Fast growing hough forest as a stable model for object detection

Antoine Tran, Antoine Manzanera

Fast Growing Hough Forest as a Stable Model for Object Detection

Antoine Tran, Antoine Manzanera
U2IS, ENSTA ParisTech
Université de Paris-Saclay
828, Bd des Maréchaux
91762 Palaiseau CEDEX - France
Email: antoine.[tran,manzanera]@ensta-paristech.fr

***Abstract*—Hough Forest is a framework combining Hough Transform and Random Forest for object detection. The purpose of the present paper is to improve the efficiency and reliability of the original framework by the mean of two contributions. First, instead of generating the image samples by drawing patches randomly from the training set, we bias this step toward the most relevant image content by selecting a proportion of patches from a geometrical criterion. Second, during the creation of non-leaf-nodes of the trees, instead of sampling uniformly the parameter space for choosing the binary tests aimed at splitting the set of image samples, we choose them according to a probability map constructed from the sample set. We aim to drastically reduce the training time without impacting the accuracy, and decreasing the variability of the produced detectors. The interest of this improved model is shown in the context of car and pedestrian detection by evaluating it on academic datasets.**

I. Introduction

Object detection is a fundamental and difficult problem of computer vision. In our context it consists in localising objects of a certain class in an input image. Typical applications of object detection are video-surveillance and image retrieval. For one given object class (such as cars or humans), the variability of all instances, due to pose, shape, and appearance changes within this class, makes the object representation very challenging. Moreover, even for a single instance, occlusions, illumination and point of view changes also make the object localisation a difficult task.

The principle of the Hough Transform is to consider a set of elements from an image (such as pixels, patches or interest points), and to make them vote for a set of parameters (related to position, scale or orientation). Historically, Hough [1] used contour points to detect lines, circles or ellipses. In object detection context, Leibe and al. [2] used a keypoint representation in their *Implicit Shape Model* (ISM) algorithm. During the training step, keypoints from example objects are collected and quantised into a codebook. Every keypoint is also associated to a displacement vector corresponding to its position with respect to the object centroid. During the detection step, all keypoints found in the image are matched with an entry from the codebook, and then vote for all displacements stored in the entry. Finally, a peak detection in the vote collecting map (the Hough transform) is used to detect objects. This detector has proven effective against occlusion and illumination change; it can cope with scale and appearance variations. However, the object representation is sparse, and to be effective, a codebook requires a large number of entries, which implies a time-consuming matching process.

On the other hand, Random Forest classifier [3] is popular in object detection. Moosmann [4] used it for multi-class detection, using visual descriptor vectors to train the random trees. Then, he used it to generate class-specific saliency maps on unknown images, obtained from sampling patches from the image.

Gall [5] proposed a patch-based appearance method, providing a denser level of representation. By using a Random Forest [3] framework associated to a sequence of binary tests related to local geometry, his algorithm is capable to detect different kinds of objects (pedestrian, cars, horse) with a low time-consuming detection step. His training step consists in generating decision trees, which recursively partition an input set of positive and negative patches. Each non-leaf node contains a binary test, while each leaf node contains displacement vectors built according to positive patches positions. During the detection step, each pixel of the image is described by its neighbouring patch, which goes through every decision tree, until it reaches a leaf. The patch then casts votes according to the displacements stored in the leaf. Hough Forest has proven a solid algorithm, giving good results on several academic datasets. It is also a strong basis for diverse enhancements: Ciolini [6] proposed different simplifications during the training and detection steps to be able to process the algorithm in embedded systems, while maintaining accuracy performances. Murai [7] proposed a method to add a weight on patches during training step, reducing the weights of positive patches which are too similar to negative ones, during the training step, in order to improve the accuracy of detection. Gall [8] also extended his original Hough Forest algorithm to multi-class and multi-view object detection.

Even though Hough Forest is a strong and versatile algorithm, training a whole forest currently lasts a couple of days following the recommended parameters in a mid-range desktop machine, which makes the parametric study and evaluation of the algorithm a complicated task. Simply decreasing some parameters may reduce the training time, however, due to the high level of randomness, the variance

of the resulting performance can increase dramatically. In this paper, we propose a one-class object detector based on Hough Forest, providing the following contributions:

- During the sampling step, to generate the positive and negative patch set for the training step, we use a geometric criterion to draw some patches in a deterministic way.
- For the construction of the Hough trees, we propose a way to drastically reduce the number of potential nodes by considering all patches used to train the non-leaf node, and generate probability maps from them.

The main objective of our contribution is to reduce the variance of the object models, while keeping accuracy close to the baseline Hough Forest, and with low time-consuming operations.

This paper is organised in three sections: after recalling the baseline Hough Forest algorithm, we present our contributions. Finally, experimentations on academic datasets will be presented and discussed.

II. Hough Forest

Random Forests [3] belong to the random decision tree classification techniques. For classic random decision trees, bagging (generating a subset of training data by randomly sampling with replacement from the full training set) is applied to increase the representativeness of the model. The difference between classic random decision trees and random forest is the use of attribute bagging (randomly choosing the feature from the full feature set).

The Hough Forest is made of binary decision trees, built from a set $\mathbf{P}$ of training patches of a constant size $W \times H$, sampled from the learning (positive and negative) images using a uniform random process. $\mathbf{P}$ is recursively partitioned during the training phase, and its resulting subsets, corresponding to the leaves of the tree, form the input of the voting process. Given a patch set $\mathbf{P}$, let $\mathbf{P}^+$ and $\mathbf{P}^-$ denote the subsets of $\mathbf{P}$ coming from the positive and negative images respectively.

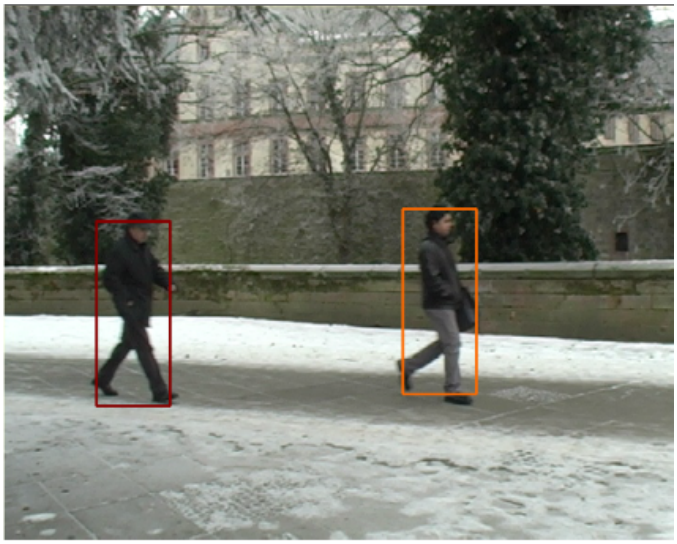
Each patch is sampled from a colour image, described with these feature channels:

- 3 components from the *Lab* colour space
- 2 absolute first derivatives: $|\frac{\partial L}{\partial x}|$ and $|\frac{\partial L}{\partial y}|$
- 2 absolute second derivatives: $|\frac{\partial^2 L}{\partial x^2}|$, $|\frac{\partial^2 L}{\partial y^2}|$
- 9 bins from the HOG [9] descriptor.

Every channel is additionally spatially enhanced by a min (erosion) and a max (dilation) filter, which finally provides $2 \cdot (3 + 2 + 2 + 9) = 32$ features. Each positive patch π is also described with the displacement d_π from its centre to the object's centroid.

Each non-leaf node is identified to a binary test T using as input a patch π, and defined from these four parameters:

- 2 pixels $(p^l, p^r) \in \pi^2$
- 1 feature channel c
- 1 threshold value τ



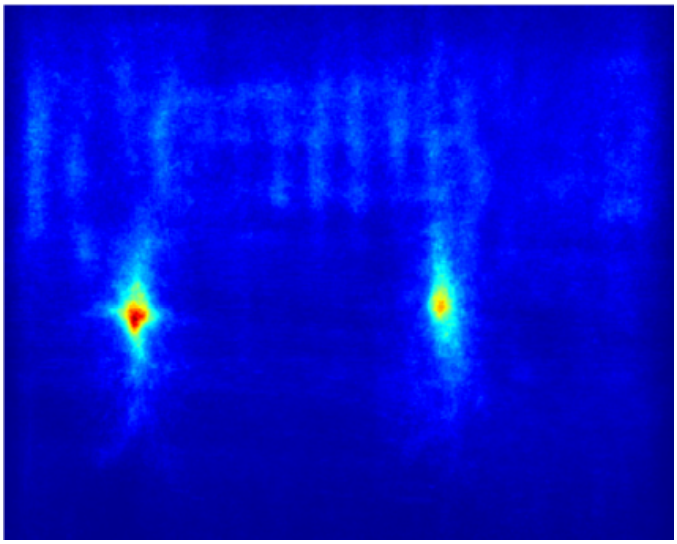


Fig. 1. Vote on one image from TUD-Pedestrian dataset [9]. The two peaks correspond to the two pedestrians.

with the following expression:

$$T(\pi) = \begin{cases} 0 & \text{if } c(p^l) > c(p^r) + \tau \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

A test T will be specified as $\{p^l, p^r, c, \tau\}$ for the remaining of this paper.

The creation of a test (i.e. a non-leaf node) T works as follows. Let $\mathbf{P}_T$ denote the subset of $\mathbf{P}$, input of the node T. The purpose is to find the test T that realises an optimal partition of $\mathbf{P}_T$ into two subsets. The training step starts by generating N binary tests $\{T_i\}_{i\in\{1,\dots,N\}}$, with, for each i, p_i^l, p_i^r, c_i drawn randomly (uniform process), and τ_i drawn using a random uniform process in the range formed by the min and the max of $\{c_i(p_i^l) - c_i(p_i^r)\}_{i\in\{1,\dots,N\}}$.

Each potential binary test T_i generates two disjoint subsets $\mathbf{P}_{T_i}^t = \{\pi \in \mathbf{P}_T | T_i(\pi) = t\}$ with $t \in \{0, 1\}$. A score

evaluating the quality of the test is then calculated, using either of the two following criteria, chosen randomly:

- One related to the class (positive or negative) repartition of the patches in the two subsets, based on the Shannon entropy H:

$$U_e(T_i) = |\mathbf{P}^0_{T_i}| \cdot H(\mathbf{P}^0_{T_i}) + |\mathbf{P}^1_{T_i}| \cdot H(\mathbf{P}^1_{T_i}) \quad (2)$$

  where $H(\mathbf{P}) = -\frac{|\mathbf{P}^+|}{|\mathbf{P}|}\log\left(\frac{|\mathbf{P}^+|}{|\mathbf{P}|}\right) - \frac{|\mathbf{P}^-|}{|\mathbf{P}|}\log\left(\frac{|\mathbf{P}^-|}{|\mathbf{P}|}\right)$, and $|\mathbf{S}|$ denotes the cardinal of set $\mathbf{S}$.
- One related to the variance of displacements associated to positive patches within the two subsets:

$$U_d(T_i) = \sum_{t\in\{0,1\}} \sum_{\pi\in(\mathbf{P}^t_{T_i})^+} (d_\pi - d_{(\mathbf{P}^t_{T_i})^+})^2 \quad (3)$$

  where $d_{\mathbf{S}}$ is the average displacement on set $\mathbf{S}$.

Finally, the best test T_{opt} is the one with the lowest score, whatever the chosen criterion.

A node is declared as a leaf if it reaches the maximal tree depth ($D_{\max} = 15$) or if the number of positive input patches is too small ($N_{\min} = 20$). A leaf contains all the displacements from the positive patches, and a weight given by $\omega_L = \frac{|\mathbf{P}^+_L|}{|\mathbf{P}_L|}$, i.e. the proportion of positive patches in the leaf L.

During the detection step, given an image test I, patches are densely extracted from I. Each patch goes through all trees of the forest, and when it reaches a leaf node, votes for all displacements stored in the leaf, with a vote weight equal to $\omega_L/|\mathbf{P}^+_L| = 1/|\mathbf{P}_L|$, such that the cumulative weight for all the votes casted from a leaf L is finally ω_L. Finally, the detection output is provided by the set of maxima into the vote map smoothed by a Gaussian filter as in [5], or by using Mean-Shift [10] algorithm, as in [11]. To handle scale variations, Gall also proposed [5] to rescale I to different sizes, and then perform the maxima search in the scale-space dimension.

III. Our contribution

Hough Forest has proven an effective and versatile framework for object detection. However, considering parameters in [5] (forest composed of 15 trees, and 20 000 candidate binary tests to create a non-leaf node), the training step is extremely time consuming. Our goal is to evaluate the performance of the algorithm with those critical parameters significantly lower (9 trees and 500 binary tests), and to propose modifications in the original Hough Forest to reduce variance in these conditions. Our two contributions provide improvements at the following levels:

- During the patch sampling step, instead of using a spatially uniform random process, a proportion of the patches is chosen in a deterministic way, using a multi-scale saliency map [12]. This guarantees the presence of relevant samples, whatever the content of the training image set.
- During the node training step, we improve the efficiency of the patch partition by drawing the binary tests according to a probability map constructed from the current patch collection, instead of choosing them by uniform random selection.

A. Patch sampling

In the baseline method [5], the patches are randomly drawn using a uniform probability on the training set. The weakness of this method is the possibility to draw patches in irrelevant (e.g. homogeneous) areas. Such patches are hard to classify using binary tests defined in (Eq. 1), and are sensitive to noise.

Based on this remark, we propose a method to guarantee a minimal number of interesting patches, chosen in a deterministic way. As a saliency measure, we use the multi-scale junction-ness function [12], defined as follows: for an image I, let I^σ_x (resp. I^σ_y) denote the estimated first derivatives at scale σ, i.e. the image I convolved with the derivative of the Gaussian kernel with variance σ^2, with respect to x (resp. y). The second derivatives I^σ_{xx}, I^σ_{xy}, and I^σ_{yy} are defined accordingly. The junction-ness map of image I at scale σ is:

$$J^\sigma_I = I^\sigma_{xx}(I^\sigma_y)^2 - 2I^\sigma_x I^\sigma_y I^\sigma_{xy} + I^\sigma_{yy}(I^\sigma_x)^2 \quad (4)$$

Let N be the number of patches to be selected to train one tree. We set the level of determinism to a rate $\alpha \in [0, 1]$, as the proportion of patches to be selected with respect to their saliency, the remaining part being chosen randomly, as in the original method. In order to get structured patches at different sizes, we choose to work with N_s different scales. Then, for a given image, and for each scale, starting from the smallest one, we take the $N_p = \frac{\alpha \cdot N}{N_s}$ pixels with the highest absolute junction-ness as patch centres. To avoid patches spatially too close, we perform non-maximum suppression for each scale, by setting to 0 all pixels at a distance from the selected centre lower than the square of the current scale.

To promote the variety in the different trees of the Hough forest, the non-maximum suppression operations done for one tree remain for the next trees.

The rationale for using the multiscale junction-ness as a saliency measure for choosing the patches is that, as a second order geometric measure, it makes all the involved feature channels significant and it allows to focus on the most discriminant points while representing a negligible time overhead, once the channels calculated. Fig. 2 shows examples of junctions.

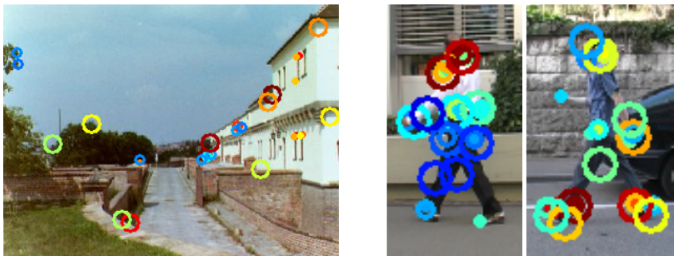


Fig. 2. Junctions shown by circles for one landscape image and two pedestrian images. The radius corresponds to the scale, and the colour to the absolute value of junction-ness (the red corresponding to the highest values).

B. Node training

In this part, we focus on the non-leaf node training process, which consists in choosing the optimal parameters

$\{p^l, p^r, c, \tau\}$ of the binary test (Eq. 1). As mentioned earlier, Gall's method [5] consists in drawing uniformly a high number (Gall mentioned 20 000) of tests, and choosing the best one. However, the number of candidate tests can only be negligible compared to the number of potential tests. Indeed, by using a unique patch size of 16×16 pixels, and considering 32 feature channels, there are $16^{2^2} \times 32 = 2\,097\,152$ potential binary tests, without mentioning the threshold, which is also randomly chosen.

Our method consists in constructing, for each node level, a particular probability law for drawing the binary tests, using the current patch collection as an input. Our goal is to dramatically reduce the number of potential binary tests required to create a node, while improving the quality of the partition with respect to the two criteria (Eq. 2 and 3).

Following the notations of Sec. II, T denotes a binary test (or a non-leaf node), which has to be created, using a patch set $\mathbf{P}_T$. Let us consider a deterministic partition of $\mathbf{P}_T$ into two subsets $\mathbf{P}_T^{\square}$ and $\mathbf{P}_T^{\blacksquare}$. The binary label denoted by the white or black square will be instanciated later: its nature depends on the criterion to optimise (entropy *vs* spatial deviation).

The first step is to create global patches (let us call them superpatches), summarising information about $\mathbf{P}_T^{\square}$ and $\mathbf{P}_T^{\blacksquare}$.

For any patch π let $\widehat{\pi}$ denote the binarised version - valued in $\{-1, +1\}$ - of π using Otsu's adaptive threshold method [13]. The role of $\widehat{\pi}$ is to provide contrast invariance, by merely indicating, for a given feature and patch, which pixels belong to the "upper" or "lower" part of the patch. Then, for $a \in \{\square, \blacksquare\}$, we define:

$$\Pi^a = \sum_{\pi \in \mathbf{P}_T^a} \widehat{\pi} \tag{5}$$

For a given label a, Π^a can be interpreted as a probability map indicating, for each feature, which pixel is more likely to be a "upper" (resp. "lower") pixel for all patches from $\mathbf{P}_T^a$. However, directly using as probability density the normalised Π^a, whose dynamics can be high, may turn the process too deterministic. We chose instead to drastically quantise the density by using the binary superpatch $\widehat{\Pi}^a$, resulting from the threshold operation of Π^a by 0 and valued in $\{-1, +1\}$.

Finally, for a binary test T, and a prior partition $\mathbf{P}_T^{\square}$ and $\mathbf{P}_T^{\blacksquare}$, we define the two superpatches Π_T^l and Π_T^r as follows:

- $\Pi_T^l = 1 + \frac{1}{2}(1 + \widehat{\Pi}^{\square}) + \frac{1}{2}(1 - \widehat{\Pi}^{\blacksquare})$
- $\Pi_T^r = 1 + \frac{1}{2}(1 + \widehat{\Pi}^{\blacksquare}) + \frac{1}{2}(1 - \widehat{\Pi}^{\square})$

Π_T^l and Π_T^r are superpatches valued in $\{1, 2, 3\}$, whose purpose is twofold:

- For a given feature channel, normalised Π_T^l (resp. Π_T^r) is used as a probability map to draw the left (resp. right) pixel member p^l (resp. p^r) used in Eq. 1.
- To select the most relevant feature channel, the (normalised) sum of the standard deviations of Π_T^l and Π_T^r is used as a probability map to draw the feature channel.

To conclude with the random selection process of the binary test, the parameters $\{p^l, p^r, c\}$ are drawn as explained above, and only the threshold τ is set the same way as in the original Gall's method.

Now the missing step consists in determining the two subsets $\mathbf{P}_T^{\square}$ and $\mathbf{P}_T^{\blacksquare}$, depending on the criterion to minimise:

- When the entropy criterion is chosen, $\{\square, \blacksquare\} = \{-, +\}$, meaning that we are using directly the labels from negative and positive example patches of $\mathbf{P}_T$.
- With the spatial deviation criterion, considering the equation (3) to minimise, we apply a k-means (with $k = 2$) clustering of the patch set $\mathbf{P}_T^+$ with respect to the displacement vector d_π. Indeed, the k-means algorithm is obviously a better way to minimise the spatial deviation criterion than Gall's random method. In this case, $\mathbf{P}_T^{\square}$ and $\mathbf{P}_T^{\blacksquare}$ are the two sets resulting of the 2-means of $\mathbf{P}_T^+$.

IV. Experiments

This section deals with the impact of our contributions by evaluating the improved models in two academic datasets:

- The UIUC-Cars dataset [14]. The training dataset is composed of 550 images of cars, approximatively of the same size (about 100×40 pixels), and 440 negative images. For the detection step, it proposes two datasets: one with 210 single scale cars in 170 images, and one with 139 cars of different scales in 108 images. All cars are taken from side viewpoints. Difficulties are due to partial occlusion, low contrast and complex backgrounds. Images are in grayscale. This dataset is only used for evaluating the impact of the patch selection; we reduce the feature channels to absolute first and second derivatives only.
- The TUD-Pedestrians dataset [15]. Its training set is composed of 400 images of people walking in different urban backgrounds, with silhouette based ground truth. Following the protocol used in [5] to compensate for the low variability of the background (4 different backgrounds), we picked negative images from the INRIA-Person dataset [9] to train our forest. We also added pedestrians images generated by y-axis symmetry from all positive images of the training set. For the detection step, the test data set is made of 250 images containing 311 pedestrians. Difficulties are due to partial occlusions, scale, pose, aspect and illumination changes.

Our objective is to show how our improved models perform with a much smaller number of potential binary tests generated for a node creation (compared to the 20 000 used for the original version). Such drastic reduction of the node parameter space sampling does not affect much the results for the UIUC-Cars dataset (all objects being similar in shape and pose). However for the TUD-Pedestrian dataset, performances can vary a lot from one experiment to the other. In this section, experiments are made following the protocols proposed by the owners of the datasets, and the results are displayed using ROC curves, with error percentage (100 - precision) in abscissa and recall percentage in ordinate.

A. Results on UIUC-Cars

For this dataset, we use a forest of 9 trees, and select 50 patches per image. Three versions of the Hough Forest were evaluated:

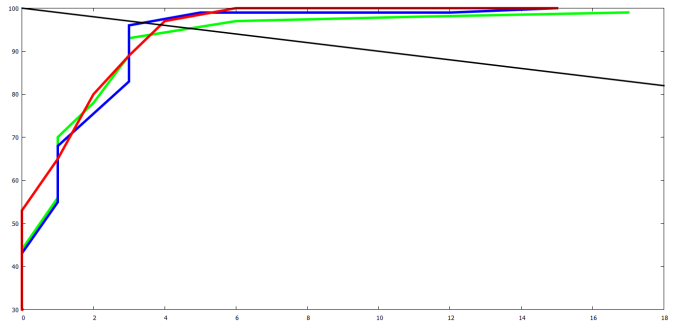


Fig. 3. ROC-curves for the UIUC-multiscale dataset. In black, the Equal-Error Rate line (EER, Precision = Recall). Green: V1, Blue: V2, Red: V3.

- V1: feature channels reduced to absolute first derivative (as Gall suggested [5])
- V2: feature channels reduced to absolute first and second derivative features
- V3: same as previous, plus our deterministic patch selection process ($\alpha = 0.1$)

Fig. 3 shows the resulting ROC-curves for the three versions on the multi-scale test dataset. It can be seen that adding the absolute second order derivative slightly improves the detector. Our model (V3) obtains the best results. However, comparing to results provided by Gall [5], particularly $EER = 98.5\%$ with 20 000 binary tests, while in our case, $EER = 96\%$ with 2 000 tests, and considering that results are very close to 100%, it turns out that this dataset is not relevant enough to evaluate our improved model.

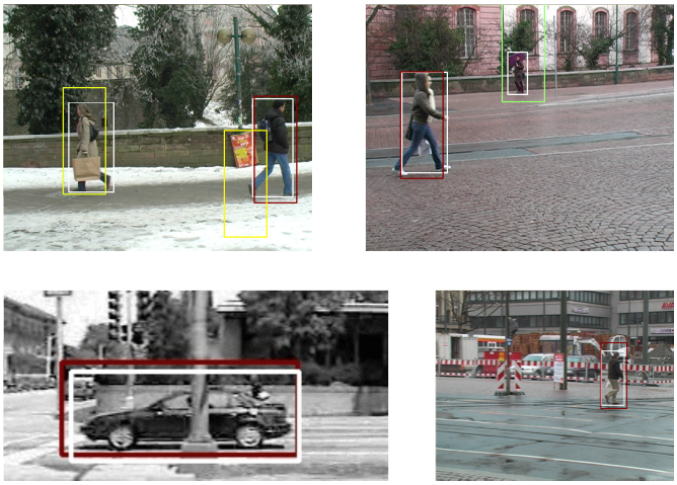


Fig. 4. Example of false and correct detections in UIUC-Cars and TUD-Pedestrian data sets. In the top-left image, the pedestrian on the left is detected after the false detection, in the middle. In the lop-right one, the scale of the pedestrian detected in the middle is wrong. Below, examples of correct detection.

B. Results on TUD-Pedestrians

The TUD-Pedestrians data set is much more challenging than the previous one, as it contains deformable objects (pedestrians) with variable aspects, sizes, and illuminations.

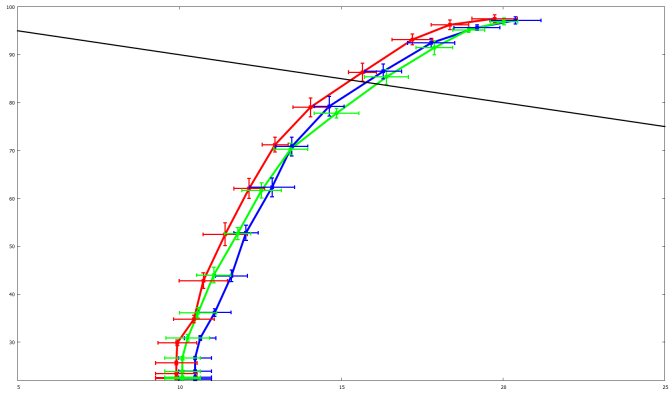


Fig. 5. Stability of Hough Forest with and without our contributions on the TUD-Pedestrian dataset. The three versions were run 10 times. Each ROC curve corresponds to the average values, while the crosses indicate the standard deviation in precision and recall for each threshold level. Va (Gall): Red, Vb (Partially improved): Green, Vc (Fully improved): Blue. Black: EER line.

Considering our set of parameters and the variability of pedestrian's aspects, performances can vary a lot. Three versions of the Hough Forest were evaluated:

- Va: Original Gall's method
- Vb: Improved model with superpatches approach (Part. III-B) but without sampling enhancement (Part. III-A)
- Vc: Improved model with both superpatches and sampling enhancement ($\alpha = 0.10$)

For each version, 9 trees per forest were constructed (instead of 15 for [5]), 25 000 negative and positive patches, and 500 generated binary tests for each non-leaf node (instead of 20 000 for [5]). Junction-ness defined by Eq. 4 is measured at three scales: 1.0, 2.0 and 4.0. For each version, we ran 10 simulations, corresponding to 10 different Hough forests. Fig. 5 and Fig. 6 summarise those simulations.

Even though our results are slightly less accurate than the one obtained by the original Hough Forest with the same parameters, the main interest of our model lies on the stability of the produced detectors in degraded parameter conditions. Indeed, considering standard deviation in Fig. 5, it can be noticed that the red curve, corresponding to the original Hough Forest is slightly the highest one from the three curves, but it is also the one with the highest variance, all along the ROC curve. Additionally, Fig. 6 displays the best and the worst ROC curves obtained within the 10 simulations. While the best results are obtained with the original Hough Forest, this method is also the one with the largest margin between the best and worst results.

If we focus on our methods, Vc provides better results than Vb, but both are more stable than the original Hough Forest Va, and the loss of performance is still very low (the average EER passes from 84.5% in Va to 84.0% in Vc). Finally, Vc results are more stable than Vb, and its accuracy is also closer to Va.

The benefits on the training time, and then on the tractability of the parameter setting, is easy to quantify, as the complexity

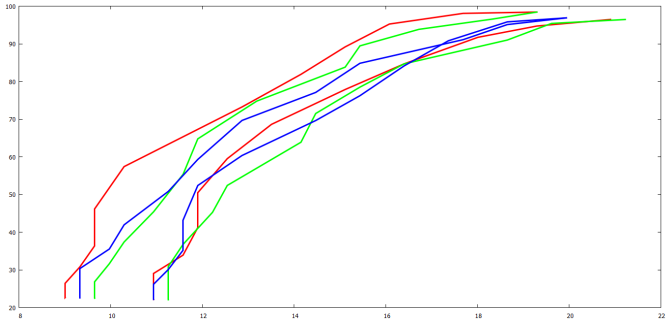


Fig. 6. Best and worst ROC-curves for each version. Colour code is the same as in Fig. 5.

of the learning phase is linear with respect to the number of sampled tests. Besides, the computational overheads of our two improvements are negligible. Thus, using 500 tests instead of 20 000 typically decreases the training phase of one tree on the TUD-Pedestrians dataset from 8 hours to 15 minutes on a standard desktop PC. With our set of parameters, our training step is slightly slower than the original one. It is mainly due to the superpatches calcutation at each node, requiring to sum every patches from one set. This operation is as slow as the number of patch is high, which is the case for nodes at low depth. Finally, the impact is limited, the time required to train the forest in our case and Gall's case are in the same order of magnitude

In term of memory consumption, we need to store one scalar saliency map for one scale and one image. Each scalar saliency map needs as much memory as one feature channel. Consequently, for one image, we pass from 32 maps for the original algorithm to 35 in ours (3 scales), which is still low.

V. Conclusion

In this paper, two contributions to improve the Hough Forest models were presented. Our goal was to drastically reduce the training time, while keeping an accuracy equivalent to the original algorithm, and reducing the variance of the model. Compared to the original Hough Forest, with the same parameters, our algorithm is slightly slower, and need a bit more memory, but is more stable. Our contributions focused on enhancing patch sampling and non-leaf node training, by reducing the randomness to the benefit of the most relevant image content. First, we proposed a method to select some patches based on the multiscale junction-ness measure [12]. Second, for a non-leaf node creation, we generated a probability map whose purpose is to summarise the landscapes corresponding to the patch subset used to create this node, and optimise the node quality criterion. These two contributions are simple and computationnally efficient. They allow to reduce the variance of the models in degraded parameters conditions, and then provide a faster and easier way to evaluate the impact of the different parameters on the detection.

Compared to the original parameters [5], we dramatically reduced the size of the forest and the number of generated tests used for creating each node. At short term, our objective is to see how our algorithm behaves when reducing another parameter which has a strong impact on the training time: the size of the patch set. We will also work on methods to enhance the way to draw the threshold τ used in Eq. 1. Then, we plan to look for the parameters providing the best results and testing our algorithm on different academic datasets, compared to the state-of-the-art. Finally, as Gall suggested [16], we will extend our contribution to different applications, such as object tracking or action recognition.

Acknowledgment

The research was supported by a DGA-MRIS scholarship.

References

[1] P.V.C. Hough. Method and means for recognizing complex patterns, Dec 1962.

[2] Bastian Leibe and Bernt Schiele. *Interleaving object categorization and segmentation*. Springer, 2006.

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

[4] Frank Moosmann, Eric Nowak, and Frederic Jurie. Randomized clustering forests for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9):1632–1646, 2008.

[5] Juergen Gall and Victor Lempitsky. Class-specific hough forests for object detection. In *Decision forests for computer vision and medical image analysis*, pages 143–157. Springer, 2013.

[6] Andrea Ciolini, Lorenzo Seidenari, Svebor Karaman, and Alberto Del Bimbo. Efficient hough forest object detection for low-power devices. In *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*, pages 1–6. IEEE, 2015.

[7] Yusuke Murai, Yuji Yamauchi, Takayoshi Yamashita, and Hironobu Fujiyoshi. Weighted hough forest for object detection. In *Machine Vision Applications (MVA), 2015 14th IAPR International Conference on*, pages 122–125. IEEE, 2015.

[8] Juergen Gall, Nima Razavi, and Luc Van Gool. An introduction to random forests for multi-class object detection. In *Outdoor and Large-Scale Real-World Scene Analysis*, pages 243–263. Springer, 2012.

[9] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[10] Yizong Cheng. Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(8):790–799, 1995.

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

[12] Tony Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116, 1998.

[13] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[14] Shivani Agarwal, Aatif Awan, and Dan Roth. Learning to detect objects in images via a sparse, part-based representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(11):1475–1490, 2004.

[15] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. People-tracking-by-detection and people-detection-by-tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[16] Juergen Gall, Angela Yao, Nima Razavi, Luc Van Gool, and Victor Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 33(11):2188–2202, 2011.