Discriminative Hough Forests for Object Detection

Paul Wohlhart
wohlhart@icg.tugraz.at
Samuel Schulter
schulter@icg.tugraz.at
Martin Köstinger
koestinger@icg.tugraz.at
Peter M. Roth
pmroth@icg.tugraz.at
Horst Bischof
bischof@icg.tugraz.at

Institute for Computer Graphics and Vision
Graz University of Technology
Austria

Abstract

Object detection models based on the Implicit Shape Model (ISM) use small, local parts that vote for object centers in images. Since these parts vote completely independently from each other, this often leads to false-positive detections due to random constellations of parts. Thus, we introduce a verification step, which considers the activations of all voting elements that contribute to a detection. The levels of activation of each voting element in the ISM form a new description vector for an object hypothesis, which can be examined in order to discriminate between correct and incorrect detections. We learn classifiers to discriminate correct and wrong part constellations and thus assign a better confidence to each detection using linear models as well as a histogram intersection kernel. Additionally, we show how to use the discriminative weights of the linear classifier, not only as a post processing step, but directly in the voting process. We evaluate the proposed approach on different datasets, showing considerable improvements of the detection results.

1 Introduction

Detecting objects of arbitrary categories in cluttered scenes is one of the main challenges in computer vision. In general, most methods in this field learn appearance and spatial relation models of the categories from labeled training images and use the obtained models to localize previously unseen instances in test images. Currently, two different approaches can be distinguished, namely sliding window and part based methods. Sliding window based methods like [11, 20] evaluate classifiers at all possible image locations, analyzing descriptors with fixed layouts like the histogram-of-gradients [4]. Although these approaches have shown to provide excellent results for rigid objects with fixed geometric properties, they yield limited performance for deformable objects.

http://dx.doi.org/10.5244/C.26.40

Thus, recently part based models have become more popular. They mainly differ in the way the spatial relations between the individual parts are defined, ranging from complex models, where each part is connected to all other parts (*e.g.*, [7]), to models without any spatial relation (bag-of-words model [17]). Recently, tree shaped models have become popular for part-based recognition, due to the highly efficient inference procedures for tree structures. The deformable part model [6], one of the most successful algorithms on the PASCAL Visual Object Class (VOC) challenge, is based on a tree shaped spatial model. One specific subtype are star shaped models, where each part is only connected to a centroid part. This is the underlying spatial representation of the *Implicit Shape Model* (ISM) [13], which constituted the basis for several extensions in the following years (*e.g.*, [12]). The ISM represents objects as a collection of a potentially large number of prototype patches, with specific appearance and a defined (but slightly varying) location relative to the object center.

More recently, Hough Forests have been presented in [8] as variant of an ISM and inspired a series of applications and extensions (*e.g.*, [5, 9, 10, 16]). In contrast to previously presented ISMs, at test time it extracts local features densely instead of just on interest points, resulting in the aggregation of more evidence and thus increased robustness. The fast identification of local object parts is made possible by the use of Random Forests [3].

One important question is how to fuse the evidence from the local parts into a joint object hypothesis score. Some authors tried to develop a fully probabilistic model for the process of object detection with an ISM such as in [7]. However, to handle the large amounts of local features that contribute to an object detection, it is necessary to factorize the joint probabilities and estimate conditional probabilities for each local feature independently. This independence assumption is questionable, since, *e.g.*, heavily overlapping input patches (around neighboring pixel in the input image) are not independent. Another problem are the weak statistics collected for the local features which are not reliable enough for use in a truly probabilistic and thus multiplicative framework. For instance the model might assign a local patch zero probability of being part on an object, and thus a multiplicative framework would not allow any detection containing that patch, although it might just be slightly occluded. Therefore, it is necessary to develop and fine-tune heuristics to account for occlusions or simple errors in the classification process of local patches.

Thus, most methods such as [12, 13, 21] rely on additive combination of the evidence (either in a generalized Hough voting procedure or using density estimation such as mean-shift or Gaussian mixture models). For more details on the robustness of additive classifier fusion versus multiplicative, we would like to refer to [12].

In this paper, we investigate the additive fusion of evidence in the generalized Hough voting process and the constellation of local object parts that jointly vote for an object. We formulate the contribution of each individual voting element to the score of an object hypothesis as a new descriptor for the hypothesis. These descriptors can be used in a discriminative classification framework in order to distinguish correct from wrong constellations. Additionally, a linear classification model allows us to use the discriminative weights learned during training directly in the voting process. The experimental evaluations demonstrate significantly improved results for both approaches.

2 Related Work

In [18] a formulation of ISM was presented, with a generative codebook of features extracted on image contours. A max-margin formulation was introduced to learn discrimina-

tive weights for each voting element. For the calculation of the activation of each codebook entry the spatial configuration was considered. However, weights were learned only per codebook entry, disregarding the relative position on the object. Additionally, the weights were constrained to be positive.

The Principled Implicit Shape model (PRISM) [12] introduces a weighting scheme for individual words. It was argued that the weighting could be defined by an arbitrary function, which would thus also allow for discriminative training. However, the specific model employed for the voting then defines the weights by a Gaussian mixtures model, which again only casts positive votes. The flexibility in choosing a weighting function is only exploited by globally scaling each mixture model, such that the maximum vote over the space is set to the probability of the corresponding visual word to belong to the foreground.

Another difference of our approach to both of the above mentioned methods is that Hough Forests do not only assign a test patch to different generative codebook entries, but, as we will explain in more detail in Section 3.1, estimate a similarity to each individual training patch. We explicitly make use of this fact by calculating a feature vector expressing the contribution of each single training sample to a detection and thus giving the classifier more fine grained control.

In [15] the activation of individual voting vectors by a detection hypothesis was determined. A Support Intersection Kernel was introduced to compare such descriptors. As we will show in Section 3.2, this definition is suitable for finding the most similar training example, but not to discriminate between correct and false detections.

3 Method

Before we introduce our novel discriminative weighting, we first briefly review the Hough Forests (HFs) of Gall and Lempitsky [8], which our method builds on; where applicable, we use the notation of [8]. Hough Forests are based on the Implicit Shape Model [13] and thus model an object as a collection of a large number of local features (image patches) $\mathcal{P}_i = \{\mathcal{I}_i, \mathbf{d}_i, y_i\}$. $\mathcal{I}_i$ is the appearance description of the patch (several feature channels calculated from the raw input pixel patch). The label of the patch, given by y_i, specifies whether a patch was extracted from a positive or negative training image. Each of the positive local patches additionally stores an offset vector $\mathbf{d}_i$ pointing to the center of the object. In contrast to ISM, which learns a flat, generative codebook of local patches, Hough Forests build an ensemble of hierarchical codebook structures $\mathcal{F} = \{\mathcal{T}_t\}_{t=1}^{T}$, where $\mathcal{T}_t$ is a randomized decision tree [3], and T is the number of trees. To train the codebook all patches (positives as well as negatives) are used.

The goal of Hough Forest training is to create a classifier that reliably estimates whether an image patch was extracted from an object of the target category or from the background. At the same time, for object patches also their location on the object should be estimated (or vice versa, where the center of the object is located relatively to the patch).

Thus, when building a Hough Forest each node either optimizes the class impurity or the offset impurity. For the first case, the information gain

$$\Delta H = -\frac{|I_l|}{|I_l|+|I_r|}H(I_l) - \frac{|I_r|}{|I_l|+|I_r|}H(I_r) \tag{1}$$

is used to find the most promising split for the local patches available in the current node. Here, I_l and I_r denote the sets of patches which will be sent to the left and the right child

node. $H(\cdot)$ is the negative entropy $-\sum_{k=1}^{K} p_k \cdot \log(p_k)$, where p_k is the probability of the current node to belong to class k, estimated from the ratio of positive to negative samples.

For optimizing the offset impurity only the positive local patches are considered (negative patches do not have offset vectors), and splits are searched such that the variance of the offset vectors in the child nodes is minimized:

$$\min \sum_{I_l} ||\mathbf{d}_i^l - \bar{\mathbf{d}}^l||^2 + \sum_{I_r} ||\mathbf{d}_i^r - \bar{\mathbf{d}}^r||^2 \ , \tag{2}$$

where $\bar{\mathbf{d}}^l$ and $\bar{\mathbf{d}}^r$ are the means of all offset vectors $\mathbf{d}_i$ falling into the left and right child nodes, respectively. Since each node in the HF is randomly chosen to optimize one of the two criteria, the leaf nodes store relevant statistics for both, appearance and offset vectors. All trees are trained until a maximum depth is reached or the leaf nodes are pure (to a level above a threshold).

During testing, for each test image $\mathcal{I}$, small local patches $\mathcal{P}(\mathbf{y})$ with appearance $\mathcal{I}(\mathbf{y})$ are densely extracted at all locations $\mathbf{y}$ and mapped onto the codebook entries, *i.e.*, the leaf nodes. Each leaf L of tree $\mathcal{T}_t$ stores the set of offset vectors D_L of all the training patches that fell into that node. In the following, $D_t(\mathbf{y})$ denotes the set of offset vectors stored in the leaf node of tree $\mathcal{T}_t$ that the test patch $\mathcal{P}(\mathbf{y})$ reaches, and $C_t(\mathbf{y})$ is the ratio of positive to negative training samples that reached this leaf node during training (*i.e.*, the probability foreground for the test patch at $\mathbf{y}$, according to tree $\mathcal{T}_t$).

Following the generalized Hough transformation procedure, each local patch in the test image casts votes for an object center. These votes are accumulated in the Hough space and local maxima indicate prospective object centers.

3.1 Activation Vector

Considering only one input location $\mathbf{y}$ and one decision tree $\mathcal{T}_t$ the score of an object hypothesis at location $\mathbf{x}$ is defined as (compare Equation (6) in [8])

$$S(\mathbf{x}|\mathbf{y};\mathcal{T}_t) = \left[\frac{1}{|D_t(\mathbf{y})|} \sum_{\mathbf{d}\in D_t(\mathbf{y})} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{||(\mathbf{y}-\mathbf{x})-\mathbf{d}||^2}{2\sigma^2} \right) \right] \cdot C_t(\mathbf{y}) \ . \tag{3}$$

By setting $w_t(\mathbf{y}) = \frac{1}{2\pi\sigma^2} \frac{C_t(\mathbf{y})}{|D_t(\mathbf{y})|}$, this can be simplified to

$$S(\mathbf{x}|\mathbf{y};\mathcal{T}_t) = w_t(\mathbf{y}) \sum_{\mathbf{d}\in D_t(\mathbf{y})} \exp\left(-\frac{||(\mathbf{y}-\mathbf{x})-\mathbf{d}||^2}{2\sigma^2} \right) \ . \tag{4}$$

To further simplify the notation later on, instead of summing over all offset vectors in the set stored with one leaf node, we can just sum over all training patches and check if the according offset vector is contained in that set:

$$S(\mathbf{x}|\mathbf{y};\mathcal{T}_t) = w_t(\mathbf{y}) \sum_{i=1}^{N} [\mathbf{d}_i \in D_t(\mathbf{y})] \exp\left(-\frac{||(\mathbf{y}-\mathbf{x})-\mathbf{d}_i||^2}{2\sigma^2} \right) \ , \tag{5}$$

where $[\cdot]$ is the Iverson bracket (*i.e.*, the expression is 1 if vector $\mathbf{d}_i$ is in the set of vectors $D_t(\mathbf{y})$ and 0 otherwise).

Integrating the information of all decision trees $\{\mathcal{T}_t\}_{t=1}^{T}$ and of all patch locations $\mathbf{y}$ in the test image gives the total score for an object hypothesis at location $\mathbf{x}$:

$$S(\mathbf{x}) = \sum_{\mathbf{y}} \frac{1}{T} \sum_{t=1}^{T} S(\mathbf{x}|\mathbf{y};\mathcal{T}_t) \ . \tag{6}$$

By combining Equations (5) and (6) and rearranging the sums, we can determine the contribution of a single offset vector $\mathbf{d}_i$ of the training patch $\mathcal{P}_i$ to this total score:

$$a_i(\mathbf{x}) = \frac{1}{T} \sum_{\mathbf{y}} \sum_{t=1}^{T} [\mathbf{d}_i \in D_t(\mathbf{y})]\, w_t(\mathbf{y}) \exp\left(-\frac{||(\mathbf{y}-\mathbf{x}) - \mathbf{d}_i||^2}{2\sigma^2} \right) \ . \tag{7}$$

We will refer to $a_i(\mathbf{x})$ as *activation* of the offset vector $\mathbf{d}_i$ for the hypothesis $\mathbf{x}$. The total *activation vector* recording the activations of all offset vectors for hypothesis $\mathbf{x}$ is then given by $\mathbf{A}(\mathbf{x}) = [a_1(\mathbf{x}), \ldots, a_N(\mathbf{x})]^T$. We additionally introduce the short notations $\mathbf{A}_j = \mathbf{A}(\mathbf{x}_j)$ and $a_{j,i} = a_i(\mathbf{x}_j)$.

Thus, the score $S(\mathbf{x})$ of a hypothesis $\mathbf{x}$ can be expressed in terms of the activations of individual training patch offset vectors:

$$S(\mathbf{x}) = \sum_{i=1}^{N} a_i(\mathbf{x}) \ . \tag{8}$$

This formulation, in which each of the offset vectors from training is treated independently, also allows for a different interpretation of Hough Forests in the context of ISMs. As stated above, in an ISM local features of the test image are mapped to codebook entries; each of them either to only one codebook entry or via a weighting to a sparse set of entries (soft assignment, *e.g.*, [18]). Along with each codebook entry (generative prototype) information of all associated training samples is summarized and then used for the voting. In a Hough Forest, a test patch is assigned a codebook entry (leaf node) by each tree. This way the RF can be seen as uniform soft assignment to a fixed number of entries from individual sub-codebooks (one per tree). However, the leaf nodes of all trees are just different partitions of the same set of all training patches. Thus, by counting in how many of the trees the test patch ends up in the same leaf node as a training patch, the Random Forest also gives an estimate of the similarity to each individual training patch, not only to the collection of training patches forming a codebook entry. This is explicitly encoded by the activation of one offset vector by one test input patch at $\mathbf{y}$ relative to the object hypothesis $\mathbf{x}$:

$$a_i(\mathbf{x},\mathbf{y}) = \frac{1}{T} \sum_{t=1}^{T} [\mathbf{d}_i \in D_t(\mathbf{y})]\, w_t(\mathbf{y}) \exp\left(-\frac{||(\mathbf{y}-\mathbf{x}) - \mathbf{d}_i||^2}{2\sigma^2} \right) \ . \tag{9}$$

3.2 Classifying valid object constellations

The activation vector can be seen as a descriptor of an object hypothesis. We can collect descriptors of positive and negative examples in order to train classifiers to discriminate between correct and incorrect detections.

In [15] a *support intersection kernel* was introduced. This way, the closest training sample for a given detection can be identified, which can be used, *e.g.*, to transfer meta data such as the object's pose from the annotated training sample to the detection. However, this

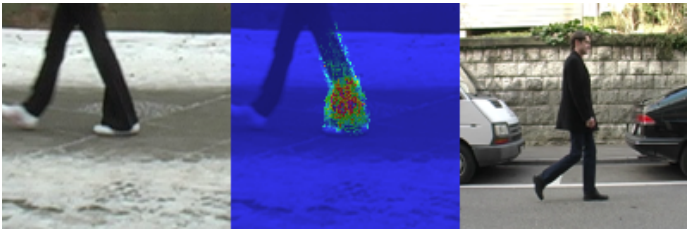

Figure 1: Matching the activations (middle) of a low scoring false positive detection (left) using the support intersection kernel [15]. Since the small area captured by the back projection is a perfect match, the distance to the training sample on the right is smaller than many distances between true positive detections and their respective closest training sample.

measure is not suitable to discriminate between correct and incorrect constellations, since it is specifically designed to also match detections with only a few active voting elements in order to handle strong occlusions. Thus, also for low scoring false positive detections often perfect matches are found, as illustrated in Figure 1.

We therefore propose to use a histogram intersection kernel. Given two activations $\mathbf{A}_1$ and $\mathbf{A}_2$, it is defined as

$$K(\mathbf{A}_1,\mathbf{A}_2) = \frac{\sum_i \min\{a_{1,i},a_{2,i}\}}{\max\{\sum_i a_{1,i},\sum_i a_{2,i}\}} \quad . \tag{10}$$

In the literature (*e.g.*, [19]) intersection kernels are often defined without the normalizing sum in the denominator of Equation (10). Thus, either the range of the values in the histogram vectors has to be limited intrinsically to reasonable values, or the individual histograms have to be normalized (usually L_1 normalization). Otherwise no meaningful results can be obtained.

In our case, we avoid normalizing each activation vector individually. In fact, a normalization would increase the noise in low scoring detection hypotheses, with very low activations of only a few offset vectors, and thus lead to similar problems as with the support intersection kernel. On the other hand, without any normalization no activation vector would ever be considered to be similar to a low scoring true positive example. Normalization with the sum over the larger of the two activation vectors returns similarity values in the well defined range of $[0,1]$, a similarity value of close to 1 also for similar low scoring true positives, and low similarity for activations of the same distribution but very different magnitude.

In our experiments we explore training of Support Vector Machines on histogram intersection kernels as well as linear SVMs to discriminate between valid and invalid object constellations. We collect a set $\{(A_j,l_j)\}_{j=1}^{M}$ of activation vectors $\mathbf{A}_j = \mathbf{A}(\mathbf{x}_j)$ from locations $\mathbf{x}_j$ in the set of training images and assign them a label $l_j = 1$ if there is an object in the ground truth at $\mathbf{x}_j$ and $l_j = -1$ for locations $\mathbf{x_j}$ in the background or on negative training images. The output of the classifier learned on this training set then defines the new score of the detection hypothesis.

3.3 Using discriminative weights in the voting process

In the last sections we have shown how to form activation vectors and learn classifiers in order to improve the total score of a hypothesis. To get the activation vector $\mathbf{A}(\mathbf{x})$ for one

object hypothesis location $\mathbf{x}$ we need to sum over all patch locations $\mathbf{y}$. Repeating this for each location $\mathbf{x}$ in the input image would be very inefficient. Therefore, the Hough Forests algorithm, as presented in [8], calculates the score for all hypotheses jointly in one run over all input patches. Each patch is passed down the trees and for each offset vector in the resulting leaf nodes its weight is added to exactly the relative hypothesis it points to (*i.e.*, $\mathbf{y}-\mathbf{d}$). The full Hough map is calculated by filtering the result once with a Gaussian.

The fact that each offset vector contributes individually to the total score gives us the possibility to, instead of just calculating an unweighted sum, introduce discriminative weights:

$$\hat{S}(\mathbf{x}) = \sum_{i=1}^{N} w_i a_i(\mathbf{x}) = \mathbf{w}^T \mathbf{A}(\mathbf{x}) \ . \tag{11}$$

Learning these weights from training data in a max-margin setup in order to optimize the final detection scores leads exactly to the linear SVM formulation as proposed in Section 3.2. However, as confirmed by experiments, to obtain good classification performance, we cannot train directly on A_j, but need to normalize each dimension of the data to zero mean and unit variance. Thus, the weight vector resulting from the SVM training (which is denoted as $\tilde{\mathbf{w}}$) cannot directly be used in Equation (11) but leads to a score defined as

$$S_{\mathrm{SVM}}(\mathbf{x}) = \sum_{i=1}^{N} \tilde{w}_i \frac{a_i(\mathbf{x}) - m_i}{s_i} \ , \tag{12}$$

where m_i and s_i are the mean and standard deviation of the activation of each vector over the training data. In order to use the discriminative weights learned in the last section directly in the voting process, such that the Hough map reflects the final score, we need to associate a weight with each individual offset vector $\mathbf{d}_i$. Looking at Equation (12) we see that

$$S_{\mathrm{SVM}}(\mathbf{x}) = \sum_{i=1}^{N} \tilde{w}_i \frac{a_i(\mathbf{x}) - m_i}{s_i} = \sum_{i=1}^{N} \frac{\tilde{w}_i}{s_i} a_i(\mathbf{x}) - \sum_{i=1}^{N} \frac{\tilde{w}_i}{s_i} m_i \ . \tag{13}$$

The last term is a constant offset that can be added after the voting. Thus, we can set $w_i = \frac{\tilde{w}_i}{s_i}$ to receive the discriminative weights as defined in Equation (11). Using the definition of $a_i(\mathbf{x})$ from Equation (7) and rearranging the terms, we see that

$$\hat{S}(\mathbf{x}) = \sum_{i=1}^{N} w_i a_i(\mathbf{x}) = \sum_{\mathbf{y}} \sum_{t=1}^{T} \sum_{\mathbf{d}_i \in D_t(\mathbf{y})} w_i w_t(\mathbf{y}) \exp\left(-\frac{||(\mathbf{y}-\mathbf{x}) - \mathbf{d}_i||^2}{2\sigma^2} \right) \ . \tag{14}$$

Thus, the final score map can be obtained during the voting process by adding a weight of $w_i w_t(\mathbf{y})$ to the hypothesis at $\mathbf{x} = \mathbf{y} - \mathbf{d}_i$ for each activation of $\mathbf{d}_i$ by a patch at $\mathbf{y}$ and Gaussian filtering of the result.

4 Experiments

We demonstrate the performance of our proposed method on two different object detection data sets, namely *TUD-pedestrian* [1] and *ETHZ-cars* [14]. Both contain quite challenging images, where objects are captured under different poses and lighting conditions and also show some occlusions. In our experiments, we want to show the relative improvement of our approach compared to the standard Hough Forest [8] without the proposed discriminative weighting of the voting vectors.

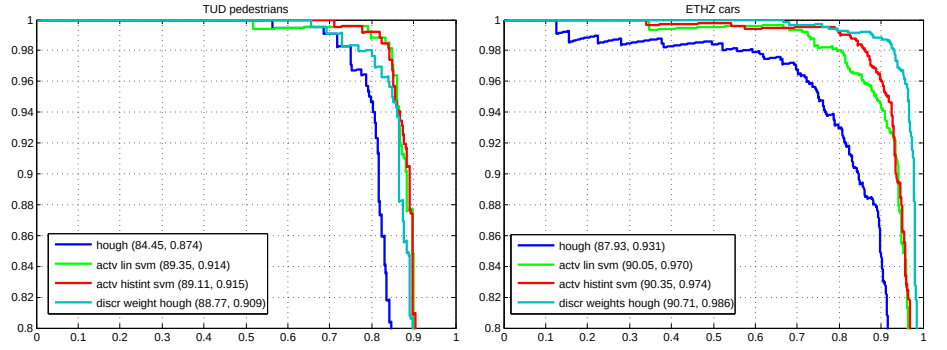

Figure 2: Precision/Recall curves for *TUD pedestrians* and *ETHZ cars* dataset. The curves represent the performance of standard Hough Forests [8] (*hough*), linear SVM on the activation vector (*actv lin svm*), histogram intersection kernel SVM on the activation vector (*actv histint svm*), and Hough voting with learned discriminative weights (*discr weights hough*).

Training In all experiments we use our own implementation of Hough Forests[8], using the same patch representation and split tests. We train 15 trees, with a maximal depth of 15. We stop early if the number of samples in a node is below 20, but keep splitting nodes that only contain positive patches on the offset vector variance criterion.

In order to collect activation vectors of positive and negative examples, we run the standard Hough Forest with uniform weights. The evaluation of the resulting detection hypotheses gives the label of the activation vector. For the evaluation of the performance of the Hough Forest we use the PASCAL overlap criterion with a threshold of 0.5. For the labeling of the activation vectors however, we do not take the most confident detection with acceptable overlap as positive, but the one with biggest overlap, regardless of its confidence.

As in [8], for handling of different scales, the feature extraction and voting is run on a series of scaled versions of the input and the resulting Hough maps are stacked into one 3D Hough space on which local maxima detection and non-maxima suppression is performed. Due to the interpolation, the resulting detection might lie between two of the tested scales. Thus, to collect the activation vectors we resize the input image to the respective scale and rerun the voting procedure, now only recording activations for the target location.

As with all max-margin classification systems, proper bootstrapping is crucial to obtain good test performance. Thus, we run several rounds of bootstrapping on a validation set during learning of the SVMs, as well as another round after we first use a preliminary version of the discriminative weights in the voting process.

Bounding Box Estimation from Back-projection The standard Hough Forest [8] delivers an estimate for the position and scale of objects in a test image. The aspect ratio of the reported detections is the mean aspect ratio of the bounding boxes from the training. If the aspect ratio varies heavily (such as for front-/back- vs. side-views of cars) this parameter has to be estimated as well. In this work, similarly to [15], we estimate the bounding box of the detection by determining where votes for the object came from (the back-projection, which we have readily available from the activation vector). To increase the robustness to noisy votes from the background, we adaptively set the minimum contribution of a pixel to 5% of the total score. The bounding box tightly captures everything above this threshold.

Figure 3: Hough maps for example test images (left) from the TUD pedestrian dataset, with discriminative weights (middle) and standard uniform weights (right). Note how in the case of discriminative voting correct detection peaks are clearly pronounced while background noise is significantly reduced.

Datasets: The *TUD-pedestrian* dataset consists of 400 training images and 250 test images containing 311 pedestrians on typical street side scenes. As a validation set, we use the related *TUD-crossing* and *TUD-campus* data sets, which consist of 201 and 71 images, respectively. The *ETHZ-cars* dataset contains images capturing cars under 7 different viewpoints. For each viewpoint, we randomly select 60 images for training and the remaining images are equally split into validation and test images. Our training set thus consists of 420 positive images (we add the same amount of negative images), the validation set contains 428 and the test set 429 images. For both datasets we also consider the horizontally flipped version of each image. Additionally, both datasets come with segmentation masks for the object. We make use of them by sampling positive training patches only within the mask and negatives patches not only on the negative dataset, but also on positive samples outside the mask. In total we sample 10 positive patches per image for *TUD-pedestrian* and 30 per image for *ETHZ-cars* and twice as many negatives.

Results Figure 2 shows the resulting Precision/Recall curves for all methods on the *TUD-pedestrian* and *ETHZ-cars* dataset. All three proposed methods show significant improvements over the baseline. The post-processing of detections with a histogram intersection kernel SVM on the activation vector achieves an Area Under the Curve (AUC) of 91.5% on *TUD-pedestrian* and 97.4% on *ETHZ-cars*, compared to 87.4% and 93.1% for the baseline. The linear classifier performs on par with the histogram intersection kernel on *TUD-pedestrian* (91.4%) and only slightly worse on *ETHZ-cars* (97.0%).

The Hough voting with discriminative weights also shows clear improvements over the baseline (90.9% and 98.6%). On the *ETHZ-cars* it even outperforms the two post-processing variants. The difference in the performance on the two datasets relative to the other methods may be attributed to the more complex background on the *TUD-pedestrian* dataset. This could probably still be improved by further rounds of bootstrapping. Notice, that the post-processing methods work on the output of the non-maxima suppression after standard Hough voting and thus, can only increase the precision but not the recall.

Figure 3 additionally visualizes some examples of Hough maps obtained from voting with the learned discriminative weights and compares them to the results of the baseline.

5 Conclusion

In this paper, we have shown how to determine how much each voting element in a Hough Forest contributes to a detection. This information can be seen as a description of the detection hypothesis and classifiers can be trained to discriminate between correct and incorrect hypotheses, based on this descriptor. Additionally, the weights for each voting element, learned for a linear classification model, can be incorporated directly into the generalized Hough voting process. The experiments on two different object classes, namely pedestrians and cars, show significant improvements. Visual inspection of the voting maps created with discriminatively learned voting weights shows much cleaner backgrounds and clearly sharpened and pronounced peaks for correct locations. This is also reflected in the detection scores.

Acknowledgments.

This work was supported by the Austrian Science Foundation (FWF) project Advanced Learning for Tracking and Detection in Medical Workflow Analysis (I535-N23) and by the Austrian Research Promotion Agency (FFG) project SHARE in the IV2Splus program.

References

[1] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *Proc. CVPR*, 2008.

[2] O. Barinova, V. Lempitsky, and P. Kohli. On the detection of multiple object instances using Hough transforms. In *Proc. CVPR*, 2010.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, 2005.

[5] M. Dantone, J. Gall, G. Fanelli, and L. van Gool. Real-time facial feature detection using conditional regression forests. In *Proc. CVPR*, 2012.

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. on PAMI*, 32:1627–1645, 2010.

[7] R. Fergus, P. Perona, and A. Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *IJCV*, 2006.

[8] J. Gall and V. Lempitsky. Class-specific Hough forests for object detection. In *Proc. CVPR*, 2009.

[9] J. Gall, A. Yao, N. Razavi, L. van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Trans. on PAMI*, 33(11):2188–2202, 2011.

[10] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *Proc. ICCV*, 2011.

[11] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE Trans. on PAMI*, 31:2129–2142, 2009.

[12] A. Lehmann, B. Leibe, and L. van Gool. Fast PRISM: Branch and bound hough transform for object class detection. *Int. J. Computer Vision*, 94(2):175–197, 2011.

[13] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Proc. ECCV Workshop on Statistical Learning in Computer Vision*, 2004.

[14] B. Leibe, N. Cornelis, K. Cornelis, and L. van Gool. Dynamic 3D scene analysis from a moving vehicle. In *Proc. CVPR*, 2007.

[15] N. Razavi, J. Gall, and L. van Gool. Backprojection revisited: Scalable multi-view object detection and similarity metrics for detections. In *Proc. ECCV*, 2010.

[16] N. Razavi, J. Gall, and L. van Gool. Scalable multi-class object detection. In *Proc. CVPR*, 2011.

[17] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, 2003.

[18] M. Subhransu and C. B. Alexander. Max-margin additive classifiers for detection. In *Proc. ICCV*, 2009.

[19] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *Proc. CVPR*, 2010.

[20] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57:137–154, 2004.

[21] O. Woodford, M. Pham, A. Maki, F. Perbet, and B. Stenger. Demisting the hough transform for 3D shape recognition and registration. In *Proc. BMVC*, 2011.