Efficient Object Detection and Segmentation with a Cascaded Hough Forest ISM

Konstantinos Rematas
ESAT-PSI / IBBT
K.U. Leuven
krematas@esat.kuleuven.be

Bastian Leibe
UMIC Research Centre
RWTH Aachen University
leibe@umic.rwth-aachen.de

Abstract

Visual pedestrian/car detection is very important for mobile robotics in complex outdoor scenarios. In this paper, we propose two improvements to the popular Hough Forest object detection framework. We show how this framework can be extended to efficiently infer precise probabilistic segmentations for the object hypotheses and how those segmentations can be used to improve the final hypothesis selection. Our approach benefits from the dense sampling of a Hough Forest detector, which results in qualitatively better segmentations than previous voting based methods. We show that, compared to previous approaches, the dense feature sampling necessitates several adaptations to the segmentation framework and propose an improved formulation. In addition, we propose an efficient cascaded voting scheme that significantly reduces the effort of the Hough voting stage without loss in accuracy. We quantitatively evaluate our approach on several challenging sequences, reaching state-of-the-art performance and showing the effectiveness of the proposed framework.

1. Introduction

In the field of robotic perception, object detection plays an important role. The autonomous entities need to understand their environment and in several cases they are supposed to interact with specific objects (object grasping, obstacle avoidance, *etc*.). Therefore, a detector that provides precise localization of objects is needed.

The recent PASCAL VOC Challenge object detection benchmarks have documented an interesting development, that simple, global sliding window representations for object detection are reaching a performance limit [9] and part-based models are again gaining popularity [12, 4, 14]. The reasons for this development are twofold. On the one hand, many object categories are poorly represented by axis-aligned bounding boxes, such that an increasingly complex learning machinery would be required in order to cope with the poor signal-to-noise ratio inherent in a holistic representation. On the other hand, part-based models have been augmented with powerful discriminative training methods [12, 19, 4, 14] that significantly increase their robustness compared to their purely generative predecessors [11, 13, 17].

At the same time, there has been a growing trend towards segmentation as an additional cue in order to support, supplement, and interpret the recognition results [15, 4, 18, 1]. This has been helped by the increasing availability of cheap, pixel-level annotations created through Amazon Mechanical Turk [23], from which category-specific segmentation methods can be trained [16]. Indeed, segmentation processes provide very useful information, since they allow recognition approaches to perform a more detailed analysis of the constituent object parts than a bounding box-based sliding window framework would permit.

Hence, there is a strong incentive to make segmentation capabilities available to approaches which have so far only been applied in a sliding window fashion. Class-specific Hough Forests [14] and their recent extensions [21, 3] are a particularly interesting case in this respect. They take up the voting idea of Implicit Shape Models (ISM) [17], but extend it with densely sampled features and a discriminative training procedure. Hough Forests have been shown to reach comparable performance to sliding window classifiers on a number of benchmark datasets [14] and are inherently capable of multi-class detection.

However, so far Hough Forests have only made use of part of their full potential. In particular, they do not include the top-down segmentation capabilities that were available in their ISM predecessor. In addition, compared to efficient sliding window implementations, the Hough voting step incurs considerable computational effort. A Hough Forest trained for a challenging object category will generate a large number of votes to be processed, limiting the approach's effective run-time.

In this paper, we propose an extension of the Hough Forest approach that addresses both of the above issues. (1) We show how top-down segmentation capabilities can be integrated into the detection process and that this integration improves the quality of the resulting detections. Compared to previous procedures for sparse feature based segmentation [17, 20], the transition to densely sampled features necessitates several adaptations. We propose an efficient algo-

rithm to compute top-down segmentations from Hough Forest detections and derive a hypothesis verification scheme that outperforms currently used schemes [14, 3]. (2) In addition, we propose a cascaded evaluation strategy that significantly improves the run-time of Hough Forest detectors without loss in accuracy. Our approach first uses a single Randomized Tree to define conservative regions-of-interest in which later features need to be evaluated. It then applies a binned vote casting strategy that limits the number of votes considered by each evaluated image patch to a small fraction of the original votes. Together, those two steps reduce the run-time of the voting step by a factor of more than two.

Related Work. Our proposed approach builds upon the class-specific Hough Forest detection framework by [14], which is in turn inspired by the ISM detector from [17]. Both of those approaches map the appearance of object parts onto visual words with specific spatial distribution. For generating the visual words, [17] clusters regions around interest points, while [14] uses the Random Forest framework [5]. This change enables the transition from sparse to dense features and the use of discriminative training, which together significantly improve the detection results. Additional improvements for vote weighting and non-maximum suppression have been proposed by [19] and [3].

The ISM detector [17] includes the capability to infer probabilistic segmentations by back-projecting the votes that contributed to a Hough-space maximum. However, the segmentation results are not very precise due to the sparse sampling only at interest points. In the recent work of [21], back-projection was also used in a Hough Forest framework, but only to reveal the support of a hypothesis in the image domain. In our method, we take full advantage of the back-projection process together with the dense sampling by Hough Forests, resulting in precise segmentations.

Cascading strategies are a well-established means to speed up object detector evaluation [24]. The idea behind cascading is that some image regions are so different from the target object class that they can be rejected already using very simple classifiers. A cascade of object/non-object classifiers was successfully applied for face detection in [24]. Another approach that uses a cascaded framework is the cascaded part-based model of [10], where each part appearance model reduces the search space of its children models. In this work, we propose a similar cascading idea in order to speed up Hough Forest voting.

2. Object Detection with Hough Forests

This section describes the necessary background of the Hough Forest framework [14] and the notation that we will use in the rest of the paper. A Hough Forest consists of a collection of randomized trees. Each image patch is passed through all trees in parallel. In each non-leaf node, a simple binary test is performed. The test is applied to each patch that arrives in the node, and its output defines the child the patch will proceed to. Once a leaf node is reached, it casts votes for possible positions of the object center in a probabilistic Generalized Hough Transform, similar to [17]. Maxima in the Hough voting space correspond to object hypotheses.

Feature Channels. The images that are used for training and testing are usually in RGB format, which in most cases is not discriminative enough. We therefore compute the following feature channels: $L * a * b$ color, first and second order derivatives in x and y, and 9 HOG-like [6] channels. In the case of pedestrians, we apply min and max filters in a spatial window of 5×5 pixels, similar to [14].

Training. The training procedure first extracts a set of object and background patches. A patch can be expressed as $f_i = (\mathcal{I}_i, c_i, \mathbf{d}_i)$, where $\mathcal{I}_i$ corresponds to the feature channel, c_i is the class label, and $\mathbf{d}_i$ is the relative position of the patch to the object center (note that $\mathbf{d}_i$ is undefined for background patches). Based on such a set of patches, the Hough Tree is then constructed recursively, starting from the root. The selection of random tests is based on how well they separate the input set of patches. The quality of the separation is measured by one of two uncertainty measures: class label uncertainty U_1 and offset uncertainty U_2

$$U_1(A) = |A| \cdot E(c_i) \quad (1)$$

$$U_2(A) = \sum_{i:c_i \neq background} (\mathbf{d}_i - \bar{\mathbf{d}}), \quad (2)$$

where A is the set of patches assigned to a node, E the class label entropy and $\bar{\mathbf{d}}$ is the mean offset of this set. The first measure tries to create two subsets of patches that are as pure as possible in terms of their class labels, while the second measure forces the patches' offsets to be spatially coherent. When the number of patches is below a certain threshold or the maximum predefined height of the tree is reached, the node is declared a leaf. For every leaf, we store the proportion of object *vs*. background patches as an indicator of the node's specificity for the given object class, as well as the spatial distribution of the offsets that reached the leaf in a non-parametric model.

Testing. The Hough Forest framework is based on the Generalized Hough Transform. In this paper we follow the probabilistic framework of [17]. Given a novel image, we first compute the same feature channels as during training. The next step is to extract patches from the image and its corresponding feature channels. As in [14], we sample the image in a pixel-wise grid with patches of size 16×16.

Let f be an extracted patch at location $\boldsymbol{\lambda}$ with appearance $\mathcal{I}(\boldsymbol{\lambda})$, namely the feature channel values. This patch is matched with visual word $\mathcal{L}$ by passing it through a Hough Tree, as each leaf represents a visual word. When the leaf $\mathcal{L}$ is activated, it casts votes for a possible object center at positions $\mathbf{x}$ with probabilities $p(\mathbf{x}|\boldsymbol{\lambda}, \mathcal{L})$. Those probabilities are estimated by the proportion $C_{\mathcal{L}}$ of object patches

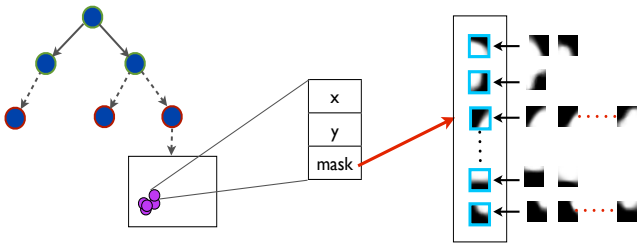


Figure 1. Visualization of the efficient top-down segmentation procedure proposed in this paper. Each leaf node of the Hough Forest contains a list of stored voting locations, together with an index to a segmentation codebook. This codebook stores local figure-ground labels that are back-projected to the image.

that were stored in leaf $\mathcal{L}$, divided by the total number of patches $D_{\mathcal{L}}$ in this leaf. Extending the case to a forest with T trees, the voting can formally be expressed as:

$$p(\mathbf{x}|\boldsymbol{\lambda}, f) = \sum_{t=1}^{T} p(\mathbf{x}|\boldsymbol{\lambda}, \mathcal{L}_t) \tag{3}$$
$$= \sum_{t=1}^{T} \left[\frac{1}{|D_{\mathcal{L}_t}|} \sum_{\mathbf{d} \in \mathbf{d}_{\mathcal{L}_t}} \exp\left\{ \frac{\|(\boldsymbol{\lambda} - \mathbf{x}) - \mathbf{d}\|^2}{2\sigma^2} \right\} \right] \cdot C_{\mathcal{L}_t}$$

The patches that are able to vote for an image location $\mathbf{x}$ are limited to a certain region around this location, and this region depends on the trained class. In Section 4, we will present an extension that takes advantage of this property to speed up detection. When all patches in the image have traversed all the trees in the forest and all votes have been cast, we smooth the obtained Hough space with a Gaussian of scale σ.

Hypothesis Selection. The final step of the algorithm is to extract maxima in the smoothed Hough space using non-maximum suppression (NMS). The maxima from the Hough space correspond to hypotheses for the object center position and thus to bounding boxes. In order to detect objects of different sizes, the test image is processed at multiple scales. The verification of hypotheses is performed by checking the overlap between two hypotheses. In line with [14], we use the bounding box intersection-over-union criterion (IoU) [9] here.

Insights in the training and testing phase. The standard training procedure for object detectors is to take an off-the-shelf training dataset or to create a new one that better captures the implicit structure of the target class. The main effort is spent on the positive images in order to obtain specific properties, such as a large number of object instances, large variance between the instances, *etc*. On the other hand, the usual criterion for choosing a negative dataset is simply that it should not contain instances of the target class. [26] has shown for holistic detectors that this is not always optimal. Following their example, we integrate into the negative dataset positive examples of object instances at larger scales from those that are used in the positive set. In this way, our part-based detector becomes more robust to false positives occurring on individual body parts, a phenomenon that is more frequent in cases where the detector searches for objects across a large number of scales (see Section 5).

After the extraction of maxima in the Hough space, the hypotheses are represented as bounding boxes in the image domain. The final set of hypotheses is selected based on their Hough scores and an overlap criterion. By varying the IoU overlap threshold, hypotheses can be discarded or accepted. However, as pointed out by [25], such an approach is not optimal. Cases where objects are close to each other have large IoU values, while small bounding boxes that overlap strongly with larger ones result in small IoU values because of the difference in the size ratio. We intend to solve this problem by making decisions on a pixel level, using the probabilistic segmentation of hypotheses, inferred by the method described in the next section.

3. Inferring top-down segmentations

The goal of the Hough Forest top-down segmentation framework is to generate a figure-ground segmentation for each hypothesis. The segmentation is expressed in a probabilistic way, meaning each pixel of the hypothesis has a certain probability of belonging to foreground or background.

3.1. Efficient Top-Down Segmentation

Training. In order to incorporate the top-down segmentation capabilities into the Hough Forest framework, it is necessary to record additional information about the votes that are stored in the leaves of the trees. This additional information comes in the form of a local figure-ground mask that corresponds to the patch sampled from the training images. This means that figure-ground segmentation masks are required for the training images. However this is not a drawback anymore, since many publicly available datasets provide fine [17, 26] or coarse [22] segmentations. Moreover, the use of Amazon Mechanical Turk [23] has opened new possibilities in the annotation process.

As a consequence of working with densely sampled image patches, storage and processing of the segmentation masks for every extracted patch requires a considerable effort in terms of memory and computation time. For this reason, we propose the generation of a mask vocabulary, based on a random subset of figure-ground patches, which are clustered using average-link agglomerative clustering with Normalized Grayscale Correlation (NGC) and a cut-off threshold set to 0.7. The cluster centers will be used as figure-ground visual words. Therefore, when a patch is extracted from a training image, the corresponding figure-ground mask is matched to a vocabulary entry and we can store only the ID of the entry (Fig. 1). In this way we reduce

the amount of memory that is used for the figure-ground masks storage.

Testing. In the voting phase of the algorithm, we first follow the same procedure as described in Sec. 2. Top-down segmentations are obtained from the back-projection of the votes that caused a maximum in the Hough space. Once a maximum is found, we retrieve all information from the votes that contributed to this maximum, namely the contribution of each vote, the position of the patch the vote was cast from, and the index to the stored mask vocabulary entry.

Knowing the position from where a vote came and its contribution, we back-project the figure-ground mask to the image space and weight it according to the corresponding vote's contribution, similar to [17]. Repeating the same procedure for every vote in a Hough-space maximum will result in a figure-ground segmentation for the hypothesis. In particular, we compute the figure and ground probabilities for each pixel $\mathbf{p}$ based on the patches $(\boldsymbol{\lambda}, f)$ containing that pixel, which were back-projected from the hypothesis. Knowing the weight and the matched figure-ground vocabulary entry for the vote that contributed to a hypothesis h at position $\mathbf{x}_h$, we derive the *figure* and *ground* probabilities as follows [17]:

$$p(\mathbf{p}=fig|\mathbf{x}_h) = \sum_{(\boldsymbol{\lambda},f)\ni\mathbf{p}} p(\mathbf{p}=fig|\boldsymbol{\lambda},f)p(\boldsymbol{\lambda},f|\mathbf{x}_h) \quad (4)$$

$$p(\mathbf{p}=gnd|\mathbf{x}_h) = \sum_{(\boldsymbol{\lambda},f)\ni\mathbf{p}} (1-p(\mathbf{p}=fig|\boldsymbol{\lambda},f))p(\boldsymbol{\lambda},f|\mathbf{x}_h) \quad (5)$$

where $p(\mathbf{p} = fig|\boldsymbol{\lambda}, f)$ is obtained from the corresponding pixel in the stored figure-ground mask. Effectively, this procedure can be realized by iteratively adding the back-projected figure-ground patches to an initially empty result image, weighted by the contribution of the corresponding votes. This can be implemented extremely efficiently on today's graphics cards. The final segmentation is then obtained as the pixel-wise likelihood ratio between the *figure* and *ground* probability maps. Some example segmentation results are shown in Fig. 2.

In the following, we denote the probability maps by $p(fig|h)$ and $p(gnd|h)$, respectively. The area Seg_h where $p(fig|h)$ is larger than $p(gnd|h)$ is considered to be the object area and this region will be used for solving the final hypothesis selection problem.

3.2. Segmentation-based Hypothesis Verification

[17] introduced a quadratic binary optimization procedure for non-maximum suppression based on the MDL principle. The main idea behind this approach was to distribute a hypothesis' score over its supporting pixels in the form of the $p(fig|h)$ probabilities, while enforcing that every pixel can only contribute to a single hypothesis. In this section, we revisit this procedure and examine how to adapt it for densely sampled image features.

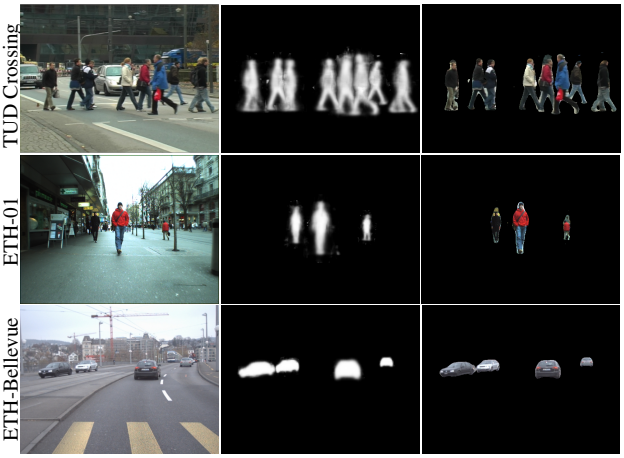


Figure 2. Example results of the Hough Forests top-down segmentation on different test sequences used in this paper.

In [17], the initial MDL score for each hypothesis was calculated based only on the *figure* probabilities $p(fig|h) \in Seg_h$. However, considering only the $p(fig|h)$ term in the Hough Forest framework results in decreased performance for some datasets. The main reason for this effect is that Hough Forests perform a dense sampling over the image and a large number of patches contribute to a hypothesis. These patches result in a large variance in the figure-ground masks, and the contribution is not proportional to the object area they cover. In particular, a patch containing only a foot of a pedestrian can have larger contribution to the Hough score than a patch that lies inside the body of the pedestrian. Nevertheless, the patch from the body will yield a higher $p(fig|h)$ contribution, as it contains more *figure* area. Hence, it is necessary to integrate also the $p(gnd|h)$ score into the total contribution. Below, we present two methods that can be used in computing the hypotheses' scores and for resolving their conflicts.

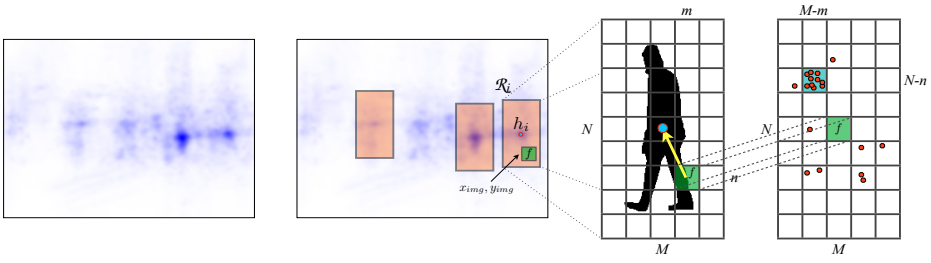
Version 1. This version of calculating scores and defining interaction terms between hypotheses is based on [17]. Once the $p(fig|h)$ and $p(gnd|h)$ probabilities have been calculated, the scores q_{ii} and interaction terms q_{ij} for the hypotheses are computed as follows:

$$q_{ii} = \frac{1}{A_{h_i}} \sum_{\mathbf{p}\in Seg_{h_i}} p(\mathbf{p} = fig|h_i) \quad (6)$$

$$q_{ij} = \frac{1}{A_{h_k}} \sum_{\mathbf{p}\in O_{ij}} p(\mathbf{p} = fig|h_k), \quad (7)$$

where $h_k, k \in \{i, j\}$ is the hypothesis with the smaller score, A_{h_k} is the expected area for this hypothesis (in our experiments, this term is equal to the size of the hypothesis' bounding box), and $O_{ij} = Seg_{h_i} \cap Seg_{h_j}$ is the overlapping area between the two hypotheses.

Version 2. The main idea behind our improved scoring scheme is to distribute the hypothesis score over *all* pixels that contributed to the Hough score. That is, we also



Hough Space of one tree | Search Space $\mathcal{A}$ | Binned Rectangle $\mathcal{R}_i$ for hypothesis h_i | Binned Leaf Distribution

Figure 3. Cascaded Hough Forests with Binned Voting. The first tree defines the regions of interest around possible hypotheses. Each of those regions and the leaf distributions are subdivided into bins. Patches belonging to one bin in a hypothesis region cast only the votes that contribute to the center of the region.

consider the contribution to a hypothesis' Hough score that was caused by background pixels around the object's border, which may be particularly discriminative.

The initial score for each hypothesis is found by back-projecting the votes around a Hough maximum (we define a 5×5 box kernel $\Phi(h)$ in our current implementation), and adding their contribution, resulting in a score similar to the Hough score. In fact, given our new definition, the score q_{ii} assigned to hypothesis h_i is the sum of the Hough scores of the bins in the hypothesis neighborhood $\Phi(h_i)$ before the Gaussian smoothing, multiplied by the patch size in pixels. In parallel, we compute the top down segmentation, as described in the previous subsection. Having an initial score for each hypothesis, the segmentation Seg_h, and the $p(fig|h)$ probabilities, the calculation of each pixel's contribution to the score is feasible.

When two hypotheses h_i, h_j overlap in an area O_{ij}, they compete for the assignment of the overlapping pixels. In the current framework, we assume that the hypothesis with the larger score is in front of the other one. Thus, if h_i has larger score, we want to penalize hypothesis h_j for the overlapping area. The part of $p(fig|h_j) \in O_{ij}$ is an indicator of how much we should penalize the second hypothesis, as these pixels will be assigned to h_i. Since the hypothesis score $q_{jj} = \phi(h_j)$ from which this penalty is subtracted now also includes a ground contribution $p(gnd|h_j)$, we need to introduce a weighting factor r, which scales the penalty according to the total contribution $\phi(h_j)$:

$$q_{ii} = \phi(h_i) = \sum_{\mathbf{p} \in Area_{h_i}} (p(\mathbf{p}=fig|h) + p(\mathbf{p}=gnd|h)) \quad (8)$$

$$q_{ij} = \sum_{\mathbf{p} \in O_{ij}} p(\mathbf{p}=fig|h_k) \cdot r, \quad (9)$$

where $Area_{h_i}$ describes the pixels where $figure$ and/or $ground$ probability exist and h_k is the hypothesis with the smaller score and

$$r = \frac{\phi(h_k)}{\sum_{\mathbf{p} \in Seg_{h_k}} p(\mathbf{p} = fig|h_k)} . \quad (10)$$

The final hypothesis selection for both versions is performed using the greedy search algorithm described in [17], which solves a quadratic binary optimization problem $\max_m m^T Q m$, with the interaction matrix $Q = \{q_{ij}\}$ and the indicator vector $m \in \{0,1\}^N$ (a global maximum solution is not always feasible).

The choice of the scoring scheme depends on the specific problem. In scenarios with constant background and large overlap between the objects, the first scoring scheme (MDL1) performs better. On the other hand, MDL2 is more suitable for most practical scenarios. A more extensive evaluation of the two scoring schemes is presented in Sec. 5.

4. Cascaded Hough Forests

The general Hough Forest framework processes the image in the following order: feature extraction, tree traversal, voting, and post-processing of the Hough space. During our experiments, we noticed that the most expensive part of the pipeline is the voting stage. Moreover, the time for voting increases with the number of trees and the number of patches that were used for training. In our approach, the voting and post processing steps require additional time and memory, because for each vote we store its contribution, the position from where the vote was cast, and the index to the associated mask vocabulary entry. Given that we sample the image in a pixel-wise grid and we typically use 15 trees, the number of votes that are cast to the Hough space is considerable. Reducing the number of training samples per tree or performing voting with fewer trees decrease the performance, especially in the complex datasets. Therefore, a more sophisticated solution that reduces computation without losing performance is necessary.

4.1. Cascaded Tree Voting

The cascaded voting scheme builds upon two ideas: 1) The first tree of the forest is able to roughly locate all the true positives, plus a number of false positives whose relative Hough scores will be reduced by the accumulated votes of the next trees. 2) The region of the image that can provide votes (support) to a hypothesis is restricted to the neighborhood around the hypothesis. Therefore, the first tree of the forest can be used as an indicator in order to find possible

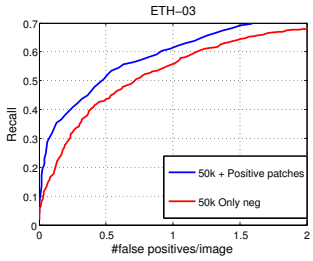


Figure 4. The performance of the Hough Forest detector increases with the integration of true positives at different scales into the negative dataset (50k patches for building a tree).

hypotheses and the remaining trees in the forest only need to process the regions around those neighborhoods.

The back-projection of the votes that contributed to a Hough maximum indicate the support of the hypothesis in the image domain. The contributing patches come from a region around the Hough maximum: During training, the patches are sampled from the training images in order to represent part of the object instance. The information from where we are able to sample patches is offered as bounding boxes or segmentation masks. However, the range of poses of objects such as pedestrians or cars is limited. Therefore, there is a rectangle $\mathcal{R}$ that bounds all the patch locations and consequently the votes in the leaf distributions.

On the other hand, an image patch can vote only in its neighboring region. This region is defined by the rectangle $\mathcal{R}$. Therefore, once the first tree finds a set of hypotheses $\mathcal{H} = \{h_i\}$, we place the rectangles $\mathcal{R}_i$ to the corresponding positions of h_i. The union of all rectangles $\mathcal{R}_i$ generates the area $\mathcal{A}$ that will serve as the region of interest for the later trees in the forest, reducing the computation time spent on patches traversing the trees and casting votes. In addition, the memory requirements are reduced, as the total number of votes is decreased significantly. The first two images from Fig. 3 visualize the proposed procedure.

4.2. Binned Voting

The cascaded scheme and the introduction of the rectangles $\mathcal{R}_i$ around possible hypotheses h_i allow for a further clipping of votes. In order to speed up the voting process for the remaining trees in the forest, we subdivide the minimum rectangle $\mathcal{R}_i$ into $M \times N$ bins. This rectangle bounds all the votes stored in the leaves of the Hough Forest, as well as the patches that can contribute to one hypothesis. Therefore, it can be used as a common framework for the bin parameterization of a patch's location relative to the object center, as well as for the votes in the leaf distributions.

Within the binned framework, the center of the rectangle $\mathcal{R}_i$ and a patch inside the rectangle are assigned to a bin. Once the patch is matched to a leaf in a Hough tree, votes are cast according to the stored spatial distribution. The votes that are cast inside $\mathcal{R}_i$ correspond to several bins. However, only the votes that are located in the bin of the center of $\mathcal{R}_i$ will contribute to the particular hypothesis.

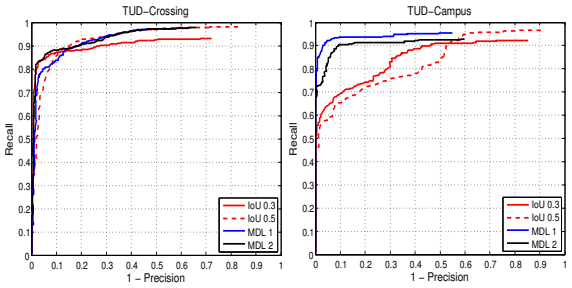


Figure 5. Comparison between different post processing methods (MDL1, MDL2, and IoU with thresholds 0.3 and 0.5) on the TUD Crossing and Campus datasets.

The votes that belong to one bin in the rectangle $\mathcal{R}_i$, also belong to one bin in the binned leaf spatial distribution. The goal of binned voting is to accept only the votes that lie inside the bin where the center of the rectangle is located. In order to find which bin in the leaf distribution corresponds to the rectangle center's bin, we use the relative position between the bin position of the patch and the bin location of the center. This mapping is necessary, because the centers of the spatial distributions in the leaves represent the part locations, while the center of rectangle $\mathcal{R}_i$ is the center of a possible object. Once the respective bin in the leaf distribution is found, we cast only the votes that belong to that bin and discard the others, as shown in Fig. 3(right).

5. Experimental Results

Training Sets. For the side-view pedestrians, we use the TUD dataset that was also used in [14, 2]. This training dataset contains 400 images of pedestrians, which we mirrored to get 800 training examples. As negative images, we used a random subset from the background images of the INRIA dataset [6], plus the negative pedestrian examples from [26] and a random subset of our true positives at larger scales (518 images total). In this way, the system is more robust to false positives generated by structures that belong to parts of the object (Fig. 4).

For the multi-view pedestrians, we integrate the TUD dataset (0° side views), with a multi-view pedestrian dataset of our own (198 images total, consisting of 55 images at 45°, 76 at 90/270°, and 67 at 135°). The negative set is the same as for the side view case. Note that the dataset is not optimal, as the number of images per view is not balanced, ranging from 55 to 400 examples.

For the multi-view car forest we used the training images from [8] (7 viewpoints, 1279 images in total) and the forest was trained similar to [21].

Test Sets. Fig. 2 lists the datasets we used for testing. The *TUD Crossing/Campus* datasets contain side views of pedestrians in different scales, and consist of 201 and 71 images respectively. The other pedestrian dataset we used in our evaluation is *ETH Person*, which contains three se-

Recall	IoU	Precision
72.15%	62.54%	82.45%

Table 1. Segmentation results on the TUD-Crossing sequence (detection EER).

quences of 999, 450, and 354 images. The bounding box used for pedestrian detection is 40×100 pixels.

In addition, we examine the behavior of the our method on the multi-view car dataset *ETH-Bellevue* from [8]. This sequence (377 images) contains 1591 annotated cars as small as 20 pixels height.

Top-down Segmentation. Fig. 2 shows example segmentations obtained with our approach (more results and videos are given in the supplementary material). In order to quantitatively evaluate the segmentation performance, we manually created ground truth segmentations for every 10^{th} frame of the TUD-Crossing sequence and applied the evaluation measures from [9]. Tab. 1 shows the corresponding segmentation performance at the approach's detection Equal-Error-Rate (EER) point.

Segmentation-based Verification. Next, we compare the performance of our top down segmentation approaches against the IoU bounding box criterion (Fig. 5).

On the *TUD Crossing* sequence, the pedestrians can be detected using only three scales and the number of overlapping pedestrians is large. Both MDL 1 and MDL 2 perform well, with MDL 2 showing better performance in the high-precision range. Here the performance of IoU 0.3 is saturated at 93% recall, because the overlap threshold rejects hypotheses close to each other.

In *TUD Campus*, 5 scales were used and the improvement by the top-down segmentation is clearly visible. True positives at small scales are correctly accepted, while false positives caused by parts of pedestrians are removed. In contrast, the IoU criterion is not able to handle strong occlusions between objects, while rejecting the false positives.

Finally, we tested the performance of our system on the *TUD Crossing* and *TUD Campus* sequences with the extended annotations of [3]. This annotation set includes every pedestrian whose head and at least one leg is visible. As Fig. 6 shows, our approach performs equally well on those more challenging annotations. For the *TUD Crossing* sequence, both MDL1 and MDL2 outperform [3]. On the *TUD Campus* sequence, MDL1 gives better results than [3] in the high-precision regime, while MDL2 performs comparably in this range. Both approaches however do not reach the very high levels of recall that [3] can achieve.

From the above experiments we can conclude that MDL1 is suitable for scenarios with relatively simple backgrounds, and objects suffering from strong occlusions. On the other hand, MDL2 is more stable in dynamic environments, where it can reliably reject false positives on the background, due to the more discriminative Hough score. For this reason, we choose MDL2 for all subsequent experiments.

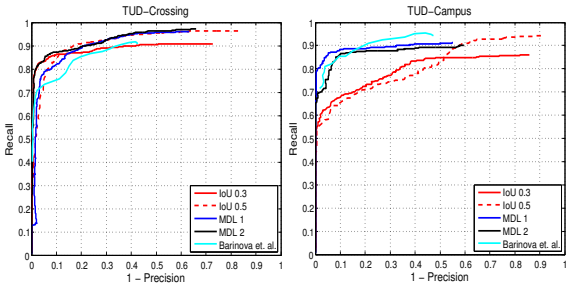


Figure 6. Effect of MDL1 and MDL2 on the updated annotations of [3] for Crossing and Campus datasets.

	Single View		Multi View	
	Original	MDL	Original	MDL
Original	14.97 s	34.21 s	41.87 s	143.09 s
Cascaded	12.20 s	24.05 s	32.60 s	38.51 s
Binned	8.49 s	14.90 s	30.22 s	31.98 s

Table 2. Times for the voting step for the different voting schemes (single-view: TUD Crossing; multi-view: ETH-Bellevue cars).

Cascaded Voting. In the next round of experiments, we demonstrate the gain of the cascaded and binned voting schemes on the ETH-Person dataset, using the same evaluation protocol and annotations as [26]. Fig. 7 shows the results of this experiment. We observe that the binned voting procedure does not introduce any loss in performance (IoU 0.3 and IoU 0.3 Binned curves). Comparing Binned MDL2 with the results of the original ISM detector [7], we clearly see that our framework outperforms the sparse feature based version in all sequences. In order to relate our approach's performance to that of a more recent state-of-the-art detector, we plot the HOG+HIKSVM performance [26]. As Fig. 7 shows, MDL2 performs only slightly worse than HOG+HIKSVM on ETH-02, but outperforms the latter on ETH-01 and ETH-03.

Multi-View Case. For this experiment, we used a multi-view Hough Forest. Fig. 7(d) shows the performance of MDL2 with cascaded voting compared to IoU 0.5 with the same voting scheme on the *ETH-Bellevue* sequence. Again, these results confirm the improvement of MDL2 compared to the simpler IoU criterion. In addition, they show that our framework can be successfully extended for multi-view/multi-class object detection.

Runtime Improvement. As pointed out before, the voting stage is currently the most expensive part of the algorithm. In order to quantify the improvement of cascaded voting, we use the single-view pedestrian and multi-view car Hough Forest detectors and measure the time spent on casting the votes (the other parts are almost negligible). Table 2 shows that both the cascaded and the binned voting bring considerable improvements in our current (unoptimized) implementation, rendering the more expensive MDL procedure affordable.

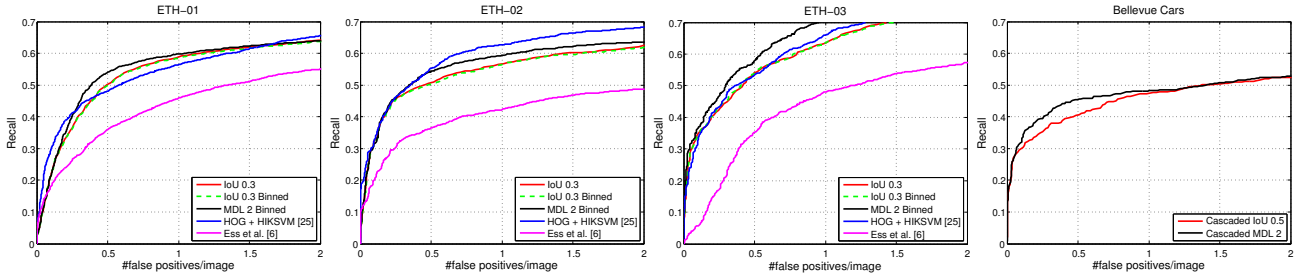


Figure 7. Performance of our approach and comparison of the proposed framework against [26, 7] on the ETH Person dataset.

6. Conclusion

In this paper, we have proposed two improvements to the Hough Forest detection framework. The first contribution is a way to integrate the ISM top-down segmentation capabilities into Hough Forest detectors. We have shown that the use of densely sampled image features requires several adaptations to the segmentation framework and proposed an improved hypothesis selection strategy building upon the segmentation results. Our second contribution is a cascaded voting strategy that reduces the effort of the Hough voting stage without loss in detection accuracy. Both improvements are general and can be readily integrated with other recent Hough Forest extensions [3, 21]. As our experimental results have shown, the resulting detector is competitive with current state-of-the-art detectors such as HOG+HIKSVM.

Acknowledgments. This project has been funded, in parts, by the EU project EUROPA (ICT-2008-231888) and the cluster of excellence UMIC (DFG EXC 89).

References

[1] N. Ahuja and S. Todorovic. Connected Segmentation Tree - A joint representation of region layout and hierarchy. In *CVPR*, 2008.

[2] M. Andriluka, S. Roth, and B. Schiele. People Tracking-by-Detection and People Detection-by-Tracking. In *CVPR*, 2008.

[3] O. Barinova, V. Lempitsky, and P. Kohli. On the Detection of Multiple Object Instances using Hough Transforms. In *CVPR*, 2010.

[4] L. Bourdev and J. Malik. Poselets: Body Parts Detectors Trained Using 3D Human Pose Annotations. In *ICCV*, 2009.

[5] L. Breiman and E. Schapire. Random forests. In *Machine Learning*, pages 5–32, 2001.

[6] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005.

[7] A. Ess, B. Leibe, and L. Van Gool. Depth and Appearance for Mobile Scene Analysis. In *ICCV*, 2007.

[8] A. Ess, K. Schindler, B. Leibe, and L. Van Gool. Object Detection and Tracking for Autonomous Navigation in Dynamic Environments. *IJRR*, 29(14):1707–1725, 2010.

[9] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 88(2):303–338, 2010.

[10] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade Object Detection with Deformable Part Models. In *CVPR*, 2010.

[11] P. Felzenszwalb and D. Huttenlocher. Pictorial Structures for Object Recognition. *IJCV*, 61(1), 2005.

[12] P. Felzenszwalb, D. McAllester, and D. Ramanan. A Discriminatively Trained, Multiscale, Deformable Part Model. In *CVPR*, 2008.

[13] R. Fergus, P. Perona, and A. Zisserman. A Sparse Object Category Model for Efficient Learning and Exhaustive Recognition. In *CVPR*, 2005.

[14] J. Gall and V. Lempitsky. Class-Specific Hough Forests for Object Detection. In *CVPR*, 2009.

[15] C. Galleguillos, B. Babenko, A. Rabinovich, and S. Belongie. Weakly Supervised Object Recognition and Localization with Stable Segmentations. In *ECCV*, 2008.

[16] D. Larlus, J. Verbeek, and F. Jurie. Category Level Object Segmentation by Combining Bag-of-Words Models and Markov Random Fields. In *CVPR*, 2008.

[17] B. Leibe, A. Leonardis, and B. Schiele. Robust Object Detection with Interleaved Categorization and Segmentation. *IJCV*, 77(1-3):259–289, 2008.

[18] F. Li, J. Carreira, and C. Sminchisescu. Object Recognition as Ranking Holistic Figure-Ground Hypotheses. In *CVPR*, 2010.

[19] S. Maji and J. Malik. Object Detection using a Max-Margin Hough Transform. In *CVPR*, 2009.

[20] M. Marszalek and C. Schmid. Accurate Object Localization with Shape Masks. In *CVPR*, 2007.

[21] N. Razavi, J. Gall, and L. Van Gool. Backprojection Revisited: Scalable Multi-view Object Detection and Similarity Metrics for Detections. In *ECCV*, 2010.

[22] B. Russell, A. Torralba, K. Murphy, and W. Freeman. LabelMe: A Database and Web-based Tool for Image Annotation. *IJCV*, 77(1-3):257–173, 2008.

[23] A. Sorokin and D. Forsyth. Utility Data Annotation with Amazon Mechanical Turk. In *CVPR'08 Workshop on Internet Vision*, 2008.

[24] P. Viola and M. Jones. Robust Real-Time Face Detection. *IJCV*, 57(2), 2004.

[25] S. Walk, N. Majer, K. Schindler, and B. Schiele. New Features and Insights for Pedestrian Detection. In *CVPR*, 2010.

[26] C. Wojek, S. Walk, and B. Schiele. Multi-Cue Onboard Pedestrian Detection. In *CVPR*, 2009.