# EFFICIENT HOUGH FOREST OBJECT DETECTION FOR LOW-POWER DEVICES

*Andrea Ciolini, Lorenzo Seidenari, Svebor Karaman, Alberto Del Bimbo*

Media Integration and Communication Center - University of Florence
andrea.ciolini@stud.unifi.it, name.lastname@unifi.it

## ABSTRACT

An important task in computer vision is object localization and recognition within images and video. Achieving real-time object localization and recognition on low-power devices is especially relevant in the context of wearable technologies. Indeed, wearable devices have a reduced size and cost and limited computational power leading to a challenging scenario for classical computer vision algorithms. This paper improves the Hough Forest approach with several contributions: a faster computation of the features and a faster evaluation of the learned model with minimal loss in accuracy. Our method is characterized by a low computational requirement and allows real-time detection on a wearable device.

***Index Terms***— Object detection, Wearable device, Wearable computing, Hough Forest

## 1. INTRODUCTION

Recently computer vision algorithms dedicated to wearable devices have risen the interest of the multimedia and pattern recognition community. Many scenarios are enabled by such computational devices such as summarization for life-logging [1, 2], assistance to visually impaired people [3], self-localization [4] and user interest profiling.

All of these applications have two important requirements: they have a hard real-time constraint and they require to run for extended periods of time. On one hand very powerful and small devices are available in the retail market nowadays [5, 6] therefore the real-time requirement can be addressed by simply boosting the computational power of wearable devices. However, the speed requirement conflicts with the running time duration. Indeed, wearable devices need to be compact and run on battery. Even with recent developments in high efficiency batteries [7] it is extremely hard to pack many milli-ampere hour (mAh) in a small portable object, hence it is difficult to have a powerful wearable device running for a long period of time.

Object detection is a central problem in computer vision since it enables many high level tasks such as scene understanding, multimedia mining and automatic image annotation. In the ego-centric vision literature, object detection is commonly used to understand actions from the user perspective. As an example detecting people is a very important task since it is the main stage of an egocentric processing pipeline to solve most high level tasks such as social behavior understanding [8, 9].

In wearable vision the most common scenarios involve occlusion and truncation of objects. Occlusion happens when an object is placed behind some fixed element of a scene or another object. Truncation happens when camera pose is such that not the whole object can be observed; this is often the case when trying to detect people at a close range. Moderate occlusion can be handled by part based models [10] or with an explicit occluder modelling [11]. For pedestrian detection occlusion can be managed by training multiple occlusion specific classifiers [12].

Recently Gall *et al.* proposed Hough Forests for object detection [13]. Their method differs from many other object detection algorithms since it does not apply a sliding window approach. Being a local patch based method, it is more robust to occlusion than holistic methods. Techniques like [12] are accelerated using GPU, integral channel features [14] and training many classifier per scale to avoid image rescaling and feature recomputation [15] but are often tailored to pedestrian detection. Deformable part models (DPM) [10] are more versatile but expensive to run. Hough Forests have been recently shown to scale and perform comparably to DPM [16] but are not still fast enough to be run on embedded hardware.

In this paper we propose a Fast Hough Forest method for object detection that improves the running time of Hough Forests with respect to [13] by $60\times$ on laptop and embedded boards with very little loss in accuracy. Our method runs at 10 FPS on a 2.3GHZ i5-2467M laptop and at 3 FPS on a SPEAR 1340 A9 ARM 600MhZ on a single thread. We show a detailed quantitative comparison on a standard pedestrian detection benchmark and qualitative results for an upper body classifier aimed at wearable vision tasks.

## 2. HOUGH FOREST FOR DETECTION

In this section we introduce the concepts of Hough Transform and Hough space and the combination with Random Forests, namely the Hough Forest, that we will use for person and upper body detection. We give details on the ensemble learning algorithm used to train Random Forests and how it is adapted

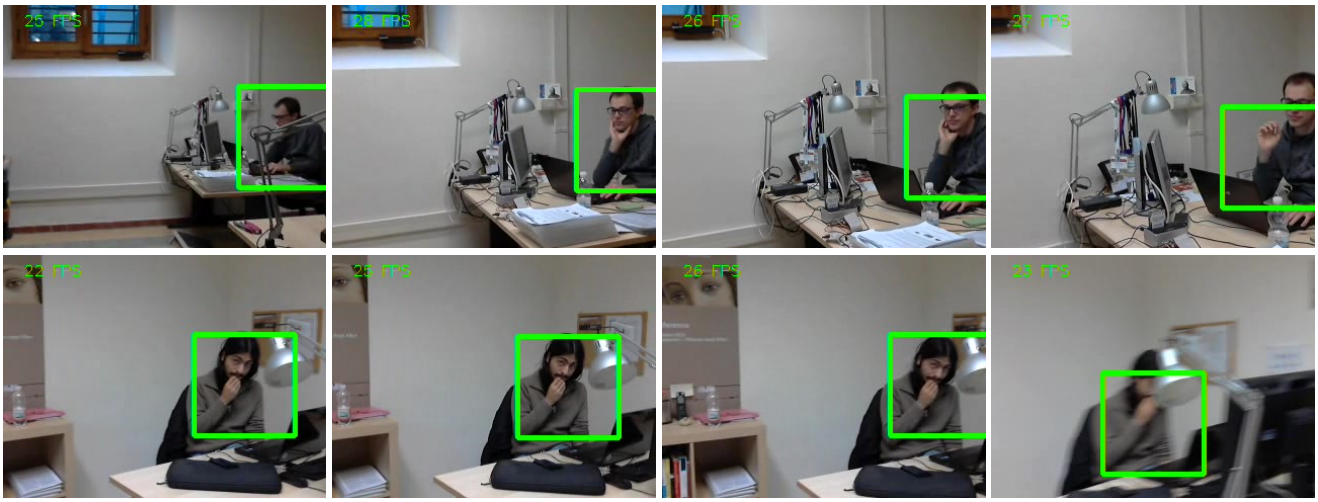


**Fig. 1**. Challenging frames from a sequence showing the method robustness under pose variation, self-occlusion, truncation and blur. Full video, processed on a i5-2467M laptop can be viewed at `http://goo.gl/04mtO2` . Note that UpperBody detection can be performed accurately with fewer scales with respect to pedestrian detection thus increasing the framerate to $\sim$ 25 FPS.

to train Hough Forests in order to use the forest as an object detector.

The Hough Forest approach shows interesting robustness properties in challenging conditions that are frequent in wearable scenarios. The fact that cameras are not handled by users means that the field-of-view may not present a proper object framing leading to object truncation, i.e. partial visibility of the object. Moreover since the camera is usually attached to the user head-gear, glasses or dress, motion blur may be generated by sudden user body motion like standing up or turning.

Object occlusion can happen because an object is occluded by another, e.g. pedestrian detection in crowds, or can be caused by the articulated nature of the object to be detected. In the case of people detection in wearable vision, both phenomena are present. In Figure 1 we show our upper body detector dealing with some of these issues. One can observe that correct detections are obtained even when the person is truncated (because on the side of the frame), occluded (by an arm or a desk lamp), or when the image is blurry due to the ego-motion.

In the following we briefly review Random Forests and Hough Forests with a discussion on the main algorithm bottlenecks. In Section 3 we will show how to remove these bottlenecks.

### 2.1. Hough Transform and Hough space

The Hough Transform [17] is a feature extraction method to find the presence of a given class of shapes in an image by a voting procedure. Originally, the Hough Transform was applied to detect the presence of lines in images by accumulating evidence of the presence of a given line parametrized by its minimal distance to the origin $r$ and the angle $\theta$ between this segment of length $r$ and the $x$-axis. The accumulation and voting procedure is done over each pixel and takes place in this ($r$,$\theta$) space [17]. The method hence relies on the accumulation of local evidence (such as gradient information) computed on the neighborhood of each pixel, to determine in the end the presence of the target shape. It was further extended to detect any shape by the Generalized Hough Transform [18]. The name Hough space is now commonly used to refer to any method that relies on the accumulation of votes from local samples to infer the presence of a higher level concept (object, person...).

### 2.2. Random Forest

A Random Forest is a machine learning technique that uses multiple Decision Trees as a single ensemble classifier. In the case of object detection, given a set of training samples labeled as positive or not, a decision tree is learnt by splitting recursively the initial set into subsets by maximizing the information gain on a set of randomly generated splitting criteria. Formally, denoting $\mathcal{S}^0$ all initial samples available, the tree learning procedure will try to split at each inner node $i$ at level $k$ the current set of samples $\mathcal{S}^i$, where $i \in \{2^{k-1}, 2^k\}$, into two subsets $\mathcal{S}_L^i$ and $\mathcal{S}_R^i$ based on the output of a splitting function $f(\tau)$. In each node, several randomly generated splitting criterions $\mathrm{T} = \{\tau\}$ are evaluated based on the information gain they provide. Without loss of generality, considering that the class labels $c \in \{0, 1\}$ we define the information

gain $G^i$ at node $i$ as:

$$G^i(\tau) = H(\mathcal{S}^i) - \sum_{j \in \{L,R\}} \frac{|\mathcal{S}^i_j|}{|\mathcal{S}^i_j|} H(\mathcal{S}^i_j), \quad (1)$$

where $H(\mathcal{S})$ is the class entropy in set $\mathcal{S}$:

$$H(\mathcal{S}^i) = - \sum_{c \in \{0,1\}} p(c|\mathcal{S}) \log p(c|\mathcal{S}) \quad (2)$$

The splitting criterion $\tau$ giving the maximum information gain is selected. This splitting procedure is carried on until the maximum depth or the minimum number of samples by node is reached. Each tree of the forest is learnt separately, potentially with different initial training samples. At test time, a sample goes down each tree and the leaves class distribution are used to estimate the class of the test sample.

### 2.3. Hough Forest

A Hough Forest is a Random Forest applied to obtain a voting map in the Hough space localizing objects in a discriminative manner. Differently from random forests the training data are tuples: $(I_i, y, d_i)$ consisting of background and foreground image patches $I_i$, their label $y$ and their displacement $d_i$ with respect to the centroid of the object they belong to. Displacement is undefined for background patches. Object detection with Hough Forests is performed classifying each patch using the forest and casting votes in the Hough voting space proportionally to the likelihood of each leaf. The main difference with respect to Random Forests at training time is the node split evaluation.

Gall *et al.* [13] propose, in addition to maximizing the information gain $G^i$, to minimize the displacement uncertainty defined as:

$$U^i = \sum_{k \in P_i} (d_k - d_P) \quad (3)$$

where $d_P$ is the mean offset of patches belonging to the object in the set of patches $P_i$ at node $i$. Alternating between $G^i$ and $U^i$ for the node split evaluation, the tree construction will both minimize the classification error and keep variation in displacements at leaf nodes low.

Denoting $P^F_l$ the set of foreground patches that reached a leaf node $l$ and $P_l$ the set of all patches in $l$, the foreground likelihood in $l$ is estimated as $\frac{|P^F_l|}{|P_l|}$. Furthermore, each leaf node $l$ will retain a set of displacements $\mathcal{D}_l$ that are used during the voting procedure.

Detection with Hough Forests is performed similarly as with Random Forests. After feature channels have been computed for the whole image. Patches are densely sampled and fed to each tree in the forest. For a patch centered at $x$ falling in leaf $l$, the value $\frac{|P^F_l|}{|P_l|}$ is added to the Hough Voting $V$ map in each displacement location:

$$V(x - d_j) = V(x - d_j) + \frac{|P^F_l|}{|P_l|} \quad (4)$$

with $d_j \in \mathcal{D}_l$. Finally to obtain the actual object locations, peaks are sought in $V(x)$. A Gaussian filtering is performed before the maxima location step. A Multi-scale Hough Forest is obtained by resizing, or upscaling if needed, the image and performing the above procedure for each image size.

### 2.4. Complexity analysis

We can break timing of detection with Hough Forests in four main operations: feature computation, forest trees traversal, voting and non-maxima suppression. Non-maxima suppression and tree traversal have a negligible cost with respect to feature computation and voting.

As it can be seen from Figure 3, the *vanilla* voting algorithm takes more than ten times the feature computation time. Voting time depends quadratically on the sampling stride and linearly in the amount of votes casted per leaf.

Extracting many feature channels to obtain as much discriminative information as possible has also a cost. These two elements are the main bottlenecks; in the following we show how we can render the voting algorithm computationally cheap and drastically reduce the feature computation time using a smaller and improved feature set.

## 3. SPEEDING UP DETECTION WITH HOUGH FOREST

In this section we explain how we speed up the detection process with a Hough Forest while maintaining a similar level of accuracy. We worked on the training procedure, reducing the computational cost of votes, and local patches sampling.

### 3.1. Training

Once any object detector has been trained, some negative samples may still be given a high score by the classifier. These samples are usually referred to as hard negatives. Hard negative mining is a technique to improve classifiers or ensembles of classifiers. Gall *et al.* train the forest as an ensemble of trees applying a bagging paradigm. The first 5 trees are trained with the whole dataset and the subsequent ones are trained on the example on which the classifier fails (false positives and false negatives) by batches of 5 trees. In the end, they obtain a forest of 15 trees. To reduce the amount of trees we also apply hard negative mining and iteratively add to the negative samples newly found false positives. Instead of adding more trees to the forest we keep only 5 trees and retrain them with the improved negative set. This has clearly a $3\times$ speed-up at runtime reducing the number of trees that must be evaluated.

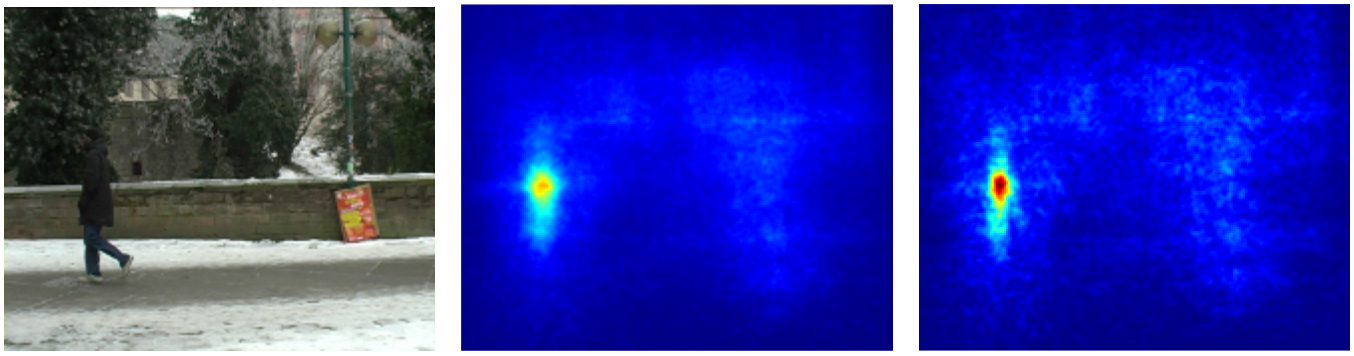

**Fig. 2**. Example of voting maps obtained on a sample pedestrian image (left) without clustering (center) and after clustering votes in the leaves (right).

### 3.2. Reducing votes with clustering and leaf selection

After the training is complete, each leaf will contain a probability of a patch reaching that leaf belonging to an object to be detected and set of displacements that localize it. There are two elements of redundancy in this structure. First, leaves with low probability will cast very low votes in the Hough space. Second, when a leaf has many patches its displacements are often clustered around few locations. Since Hough Forests leaves can be thought as a discriminatively trained codebook all the leaves belonging to the same word (object part) will have similar displacements.

Considering the first issue we can safely remove low probability leaves to drastically reduce the amount of votes to be casted. We set the discarding probability to $0.5$.

Regarding the second issue, we synthesize all the votes by clustering the displacements at each leaf. With this simple idea it is possible to reduce and limit the complexity of the voting procedure. Once the clusters are computed, for each leaf the centroids $\mu_k$ of each cluster $c_k \in \mathcal{C}_l$ are stored. The fast voting procedure is now performed casting votes at the average location of the cluster $\mu_k$ with a weight equal to the leaf likelihood multiplied by the number of elements in the cluster:

$$V(x-\mu_k) = V(x-\mu_k) + |c_k|\frac{|P_l^F|}{|P_l|} \tag{5}$$

where $|c_k|$ is the cardinality of cluster $c_k \in \mathcal{C}_l$.

With this approximation we obtain coarser voting maps but with identical peak locations as can be seen in Figure 2.

### 3.3. Patch subsampling

We also use patch subsampling to reduce the amount of votes to be casted. The method proposed by Gall *et al.* relies on a dense sampling of the image, meaning that for *every* pixel of the image features are extracted from its local neighborhood. There is little to no changes from one pixel to another and hence there is an amount of useless computation since two neighbor pixels are very likely to vote with the same foreground probability for the same displacement vectors. Therefore, we investigate applying subsampling with factors of 2, 4 and 8 to further speed up the detection process while analyzing the potential loss in accuracy.

## 4. FEATURES FOR OBJECT DETECTION

In order to obtain high accuracy Gall *et al.* used a diverse set of features channels: 3 $Lab$ image channels, the absolute values of the first and second image derivatives and 9 HOG like channels. They use the max/min pooled versions of this 16 channels yielding 32 channels in total. This pooling is performed over regions of $5\times5$ in order to increase the invariance of channels under noise. For efficiency they do not compute an actual orientation histogram but apply a Gaussian blur with a $5 \times 5$ kernel on each channel thus approximating a HOG.

To reduce the feature computation time, we seek first a reduction in feature size. In this work we propose to use an actual histogram of orientations of the image gradient computed on 9 bins. To keep the evaluation of these 9 channels efficient we use the Integral HOG (IHOG).

We obtain gradient images $I_x$ and $I_y$ with a Sobel filtering and compute 9 orientations intensity channels quantizing $I_\theta = \arctan\left(\frac{I_y}{I_x}\right)$. We then compute integral images for each orientation image. When we evaluate a histogram of orientations for a given patch, this can be done with $4 \times 9$ additions independently of the patch size. The approximation computed by Gall *et al.* using the Gaussian blur requires $5\times5\times9$ kernel evaluations and summations. Roughly speaking we reduce the amount of operations by a factor of 4. The final speed-up will be lower since the integral image computation represents an additional cost in the channel computation.

This approach has two advantages. First it allows to reduce the redundant computation of many overlapping patches and avoids kernel evaluations. Second it avoids using an approximation of the HOG, potentially leading to better performance. In our case we can have the same performance as [13]

but reducing the features from 32 to 9. To further speed-up the method, $I_\theta$ is computed with the aid of a pre-computed look-up-table (LUT) avoiding floating point operations to compute the $\arctan(\cdot)$ function.

## 5. EXPERIMENTS

We tested our algorithm on the TUD Pedestrians dataset [19]. The dataset is composed of 400 training frames and 250 test $720 \times 576$ frames. We followed the same data split as [13] in order to obtain the fairest comparison. In the following experiments, we first detail the speed-up effect of each proposed improvement. Then we show how each of our speed enhancing techniques affect the detection accuracy. We evaluate the accuracy with precision/recall curves and also give the equal error rate (EER) value.

### 5.1. Speed-up analysis

All the proposed methods concur in consistently reducing the time complexity of our method. We selected the best trade-off speed/accuracy for each proposed improvements and evaluated the speed-up from the baseline (B) Hough forest approach.

In Figure 3 it is clear that the voting is taking more than 90% of the processing time in the approach from [13]. Applying the Foreground Selection (F) to the baseline full (B) forests halves the time. Moreover, using our more compact forest (S) we can reduce by a factor of 3 the computation with respect to the full forest. Combining this two ideas we go from $\approx$ 6s per image to less than 1s (S+F). Applying clustering (C) alone halves the computational cost of voting. Using also patch sub-sampling (P) the voting step is now clearly dominated by the features computation time. Switching to our features based on Integral HOG (I) using the LUT we reach a processing time of only 0.1s. This represents a 60X speed-up over (B) which is the algorithm proposed in [13]. Our final model applies Foreground Selection, uses 2 pixels subsampling and 8 clusters per leaf.

### 5.2. Impact on accuracy

As a first experiment we compare the hard-example mining of [13] with our own that keeps the forest compact and remove low probability leaves. It can be seen from Figure 4 (left) that there is no significant drop in performance. The curves are almost overlapped and the EER only slightly decreases from 0.89 to 0.87. Hence without significant loss of accuracy, this enables a 6X speed-up with regardes to the performance of [13], see column S+F in Figure 3.

The sampling step as explained in section 3.3 affects the timing quadratically. Unfortunately sampling votes too coarsely affects the accuracy a lot. We can see from Figure 4 (center) that sampling with a stride of 2 pixels is equivalent to full dense sampling, while for a sampling step of 4 we loose 3 precision-recall EER points, going further to 8 pixels stride detection precision-recall EER drops to less than 0.60. Hence, only the subsampling every 2 pixels is viable for the speed-up/accuracy trade-off.

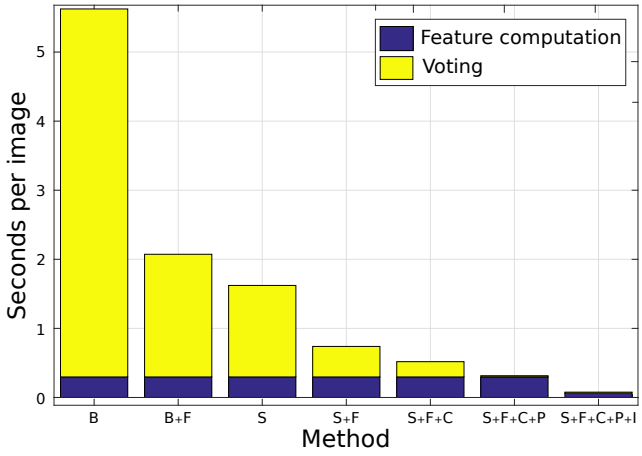


**Fig. 3**. Evolution of processing time per image applying the techniques proposed in the paper. B: Baseline [13]; F: Foreground selection; S: Smaller forest; C: Clustering of displacements votes; P: Patch Subsampling; I: Integral HOG features. We obtain a 60X speed-up total with respect to [13].

The foreground selection obtains a consistent speed-up but the voting procedure is still aggravating. We individuated the redundancy of casted location votes as a source of inefficiency. In Figure 4 (right) we show how using 8 clusters per leaf does not affect the accuracy at all; we start to lose accuracy using very coarse clustering like 2 with the maximum loss using the extreme case of a single vote per leaf.

## 6. DISCUSSION

In this paper, we have presented an object detection method based on Hough Forest targeted for the constraints of low powered wearable devices. We have shown how our method can speed up the detection process enabling real time processing even on a low computational power board. A thorough analysis of the computation cost for this family of algorithms has been performed. This analysis shed light on the main computational bottlenecks: voting and feature computation. We propose a better, faster and more compact features set and an improved voting algorithm that retain high accuracy with a final 60X speed up.

## 7. REFERENCES

[1] S. Hodges, E. Berry, and K. Wood, "Sensecam: A wearable camera that stimulates and rehabilitates autobio-

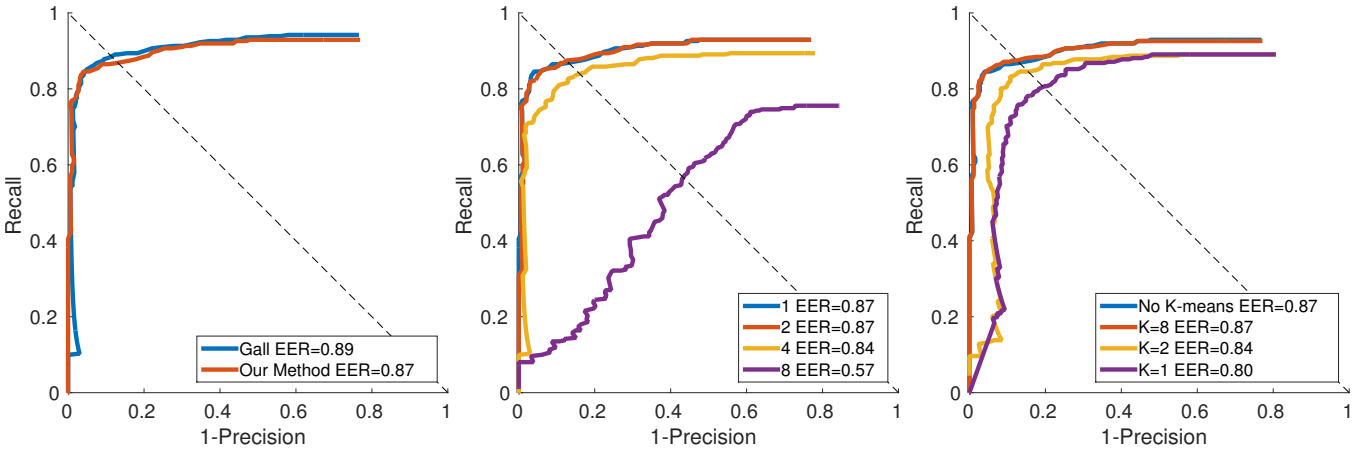


**Fig. 4**. ROC curves on TUD. Left: our hard mining training with a 5 trees forest and foreground selection vs. the 15 trees forest of [13], center: ROC curves varying the sampling step and on the right the displacements vectors clustering.


graphical memory," *Memory*, vol. 19, no. 7, pp. 685–696, 2011.

[2] S. Karaman, J. Benois-Pineau, V. Dovgalecs, R. Mégret, J. Pinquier, R. André-Obrecht, Y. Gaëstel, and J.-F. Dartigues, "Hierarchical hidden markov model in detecting activities of daily living in wearable videos for studies of dementia," *Multimedia Tools and Applications*, vol. 69, no. 3, pp. 743–771, 2014.

[3] V. Pradeep, G. Medioni, and J. Weiland, "Robot vision for the visually impaired," in *Proc. of CVPR Workshops*, 2010, pp. 15–22.

[4] R. O. Castle, D. J. Gawley, G. Klein, and D. W. Murray, "Video-rate recognition and localization for wearable cameras," in *Proc. of BMVC*, 2007, pp. 1100–1109.

[5] "Hardkernel odroid xu3," http://www.hardkernel.com/, Accessed: 2014-04-10.

[6] "Nvidia jetsonk developer kit," http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html, Accessed: 2014-04-10.

[7] Z. Lin, G. Li, Z. Li, and B. Zhang, "Developments of electrolyte systems for lithium-sulfur batteries: A review," *Name: Frontiers in Energy Research*, vol. 3, no. 5, 2015.

[8] A. Fathi, J.K. Hodgins, and J.M. Rehg, "Social interactions: A first-person perspective," in *Proc. of CVPR*, 2012, pp. 1226–1233.

[9] S. Alletto, G. Serra, S. Calderara, Solera F., and R. Cucchiara, "From ego to nos-vision: Detecting social relationships in first-person views," in *Proc. of CVPR Workshops*, 2014.

[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[11] B. Pepik, M. Stark, P. Gehler, and B. Schiele, "Occlusion patterns for object class detection," in *Proc. of CVPR*, 2013.

[12] M. Mathias, R. Benenson, R. Timofte, and L. Van Gool, "Handling occlusions with franken-classifiers," in *Proc. of ICCV*, 2013, pp. 1505–1512.

[13] J. Gall and V. Lempitsky, "Class-specific hough forests for object detection," in *Proc. of ICCV*, 2009.

[14] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," in *Proc. of BMVC*, 2009.

[15] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool, "Pedestrian detection at 100 frames per second," in *Proc. of CVPR*, 2012, pp. 2903–2910.

[16] N. Razavi, J. Gall, and L. Van Gool, "Scalable multi-class object detection," in *Proc. of CVPR*, 2011.

[17] R.O. Duda and P.E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[18] D.H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[19] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *Proc. of CVPR*, 2008.