Implicit Shape Kernel for Discriminative Learning of the Hough Transform Detector

Yimeng Zhang
yz457@cornell.edu
Tsuhan Chen
tsuhan@ece.cornell.edu

School of Electronic and Computer Enginerring
Cornell University
Ithaca, NY, USA

Abstract

The Hough transform provides an efficient way to detect objects. Various methods have been proposed to achieve discriminative learning of the Hough transform, but they have usually focused on learning discriminative weights to the local features, which reflect whether the features are matched on the object or the background. In this paper, we propose a novel approach to put the *whole* Hough transform into a maximum margin framework, including both the weights for the local features and their locations. This is achieved through the kernel methods of the SVM. We propose a kernel that can be used to learn a SVM classifier that determines the presence of the object in a subimage. The kernel is designed such that during testing, the standard Hough transform process can be used to obtain the exact decision scores of the SVM at every location and scale of a test image. The experiment results show that our approach significantly improves the detection performance over previous methods of learning the Hough transform.

1 Introduction

The sliding window approach has been widely used for object detection because it provides a simple way to apply object recognition techniques to the detection task. A binary classifier is evaluated at every scale and location in an image to determine the presence of the object of interest in possible subwindows. The approach has been successful with various kinds of features and classifiers [3, 4, 6, 7, 10, 15]. Despite its effectiveness, though, the exhaustive search makes the approach inefficient in the case of a non-trivial classifier. The branch and bound techniques [16, 17] have been proposed in recent years to avoid the exhaustive search and still find the global optimal.

The Hough transform [2, 12, 13, 19, 22, 25] provides an alternative way to perform the detection task in a much more efficient manner. The Hough transform based detector has three main steps, as illustrated in Figure 1: 1) Each local patch in a test image is assigned to a codeword. 2) According to the spatial information of the codebook learned from the training images, each patch will cast weighted votes to the object locations and scales, and obtain the initial Hough image. 3) In order to tolerate shape deformation, kernel density estimation, such as Gaussian filtering in [13] or Mean-shift modes estimation in [14, 19], is applied to the Hough image. This process gives us the final Hough image, and the peaks in the Hough image are extracted as the detection hypotheses. Figure 1 also shows the three

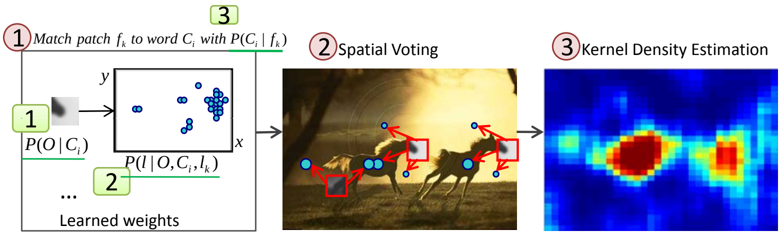

1 Match patch f_k to word C_i with $P(C_i \mid f_k)$
3
y
x
1
$P(O \mid C_i)$
$P(l \mid O, C_i, l_k)$
2
...
Learned weights
2 Spatial Voting
3 Kernel Density Estimation

Figure 1: The illustration for the Hough transform. The three steps for testing an image are indexed with red circles. The three weights we need to learn are indexed with green circles.

weights we need to learn. The implicit shape model [19] puts the Hough transform into a probabilistic formulation by learning the locations of the codewords (weight 2) based on their spatial distribution in the training images. The approach has been used successfully in detecting objects because of its flexibility in combining different parts of training examples and its efficiency in detection.

Despite the success of the implicit shape model, it has two drawbacks. First is its discrimination power. Usually a discriminatively learned classifier performs better than a generatively learned one. There have been several works these years that dealt with this issue. The max-margin Hough transform (M^2HT) [22] learns discriminative weights for the codewords in a maximum margin framework. The Hough forest [13] and discriminative generalized Hough transform (DGHT) [25] generate a discriminative codebook using the random forest methods. All of these methods have significantly improved the implicit shape model. However, they only make discrimination on the codewords (weight 1 or 3 in Figure 1), while the spatial weights are learned generatively as the spatial distribution of the codewords in positive training examples (weight 2). The second drawback is that it is difficult to interpret the scores in the final Hough image, especially after the kernel density estimation, so it is difficult to tell what function the learning process is optimizing. The principled implicit shape model [18] reasons the Hough transform in a sliding window manner. From this reasoning, it derives a generative model, which is similar to the implicit shape model, but avoids the kernel density estimation and has better interpretation of the Hough image.

In this paper, we propose a novel approach for learning the Hough transform. The approach puts the *whole* of the Hough transform into a maximum margin formulation by connecting the Hough transform with the SVM through the kernel methods. We design a kernel particularly for the Hough transform detector and call the kernel "Implicit Shape Kernel". During training, we use the kernel to train a SVM classifier, which determines the presence of the object of interest in a subwindow. During testing, we can follow the standard Hough transform process for the kernel calculations, and the final Hough image will provide the exact the output scores of the SVM at every location and scale.

2 Approach

2.1 Probabilistic Hough Transform

We start by reviewing the implicit shape model and some other works on the Hough transform based detection. Let f_k be a local patch extracted from the input image at location x_k.

Let x_k denote the position and scale of an object. By matching it to the codebook, we obtain a set of codewords C_i with probability $p(C_i|f_k,x_k)$. We usually generate codewords based only on the appearance of the local patch. Therefore, the confidence reduces to $p(C_i|f_k)$. For every C_i, we cast weighted votes to the locations and scales of object O. Let $V(O,x|f_k,x_k)$ denote the votes collected for a particular location x on the input image from the patch f_k.

$$V(O,x|f_k,x_k) = \sum_i p(O,x|C_i,x_k)p(C_i|f_k) = \sum_i p(O|C_i,x_k)p(x|O,C_i,x_k)p(C_i|f_k) \tag{1}$$

The first term is the weight specifying the confidence that the codeword C_i at location x_k matches the object as opposed to the background. The second term is the probabilistic Hough vote for the object position given a codeword and its location. This term is obtained based on the spatial distribution of each codeword on the positive examples. This distribution is usually obtained by storing all observations of the codeword, and then normalizing to one. Most studies for the Hough transform differ from each other in terms of how to learn the weights for the first [22] and third terms [13, 25] in order to discriminate the codewords from the object of interest and the codewords from the background.

The final score $S(O,x)$ for object O at location x is the summation of the votes collected from all local patches. In order to allow for small shape deformation, kernel density estimation is usually used to estimate $S(O,x)$.

$$S(O,x) = \sum_k \sum_j V(O,x_j|f_k,x_k) K_w(\frac{x-x_j}{b}) \tag{2}$$

we use K_w to denote the window function with bandwidth b to differentiate from our designed kernel which is denoted as K in later sections. A Gaussian function can be used for Gaussian filtering on the Hough image, and an Epanechnikov function can be used for a Mean-shift mode search.

2.2 Hough Transform as Kernel Calculation

We reformulate the Hough transform so as to fit it to the kernel classifier. The main idea is to write the Hough voting score for a location x as a weighted summation of a function K of the subimage at x and each training example. In this way, we can use the function K as a kernel to learn a SVM. To simplify the explanation, we consider only the voting for the location of the object and ignore the scale for now. We explain how to make multi-scale detection in Section 2.4.

For each codeword C_i, we assume that we have its learned weight at each location l_k relative to the object center, which we denote as $w_{l_k,i}$. This weight includes both the appearance and spatial weights of the codeword. For the implicit shape model, $w_{l_k,i}$ is equal to the product of the spatial distribution of the codeword C_i and $P(O|C_i)$. For codeword creation, we match each region to the nearest codeword, that is, $p(C_i|f_k) = 1$ when C_i is the nearest cluster for f_k, and otherwise 0. Our algorithm can be easily extended to the case that a patches matches multiple codewords.

The votes for location x_j collected from the patch f_k would be $w_{x_k-x_j,C(f_k)}$, where $x_k - x_j$ is the relative location when x_j is the center of the object, and $C(f_k)$ is the codeword assignment of patch f_k. Thus the score $S(O,x)$ of object O at location x from equation 2 can be written as:

$$S(O,x) = \sum_k \sum_j w_{x_k-x_j,C(f_k)} K_w(\frac{x-x_j}{b}) = \sum_i \sum_{k:C(f_k)=C_i} \sum_j w_{x_k-x_j,i} K_w(\frac{x-x_j}{b}) \tag{3}$$

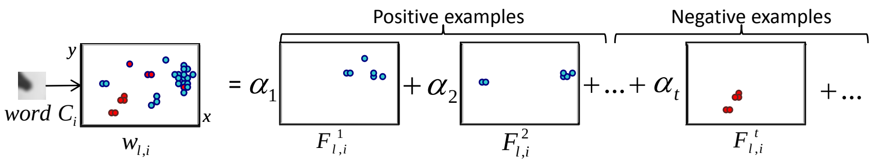


Figure 2: Illustration for the weights of a codeword as a weighted combination of the observations of training examples

We assume that $w_{l_k,i}$ is the weighted summation of the observations from the training examples as illustrated in Figure 2. Positive examples are the subimages inside the bounding boxes of objects. Negative examples are sampled from the training images. Let $F^t_{l_k,i}$ be the number of occurrence of codeword C_i at location l_k in training example I_t. Let α_t be the coefficient for training example I_t. The weight $w_{l_k,i}$ is calculated as $w_{l_k,i} = \sum_t \alpha_t F^t_{l_k,i}$.

For the implicit shape model, $\alpha_t = 1/|occ[i]|$ for all positive examples and 0 for all negative examples. Here, $|occ[i]|$ denotes the total number of occurrences of codeword C_i from all positive training examples. In our case, α_t will be learned through SVM, which will be explained later. Combining with equation 3, we can rewrite the score $S(O,x)$ as:

$$S(O,x) = \sum_t \alpha_t \sum_i \sum_{k:C(f_k)=C_i} \sum_j F^t_{x_k-x_j,i} K_w\left(\frac{x-x_j}{b}\right) \tag{4}$$

Next, we write all locations in the expression as opposed to the same object center x. We use a new variable $x_{k'}$ to replace x_j, and define $x_{k'}$ such that $x_{k'} - x = x_k - x_j$ as illustrated in Figure 3.

$$\begin{aligned} S(O,x) &= \sum_t \alpha_t \sum_i \sum_{k:C(f_k)=C_i} \sum_j F^t_{x_{k'}-x,i} K_w\left(\frac{x_k - x_{k'}}{b}\right) & (5) \\ &= \sum_t \alpha_t \Big[\sum_i \sum_{k:C(f_k)=C_i} \sum_{k':C(f^t_{k'})=C_i} K_w\left(\frac{x_k - x_{k'}}{b}\right)\Big] & (6) \end{aligned}$$

where $f^t_{k'}$ is the local region of training example I_t. Now we define the kernel function $K(I_x, I_t)$ between two the subimage at location x, I_x and the example I_t as:

$$K(I_x, I_t) = \sum_i \sum_{k:C(f_k)=C_i} \sum_{k':C(f_{k'})=C_i} K_w\left(\frac{x_k - x_{k'}}{b}\right) \tag{7}$$

The score $S(O,x)$ for the subimage centered at location x would be $S(O,x) = \sum \alpha_t K(I_x, I_t)$. This is exactly the expression of the decision function for subimage I_x of a SVM with kernel $K(I_x, I_t)$, where α_t is the coefficient for the t^{th} support vector.

2.3 Implicit Shape Kernel

We define a kernel as in equation 7 for the Hough transform and call this kernel the implicit shape kernel (ISK). The intuition of this kernel is illustrated in Figure 4. For each pair of regions from two examples, if they are matched to the same codeword, we calculate the similarity value of their locations relative to the object centers. The similarity value

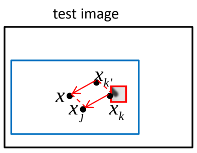


Figure 3: Rewrite locations relative to the center of a evaluating window x.

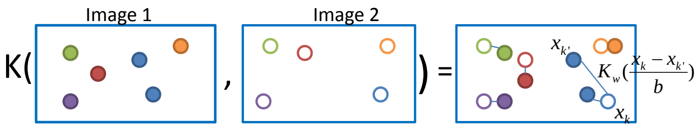


Figure 4: The illustration of the proposed kernel. The circles represent local patches. Different colors represent different codeword assignments.

is calculated using the window function in the kernel density estimation (Equation 2). The kernel value of the implicit shape kernel is the summation of the similarities of all such pairs. This kernel implicitly defines the deformation between the shapes of the two examples. If the two examples are the same image, the locations of the codewords will match exactly. Based on the property of the window function K_w, we get the highest similarity value for each pair of words. As the shape deforms, the value will decrease in a way defined by the window function.

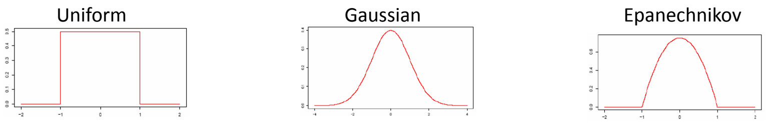
We consider multiple window functions to reflect the deformation of the shapes. Different functions also define different processes at the runtime of detection with the Hough transform. Figure 5 shows three types of functions: 1) The uniform function - The Hough transform spreads the votes uniformly to the neighbors at the third step (Fig. 1). This is also similar to giving a binned estimation for the spatial weights of the codewords (used in [22]). 2) The Gaussian function - This corresponds to performing a Gaussian filtering at the third step (used in [13]); 3) The Epanechnikov function - This corresponds to a standard Mean-shift local maximum search at the runtime (used in [14, 19]).

Discussion

The proposed kernel provides several good properties: 1) By using the proposed kernel to train a maximum margin classifier (i.e. SVM), we separate the scores of object and non-object in equation 3 in a maximum margin manner. Thus our method provides full discriminative learning of the Hough transform, including both appearances of the local patches and their spatial weights. The learning also takes the kernel density estimation into account. 2) During testing, we can apply the standard Hough transform process by recovering $w_{l_k,i}$ with the learned SVM, and use the same window function K_w as that for training. Therefore, the method benefits from the detection efficiency of the Hough transform. 3) Through the window function K_w, the implicit shape kernel avoids hard quantization of the image space when modeling the spatial information of the codewords, and therefore retains the flexibility of the implicit shape model [19]. 4) From the derivation in Section 2.2, it is clear that the scores in the final Hough image (after subtracting the bias term β in the SVM) give the exact decision scores of the classifier for every location of the object in a testing image. Therefore, we are directly optimizing the classification scores during training.

Mercer's Condition

Only the kernels that satisfy the Mercer's condition (positive semi-definite) guarantee a global optimal solution to kernel based algorithms based on convex optimization, including SVM. According to [21], for two local feature sets, $F_x = \{f_{x_1},, f_{x_{|F_x|}}\}$ and $F_y = \{f_{y_1},, f_{y_{|F_y|}}\}$ of two images I_x and I_y, the following kernel satisfies the Mercer's condition, if $K_w(f_{x_i}, f_{y_j})$



$$K_U(\mathbf{x}) = \begin{cases} c & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases} \qquad K_N(\mathbf{x}) = c \cdot \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \qquad K_E(\mathbf{x}) = \begin{cases} c\left(1-\|\mathbf{x}\|^2\right) & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$$

Figure 5: Different types of window functions for kernel density estimation

is a Mercer's kernel:

$$K(I_x, I_y) = \sum_{i=1}^{|F_x|} \sum_{j=1}^{|F_y|} K_w(f_{x_i}, f_{y_j}) \tag{8}$$

In our case, we represent each local feature as its location (x and y coordinates) on the image. Let F_x^i and F_y^i be the sets of local features from image I_x and I_y respectively matched to codeword C_i. Since the summation of multiple Mercer's kernels is still a Mercer's kernel, the kernel defined in equation 7 is a Mercer's kernel if the window function K_w is a Mercer's kernel.

We know that a Gaussian kernel is a Mercer's kernel. Therefore, the implicit shape kernel defined using the Gaussian function as the window function is a Mercer's kernel. Since neither the uniform function nor Epanechnikov function is a Mercer's kernel, the implicit shape kernel with these two functions does not guarantee a global optimal solution for the SVM learning. However, in practice, a non-Mercer's kernel usually reports promising empirical results [26]. This is also seen in our experiments.

2.4 Multi-scale Hough Transform

Training examples are normalized to the same width W and height H. To handle object detection at multiple scales, we resize the features of a test image and their locations by a set of scale factors $\hat{s}_1, \ldots, \hat{s}_S$. Specifically, the size s_f of each local feature f is resized to $\hat{s}_i s_f$ for the i^{th} scale factor, and the x, y coordinates of its location is resized to $\hat{s}_i x_f$ and $\hat{s}_i y_f$. After that, the Hough transform is performed for each resized image, and detection scores are obtained at every scale. The bounding box detected at scale s_i would have the size $(W/s_i, H/s_i)$. During voting, we match a pair of local features from two examples only when their scales are similar. That is, let the scale of a local feature f_i in image I_x be $s^x_{f_i}$, and a local feature f_j in image I_y be $s^y_{f_j}$. We calculate a vote only when $1/\theta < s^x_{f_i}/s^y_{f_j} < \theta$. This will give similar scores as the Hough voting on the 3D space (location and scale).

2.5 Local Normalization

Normalization of the feature histograms for the Hough transform based detectors is not as straightforward as a sliding window detector because of its additive voting during detection. In our experiment, we found that a non-normalized detector tends to give higher scores to larger-scale bounding boxes in multi-scale detection. To deal with this problem, we give a weight v_j to each local feature f_j, calculated as $v_j = 1/\sum_{k \in F} K_w(\frac{x_k - x_j}{b})$, where x_j denote the location of the local feature f_j , and F is the set of local features in the image. Intuitively, the weight of each local feature is normalized by the number of features around it. We use the same window function and bandwidth as for the kernel density estimation. Adding this weight, the implicit shape kernel of examples I_x and I_t is defined as:

$$K(I_x, I_t) = \sum_i \sum_{k:C(f_k)=C_i} \sum_{k':C(f_{k'})=C_i} v_k^x v_{k'}^t K_w\left(\frac{x_k - x_{k'}}{b}\right) \tag{9}$$

Obviously this is still a Mercer's kernel when the window function is positive semi-definite. This local normalization is also performed during the voting at testing time.

3 Experiment

We evaluate the proposed approach on Pascal VOC [9][8], INRIA horse [11] and UIUC car [1] datasets. We compare the detection performance favorably to other methods for learning the Hough transform. In other works, there is usually a verification step after the Hough transform with some non-linear classifier [22][12]. We would like to compare the pure Hough transform detectors learned with different methods; therefore, we do not perform the verification step in the experiments.

During training, negative examples are originally generated as false positives of the implicit shape model. An initial model is learned. Then we apply the model to the training images and add new false positives to the negative example set. The model is retrained using the augmented set to produce the final detector. For all datasets, we extract Harris-hessian regions [23] from gray scale images and use SIFT features [20] to represent the regions. The codebook is created by clustering the features from training images with K-means. We use $k = 2000$ for the Pascal dataset, $k = 500$ for the INRIA horse and UIUC car datasets. To remove multiple detections on the same object, we delete the bounding boxes that are at least 50% covered by another bounding box with a higher confidence score.

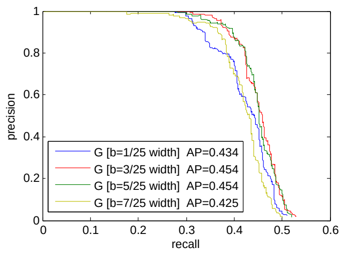
3.1 Pascal Datasets

For each category, we learn two models with different aspect ratios for each category by clustering the positive examples into two sets. Thus, we only need to vote for different locations and scales for each model during detection. We tune the parameter C for the SVM, and the bandwidth of the window function on the validation dataset for each category respectively.

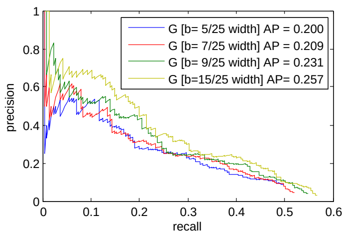
Bandwidth and different window function

We analyze the impact of different window functions and the bandwidth on the detection performance. Figure 6 shows the precision-recall curves on the verification data for the car and cat categories of Pascal 06 dataset. The choice of bandwidth affect the performance as in many methods with kernel density estimation [5]. For the car category, a relatively small bandwidth is preferred. However, when the bandwidth is too small, the performance is not good either, since we lose the generality of the model. The same effect is also observed for other categories with rigid shapes. On the other hand, for the cat category, whose shape is loose, a larger bandwidth leads to better detection performance. With the same bandwidth, the Gaussian function and the Epanechnikov function performs much better than the uniform function on the car category. For the cat category, the three functions lead to similar performance, with the Gaussian function slightly better than the others.

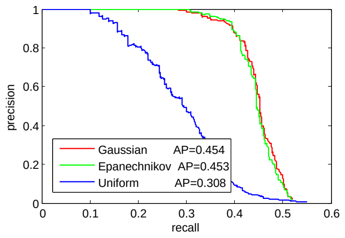
Comparison with other Hough transform methods



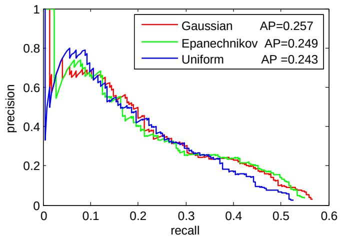
(a) car (Gaussian)



(b) cat (Gaussian)



(c) car (b=7/25width)



(d) cat (b=15/25width)

Figure 6: Precision-Recall curves on the Pascal 06 dataset. (a) and (b) are the detection results for car and cat categories with different bandwidth. The bandwidth is specified as the ratio of the width of the model. The Gaussian function is used here. (c) and (d) are the results with different window functions.

Table 1: Average Precisions (%) on Pascal VOC 2006 dataset using the Hough transform detectors learned with different methods

	bike	bus	car	cat	cow	dog	horse	mbike	person	sheep
ISM[19]	35.6	1.8	22.4	4.0	2.7	2.9	1.6	5.1	1.4	1.8
M^{2}HT[22]	43.9	7.7	24.3	12.6	16.0	3.2	1.8	11.9	5.3	12.2
ISM+norm	41.4	5.6	32.6	3.6	15.5	3.9	4.0	16.9	2.9	13.3
M^{2}HT+norm	45.6	18.2	25.3	13.0	16.5	6.4	12.7	21.1	11.6	14.9
Ours	**53.6**	**29.3**	**45.1**	**25.6**	**26.7**	**18.5**	**20.9**	**41.7**	**12.4**	**26.5**

Table 1 shows the average precisions for each category on the test data of Pascal 06 dataset. We implemented the implicit shape model (ISM) and the max-margin Hough transform (M^2HT) by ourselves. To make a fair comparison, we keep everything the same except the learning algorithm for each model. We also present the results of ISM and M2HT with local normalization with the method described in Section 2.5.

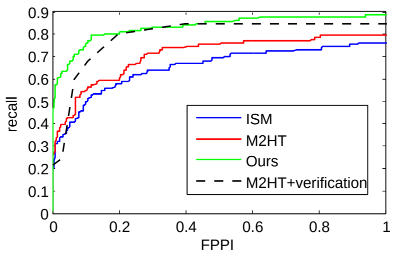
According to the table, making discriminative learning for the weights of codewords with M^2HT leads to higher AP scores than ISM. Performing local normalization consistently improves both ISM and M^2HT. Learning the Hough transform with our approach further outperforms M^2HT with local normalization on all the categories. This shows that making discriminative learning on the locations of the codewords significantly improves the detection performance of the Hough transform that is only learned with per-word discriminative weights.

Comparison with non Hough transform methods

It would be interesting to compare the Hough transform detector with some related sliding window methods. A SVM with the proposed implicit shape kernel (ISK) is in many ways similar to a linear SVM with the spatial pyramid kernel, except that the ISK does not do hard

Table 2: Comparison of Average Precision (%) on Pascal VOC 2007 dataset using our method and related non Hough transform methods

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow
Ours	**24.6**	**32.1**	5.0	**9.7**	**9.2**	23.3	**29.1**	11.3	9.1	**10.9**
Linear SPK[15]	16.9	21.2	4.9	4.8	7.3	**25.2**	28.4	6.9	**9.8**	10.3
Linear BoW[16]	15.2	15.7	**9.8**	1.6	0.1	18.6	12.0	**24.0**	0.7	6.1
	table	dog	horse	motor	person	plant	sheep	sofa	train	tv
Ours	8.1	**13.0**	**31.8**	**29.5**	**16.6**	6.1	7.3	11.8	**22.6**	21.9
Linear SPK[15]	6.7	6.9	30.5	26.6	13.1	**9.4**	**12.5**	12.1	17.0	**28.5**
Linear BoW[16]	**9.8**	16.2	3.4	20.8	11.7	0.2	4.6	**14.7**	11.0	5.4



(a) INRIA horse

Methods	EER
Ours	99.5%
ISM [19]	91.0%
ISM+verification [19]	97.5%
M^2HT+verification [22]	97.5%
Hough Forest [13]	98.5%
DGHT [25]	98.5%
Mutch & Lowe [24]	99.6%
ESS [16]	98.5%

(b) UIUC car

Figure 7: (a) Precison-recall curves for the INRIA horse dataset, we show the results of M^2HT implemented by ourselves, and the curve reported in [22] for the M^2HT with a non-linear SVM for verification. At 1.0 false positive per image, the recall of our approach is 89.20%, and recall for $M^2HT + verification$ is 85.27%. (b) The equal error rates on the UIUC car dataset with both the Hough transform methods, and non-Hough transform methods [24][16]

quantization on the image space. With a large bandwidth for the window function, the ISK is also similar to the bag of words model. On the Pascal 07 dataset, we compare the AP scores of the Hough transform learned with our method with the sliding window approach with linear spatial pyramid kernel [15] and the efficient subwindow search method with bag of words [16]. We outperform the linear SVM with SPK and bag of words on most of the categories. In [15], a non-linear SVM with χ^2 kernel which is much slower for evaluation is used to verify the candidates generated by the linear SVM, which produces better AP scores (also than our approach). It is noteworthy that the Hough transform detector is designed mainly for a fast detection. The same verification classifier in [15] can be used after the Hough transform but with more accurate candidates, which are extracted even faster than the linear SVM with SPK.

Computation time

On a server with 8 core 2.26GHz CPU, the detection on all scales takes 100ms to 200ms per image depending on different categories.

3.2 INRIA horse and UIUC car

To directly compare with the performance reported on the papers of other Hough transform methods, we did experiments on some of the datasets used by those papers. We use the same experiment setting with M^2HT [22] for the INRIA horse dataset, and the same experiment setting as [1] for the UIUC car dataset (single-scale). Figure 7 (a) shows the recall-fppi (false positive per image) curve comparing to the M^2HT implemented by ourselves and the curve reported in [22] for M^2HT with a non linear SVM for verification. The results verified that making spatial discrimination of the codewords improves the Hough transform detector.

4 Conclusion

We proposed an approach that enables maximum margin learning for both appearances and locations of the codewords for the Hough transform detector. We designed a kernel in the way that the Hough transform can be used to obtain the scores of the SVM classifier learned with this kernel. The Hough transform detector learned with our approach leads to better performance than the ones learned with previous methods. For the future work, we are interested in learning of the bandwidth in the window function automatically from training data. Moreover, we would like to explore other types of kernels that can be evaluated with the Hough transform.

References

[1] Shivani Agarwal, Aatif Awan, and Dan Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 26:1475–1490, 2004.

[2] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. People-tracking-by-detection and people-detection-by-tracking. In *CVPR*, 2008.

[3] Matthew B. Blaschko and Christoph H. Lampert. Learning to localize objects with structured output regression. In *ECCV*, 2008.

[4] Ondrej Chum and Andrew Zisserman. An exemplar model for learning object classes. In *CVPR*, 2007.

[5] Dorin Comaniciu, Peter Meer, and Senior Member. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[7] Chaitanya Desai, Deva Ramanan, and Charless Fowlkes. Discriminative models for multi-class object layout. In *ICCV*, 2009.

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, .

[9] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf, .

[10] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[11] Vittorio Ferrari, Loic Fevrier, Frederic Jurie, and Cordelia Schmid. Groups of adjacent contour segments for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):36–51, 2008.

[12] Mario Fritz, Bastian Leibe, Barbara Caputo, and Bernt Schiele. Integrating representative and discriminant models for object category detection. In *ICCV*, 2005.

[13] Juergen Gall and Victor Lempitsky. Class-specific hough forests for object detection. In *CVPR*, 2009.

[14] Chunhui Gu, Joseph J. Lim, Pablo Arbelaez, and Jitendra Malik. Recognition using regions. In *CVPR*, 2009.

[15] Hedi Harzallah, Frederic Jurie, and Cordelia Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009.

[16] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, 2008.

[17] Alain Lehmann, Bastian Leibe, and Luc van Gool. Feature-centric efficient subwindow search. In *ICCV*, 2009.

[18] Alain Lehmann, Bastian Leibe, and Luc van Gool. Prism: Principled implicit shape model. In *BMVC*, 2009.

[19] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77 (1-3):259–289, 2008.

[20] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[21] Siwei Lyu. Mercer kernels for object recognition with local features. In *CVPR*, 2005.

[22] Subhransu Maji and Jitendra Malik. Object detection using a max-margin hough transform. In *CVPR*, 2009.

[23] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10): 1615–1630, 2005.

[24] Jim Mutch and David G. Lowe. Multiclass object recognition with sparse, localized features. In *CVPR*, 2006.

[25] Ryuzo Okada. Discriminative generalized hough transform for object detection. In *ICCV*, 2009.

[26] Cheng Soon Ong, Stephane Canu, and Alexander J. Smola. Learning with non-positive kernels. In *ICML*, 2004.