training images. However, in practice, detectors are applied to large images with multiple pedestrians in different locations and sizes. To apply a single size pedestrian detector, the input image is, typically, scanned several times with different window sizes. It is not clear how the detection performance would be affected by such multiple-size scanning technique. Moreover, to implement such a technique, one is faced with a multitude of design choices, each of which potentially affects the performance of the detector. The contribution of this paper is to assess and reason about the differences in detection performance of two state of the art detectors across changes in modality (visible or near infrared), evaluation method (on cropped or whole images), the effect of different design choices (resizing features or images, smoothing or not).**

***Index Terms*—Human Detection, Performance Evaluation, Near Infrared.**

I. Introduction

Pedestrian detection is one of the most challenging tasks in computer vision with a long list of fundamental applications from intelligent vehicles and video surveillance to interactive environments.

Pedestrian detection methods can be categorized into two groups based on the camera setup. For static camera setups, object motion is considered as the distinctive feature. A motion detector, either a background subtraction or image segmentation method, is applied to the input video to extract the moving regions and their motion statistics. A real time moving human detection algorithm that uses Haar wavelet descriptors extracted from space-time image differences was described in [1]. Using AdaBoost, the most discriminative frame difference features were selected, and multiple features were combined to form a strong classifier. A rejection cascade that is constructed by strong classifiers to efficiently reject negative examples is adopted to improve the detection speed. A shortcoming of the motion based algorithms is that they fail to detect stationary pedestrians. In addition, such methods are highly sensitive to view-point and illumination changes.

The second category of methods is based on detecting human appearance and silhouette, either applying a classifier at all possible subwindows in the given image, or assembling local human parts [2]–[6] according to geometric constraints to form the final human model. A popular appearance based method is the principal component analysis (PCA) that projects given images onto a compact subspace. While providing visually coherent representations, PCA tends to be easily affected by the variations in pose and illumination conditions. To make the representation more adaptive to changes, local receptive fields (LRF) features are extracted from silhouettes using multi-layer perceptrons by means of their hidden layer [7], and then are provided to a support vector machine (SVM). In [8], a polynomial SVM was learned using Haar wavelets as human descriptors. Later, the work was extended to multiple classifiers trained to detect human parts, and the responses inside the detection window are combined to give the final decision [9]. In [10], an SVM classifier, that was shown to have false positive rates of at least one-two orders of magnitude lower at the same detection rates than the conventional approaches, was trained using densely sampled histograms of oriented gradients (HOG) inside the detection window. In a similar approach [11], near real time detection performances were achieved by training a cascade model.

Despite the abundant work in pedestrian detection, little effort has been devoted to algorithm evaluation for practical purposes. Typically, detectors are evaluated on color images that are cropped to have the same size as the images used to train the detector. However, in practice, the input of the pedestrian detection algorithm is a video frame with possibly multiple pedestrians in different spatial locations and different sizes, which requires different scanning techniques. Besides, other imaging modalities such as thermal IR may be of interest in certain applications. It is not clear how the detection performance is affected by the adopted scanning strategy or the image modality.

Here, we aim to assess and discuss the differences in detection performance of two state of the art appearance-based detectors [11], [12] across changes in modality (visible or near infrared), evaluation method (on cropped or whole images), the effect of different design choices (resizing features or images, smoothing or not). We experiment on two different datasets:

one dataset consisting of color and another of near infrared images.

The paper is organized as follows. In Section II, we briefly describe the two pedestrian detectors that we analyzed. In Section III, we discuss the datasets, the training and testing parameters, evaluation methods and metrics, and give a detailed comparison. Finally, we present our main conclusions in section IV.

II. Evaluated Detectors

The two human detectors which we used for evaluation are based on a rejection cascade of boosted feature regions. They differ in how they describe the feature regions and in how the boosting algorithm works. One detector uses Region Covariance to describe feature regions and uses boosting on manifolds for classification [12]. We refer to this detector as RCM, for Region Covariance on Manifolds. The other detector uses Histograms of Oriented Gradients (HOG) to describe feature regions and uses conventional boosting for classification [11]. We refer to this detector as HOG. For the sake of completeness, we briefly describe here the notion of a rejection cascade of boosted feature regions, as well as the descriptors used by the two classifiers. The reader is referred to the original papers for details.

A. Rejection Cascade of Boosted Feature Regions

Rejection cascades of boosted feature regions were popularized by their success in the area of face detection [13]. They are based on two main concepts: boosted feature regions, and rejection cascades.

Boosting [14] works by combining weak classifiers to build a strong classifier. Boosting feature regions can be understood as combining simple feature regions to build a strong representation of the object which is a representation that can be used to best discriminate between positive and negative training examples. Feature regions in our case are rectangular subregions from feature maps of input images.

A rejection cascade is built of a number of classification layers. A test pattern is examined by layers of the cascade one after another until it is rejected by one of them, or until it is accepted by the final layer, in which case it is classified as a positive example. During training of the cascade, the first layer is trained on all positive examples and a random sample of negative examples. Each subsequent layer is trained on all positive examples and only negative examples that are wrongly classified as positives by the preceding layers. In this way, each layer handles harder negative examples than all the preceding layers. The benefit of this mechanism is two fold. One is the possibility of usage of a huge number of negative examples in training the classifier, which is not possible in training a traditional single layer classifier. The other is that, during testing, most negative examples are rejected quickly by the initial layers of the cascade and only hard ones are handled by later layers. Since, in our applications, most of the examined patterns are negative, the rejection cascade is computationally very efficient since it quickly rejects easy negative examples while spending more time on hard negative or positive examples. In our implementation, each cascade layer is trained using the Logit Boost algorithm [14]. In the case of Region Covariance, we use the variant that supports manifolds [12].

B. Region Covariances

Region covariances were first introduced as descriptors in [15] and then used for human detection [12], which outperformed other state of the art classifiers. Let I be a $W \times H$ one-dimensional intensity or a three-dimensional color image, and F be a $W \times H \times d$ dimensional feature image extracted from I

$$F(x, y) = \Phi(I, x, y) \tag{1}$$

where the function Φ can be any mapping such as intensity, color, gradients, filter responses, etc. For a given rectangular region $R \subset F$, let $\{\mathbf{z}_i\}_{i=1..S}$ be the d-dimensional feature points inside R. The region R is represented with the $d \times d$ covariance matrix of the feature points.

For the human detection problem, the mapping $\Phi(I, x, y)$ is defined as

$$\left[x \;\; y \;\; |I_x| \;\; |I_y| \;\; \sqrt{I_x^2 + I_y^2} \;\; |I_{xx}| \;\; |I_{yy}| \;\; \arctan\frac{|I_x|}{|I_y|}\right]^T \tag{2}$$

where x and y represent pixel location, $I_x, I_{xx}, ..$ are intensity derivatives, and the last term is the edge orientation. With the defined mapping the input image is mapped to a $d = 8$ dimensional feature image. The covariance descriptor of a region is an 8×8 matrix and due to symmetry only the upper triangular part is stored, which has only 36 different values. The descriptor encodes information of the variances of the defined features inside the region, their correlations with each other. These variances and correlations, in turn, encode the spatial layout of the region.

Region covariances can be computed efficiently, in $O(d^2)$ computations, regardless of the region size, using integral histograms [16] [15]. Covariance matrices, and hence region covariance descriptors, do not form an Euclidean vector space, which necessitates modifications to conventional machine learning techniques. But, since covariance matrices are positive definite matrices, they lie on a connected Reimannian manifold. Therefore, instead of using a conventional boosting algorithm, boosting on manifolds is used [12].

C. Histograms of Oriented Gradients

Histograms of Oriented Gradients were first applied to human detection in [10], which achieved a significant improvement over other features used for human detection at that time. Histograms of Oriented Gradients were used in a rejection cascade of boosted feature regions framework in [11] to deliver comparable performance to [10] but at a much higher speed.

To compute the Histogram of Oriented Gradients descriptor of a region, the region is divided into 4 cells, in a 2 layout. A 9 bin histogram is built for each cell. Histogram bins correspond

to different gradient orientation directions. Instead of just counting the number of pixels in each bin, gradient magnitudes at the designated pixels are accumulated. The four histograms are then concatenated to make a 36-dimensional feature vector, which is then normalized. In our implementation, we use L_2 normalization for HOG features.

Like Region Covariance descriptors, HOG descriptors can be computed fast using integral histograms. The number of integrals needed for HOG is equal to the number of orienation bins, which is typically fewer than integrals needed for Region Covariance. In our experiments, we found that constructing integrals for the HOG descriptors takes around 36% of the time needed to construct integrals for the Region Covariance descriptors, when 9 orientation bins are used.

III. Experimental Evaluation

We start by describing the two datasets we used and the differences between them. Then, we list the parameters used in training and testing the two classifiers. Afterwards, we explain the two evaluation methods: evaluation on cropped windows and evaluation on whole images. Then, we explain the metrics we compute in each evaluation method. Finally, we present the evaluation results.

A. Datasets

We evaluated the detectors on two different datasets, INRIA-Person and MERL-NIR. The INRIA dataset was introduced in [10], and subsequently used to evaluate several human classifiers, e.g. [11] and [12]. It consists of close to 2600 images, around 900 of which contain full-body human subjects. The remaining images are background images. Images are progressive scan color images of different sizes and aspect ratios. Images containing humans are of different people and locations. Some people appear in multiple images, but, most of them appear only once. In total, the INRIA dataset contains around 1800 different human images. Human images range from 48 to 832 pixels high. Sample whole images and cropped human images, that are resized so that the human's height is 96 pixels, are shown in figure 1.

The MERL-NIR dataset consists of around 60000 frames from a long video sequence. The video is shot from a vehicle touring an Asian city. The video is a near infrared video with interlaced scanning. Close to 10000 frames of the video contain annotated full-body human subjects. In total, the sequence contains close to 12000 human images. Because it is a video sequence, the same person can appear in many consecutive frames. Human subjects in the video compose 285 different tracks. Sample whole images and cropped human images, that are resized so that the human's height is 36 pixels, are shown in figure 2.

B. Training and Testing Parameters

Each dataset is divided into two portions, one for testing and one for training. For the INRIA dataset, we use the standard training and testing portions that come with the dataset. Images that contain humans are divided as 614 for training and 287 for testing. Background images are divided as 1218 for training and 453 for testing. Training is performed on positive samples cropped from images. Each positive example is resized so that the height of the human subject is 96 pixels. Then, a margin is added from background pixels to make the windows 128 pixels high. The width is set to half of that value, i.e. 64 pixels. The bottom row of figure 1 shows examples of these images.

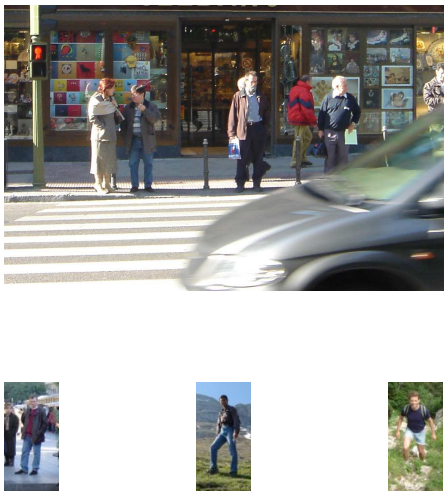


Fig. 1. Sample whole and cropped pedestrian images from INRIA dataset.

Fig. 2. Sample whole and cropped pedestrian images from MERL-NIR dataset.

For the MERL-NIR dataset, we excluded parts of the sequence that are too dark for a human inspector to locate people. These constituted roughly the last 15000 frames of the sequence. The other part is divided as 35000 frames for training and 10000 frames for testing. From the training portion, 5435 frames contained annotated humans. From the testing portion, 2818 frames contained annotated humans. Since consecutive frames are similar to one another, we sampled 1250 frames from each group to use in training

and testing the classifiers. Annotated human subjects in the sampled frames are cropped and resized so that the human subject's height is 36 pixels. Then, a margin is added from background pixels to make the windows 48 pixels high. The width is set to 24 pixels. Due to a shortage of large human subjects in the dataset, we could not use the same size as in INRIA dataset. The bottom row of figure 2 shows examples of these images.

For each cropped positive window, in both datasets, another copy is made by flipping the window around its central vertical axis. Negative samples for training are created by scanning background images with different window sizes. We use 9000 negative samples in training each layer of the cascade classifier. The target detection rate for each layer of the cascade was set to 0.998. The target overall false alarm rate was set to 10^{-6}. The target layer false alarm rate and the number of layers was not the same for the two datasets. For the INRIA dataset, the number of layers was set to 30 and the target layer false alarm rate was set to 0.65. For the MERL-NIR dataset, the number of layers was set to 50 and the target layer false alarm rate was set to 0.75. The difference between the two settings is in the level of control over operating point selection and should have no effect of the detection results.

C. Evaluation Method

We evaluated the classifiers on both cropped windows and whole images.

1) Evaluation on Cropped Windows: Positive examples are the cropped and resized windows around human subjects. Negative examples are windows of any size that are cropped from any location of background images.

2) Evaluation on Whole Images: We use the testing images that contain humans subjects. Each image is scanned with 9 different window sizes. In the case of the INRIA dataset, the smallest scanning window size is 48×96. In the case of the MERL-NIR dataset, the smallest scanning window size is 24×48. Subsequent window sizes are generated by multiplying the smallest sizes by powers of 1.25 in the case of the INRIA dataset, and 1.7 in the case of the MERL-NIR dataset. The scanning step is set to 5% of the scanning windows side length. When we evaluate on whole images, the same test image contains both negative and positive examples.

Since we train each classifier on single size images, in the case of whole images which contain humans of possibly different sizes, we have two options. One is to resize the images so that our scanning window size becomes the same as the training size and then scan the resized image with the training size. For example, if we want to scan a 640×480 image with a window of size 128×256, while the classifier is trained on size 64×128, then we resize the image to 320×240 and then scan with the training size. We will refer to this technique as *resizing images*. The other option is to resize the features selected by the classifier while maintaining their relative sizes to the scan window. For example, if a feature of size 8×8 at location $(4, 4)$ was selected by the classifier when trained on a window of size 64, then, if we scan an image with a window of size 128×256, the same feature is computed at location 8×8 with size 16×16. We will refer to this technique as *resizing features*.

Resizing features is generally faster since gradients and integral histograms need to be computed only once, while in resizing images, these computations need to be repeated for each size. But, as we will see in section III-E2, resizing features is less accurate.

D. Evaluation Metrics

To evaluate the two classifiers, we use the DET (Detection Error Tradeoff) curves as in [10]. The DET curves relate the *miss rate* to the *false alarm rate* on a log scale for both axes.

In the case of evaluation on cropped windows, the positive and negative examples are well defined. All positive examples contain a human subject that is centered in the window, and the size of the window is adjusted to be the same as training examples. Also, all negative examples do not contain any part of a human since they are all cropped from background images. Therefore, in the case of evaluation on cropped windows, computing the miss rate and false alarm rate is straight forward.

On the other hand, in the case of evaluation on whole images, the situation is different. An image, that can contain any number of humans in random locations and sizes, is entirely scanned several times, each time with a different window size. Scanned windows are not all perfect positive or negative examples. A scanned window may contain a human but the relative size of the human with respect to the window size is not the same as the relative size used for training, or it may contain a human that is not centered. In either case, the classifier may make a wrong decision because the relative locations of features are not exactly the same as learned during training. From a practical point of view, in many applications, detections are acceptable even if slightly shifted, or slightly smaller or larger than the subject. Therefore, we should not consider such windows as negative examples and penalize the classifier if it classifies them as positives. However, if we consider all scanned windows that are close to a human subject as positive examples, we will be penalizing the classifier if it misses any of them, although detecting just one of them is good enough in practice.

Based on these considerations, in the case of evaluation on whole images, we need a different measure for what is a missed detection and what is a false alarm. We consider any scanned window that is "significantly" (see below) far from all annotated human subjects in the image as a negative example, and hence count a positive classification of it as a false alarm. A missed detection is counted if an annotated human subject is "significantly" far from all scanned windows that are classified as positive by the classifier.

We now define how to determine whether a scanned window is significantly close to or far from an annotated human subject. We require a distance measure that is minimum when the scanned window is perfectly aligned with an annotated human subject, and is maximum when there is no overlapping

at all between them. One measure that has this property is the *overlap distance*. Let $|R|$ be the area of a region R. Consider two regions R_1 and R_2. The overlap distance between R_1 and R_2 is defined as

$$OLD(R_1, R_2) = \frac{|R_1 \cup R_2|}{|R_1 \cap R_2|} \ . \quad (3)$$

This is the ratio between the area of the union to the area of the intersection of the two regions. Given this definition, the overlap distance takes the minimum value of 1, when the two regions are perfectly aligned, and takes the value ∞ when there is no overlap at all between them.

In our evaluation, we consider a scan window negative if its overlap distance to the closest annotated human subject is above 16. We count a miss detection if all scanned windows within overlap distance of 2 around an annotated human subject are all classified as negatives.

E. Evaluation Results

We first present the baseline performance of the two classifiers on the two datasets. The baseline performance is the performance under the default parameters, which are the best parameters. Then, we will introduce a number of changes to the default parameters and describe how performance differs. Whenever a change is introduced, curves of the default parameters are reprinted besides the new curves to make comparisons easier.

1) Performance Using Default Configurations: Besides the difference between the two datasets in training and testing parameters mentioned in sections III-C and III-D, in the case of evaluation on whole images, the default setting for the INRIA dataset is to smooth the image after resizing it. But, the default for the MERL-NIR is to resize without smoothing afterwards. The reason will be clear in section III-E3.

Figure 3 compares the performance of the two classifiers on the two datasets. The top row shows performance on cropped windows. The bottom row shows performance on whole images. The left column shows performance on INRIA dataset. The right column shows performance on MERL-NIR dataset.

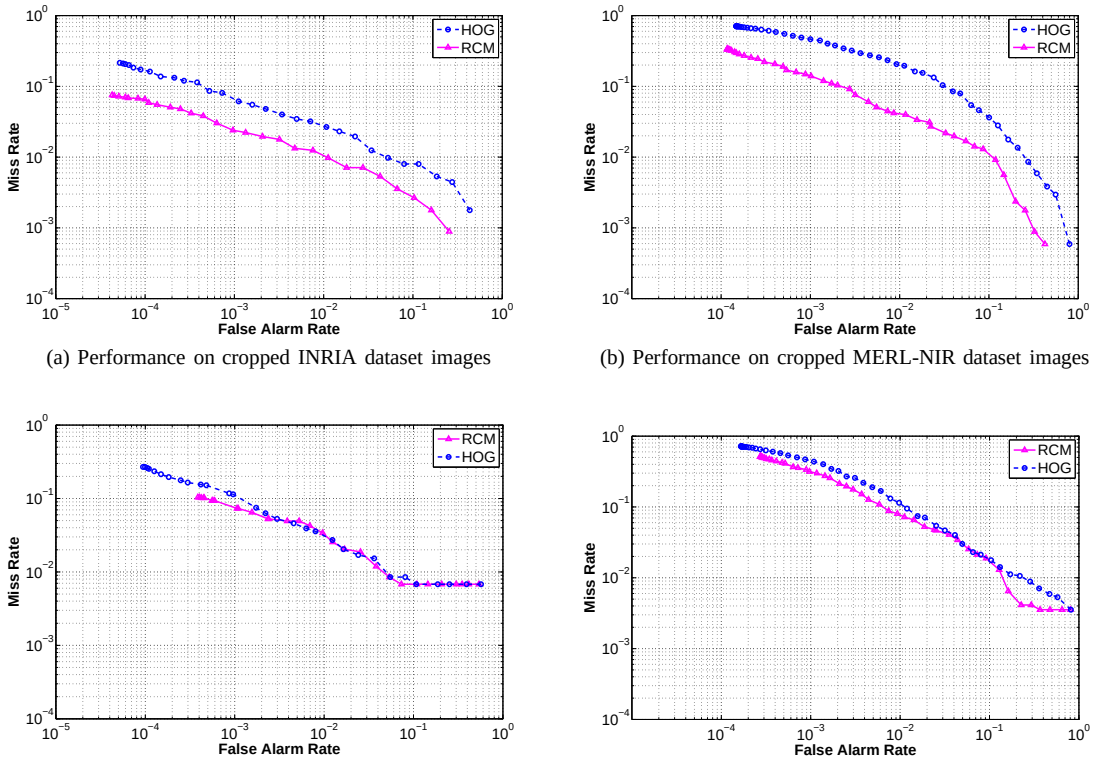
In comparing the two classifiers, RCM is almost always performing better than HOG on the two datasets and with the two methods of evaluation. That is consistent with the results reported in [12]. However, in the case of evaluation on whole images, HOG is able to reach levels of false alarm rate that are not reached by RCM. That comes at the cost of higher miss rate.

In comparing the two datasets, performance on the INRIA dataset is consistently better than performance on the MERL-NIR dataset for the two classifiers and for the two methods of evaluations. That is expected because of the difference between the two datasets. One important difference is the difference in size of training windows. A training window of size 24×48 is used for the MERL-NIR dataset while a training window of 64×128 is used for the INRIA dataset. That means in training for the MERL-NIR dataset, the classifier has less information about the object it trains on. Another important difference between the two datasets is image quality. The MERL-NIR dataset suffers from interlacing, motion blur, and poor focus. Another difference between the two datasets is the variability within the training dataset. The INRIA dataset has more variability in terms of number of people appearing in the dataset, and the places they appear.

To understand the difference between performance on cropped windows and whole images, we need to consider the difference between the miss rates and false alarm rates. For the false alarm rate, the negative windows used differ in the two evaluation methods. When evaluating on cropped images, we use cropped windows from background images that do not contain any human. When evaluating on whole images, we use images that contain humans and avoid the areas close to existing humans. Therefore, the results are not perfectly comparable in the case of false alarm rate. But, we can see from figure 3 that the false alarm rate is generally higher in the case of evaluation on whole images than cropped windows. The reason for that is probably lack of annotation of some human subjects. This occurs in some images of the INRIA dataset when there are many people in the background. It also occurs in the MERL-NIR dataset when people are partially occluded and not annotated as human subjects.

In the case of miss rate, there are two factors that explain the difference between the two methods. One factor is that for cropped images, the positive images are perfectly aligned around a human subject whose size is adjusted to make the relative size of the human subject inside the windows the same as the training size. On the other hand, in evaluation on whole images, we use fixed size windows and scan the image with a fixed step. Therefore, it is unlikely that we obtain windows that are perfectly aligned and resized as for cropped windows. That results in a higher miss rate in the evaluation on whole images. The second factor is, in the evaluation on cropped images, each annotated human subject has exactly two cropped positive windows, one in the original position and one that is flipped around its central vertical axis. Each of these two images are dealt with independently so that missing any of them counts towards the miss rate. On the other hand, in the case of evaluation on whole images, each human subject can have many windows surrounding it that are all evaluated. But they are not all dealt with independently. Missing all of them counts as one missed detection. But, detecting at least one of them counts as a true detection. Therefore, a single human subject has more chances to get detected than in the case of cropped windows. That results in lower miss rate in the case of evaluation on whole images. Hence, we have two conflicting factors: one can increase the miss rate and one can decrease it in each evaluation method. This probably explains the discrepancy in our results. In figure 3, we find that in general the miss rate is higher in evaluation on whole images. But, we can see that the HOG classifier has actually a lower miss rate when evaluated on whole images in the case of the MERL-NIR dataset.

For running times, on the MERL-NIR dataset, the RCM



(a) Performance on cropped INRIA dataset images

(b) Performance on cropped MERL-NIR dataset images

(c) Performance on whole INRIA dataset images

(d) Performance on whole MERL-NIR dataset images

Fig. 3. Detection performance of the two classifiers, using default parameters for each, on both INRIA and MERL-NIR datasets.

classifier takes around 22.14 seconds per frame while HOG takes around 2.16 seconds. For the INRIA dataset, RCM takes around 5.9 seconds per image, while HOG takes around 0.8 seconds. Therefore, RCM is around 10 times slower than HOG. The times are different from the INRIA dataset to the MERL-NIR dataset due to the larger number of false alarms in the MERL-NIR dataset, which means more scanning windows are examined by all layers of the cascade.

2) Effect of Resizing Features vs Resizing Images: As explained in section III-C, when we evaluate on whole images, we have one of two options: resizing images or resizing features.

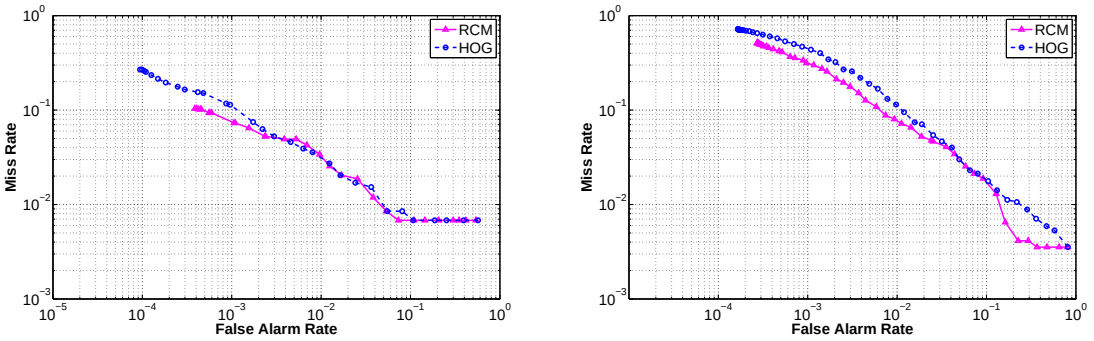
Figure 4 compares the performance of the two classifiers on the two datasets when resizing images versus resizing features. The top row plots show performance when resizing images. The bottom show performance when resizing features. It is clear that the miss rate is significantly higher for resizing features than for resizing images, for the two classifiers on the two datasets. That is understandable since when we resize the features, we use the classifier for feature sizes that differ from the sizes used during training. On the other hand, we can observe that the false alarm rate slightly decreases in the case of resizing features than in the case of resizing images.

The two classifiers have similar behavior whether we resize features or images. While RCM typically outperforms HOG, when resizing features, it became very close to or even worse than HOG. This is possibly due to the strong normalization used in computing HOG features, which can make the effect of changed feature size less severe.

For timing, with resizing features, RCM takes around 19.4 seconds per frame on the MERL-NIR dataset, and 4.43 seconds per image on the INRIA dataset. The HOG classifier takes around 1.87 seconds per frame on the MERL-NIR dataset, and 0.46 second per image on the INRIA dataset.

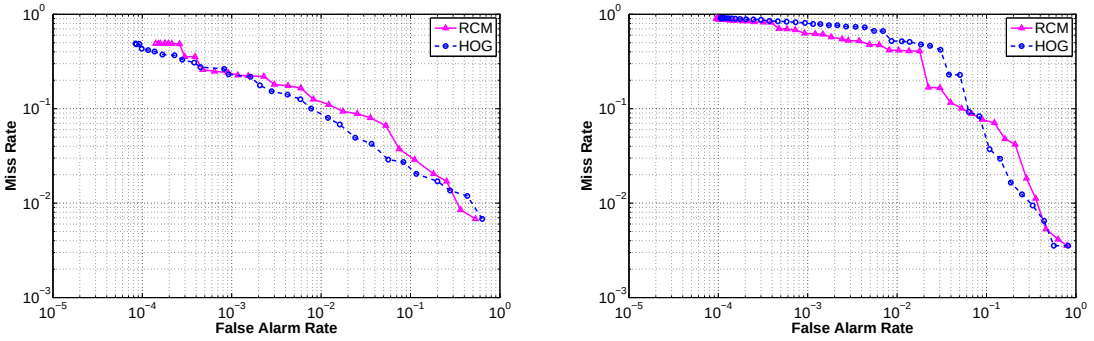
3) Effect of Smoothing: In the case of evaluation on whole images with resizing images, we found that whether to smooth images after resizing or not is an important factor. In all cases, we use bi-cubic interpolation to resize images, using the function provided with the OpenCV library [17]. When we use smoothing, we use a 3×3 Gaussian smoothing kernel, with standard deviation of 0.8. From our experiments, with results shown in figure 5, the effect of smoothing is not the same for the two datasets or the two classifiers.

In the case of the INRIA dataset, not smoothing increases the miss rate of the two classifiers. However, the RCM classifier is much more affected than HOG although when we use smoothing the miss rate of HOG is always higher. In the case of the MERL-NIR dataset, we make the same observation. The miss rate increases when smoothing is not used. But, in this case RCM has a lower miss rate than HOG even without smoothing. The differences between the two datasets are possibly the reason for the difference in classifiers' behavior. The INRIA dataset has detailed images with a wide range of sizes. When using bi-cubic interpolation



(a) Performance on whole INRIA dataset images with resizing images.

(b) Performance on whole MERL-NIR dataset images with resizing images.



(c) Performance on whole INRIA dataset images with resizing features.

(d) Performance on whole MERL-NIR dataset images with resizing features.

Fig. 4. Comparing resizing images versus resizing features.

without following it by smoothing, some artifacts emerge in the images, which affects the computed gradients and result in a higher miss rate. However, in the case of the MERL-NIR dataset, images are blurred and with poor focus, and the range of sizes is not as wide as the INRIA dataset. That reduces the artifacts resulting from resizing the images and hence reduces the benefit of smoothing. The reason why the effect of smoothing is not as severe in the case of HOG as it is in the case of RCM is probably the strong normalization in constructing HOG features.

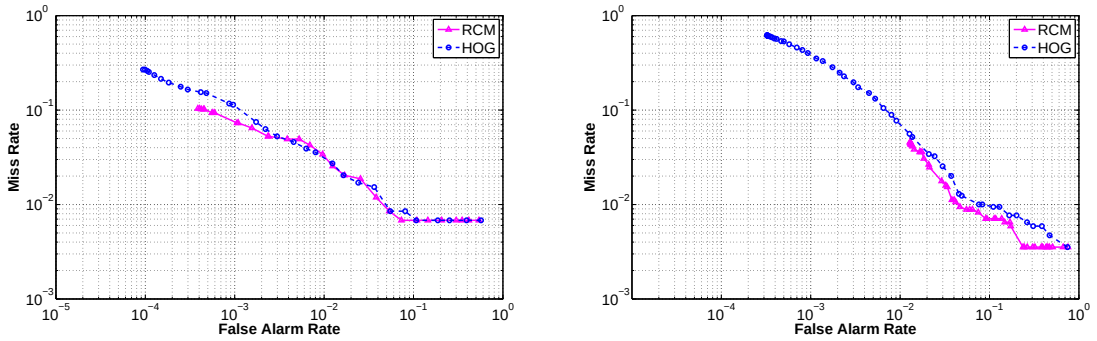
For the false alarm rate, smoothing results in increasing the false alarm rate. That is because lack of smoothing results in artifacts. These artifacts are likely to result in rejection of positive examples. Therefore, it is not unexpected to result in rejection of the negative examples as well. But, the increase in false alarm rate in the case of RCM is much more than in the case of HOG. That is probably due to the fact that smoothing tends to reduce the correlation between gradient magnitude of pixels in a region and their relative positions. To see that, consider the extreme case of smoothing when all the pixels become the same value. In this case there is no correlation between gradient magnitude and relative position. In the case of the MERL-NIR dataset, where the effect of smoothing in reducing artifacts is not really important, the effect of smoothing in reducing correlation between pixel positions and gradient magnitudes becomes much more visible. Hence, we get the results shown in figure 5b.

IV. Conclusion

We presented an evaluation of two state of the art pedestrian detectors. The goal of our study was to evaluate the detectors in practical scenarios. Our evaluation was conducted on two datasets with different imaging modalities, human size distribution, and other factors. We conducted our evaluation on whole images as well as cropped windows. We studied effects of two parameter settings: resize images vs. resizing features, and smoothing images after resizing. Our study is distinguished by its focus on effect of change in image modality, and on evaluation on whole images and the effect of different parameters associated with it.

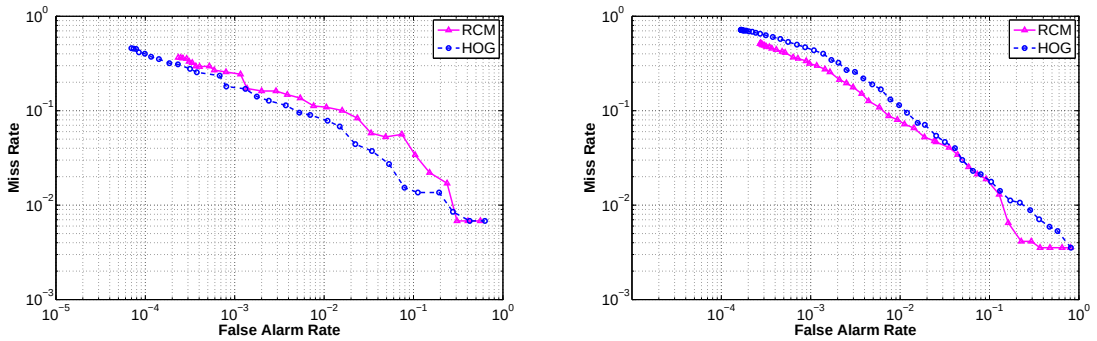
Although the practical application of pedestrian detection is on whole images, most evaluations focused on cropped images. Unfortunately, detection performance on cropped images is sometimes illusive and does not reveal the actual classification performance on practical applications. In our experiments, we observed up to one order of magnitude increase in false alarm rate and 25% increase in miss rate when evaluating on whole images instead of cropped windows

When evaluating on whole images, an image is typically scanned with multiple window sizes using a classifier that is trained on a single size. This is accomplished by either resizing features or resizing images. In our experiments, we found



(a) Whole INRIA dataset images **with** smoothing of images after resizing.

(b) Whole MERL-NIR dataset images **with** smoothing of images after resizing.



(c) Whole INRIA dataset images **without** smoothing of images after resizing.

(d) Whole MERL-NIR dataset images **without** smoothing of images after resizing.

Fig. 5. Effect of smoothing images after resizing them on detection performance.

that resizing feature can result in a significant deterioration in detection performance than resizing images. That suggests one of two techniques: either to always resize images so that scanning window size is the same as the classifier's training window size, or to train multiple classifiers for multiple sizes.

Implementation details make a difference in detection performance. In our experiments, we found that smoothing an image, after resizing, an important factor in detection performance, especially in the case of the RCM classifier, where features are more sensitive to resizing artifacts. In our experiments, for color images, smoothing was important to obtain the best performance. But, for near infrared images, smoothing resulted in worse performance.

REFERENCES

[1] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, New York, NY, vol. 1, 2003, pp. 734–741.

[2] P. Felzenszwalb and D. Huttenlocher, "Pictorial structures for object recognition," in *Intl. J. of Computer Vision*, vol. 61, no. 1, 2005.

[3] S. Ioffe and D. A. Forsyth, "Probabilistic methods for finding people," *Intl. J. of Computer Vision*, vol. 43, no. 1, pp. 45–68, 2001.

[4] R. Ronfard, C. Schmid, and B. Triggs, "Learning to parse pictures of people," in *Proc. European Conf. on Computer Vision*, Copehagen, Denmark, vol. 4, 2002, pp. 700–714.

[5] K. Mikolajczyk, B. Leibe, and B. Schiele, "Multiple object class detection with a generative model," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, New York, NY, vol. 1, 2006, pp. 26–36.

[6] A. Opelt, A. Pinz, and A. Zisserman, "Incremental learning of object detectors using a visual shape alphabet," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, New York, NY, vol. 1, 2006, pp. 3–10.

[7] D. Gavrila and V. Philomin, "Real-time object detection for smart vehicles," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Fort Collins, CO, 1999, pp. 87–93.

[8] P. Papageorgiou and T. Poggio, "A trainable system for object detection," *Intl. J. of Computer Vision*, vol. 38, no. 1, pp. 15–33, 2000.

[9] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-based object detection in images by components," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 4, pp. 349–360, 2001.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[11] Q. Zhu, S. Avidan, M.-C. Yeh, and K.-T. Cheng, "Fast human detection using a cascade of histograms of oriented gradients," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York, June 2006.

[12] O. Tuzel, F. Porikli, and P. Meer, "Human detection via classification on riemannian manifolds," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[13] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[14] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Annals of Statistics*, vol. 28, 2000.

[15] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *European Conference on Computer Vision (ECCV)*, 2006.

[16] F. Porikli, "Integral histogram: A fast way to extract histogram features," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[17] *Open Source Computer Vision Library*.