Class-Specific Hough Forests for Object Detection

Juergen Gall
BIWI, ETH Zürich and MPI Informatik*
gall@vision.ee.ethz.ch

Victor Lempitsky
Microsoft Research Cambridge
victlem@microsoft.com

Abstract

We present a method for the detection of instances of an object class, such as cars or pedestrians, in natural images. Similarly to some previous works, this is accomplished via generalized Hough transform, where the detections of individual object parts cast probabilistic votes for possible locations of the centroid of the whole object; the detection hypotheses then correspond to the maxima of the Hough image that accumulates the votes from all parts. However, whereas the previous methods detect object parts using generative codebooks of part appearances, we take a more discriminative approach to object part detection. Towards this end, we train a class-specific Hough forest*, which is a random forest that directly maps the image patch appearance to the probabilistic vote about the possible location of the object centroid. We demonstrate that Hough forests improve the results of the Hough-transform object detection significantly and achieve state-of-the-art performance for several classes and datasets.*

1. Introduction

The appearance of objects of the same class such as cars or pedestrians in natural images vary greatly due to intraclass differences, changes in illuminations, and imaging conditions, as well as object articulations. Therefore, to ease the detection (localization) most of the methods take the bottom-up, part-based approach, where the detections of individual object parts (features) are further integrated to reason about the positioning of the entire objects.

Toward this end, the Hough-transform based method of Leibe et al. [10, 11] learns the class-specific *implicit shape model (ISM)*, which is essentially a codebook of interest point descriptors typical for a given class. After the codebook is created, each entry is assigned a set of offsets with respect to the object centroid that are observed on the training data. At runtime, the interest point descriptors in the image are matched against the codebook and the matches cast probabilistic votes about possible positions of the object in the scale-space. These votes are summed up into a Hough image, the peaks of it being considered as detection hypotheses. The whole detection process can thus be described as a generalized class-specific Hough transform [4].

Implicit shape models can integrate information from a large number of parts. They also demonstrate good generalization as they are free to combine parts observed on different training examples. Furthermore, the additive nature of the Hough transform makes the approach robust to partial occlusions and untypical unseen part appearances. However, such codebook-based Hough transform comes at a significant computational price. On the one hand, a large generative codebook is required to achieve good discrimination. On the other hand, the construction of large codebooks involves solving difficult, large-scale clustering problems. Finally, matching with the constructed codebook is time-consuming, as it is typically linear in the number of entries.

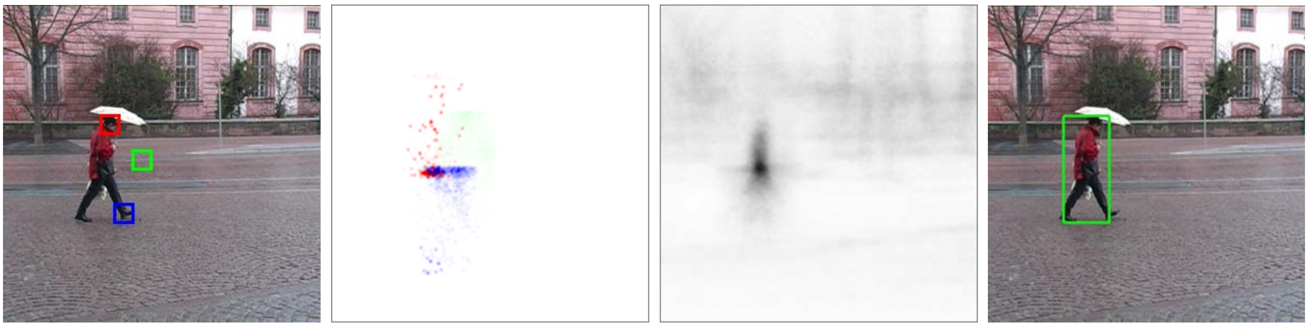
In the paper, we develop a new Hough transform-based detection method, which takes a more discriminative approach to part detection. Rather than using an explicit codebook of part appearances, we learn a direct mapping between the appearance of an image patch and its Hough vote. While learning such a mapping cannot be formalized as a standard classification or regression problem, we demonstrate that it can be efficiently accomplished within the *random forest* framework [2, 7]. Thus, given a dataset of training images with the bounding-box annotated samples of the class instances, we learn a class-specific random forest that is able to map an image patch to a probabilistic vote about the position of an object centroid. At runtime, such a class-specific *Hough forest* is applied to the patches in the test image and the resulting votes are accumulated in the Hough image, where the maxima are sought.

Random forests have recently attracted a lot of attention in computer vision [6, 12, 20, 25, 27]. Related to our work, the idea of replacing generative codebooks with random forests has been investigated in the context of image classification and semantic segmentation in [13, 14, 20, 25]. Most similar to Hough forests are the classification random forests used to obtain the unary potentials within the LayoutCRF method [27].

While Hough forests are in many aspects similar to other random forests in computer vision, they possess several interesting specific properties, motivated by their use within the generalized Hough transform framework:

- The set of leaf nodes of each tree in the Hough-forest can be regarded as a discriminative codebook. Each leaf node makes a probabilistic decision whether a patch cor-

[*] The major part of the research project was undertaken when Juergen Gall was an intern with Microsoft Research Cambridge.



(a) – Original image with three sample patches emphasized

(b) – Votes assigned to these patches by the Hough forest

(c) – Hough image aggregating votes from all patches

(d) – The detection hypothesis corresponding to the peak in *(c)*

Figure 1. For each of the three patches emphasized in *(a)*, the pedestrian class-specific Hough forest casts a vote about the possible location of a pedestrian centroid *(b)* (each color channel corresponds to the vote of a sample patch). Note the weakness of the vote from the background patch (green). After the votes from all patches are aggregated into a Hough image *(c)*, the pedestrian can be detected *(d)* as a peak in this image.

responds to a part of the object or to the background, and casts a probabilistic vote about the centroid position with respect to the patch center.

- The trees in a Hough forest are built directly to optimize their voting performance. In other words, the training process builds each tree so that the leaves produce probabilistic votes with small uncertainty.
- Each tree is built based on the collection of patches drawn from the training data. Importantly, the building process employs all the supervision available for the training data: namely, whether a patch comes from a background or an object, in the latter case, which part of the object does it come from.

Our method also benefits from the advantages typical to other random forests applications. Thus:

- Random forests can be trained on large, very high-dimensional datasets without significant overfitting and within a reasonable amount of time (hours). For our method, this permits the use of a discriminative, high-dimensional (up to 8192D) patch appearance descriptor and large training datasets.
- Random forests are very efficient at runtime, since matching a sample against a tree is logarithmic in the number of leaves. Therefore, rather than restricting our attention to the interest points as in [10, 11], our method is able to sample patches densely, while maintaining similar or better computational performance.
- Random forests can tolerate a significant amount of labeling noise and errors in the training data. Therefore, our method permits the use of bounding box-annotated training data as opposed to pixel-accurate segmentations used by previous Hough-based methods [10, 11].

After discussing related work, we describe how class-specific Hough forests are constructed and demonstrate how they can be used to cast probabilistic votes within the generalized Hough transform. We conclude with the experimental validation on several benchmark datasets, where our approach outperforms related approaches and in many cases achieves state-of-the-art detection performance.

2. Related work

The set of leaves of each tree in a Hough forest can be regarded as a discriminative Hough-voting codebook. Importantly, while generative codebooks for ISMs [10, 11] are constructed via *unsupervised* clustering of appearances, each tree in a Hough forest is constructed in a supervised way. Such a supervision allows to optimize the codebook entries to produce more reliable votes in Hough space.

Opelt et al. [17] also investigated the use of the supervised construction of Hough-voting codebooks with the emphasis on contour shape features. Their generative codebook is constructed by picking the exemplars that tend to produce more reliable votes at train time. They further increase the discriminative power of their model by picking the small ensemble of original entries and combining them within the boosting framework. Our approach, therefore, shares the idea of supervision for the voting codebook construction with [17], but does this within a discriminative random forest framework.

Similarly to our approach, Marée et al. [13] and Moosmann et al. [14] as well as Shotton et al. [25] and Schroff et al. [20] train random forests on image patches in order to use them as discriminative codebooks. Those codebooks, however, are employed for image categorization or semantic segmentation rather than Hough-based object detection. As such, no geometric information but only class labels are stored at leaves and are used as a supervision for trees construction, as opposed to our method.

Finally, Winn and Shotton [27] build random forests in order to distinguish between the patches from different parts of the object as well as from background (similar purpose to Hough forests). They, however, consider a pure classification problem by splitting the object into a predefined number of parts treated as independent classes. This is because the output of their forests is used on a later stage as unary terms for a discrete-labeled conditional random field. On the contrary, as Hough forests are used for voting, their output are essentially the votes in the continuous domain. Thus, unlike [27], we avoid splitting objects into independent parts.

3. Hough Forests

Random forests [2, 7] have been used for a large number of classification as well as regression tasks. A typical random forest consists of a set of binary decision trees [19]. During training, each non-leaf node in each tree is assigned a binary test that is applicable to any data sample. Depending on the result of the test, a sample can go to one of the two children of a given non-leaf node. This way, a sample can be passed through each of the trees, starting from its root and ending up in one of its leaves.

Random forests are trained in a supervised way. Training involves tree construction as well as assigning each leaf node the information about the training samples reaching this leaf node, e.g. the class distribution in the case of classification tasks. At runtime, a test sample is passed down all the trees of the forest, and the output is computed by averaging the distributions recorded at the reached leaf nodes.

It has been shown [2, 7] that assembling together several trees trained in a randomized way achieves superior generalization and stability compared to a single deterministic decision tree. The randomization is achieved, firstly, by training each tree on a random subset of the training data, and, secondly, by considering a random subset of possible binary tests at each non-leaf node. Among this random subset, the training procedure picks the binary test that splits the training samples in the optimal way, where the measure of optimality is typically application-specific.

3.1. Building Hough Forests

Training data and leaf information. For Hough forests, each tree $\mathcal{T}$ is constructed based on a set of patches $\{\mathcal{P}_i{=}(\mathcal{I}_i, c_i, \mathbf{d}_i)\}$, where $\mathcal{I}_i$ is the appearance of the patch, c_i is the *class label* of the patch, and $\mathbf{d}_i$ is the *offset* of the patch. We describe each of these components and their use below, starting with the class label and the offset.

The training patches are sampled from the training collection of images, some of them containing examples of the class of interest with known bounding boxes. The patches sampled from the background (*background patches*) are assigned the *class label* $c_i{=}0$, while the patches sampled from the interior of the object bounding boxes (*object patches*) are assigned $c_i{=}1$. Each object patch is also assigned a 2D *offset vector* $\mathbf{d}_i$ equal to the offset from the centroid of the bounding box to the center of the patch. For a background patch, $\mathbf{d}_i$ is undefined. In our current implementation, scale invariance is not introduced during training, and therefore the object patches are sampled from the pre-scaled object images to have approximately the same size. To achieve the scale invariance at runtime, we apply Hough forests at several scales as described in Section 4.

For each leaf node L in the constructed tree, the information about the patches that have reached this node at train time is stored. Thus, we store the proportion C_L of the object patches (e.g., $C_L = 1$ means that only object patches have reached the leaf) and the list $D_L{=}\{\mathbf{d}_i\}$ of the offset vectors corresponding to the object patches. The leaves of the tree thus form a discriminative codebook with the assigned information about possible locations of the centroid (Figure 2). At runtime, this information is used to cast the probabilistic Hough votes about the existence of the object at different positions (Section 4).

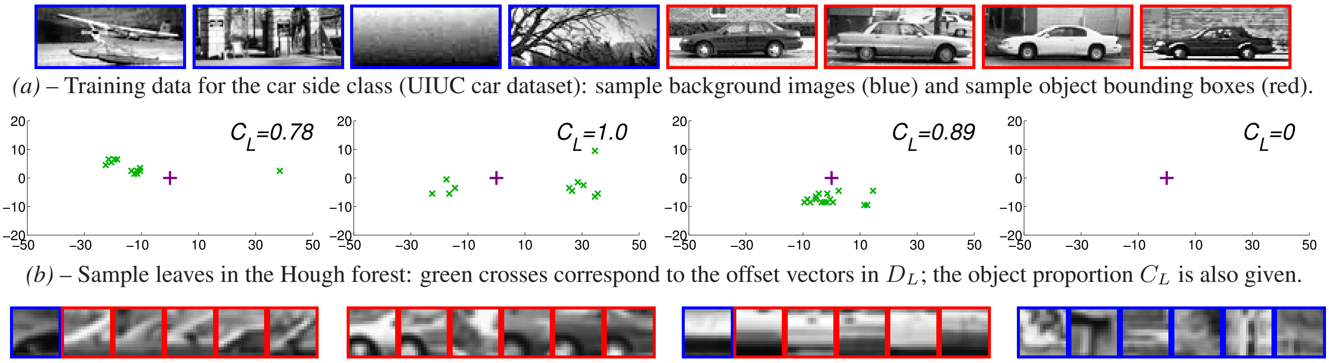
Patch appearance and binary tests. During training, each non-leaf node in each tree is assigned a binary test applicable to the appearance of a patch $\mathcal{I}_i$. At both train and test time, the patches have a fixed size, e.g. 16-by-16 pixels, and the appearance is defined by the extracted feature channels, which may correspond to raw intensities, derivative filter responses, etc. Thus, the appearance of the patch can be written as $\mathcal{I}_i = (I_i^1, I_i^2, \ldots I_i^C)$, where each I_i^j is a 16-by-16 image and C is the number of channels.

The binary tests on a patch appearance $t(\mathcal{I}) \rightarrow \{0,1\}$ can be defined in several ways, but we have chosen simple pixel-based tests. Such a test is defined by a channel $a \in \{1, 2, \ldots, C\}$, two positions (p,q) and (r,s) in the 16-by-16 image, and a real handicap value τ. The test $t_{a,p,q,r,s,\tau}(\mathcal{I})$ is then defined as:

$$t_{a,p,q,r,s,\tau}(\mathcal{I}) = \begin{cases} 0, & \text{if } I^a(p,q) < I^a(r,s) + \tau \\ 1, & \text{otherwise.} \end{cases} \quad (1)$$

Such a test simply compares the values of a pair of pixels in the same channel with some handicap.

Tree construction. In general, the tree construction for Hough forests follows the common random forest framework [7]. Each tree is constructed recursively starting from the root. During construction, each node receives a set of training patches. If the depth of the node is equal to the maximal one ($d_{Max} = 15$) or the number of patches is small ($N_{\min} = 20$), the constructed node is declared a leaf and the leaf vote information (C_L, D_L) is accumulated and stored. Otherwise, a non-leaf node is created and an optimal binary test is chosen from a large pool of randomly generated binary tests. The training patch set that has arrived to the node is then split according to the chosen test into two



(a) – Training data for the car side class (UIUC car dataset): sample background images (blue) and sample object bounding boxes (red).

(b) – Sample leaves in the Hough forest: green crosses correspond to the offset vectors in D_L; the object proportion C_L is also given.

(c) – Sample patches that fall inside each of the leaves above during training (red corresponds to object patches).

Figure 2. For the set of training images shown in *(a)*, we visualize the data recorded in some of the leaves of the constructed class-specific Hough forest in *(b)*. This data consists of the object patch proportion C_L and the list of the offset vectors for object patches D_L. Note that the leaves of the Hough forest form a discriminative class-specific codebook as shown in *(c)*: the training examples falling inside each of the first three leaves can be associated with different parts of a car.

subsets that are passed to the two newly created children nodes; after that the recursion proceeds[1].

The key moment specific to Hough forests is the way the quality of binary tests is evaluated. At runtime, both the class proportions and the offset lists are used to cast the votes. The high-level idea, therefore, is to pick the tests, so that the uncertainties in both the class labels and the offset vectors decrease towards the leaves. To achieve such a reduction, we define two measures of the uncertainty for a set of patches $A = \{\mathcal{P}_i = (\mathcal{I}_i, c_i, \mathbf{d}_i)\}$. The *class-label uncertainty* measuring the impurity of the class labels c_i is defined as:

$$U_1(A) = |A| \cdot \mathit{Entropy}(\{c_i\})\,, \tag{2}$$

where the entropy of the set of binary values $\{c_i\}$ with the mean value $c \in [0,1]$ is defined in a standard way: $\mathit{Entropy}(\{c_i\}) = -c \cdot \log c - (1-c) \cdot \log(1-c)$. The second measure called the *offset uncertainty* and corresponding to the impurity of the offset vectors $\mathbf{d}_i$ is defined simply as:

$$U_2(A) = \sum_{i:c_i=1} (\mathbf{d}_i - \mathbf{d}_A)^2, \tag{3}$$

where $\mathbf{d}_A$ is the mean offset vector over all object patches in the set[2]. Note that the background patches corresponding to $c_i = 0$ are ignored here.

Given the two uncertainty measures, the binary test is chosen as follows. Given a training set of patches $\{\mathcal{P}_i = (\mathcal{I}_i, c_i, \mathbf{d}_i)\}$, we first generate a pool of pixel tests $\{t^k\}$ by sampling a, p, q, r, and s uniformly. The handicap value τ for each test is chosen uniformly at random from the range of differences observed on the data. Then, the randomized decision is made whether the node should minimize the class-label uncertainty or the offset uncertainty. In general, we choose this with equal probability unless the number of negative patches is small ($< 5\%$), in which case the node is chosen to minimize the offset uncertainty. Finally, we pick the binary test with the minimal sum of the respective uncertainty measures for the two subsets, it splits the training set into:

$$\underset{k}{\operatorname{argmin}} \Big(U_\star(\{p_i |\, t^k(\mathcal{I}_i){=}0\}) + U_\star(\{p_i |\, t^k(\mathcal{I}_i){=}1\}) \Big) \tag{4}$$

where $\star = 1$ or 2 (depending on our random choice).

By interleaving the nodes that decrease the class-label uncertainty with the nodes that decrease the offset uncertainty, the tree construction process ensures that the sets that reach the leaves have low variations in both class labels and offsets (as Figure 2 demonstrates). As a result, the leaves in a Hough forest are able to cast probabilistic votes with low uncertainty about the presence of the object centroids near the patch. This is used for object detection as discussed in the next section.

4. Object Detection with Hough Forests

Class-specific Hough forests can be used to localize (detect) the bounding boxes of the instances of a class in a test image using Hough transform. Let us, first, assume that the size of the object bounding boxes is fixed to $W \times H$ during both training and testing. Under this assumption, the only parameter defining an object bounding box is the centroid.

Consider a patch $\mathcal{P}(\mathbf{y}) = (\mathcal{I}(\mathbf{y}), c(\mathbf{y}), \mathbf{d}(\mathbf{y}))$ centered at the position $\mathbf{y}$ in the test image. Here, $\mathcal{I}(\mathbf{y})$ is the ob-

[1] In general, the trees are not guaranteed to be balanced and are not perfectly balanced in practice. There is, however, a bias towards balanced trees, as both our splitting criterions (2) and (3) have biases towards equal-size partitions.

[2] An information-theoretic offset uncertainty measure based on the entropy of a kernel-density estimate for the offset vectors distribution was also tried. While it achieved a similar level of performance, the simpler measure defined above was preferred for computational efficiency reasons.

served appearance of the patch, $c(\mathbf{y})$ is the hidden class-label (whether $\mathbf{y}$ lies inside the object bounding box or not), and $\mathbf{d}(\mathbf{y})$ is the hidden offset vector from the center of the object bounding box to y (meaningful only in the case $c(\mathbf{y}) = 1$). Furthermore, $E(\mathbf{x})$ denotes the random event corresponding to the existence of the object centered at the location $\mathbf{x}$ in the image.

We are now interested in computing the probabilistic evidence $p\big(E(\mathbf{x})|\,\mathcal{I}(\mathbf{y})\big)$ that the appearance $\mathcal{I}(\mathbf{y})$ of the patch brings about the availability $E(\mathbf{x})$ at different positions x in the image. We will distinguish between the two cases: whether $\mathbf{y}$ belongs to the bounding box $B(\mathbf{x})$ centered at $\mathbf{x}$ or not. If $\mathbf{y} \notin B(\mathbf{x})$, then we assume that $\mathcal{I}(\mathbf{y})$ is not informative about $E(\mathbf{x})$, putting $p\big(E(\mathbf{x})|\,\mathcal{I}(\mathbf{y})\big) = p\big(E(\mathbf{x})\big)$. This is, of course, a simplifying assumption, and such "long-range" *context* information has been proven useful for object recognition [26]. In fact, it has even been exploited for semantic segmentation in random forest based methods [20, 25] and it can be incorporated in our system in a similar way.

In the paper, however, we consider the evidence coming from the patches within the bounding box only and thus focus on the second case when $\mathbf{y} \in B(\mathbf{x})$. Here, the existence of an object centered at $\mathbf{x}$ inevitably implies $c(\mathbf{y}) = 1$ by our definition of the class label. As a result, one gets:

$$\begin{aligned} p\big(E(\mathbf{x})|\,\mathcal{I}(\mathbf{y})\big) \;=\; p\big(E(\mathbf{x}),c(\mathbf{y}){=}1|\,\mathcal{I}(\mathbf{y})\big) =\\ p\big(E(\mathbf{x})|\,c(\mathbf{y}){=}1,\mathcal{I}(\mathbf{y})\big)\cdot p\big(c(\mathbf{y}){=}1|\,\mathcal{I}(\mathbf{y})\big) =\\ p\big(\mathbf{d}(\mathbf{y}) = \mathbf{y}-\mathbf{x}|\,c(\mathbf{y}){=}1,\mathcal{I}(\mathbf{y})\big)\cdot p\big(c(\mathbf{y}){=}1|\,\mathcal{I}(\mathbf{y})\big). \end{aligned} \tag{5}$$

Both factors in (5), can be estimated by passing the patch appearance $\mathcal{I}(\mathbf{y})$ through the trees in the class-specific Hough forest. Let us assume that for a tree $\mathcal{T}$ the patch appearance ends up in a leaf L. The first factor can then be approximated using the Parzen-window estimate based on the offset vectors D_L collected in the leaf at train time, while the second factor can be straightforwardly estimated as the proportion C_L of object patches at train time. For a single tree $\mathcal{T}$, the probability estimate can be written as:

$$\begin{aligned} &p\big(E(\mathbf{x})|\,\mathcal{I}(\mathbf{y});\mathcal{T}\big) =\\ &\left[\frac{1}{|D_L|}\sum_{d\in D_L}\frac{1}{2\pi\sigma^2}\exp\left(-\frac{||(\mathbf{y}-\mathbf{x})-d||^2}{2\sigma^2}\right)\right]\cdot C_L \end{aligned} \tag{6}$$

where $\sigma^2\mathbf{I}_{2\times 2}$ is the covariance of the Gaussian Parzen window. For the entire forest $\{\mathcal{T}_t\}_{t=1}^{T}$, we simply average the probabilities (6) coming from different trees [2, 7], getting the forest-based estimate:

$$p\big(E(\mathbf{x})|\,\mathcal{I}(\mathbf{y});\{\mathcal{T}_t\}_{t=1}^{T}\big) = \frac{1}{T}\sum_{t=1}^{T} p\big(E(\mathbf{x})|\,\mathcal{I}(\mathbf{y});\mathcal{T}_t\big). \tag{7}$$

The Equations (6) and (7) define the probabilistic vote cast by a single patch about the existence of the objects in nearby locations. To integrate the votes coming from different patches, we accumulate them in an (admittedly non-probabilistic) additive way into a 2D Hough image $V(\mathbf{x})$, which for each pixel-location $\mathbf{x}$ sums up the votes (7) coming from the nearby patches:

$$V(\mathbf{x}) = \sum_{y\in B(x)} p\big(E(\mathbf{x})|\,\mathcal{I}(\mathbf{y});\{\mathcal{T}_t\}_{t=1}^{T}\big)\,. \tag{8}$$

The detection procedure simply computes the Hough image V and returns the set of its maxima locations and values $\{\hat{x}, V(\hat{x})\}$ as the detection hypotheses. The $V(\hat{x})$ values serve as the confidence measures for each hypothesis.

The computation of the Hough image using the order of operations as suggested by (6)–(8) would be inefficient. Instead, the same image (upto a constant multiplicative factor and minor pixel discretization issues) can be computed by going through each pixel location $\mathbf{y}$, passing the patch appearance $\mathcal{I}(\mathbf{y})$ through every tree in the Hough forest, and adding the value $\frac{C_L}{|D_L|}$ to all pixels $\{\mathbf{y}-\mathbf{d}|\,\mathbf{d}\in D_L\}$. The Hough image $V(\mathbf{x})$ is then obtained by Gaussian-filtering the vote counts accumulated in each pixel. An alternative way to find the maxima of the Hough image would be to use the mean-shift procedure as it is done in other Hough voting-based frameworks [10, 11, 17].

Handling variable scales. To handle scale variations, we resize a test image by a set of scale factors $s_1, s_2, \ldots s_S$. The Hough images $V^1, V^2, \ldots V^S$ are then computed independently at each scale. After that, the images are stacked in a 3D scale-space frustum, the Gaussian filtration is performed across the third (scale) dimension, and the maxima of the resulting function are localized in 3D. The resulting detection hypotheses have the form $(\hat{\mathbf{x}}, \hat{s}, V^{\hat{s}}(\hat{\mathbf{x}}))$. The hypothesized bounding box in the original image is then centered at the point $\frac{\hat{\mathbf{x}}}{\hat{s}}$, has the size $\frac{W}{\hat{s}} \times \frac{H}{\hat{s}}$, and the detection confidence $V^{\hat{s}}(\hat{\mathbf{x}})$. Similar ideas can be applied if significant variations of the aspect ratio are expected as is briefly discussed in Section 5.

5. Experiments

We evaluated the Hough forests on several challenging datasets (Figure 3), where we provide a performance comparison with the related detection methods as well as with the best (as far as we know) previously published results. The performance curves were generated by changing the acceptance threshold on the hypotheses vote strength $V(\hat{x})$. We adhered to the experimental protocols and detection correctness criteria established for each of the datasets in previous works. When generating recall-precision curves, we rejected the detection hypotheses with centroids inside the bounding boxes detected with higher confidence in order to avoid multiple detections of the same instance.

The training settings were as follows. During training,

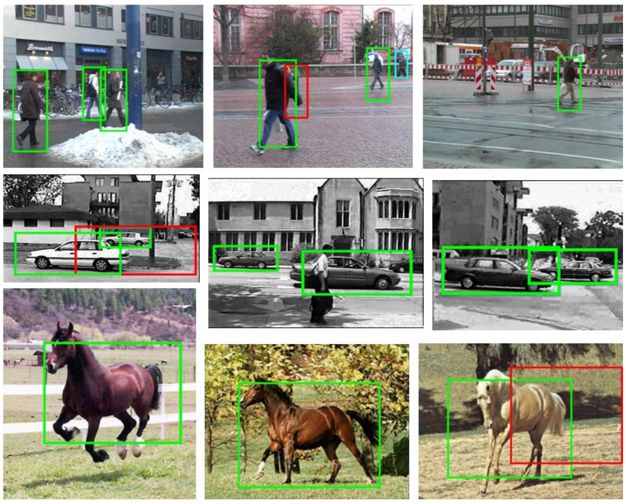


Figure 3. The results of our detector at equal recall-precision rates on challenging images from TUD pedestrian, UIUC-Scale, and Weizmann Horse datasets (green = correct, red = false positive, cyan = missed detection).

the positive examples were rescaled to the same height, chosen so that the larger bounding box dimension (width or height) was equal to 100 pixels on average over a dataset. 20 000 random binary tests were considered for each node. Each tree was trained on 25 000 positive and 25 000 negative patches. To bias our training to work better on hard examples, we used the following procedure. For the first 5 trees, the patches were sampled with uniform probabilities from all available positive and negative examples. Then the constructed random forest was applied to the training data and the positive and negative instances that were harder to classify were acquired. These were used to construct the next 5 trees added to the previous 5. We applied this procedure once more, ending up with the forest having 15 trees.

For detection, we used a Parzen window with $\sigma^2 = 9$. In a multi-scale setting, the additional third dimension was filtered with $\sigma^2 = 1$. Typically, 4–5 scales with equal spacing were used to handle the variety of scales in the test data.

UIUC cars. The UIUC car dataset [1] contains images of side views of cars. The test data are split into the set of 170 images with 210 cars of approximately same scale (*UIUC-Single*) and the set of 108 images containing 139 cars at multiple scales (*UIUC-Multi*). The sets include partially occluded cars, cars with low contrast, images with multiple car instances, cluttered backgrounds, and challenging illumination. The shape of the objects remains, however, mostly rigid, which makes the detection task easier.

On the available 550 positive and 450 negative training images, we trained a class-specific Hough forest. For patch appearance, 3 channels were used (intensity, absolute value of x- and y-derivatives). Applying this forest for the detection achieved an impressive 98.5% EER for UIUC-Single and 98.6% for UIUC-Multi, thus exactly matching the state-of-the-art performance reported recently in [9] (Table 1).

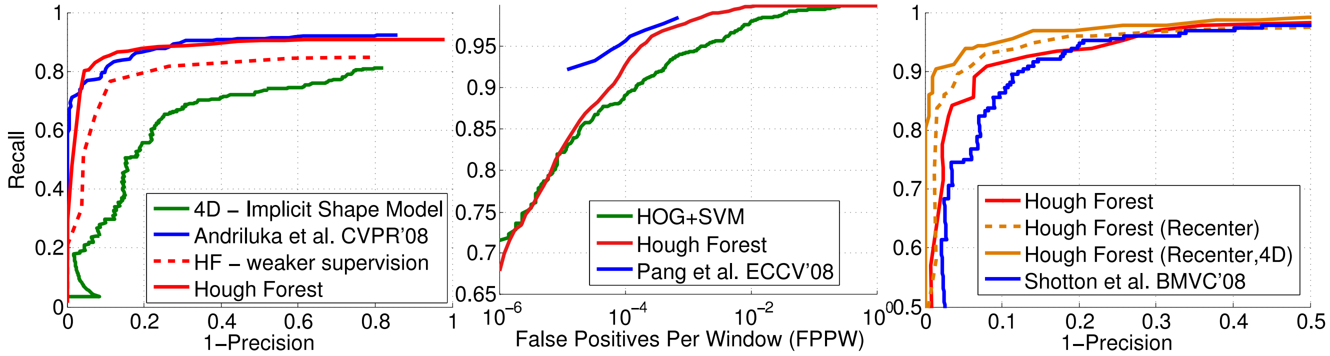
Methods	UIUC-Single	UIUC-Multi
Hough-based methods		
Implicit Shape Model [10]	91%	–
ISM+verification [10]	97.5%	95%
Boundary Shape Model [17]	85%	–
Random forest based method		
LayoutCRF [27]	93%	–
State-of-the-art		
Mutch and Lowe CVPR'06 [15]	99.9%	90.6%
Lampert et al. CVPR'08 [9]	98.5%	98.6%
Our approach		
Hough Forest	98.5%	98.6%
HF - Weaker supervision	94.4%	–

Table 1. Performance of different methods on the two UIUC car datasets at recall-precision equal error rate (EER). Hough Forest outperforms the previous Hough-based and random forest based methods and achieves the state-of-the-art.

Importantly, it outperformed considerably the Hough-based implicit shape model approach [10] (even with an additional MDL verification step) and boundary-shape model approach [17] as well as the random-forest based LayoutCRF method [27]. It has to be mentioned, at the same time, that these related methods used smaller subsets of the provided training data. In the case of the ISM and the LayoutCRF, this is due to the necessity of obtaining pixel-accurate annotations. Additionally, in the case of ISM and the Boundary Shape Model [17] this might be due to the computational burden of constructing and processing generative codebooks. As Hough Forests are not limited by these factors, we used the provided training data completely, possibly accounting for some part of the improvement.

Another distinguishing factor is that our method samples patches densely, whereas ISM methods consider sparse interest points, which is likely to give our method a significant advantage [16]. We therefore investigated the performance of our method on the single scale dataset as the density of patch sampling is decreased. The graceful degradation of the performance in Figure 4, as the number of patches is decreased down to $1/256$ of the original, suggests that the relative accuracy of the Hough forest detection is not only due to a very large number of patch votes, but also has to do with the discriminative training of the codebook.

TUD pedestrians, INRIA pedestrians, Weizmann Horses. To assess the performance of our method for more challenging, articulated classes we evaluated it on two pedestrian datasets: a recent one from TU Darmstadt introduced in [3] containing mostly side views and a more established from INRIA [8] containing mostly front and back views. Both datasets contain partial occlusions and variations in scales, poses, clothing styles, illumination, and weather conditions.



(a) TUD Pedestrian dataset *(b)* INRIA Pedestrian dataset *(c)* Weizmann Horse dataset

Figure 5. Hough forests (red and orange curves) demonstrate a competitive performance with respect to the previous state-of-the-art methods (blue curves) on several challenging datasets. See text for a more detailed discussion.

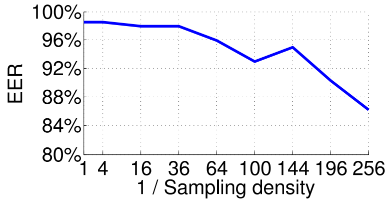


Figure 4. As the sampling density is decreased, the recall-precision equal error rate (EER) of our method on UIUC-Single degrades gracefully.

For the TUD dataset 400 training images with pedestrians are provided and, as the diversities of the backgrounds were low, we augment it with training background images from the INRIA dataset. Otherwise, we followed the experimental protocol of [3] and tested it on 250 images with 311 pedestrians in it. For the INRIA dataset, we used the provided training data of 614 images with pedestrians and 1218 background images. According to [8], we applied our method as a classifier on 288 cropped, pre-scaled images with pedestrians and 453 images without them.

We have also considered the Weizmann Horses dataset [5] containing the near-side views of horses in natural environments under varying scale and strongly varying poses. We used the training-testing split (100 horse images and 100 background image for training, 228 horse images and 228 background images for testing) as suggested in [24].

For all three datasets we used the same color channels. We have considered the following 16 feature channels: 3 color channels of the *Lab* color space, the absolute values of the first-order derivatives $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$, the absolute values of the two second-order derivatives $\frac{\partial^2}{\partial x^2}$, $\frac{\partial^2}{\partial y^2}$, and the nine HOG-like [8] channels. Each HOG-like channel were obtained as the soft bin count of gradient orientations in a 5-by-5 neighborhood around a pixel. To increase the invariance under noise and articulations of individual parts, we further processed the above-introduced 16 channels by applying the min and the max filtration with 5-by-5 filter size, yielding $C = 32$ feature channels (16 for the min filter and 16 for the max filter).

The performance of different methods including ours is shown in Figure 5. For TUD pedestrians our method (recall-precision EER = 86.5%, AUC = 0.87, recall at 90% precision = 85%) performed on a par with the state-of-the-art method [3] and significantly better than the implicit shape model-based method [21] (reproduced from [3]). Furthermore, it should be noticed that both competitors require additional annotation for training (the ISM-based approach, however, was again trained on a smaller training set).

For an image from the TUD dataset, our system requires 6 seconds[3] where each of the four operations, namely feature extraction, passing patches through the trees, casting the votes, and processing the Hough images, takes around a quarter of the computation time.

For the INRIA dataset, the Hough forest performance (recall = 93% at FFPW=10^{-4}) was not as good as that of the state-of-the-art method [18]. Yet it is still quite competitive and, in particular, performs better than the SVM-based detection for similar features (HOG) [8]. It may be also argued that the non-standard testing protocol for this dataset favors sliding-window approaches to some extent. We also used the dataset to measure the impact of the min and max filtration of the original channels, where we got a decrease of around 10% recall at 10^{-4} FPPW when working with the 16 unfiltered channels. This suggests that min and max filtration is needed to make the response of pixel-based comparison tests (1) more stable, at least when working with such deformable classes as pedestrians.

Finally, for the Weizmann Horse dataset the performance of the Hough forest was clearly better than the previous state-of-the-art [23]. Nevertheless, we have tried two more improvements addressing the two challenges of this dataset. Firstly, the position of bounding box centroids are not stable with respect to the horse bodies, which leads to a cer-

[3] 720×576 pixel resolution; 4 scales (0.3, 0.4, 0.5, 0.6); modern CPU

tain smearing of votes. To address this, we ran our detector on the positive training images and recentered the bounding boxes to the peaks of the response. After that the forest was retrained. Secondly, the aspect ratios of the boxes varied considerably due to the articulations and variations in the viewpoint. To address this, we performed voting in 4D Hough space, where the 4th dimension corresponded to the aspect ratio multiplier (the number of patch-against-tree matching operations were not increased though, as the votes were reused between different ratios). As can be seen from Figure 5*(c)*, both improvements increased the performance considerably (recall-precision EER went from 91% to 93.9%, AUC from 0.96 to 0.98, recall at 90% precision from 91.5% to 95.1%) obtaining a substantial margin over the previous state-of-the-art.

Does offset supervision matter? Quite a few previous approaches have used random forests as discriminative codebooks [12, 13, 14, 20, 25]. Hough forests differ from them as they store the patch offsets at leaves and use them at runtime to perform voting. Furthermore, the offset information is used as supervision during training of Hough forests since half of the binary tests are chosen to minimize the offset uncertainty (3). We therefore addressed the question whether such additional supervision matters. Thus, for the datasets UIUC-Single and TUD, we built forests where all binary splits were chosen to minimize the class uncertainty (2) (a similar criteria drove forest constructions in the above-mentioned works). The leaf information and the detection procedure remained as before.

The performance of the new forests form the *'HF-weaker supervision'* entries in Table 1 and Figure 5*(a)*. A considerable drop in performance compared to fully-supervised Hough forests is observed, suggesting that offset vectors were a valuable supervision during training.

6. Discussion

We have introduced the *Hough forests* approach for object detection. It builds discriminative class-specific part appearance codebooks based on random forests that are able to cast probabilistic votes within the Hough transform framework. Such forests can be efficiently used to detect instances of classes in natural images, with the accuracy that is not only superior to previous methods using related techniques but also improves the state-of-the-art for several datasets. Apart from the accuracy, the use of random forests potentially allows a very time-efficient implementation. While our current unoptimized CPU version takes several seconds per image, the speed-up factors reported for the GPU implementation of random forests in [22] suggest that near real-time performance is attainable. Our future plans also include building multi-class Hough forests, as well as testing on challenging PASCAL VOC datasets.

Acknowledgements. The advice from Toby Sharp, Jamie Shotton, and other members of the Computer Vision Group at MSRC is gratefully acknowledged. We would also like to thank all the researchers, who have collected and published the datasets we used for the evaluation.

References

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *TPAMI*, 26(11):1475–1490, 2004.

[2] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997.

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

[4] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[5] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. *ECCV*, pp. 639–641, 2002.

[6] A. Bosch, A. Zisserman, and X. Muñoz. Image classification using random forests and ferns. *ICCV*, pp. 1–8, 2007.

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

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR (1)*, pp. 886–893, 2005.

[9] C. Lampert, M. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. *CVPR*, 2008.

[10] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 77(1-3):259–289, 2008.

[11] B. Leibe and B. Schiele. Interleaved object categorization and segmentation. *In BMVC*, pp. 759–768, 2003.

[12] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. *CVPR (2)*, pp. 775–781, 2005.

[13] R. Marée, P. Geurts, J. H. Piater, and L. Wehenkel. Random subwindows for robust image classification. *CVPR (1)*, pp. 34–40, 2005.

[14] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. *NIPS*, 2006.

[15] J. Mutch and D. G. Lowe. Multiclass object recognition with sparse, localized features. *CVPR (1)*, pp. 11–18, 2006.

[16] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. *ECCV (4)*, pp. 490–503, 2006.

[17] A. Opelt, A. Pinz, and A. Zisserman. Learning an alphabet of shape and appearance for multi-class object detection. *IJCV*, 2008.

[18] J. Pang, Q. Huang, and S. Jiang. Multiple instance boost using graph embedding based decision stump for pedestrian detection. *ECCV*, pp. 541–552, 2008.

[19] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[20] F. Schroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests. *BMVC*, 2008.

[21] E. Seemann and B. Schiele. Cross-articulation learning for robust detection of pedestrians. *DAGM*, pp. 242–252, 2006.

[22] T. Sharp. Implementing decision trees and forests on a GPU. *ECCV (4)*, pp. 595–608, 2008.

[23] J. Shotton, A. Blake, and R. Cipolla. Efficiently combining contour and texture cues for object recognition. *BMVC*, 2008.

[24] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *PAMI*, 30(7):1270–1281, 2008.

[25] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. *CVPR*, 2008.

[26] A. B. Torralba and P. Sinha. Statistical context priming for object detection. *ICCV*, pp. 763–770, 2001.

[27] J. M. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. *CVPR (1)*, pp. 37–44, 2006.