Fast Pedestrian Detection by Cascaded Random Forest with Dominant Orientation Templates

Danhang Tang
http://www.iis.ee.ic.ac.uk/~dtang
Yang Liu
http://www.iis.ee.ic.ac.uk/~yliu
Tae-Kyun Kim
http://www.iis.ee.ic.ac.uk/~tkkim

Department of Electrical Engineering,
Imperial College,
London, UK

Abstract

In this paper, we present a new pedestrian detection method combining Random Forest and Dominant Orientation Templates(DOT) to achieve state-of-the-art accuracy and, more importantly, to accelerate run-time speed. DOT can be considered as a binary version of Histogram of Oriented Gradients(HOG) and therefore provides time-efficient properties. However, since discarding magnitude information, it degrades the detection rate, when it is directly incorporated. We propose a novel template-matching split function using DOT for Random Forest. It divides a feature space in a non-linear manner, but has a very low complexity up to binary bit-wise operations. Experiments demonstrate that our method provides much superior speed with comparable accuracy to state-of-the-art pedestrian detectors. By combining a holistic and a patch-based detectors in a cascade manner, we accelerate the detection speed of Hough Forest, a prior-art using Random Forest and HOG, by about 20 times. The obtained speed is 5 frames per second for 640×480 images with 24 scales.

1 Introduction

Pedestrian detection is one of the most competitive domains in computer vision community. For the last few years, researchers has been pushing the upper bound of accuracy and efficiency to new levels [9, 12]. According to a recent survey [10], the latest version of LatentSVM [12] can achieve 20% of miss rate whilst having only 10^{-1} false positives per image(fppi). Its speed however is still under 1.5 frame per second(fps). Another state-of-the-art method, the Fastest Pedestrian Detector in the West(FPDW) [9] achieves a close real-time speed of 5 fps whilst keeping a comparable accuracy against LatentSVM. A more recent piece of work by Benenson et al. [3] even push the limit to 50 fps for monocular images using GPU. The common optimising direction of [3, 9] is to reduce the detection workload by handling multiple scales smartly.

In this paper, we approach this problem from a different angle by proposing a Random Forest(RF) based detector and optimising its infrastructure. RF has been proven to be an

 http://dx.doi.org/10.5244/C.26.58

effective classifier owing to its accuracy, scalability and run-time efficiency. Effort has been made applying it on pedestrian detection problem to obtain state-of-the-art accuracy [13], where the technique is called Hough Forest(HF). And yet speed wise it's much behind [9]. To enhance the efficiency as well as keeping accuracy, we bring up these major contributions:

- **Adoption of Dominant Orientation Templates(DOT):** After analysing existing methods, we address that the bottleneck in speed often lies in densely-sampled features, especially when performing detection under a dense scale pyramid. However, it is not easy to solve the problem since these features are often key reasons of their success in accuracy. In this work, we successfully replace Histogram of Oriented Gradients(HOG) with DOT [17], which not only provides an efficient binary form feature, but also allows us to down-sample the search space. Furthermore, we extend DOT to colour domain to incorporate multi-cue features.

- **Novel split function:** Albeit an efficient feature, DOT loses quite discriminative information against HOG, which leads to a significant decrease in terms of accuracy. To compensate, we propose a novel split function for RF to exploit more feature dimensions. This function provides a non-linear hyperplane for better dividing the feature space whilst having only complexity similar to an axis-aligned split function. In the experiments, the novel split function radically improved the detection accuracy of RF with 2 pixel tests on DOT.

- **Coarse-to-fine cascade:** We propose a 2-level cascade architecture to further accelerate the method. For the 1^{st} level, we design and train a holistic RF detector which can filter out most of the unlikely regions. And a patch-based RF detector with hough voting is performed as the 2^{nd} level to produce a higher detection rate. The proposed cascade accelerates the original HF [13] by 20 times.

- **RF training revisit:** In object detection tasks, we often have a huge number of negative training samples available, which demands a highly scalable and easy-to-bootstrap algorithm. The parallel structure of RF, as an ensemble of random trees, facilitates a larger-scale training. Section 4 addresses two major issues, bagging and bootstrapping for RF. For each of them we revisit and compare two different methods.

2 Related Work

Architecture. State-of-the-art pedestrian detectors can be largely classified into 3 categories: (a) Holistic: It takes full-human windows as input and makes decisions. A major breakthrough dates back to 2005, when Dalal and Triggs invented HOG feature [8]. Since then significant progress has been made e.g. [9, 31]. (b)Part/patch-based: To achieve robustness in terms of occlusion and pose variation, attention has been paid to part/patch-based methods. Body parts have been exploited as a strong cue in human related problems [2, 4, 5, 26, 27]. Inspired by the Implicit Shape Model [19], Gall et al. [13] train a Random Forest codebook on patches along with spatial information, and then perform a hough voting stage to determine object centers. (c) Cascade: Many existing detectors have a cascade structure. Some are about accuracy [11, 24, 30] while others are for reducing a search space to improve efficiency [12, 22, 32, 33]. Among them Felzenszwalb et al. [12], which is a cascade of holistic and part-based detectors, has been widely considered as a state-of-the-art baseline.

Feature. In recent years, HOG [8] has dominated the area of pedestrian detection. The concept behind is to transform the whole object into dense grids of gradient histograms, which captures not only the appearance but also spatial information to some degree. Moreover, HOG normalizes feature vectors at a block level, which makes it more robust to illumination changes. These characteristics are particularly in favour of variations in pedestrian detection problems.

However, since computing HOG is time-consuming, researchers have been searching for a replacement. So far the trend is to integrate different simple features. Being laid as a foundation, Wojek et al. [31] provided an empirical proof that combining different features with HOG can improve the performance drastically. Likewise, Wang et al. [30] augmented LBP with HOG to obtain state-of-the-art accuracy. Hattori et al. [16] discussed how to integrate stereo information with HOG properly. Later Dollár et al. [9] took a further step by integrating colour(LUV), gradient information, edge feature, etc., to approximate the performance of HOG successfully. Also, some literatures have added motion as a strong cue to boost the performance [1, 20, 28]. Furthermore, researchers have started employing 3D synthetic models to help train a better detection model [21, 23], which is a similar approach to [25] except the latter is for 3D pose estimation problem.

One literature that inspires our work is the Dominant Orientation Template(DOT) [17]. Similar to HOG, it describes the target as a whole based on gradient information. However, instead of computing histograms, DOT only considers those most dominant orientations within each grid and transforms them into a binary form. This representation allows further acceleration with CPU. In their experiments with texture-less objects, DOT shows significantly improvement in terms of speed. Nevertheless, no effort has been put in applying DOT to human detection yet.

Classifier. Randomised Decision Forest(RF), an ensemble of decision trees, is an emerging technique in the field. It nicely scales up to a large train set, which leads to a more complete training for better recognition accuracy. Randomisation embedded in this technique conveniently offers good generalisation performance on new unseen samples. Owing to its scalability and real-time performance, RF has made an enormous success in human pose estimation [25] for console game user interfaces. It has been very successful in semantic image segmentation, key-point tracking, and object categorisation problems as a fast discriminative codebook. However, traditional classifiers like SVM [12, 30] and Boosting [9, 15] are still popularly adopted in the human detection domain, where RF is yet under-explored except for a few studies. Among them Hough Forest(HF) is a successful case [13, 14]. The key difference from RF is that HF incorporates a new split function to measure the offset uncertainty between current patches and object center. Working along with the appearance term, it clusters patches close to each other both in appearance and spatial space. At a testing stage, a hough voting scheme is performed with these offset vectors to predict object centers. One drawback of HF is, to provide state-of-the-art accuracy, dense sampling [18, 29] is needed, which leads to at least 1 order of magnitude away in detection speed from the fastest one [9].

3 Proposed Method

3.1 Overview

Our method consists of a 2-level cascade: At first the scale space is divided into G overlapped groups, and a holistic detector(1^{st} level), designed with RF and our novel split function based

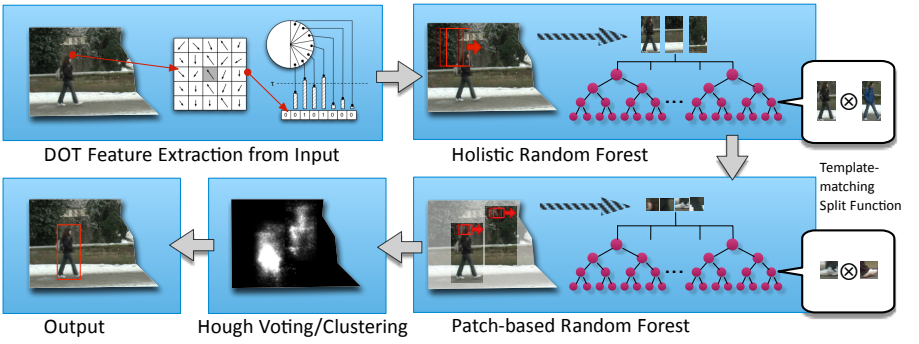


Figure 1: Overview of the proposed method.

on DOT, is applied on the first layer of each group and thus areas of interests are identified. After that a patch-based detector(2^{nd}), improved from HF [13] using the novel split function and clustering votes, is applied within these areas to achieve final bounding boxes. See Figure 1.

The idea is actuated by the fact that holistic detectors are normally faster than patch-based detectors due to a larger step size to use for sliding-window. Whereas patch-based detectors can provide better accuracy and are more robust to pose variation. In the following sections, we will explain our novel split function and discuss how to design and train a holistic detector, a patch-based detector and their combined using the split function.

3.2 Binary template-based split function

The essential part of designing the RF-based detectors (both the holistic and part-based) is to choose an effective split function. It has been discussed that a non-linear hyperplane outperforms axis-aligned or linear ones [7]. Unfortunately due to its time cost, people seldom adopt it in practice. In our case, we observe that the popular 2-pixel test [13, 25] doesn't give satisfying results working with Dominant Orientation Templates(DOT). To tackle this, we propose a novel non-linear template-matching scheme, which yet has the complexity similar to an axis-aligned split function, see Figure 2.

Feature. The prerequisite of this function is to adopt DOT [17] as a descriptor, for its binary form is apt for bitwise operation. To start with, we generate the feature as described in [17]: each pixel encodes the most dominant orientations of its K neighbours into one byte; the most significant bit indicates whether this region is uniform; and the most dominant orientations are quantized into the rest 7 bits.

We also encode another cue, colour information in this manner: first transform the colour image into HSV space; if V is less than a threshold, we set the most significant bit to 1, or else 0; and then the H value is quantized into the rest 7 bits. With this we have 2 channels for our descriptor, one for dominant colours and the other for dominant orientations.

DOT has several benefits over HOG. Firstly, DOT only considers the most dominant orientations whilst discards the magnitudes, which is faster than calculating a histogram. Secondly, since no magnitude information is involved, normalisation step can be skipped. Thirdly, only considering most dominant orientations allows down-sampling in scanning-windows. On the flip side, since DOT encodes less information than HOG, it loses some discriminative information degrading the detection rate. To compensate, we propose a novel

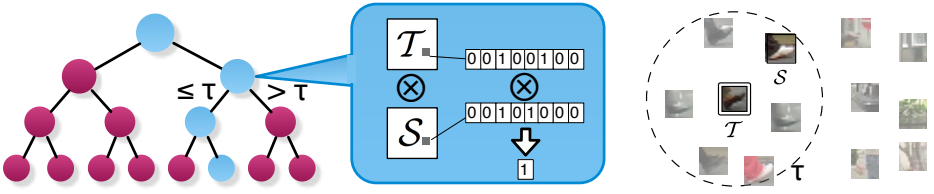


Figure 2: Visualisation of our template matching split function. $\mathcal{T}$ is a DOT template chosen randomly at each split node and $\mathcal{S}$ is any other sample. Image patches (right) represent data samples. Those within τ are passed left and others are passed right.

similarity measurement to incorporate more dimensions. The novel split function drastically improves both the detection *accuracy* of RF with 2-pixel tests on *DOT*, and the detection *speed* of RF with 2-pixel tests on *HOG* (see Section 5).

Similarity measurement. We define a template $\mathcal{T}$ as a n-dimension DOT sample selected from a positive training set $\mathbf{S}^P$. The similarity between $\mathcal{T}$ and any sample $\mathcal{S}$ can be measured with:

$$F(\mathcal{S},\mathcal{T}) = \sum_{\substack{P_d^{\mathcal{S}} \in \mathcal{S} \\ P_d^{\mathcal{T}} \in \mathcal{T}}} \delta(P_d^{\mathcal{S}} \otimes P_d^{\mathcal{T}} \neq 0), d = 1,...,n \tag{1}$$

where $P_d^{\mathcal{S}}$ and $P_d^{\mathcal{T}}$ refer to the d^{th} dimension of $\mathcal{S}$ and $\mathcal{T}$ respectively. $\otimes$ is the bitwise AND operation. δ is an impulse function that is zero except when any bit in $P_{\mathcal{S}} \otimes P_{\mathcal{T}}$ is 1, which can also be interpreted as counting the number of non-zero matchings within $\mathcal{S}$ and $\mathcal{T}$. To accelerate this matching process, SSE optimization is employed to compare 16 bytes in a batch, similar to [17]. Therefore although this function measures the distance between samples in a feature space as a non-linear split, it is efficient since only binary bitwise operations and addition are involved.

Split function. With the measurement above, we can then define the split function h_i as:

$$h_i(\mathcal{S}) = \begin{cases} 0, & F(\mathcal{S},\mathcal{T}_i) \leq \tau_i \\ 1, & F(\mathcal{S},\mathcal{T}_i) > \tau_i \end{cases}, \tag{2}$$

where $\mathcal{T}_i$ means a chosen template and τ_i is a threshold stored at the i^{th} node.

During training, a set of positive $\mathbf{S}^P$ and negative $\mathbf{S}^N$ samples are used to construct a set of randomised decision trees. At the i-th non-leaf node, we have these parameters:

$$\theta_i = \{\mathcal{T}_i, \tau_i\}, \quad \mathcal{T}_i \in \mathbf{S}_i^P, \tag{3}$$

where $\mathcal{T}_i$ is a template chosen from positive samples, and τ_i is a threshold. We randomly generate a set of θ_i, and select the optimal one in terms of information gain.

3.3 Cascade of RF detectors with DOT

Although adopting DOT allows down-sampling in scanning-windows, we still need to perform dense classification and voting to obtain satisfying results. Thus it is a natural option to construct a cascade to filter out unlikely regions before performing the patch-based detector. First we tried a sparsely-sampled patch-based detector as the filtering step but the result is not satisfying. Therefore we propose a novel holistic detector which has a better reject rate. See Section 5.

Holistic detector. We design the holistic detector based on a classification forest with Equation 2 as a split function. We grow trees by recursively splitting nodes. For each split node, the optimal one among a set of θ randomly generated is chosen in terms of information gain. We store the normalised ratio of positive samples that arrived in leaf nodes as a score.

During testing, we propagate each sample S down T trees, and use the average score as the final probability of this sample.

$$p(C=P|S) = \frac{1}{T}\sum_{t=1}^{T}\frac{\mathbf{S}_t^P}{\mathbf{S}_t^P + \mathbf{S}_t^N} \tag{4}$$

After classification, we greedily select those windows with highest scores at centre and apply the patch-based detector on them.

Patch-based detector. Our patch-based detector inherits the structure of Hough Forest(HF) [13]. Since the choice of feature is also DOT instead of HOG in the original HF, the extracted features for the holistic detector can be shared throughout the cascade. While the sample dimension is reduced to patch-size, our split function still radically improves the speed of HF, exhibiting a comparable accuracy.

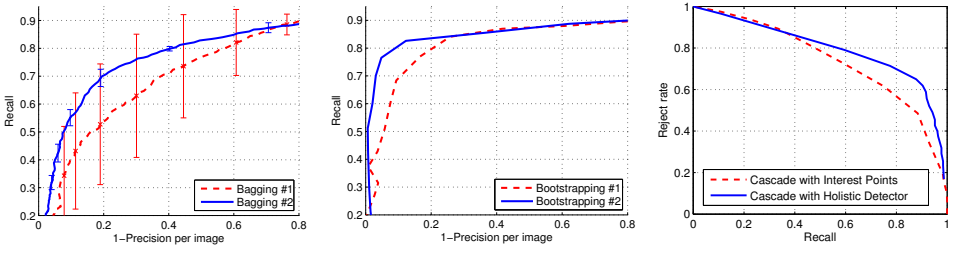
Instead of adopting Equation 4 as the final score, HF has an extra voting step to identify possible object centres. The original HF stores all the offset vectors extracted from training data and exploits them for voting during testing. Depending on how many positive training patches we have, average votes per leaf node could go from tens to hundreds. To speed up, we incorporate a clustering preprocessing step to reduce votes per node. Similar to [14], the modes of votes are found by Meanshift and sizes of distributions are employed as weights while accumulating votes. In practice, we find that clustering reduces the discriminative power since the correct positions get less votes. To avoid this side effect, we aggregate scores of neighbourhood positions by voting in a down-sampled space.

4 RF Training

Training for object detection tasks involves a large number of images, from which myriads of pixels or patches around them are cropped to serve as training instances. The number of training instances is often too large to fit into memory at a single time, and scalability of machine learning algorithms is another relevant issue. People do bootstrapping to augment an initially small training dataset. Random Forest is a highly scalable algorithm owing to its efficient tree structure, in addition to that, as an ensemble of random trees, it affords to a larger-scale training. We revisit bagging and bootstrapping for RF and suggest some heuristics, in order to maximise the performance of object detection.

Bagging. A common practice of bagging(#1) for RF [6] is to extract a training sample set $\mathbf{S}$ from training images $\mathbf{I}$. And then for each tree t, we sub-sample with replacement from $\mathbf{S}$ to form a training set $\mathbf{S}_t$. We normally have $\mathbf{S} \ll \mathbf{I}$ due to memory restrictions, and have our training set for the whole forest(T trees) yet much smaller than $\mathbf{I}$, i.e. $\mathbf{S} < \bigcup_{t=1}^{T}\mathbf{S}_t \ll \mathbf{I}$. This will lead to larger variations in performance. The other way of doing bagging(#2) for RF is to randomly draw samples from available training data $\mathbf{S}_t^{'}$ directly from $\mathbf{I}$. This way, we may access more pixels for the whole forest i.e. $\mathbf{S} \ll \bigcup_{t=1}^{T}\mathbf{S}_t^{'} < \mathbf{I}$, if $\mathbf{S}_t^{'} \approx \mathbf{S}$. This leads to a much stabler result.

Bootstrapping. The traditional way(#1) [8, 12] is to uniformly draw samples $\mathbf{S}^{1st}$ to train the first version of detector (the whole forest), then apply it to the complete training set $\mathbf{I}$,



(a) (b) (c)

Figure 3: Results of different bagging(a) and bootstrapping(b) practices for RF. See text. (c) Demonstration of Reject Rate(ratio of rejected pixels over all pixels) against Recall.

augment those false positives with $\mathbf{S}^{1st}$ to form $\mathbf{S}^{2nd}$, and train the second version of detector. It keeps repeating this process until the performance converges or $\mathbf{S}^{nth}$ cannot be fitted into memory. Gall et al. adopted a different practice(#2) in [13]. Since Random Forest is an ensemble classifier, they train the first 5 trees with uniformly-drawn samples, use them to obtain difficult positive and negative sample to train the next 5 trees, and repeat this process. With this, each group of trees has its bias and together they can achieve a better performance.

5 Experiments

In this section, we quantitatively evaluate our method. All experiments were performed using the TUD pedestrian dataset, which contains 400 training images, and 250 test images with pedestrians in different poses, clothes, sizes, occlusions in different street scenes. For comparisons with some other major detectors, we followed the per-image evaluation procedure as described in [13]. All experiments were performed with an Intel Core™ i5 2.90GHz CPU and 8GB memory.

5.1 RF Training

As a preliminary step, we experiment on how to train an RF detector w.r.t. bagging and bootstrapping, using the patch-based detector in Section 3.3.

First we compare two different ways of doing bagging. To demonstrate the randomness of sample sets, we solely perform this experiment on the patch-based detector without bootstrapping. For bagging #1, we extract 24000 positive and 24000 negative patches as a training set and perform the standard bagging on it to train 5 trees. As to bagging #2, we sample different training sets(24000 positive and 24000 negative) directly from images when training each tree. For both methods we train the detector 10 times to see their performance variance. Figure 3(a) shows the result: comparing to bagging #1, bagging #2 behaves much more consistently, and has a much better mean curve, as expected. Therefore it is a better option as a first step for bootstrapping.

And then we compare the performance of two different bootstrappings based on bagging #2. For the standard bootstrapping #1, we re-train the whole detector(15 trees) each round. Whereas with bootstrapping #2, we train the first 5 trees with randomly drawn samples, use them to find 24000 difficult negative samples and then augment them to the training set to train the next 5 trees. We keep doing this until we have 15 trees. Figure 3(b) shows that bootstrapping #2 gives more discriminative results(10% higher at 0.1 fppi).

Method	Time(s)
Original Hough Forest	4.15
HF with Fast corner cascade	2.84
HF with Fast corner cascade and clustering	1.11
Our method(patch-based+DOT)	0.62
Our method(cascade)	0.33
Our method(cascade+clustering)	0.20

Table 1: Comparison of efficiency with 24 scales(0.17∼0.87) on 640x480 images.

5.2 Benchmark

For benchmarking we follow this procedure: with Bagging #2 and Bootstrapping #2 as our training protocol, we train 15 trees for holistic and patch-based detector separately. For first-round holistic training data, we crop 400 positive and 2000 negative windows from TUD dataset; whilst for patch-based, we draw 24000 positive and 24000 negative patches. Difficult samples are added for next rounds. According to amounts of training data, we set the maximum depth of patch-based RF to 15 whilst 8 for holistic. Feature-wise, we down-sample images by skipping every other pixel in both x and y directions when sliding windows for patch-based detector. For holistic we skip even more(every 3 pixels).

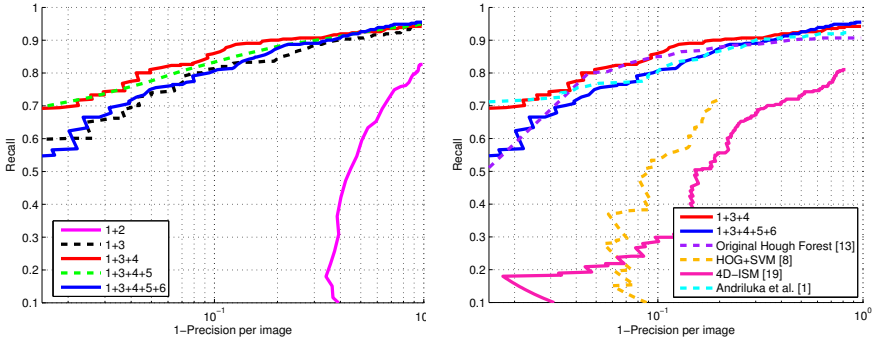
Firstly we compare our novel split function against the widely-used 2-pixel test under the framework of patch-based RF and DOT. Figure 4 (a) shows that our method outperforms the 2-pixel test drastically to the State-of-the-art level.

In terms of features, Figure 4 shows that adopting DOT with orientation only is not as discriminative as the original HF with HOG when 1-Precision is between 0 and 0.3. However, incorporating the colour channel successfully boosts up the accuracy to even better than HF, as shown in Figure 4 (b). Note that here down-sampling images doesn't degrade the performance at all.

As a control group for cascade architecture, we compare our detector with a more intuitive cascaded version of HF: On the 1^{st} level we apply Fast corner detector to identify interest points. We then classify the patches centred at these points and let them vote for possible regions. This can be considered as a sparse-sampling technique. The 2^{nd} level we apply the dense-sampling HF to those identified regions as it is. As shown in Figure 3(c), under the same recall rate(90%), our cascade can reject 20% more unlikely pixels than Fast-corner cascade. Table 1 further proves this by showing that our cascade enhances the runtime by the factor of 12.57 whilst Fast-corner can only speed up by the factor of 1.46, over the original HF. This is because Fast Corner detector normally produces more interest points in clutter, which leads to outvoting those pedestrian windows.

We can also see that the clustering/discretizing idea for hough voting can further accelerate the speed to 0.2s per image, which is *20.75 times faster* than the original HF. Accuracy loss about 5% at 10^{-1} 1-Precision is relatively minor.

Figure 5 shows some result images demonstrating the performance of our detector in different environments. Since the TUD training set mainly contains side views, to test on our own street scene sequences, we augmented the ETH dataset with TUD to accommodate more pose and environment variations. More details demonstrating runtime performance can be seen in our supplement videos.



(a) (b)

Figure 4: (a) Comparison of accuracy with different components: 1. Dominant Orientations; 2. 2-Pixel Test; 3. Template Matching; 4. Dominant Colours; 5. Cascade; 6. Clustered Votes. (b) Comparison between our detector and State-of-the-art. Note that [1] performs detection by tracking. Best viewed in colour.

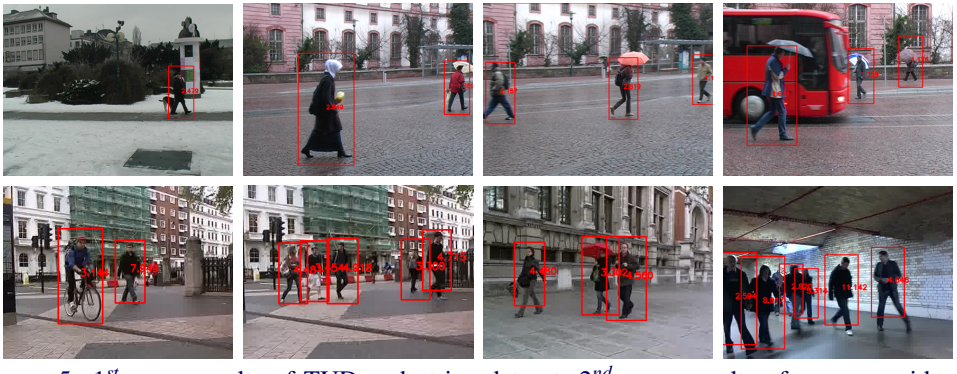


Figure 5: 1^{st} row: results of TUD pedestrian dataset; 2^{nd} row: results of our own video sequences. Numbers within bounding boxes indicate the voting scores.

6 Conclusion

We have introduced a novel standalone solution for fast pedestrian detection adopting an efficient feature DOT and Random Forest. The novel split function for RF using DOT maintains sufficient discriminative information, while drastically improving the run-time speed of the original Hough Forest.

We have demonstrated that our method delivers state-of-the-art detection accuracy, while largely boosting the speed of similar frameworks. The runtime speed obtained is 5 frames/sec for 24 scales(0.17∼0.87) on 640x480 images, which is about 20 times faster than the Hough Forest and is comparable to the Fastest Pedestrian Detector in the West(FPDW) [9]. Note that our speed optimisation is mainly done about features rather than scales, therefore it can be combined with those works [9] and obtain further speed-up. Also, we emphasise the inherent benefits of our RF framework for scalability, quick training, multi-class or multi-part detection.

References

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

[2] A. Bar-Hillel, D. Levi, E. Krupka, and C. Goldberg. Part-based feature synthesis for human detection. In *ECCV*, 2010.

[3] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool. Pedestrian detection at 100 frames per second. In *CVPR*, 2012.

[4] Y. Bo and C. Fowlkes. Shape-based pedestrian parsing. In *CVPR*, 2011.

[5] L. Bourdev, S. Maji, T. Brox, and J. Malik. Describing people: A poselet-based approach to attribute classification. In *ICCV*, 2011.

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

[7] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. *Tech. Rep., Microsoft Research, Cambridge*, 2011.

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

[9] P. Dollár and P. Perona. The fastest pedestrian detector in the west. In *BMVC*, 2010.

[10] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *PAMI*, 34(4):743–761, 2012.

[11] G. Duan, H. Ai, and S. Lao. A structural filter approach to human detection. In *ECCV*, 2010.

[12] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010.

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

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

[15] G. Gualdi, A. Prati, and R. Cucchiara. Multi-stage sampling with boosting cascades for pedestrian detection in images and videos. In *ECCV*, 2010.

[16] H. Hattori, A. Seki, M. Nishiyama, and T. Watanabe. Stereo-based pedestrian detection using multiple patterns. In *BMVC*, 2009.

[17] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab. Dominant orientation templates for real-time detection of texture-less objects. In *CVPR*, 2010.

[18] F. Jurie and W. Triggs. Creating efficient codebooks for visual recognition. In *ICCV*, 2005.

[19] B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *BMVC*, 2003.

[20] Y. Liu, S. Shan, X. Chen, J. Heikkila, W. Gao, and M. Pietikainen. Spatial-temporal granularity-tunable gradients partition (stggp) descriptors for human detection. In *ECCV*, 2010.

[21] J. Marín, D. Vazquez, D. Geronimo, and A.M. Lopez. Learning appearance in virtual scenarios for pedestrian detection. In *CVPR*, 2010.

[22] M. Pedersoli, A. Vedaldi, and J. Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *CVPR*, 2011.

[23] L. Pishchulin, T. Thorm, C. Wojek, and M P I Informatics. Learning people detection models from few training samples. *Pattern Recognition*, 2011.

[24] M. Rodriguez, I. Laptev, J. Sivic, and J.-Y. Audibert. Density-aware person detection and tracking in crowds. In *ICCV*, 2011.

[25] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, 2011.

[26] Z. Song, M. Wang, X. Hua, and S. Yan. Predicting occupation via human clothing and contexts. In *ICCV*, 2011.

[27] T.-P. Tian and S. Sclaroff. Fast multi-aspect 2d human detection. In *ECCV*, 2010.

[28] S. Walk and N. Majer. New features and insights for pedestrian detection. In *CVPR*, 2010.

[29] H. Wang. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.

[30] X. Wang, T. X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *ICCV*. IEEE, 2009.

[31] C. Wojek, S. Walk, and B. Schiele. Multi-cue onboard pedestrian detection. In *CVPR*, 2009.

[32] R. Xu, B. Zhang, Q. Ye, and J. Jiao. Cascaded l1-norm minimization learning (clml) classifier for human detection. In *CVPR*, 2010.

[33] Z. Zhang, J. Warrell, and P. Torr. Proposal generation for object detection using cascaded ranking svms. In *CVPR*, 2011.