# Unsupervised Moving Object Detection with On-line Generalized Hough Transform

Jie Xu, Yang Wang, Wei Wang, Jun Yang, and Zhidong Li

National ICT Australia
University of New South Wales
{jie.xu,yang.wang,jun.yang,zhidong.li}@nicta.com.au,
weiw@cse.unsw.edu.au

**Abstract.** Generalized Hough Transform-based methods have been successfully applied to object detection. Such methods have the following disadvantages: (i) manual labeling of training data ; (ii) the off-line construction of codebook. To overcome these limitations, we propose an unsupervised moving object detection algorithm with on-line Generalized Hough Transform. Our contributions are two-fold: (i) an unsupervised training data selection algorithm based on Multiple Instance Learning (MIL); (ii) an on-line Extremely Randomized Trees construction algorithm for on-line codebook adaptation. We evaluate the proposed algorithm on three video datasets. The experimental results show that the proposed algorithm achieves comparable performance to the supervised detection method with manual labeling. They also show that the proposed algorithm outperforms the previously proposed unsupervised learning algorithm.

## 1 Introduction

The detection of moving objects in videos, especially pedestrians or vehicles, is an important task in many vision applications, such as video compression, video surveillance, and content-based video retrieval. Numerous approaches have been proposed in the literature for object detection. Currently the predominant approach for object detection is the sliding window approach [1], [2], in which a learned classifier examines the image features over locations and scales to predict the presence of objects in subwindows. Though it has been demonstrated effective in many cases, it can be easily affected by background clutters and occlusions. To cope with the occlusion problem, part-based approaches [3], [4] which model objects as collections of parts are proposed.

The Generalized Hough Transform based methods [5], [6] can be categorized as part-based approaches. Each of them requires a class-specific codebook to cast probabilistic votes for object hypotheses. The codebook can be generated using generative clustering methods [5], and discriminative clustering methods [6]. Each cluster centroid corresponds to one codebook instance. At runtime, feature descriptors from the testing data are matched against the codebook instances, and valid matches then cast probabilistic votes for object hypotheses.

R. Kimmel, R. Klette, and A. Sugimoto (Eds.): ACCV 2010, Part III, LNCS 6494, pp. 145–156, 2011.

The additive nature of Generalized Hough Transform makes the detector robust to partial occlusions. However, these methods have the following disadvantages: (i) manual labeling of training data is required for the codebook construction; (ii) the codebook is constructed in an off-line manner, which cannot adapt to new data after the construction ends.

Several approaches have been proposed to tackle the problem of manual labeling of training data. The idea of *co-training* is proposed to incrementally generate a large amount of labeled data automatically from a small manually labeled set [7]. Given a small hand labeled set, a *pair* of classifiers are trained on two independent "views" of the data [7]. Co-training then generates the additional training data from the unlabeled data, by using each classifier's prediction to enlarge the other classifier's training set [8]. Alternatively, Wu *et al.* uses a small labeled training set to train an automatic labeler, which is then used to collect the training samples for the on-line boosting in [9]. Both approaches require hand labeled sets for initialization. To overcome the limitation of hand labeling, the idea of automatic labeling is proposed. Nair *et al* employs the motion based object detector as the labeler in [10]. However, motion based object detector is not robust, and can be affected by shadows, reflections and illumination changes. To improve such labeler, Roth *et al.* uses the PCA-based reconstructive model [11], to verify the motion detection results. As for the codebook construction for the Generalized Hough Transform, tree-based codebooks have become popular recently. The Extremely Randomized Trees [6], and the Random Forests [12] have been demonstrated to improve the performance of the Generalized Hough Transform. Such trees are usually learned offline, however Saffari *et al.* recently propose an on-line algorithm to enable the on-line learning of Random Forests [13].

In this paper, we propose an unsupervised moving object detection algorithm, with on-line Generalized Hough Transform. Our contributions are two-fold: (i) an unsupervised on-line training data selection algorithm based on Multiple Instance Learning (MIL); (ii) an on-line Extremely Randomized Trees construction algorithm for on-line codebook adaptation. The most related algorithm to our automatic training data selection algorithm is the co-training algorithm [7], and also the conservative learning algorithm [11]. Unlike the co-training algorithm, our algorithm does not require any hand labeling. In the conservative learning algorithm, a reconstructive model is employed to verify the motion detection results. Only the sufficiently consistent motion detections would be used to build the reconstructive model, and hence it might result in a biased training set. In contrast, our algorithm employs an instance selection scheme to produce a training set with less selection bias. For our proposed on-line Extremely Randomized Trees algorithm, the most related work is the on-line Random Forest algorithm by Saffari *et al.* in [13]. Different from the on-line Random Forest, our on-line Extremely Randomized Trees do not require the bootstrapping, and hence it is more computationally efficient.

The rest of the paper is organized as follows: the proposed work is described in Section 2, followed by the experimental results in Section 3. Our final conclusions are presented in Section 4.

## 2 Proposed Work

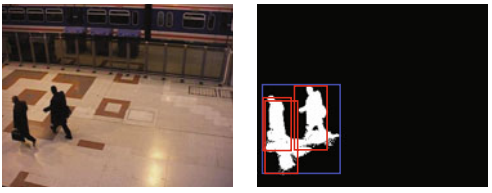
In this section, we present our unsupervised moving object detection algorithm with on-line Generalized Hough Transform. We design our automatic labeler based on Multiple Instance Learning for training sample selection. Given a set of noisy detection results from the background subtraction, the automatic labeler selects training samples automatically and unbiasedly. The on-line learning algorithm then uses the selected samples for codebook adaptation.

### 2.1 Automatic On-line Instance Selection

We present our automatic labeler design in this section. An automatic labeler is actually an object detector, which selects sub-windows that contain objects. Generally speaking, there are two issues in the design of a labeler for object detection. One issue is the labeler's error, which can be categorized as the alignment error and the labeling error. An alignment error occurs when the sub-window selected by the labeler contains an object with inaccurate size of positions, whereas a labeling error occurs when the selected sub-window contains no object. The other issue is the labeler's bias. The labeler should not introduce any bias into the produced training data, otherwise it may mislead the detector. For instance, if the labeler systematically fails to collect some certain type of training sample, the detector would not be able to recognize the corresponding object. We will show how our design can cope with these two issues.

We begin our design with background subtraction. Given a video, background subtraction generates a set of foreground blobs, which comprise the training samples for selection. Since background subtraction is not robust against environmental factors, alignment and labeling error might occur. To handle the errors, we introduce the Multiple Instance Learning (MIL) to our labeler design. In MIL, the training data comes in the form of "bags", where all the instances in one bag share a label. A positive bag means it contains at least one positive instance, whereas a negative bag means all the instances are negative. The advantage of MIL is that, it can handle both the ambiguity and noises in the instance labeling. In our problem, each foreground blob corresponds to a positive bag. Given one foreground blob, in order to locate the possible locations of individual persons, a smoothed histogram of foreground heights over the $x$-axis is computed. We assume that the tops of objects correspond to the peaks of the histogram. After the peaks are located, we crop the corresponding instances using bounding boxes. Figure 1a depicts the image frame, and Figure 1b demonstrates the detected foreground blob and its bag formulation. In Figure 1b, the blue rectangle is the foreground blob, while the red rectangles correspond to the instances inside the bag. As shown in Figure 1b, the foreground blobs contains two pedestrians, but there are more than two instances found in the corresponding bag due to the noisy motion detection result.

To deal with the noisy detection, we propose to use the following scheme to select instances from all the bags. Let $\mathbb{B} = \{B_1^+, B_2^+, ..., B_n^+\}$ be the set of all positive bags. The goal of the selection is to select the instance $B_{gh}$ that has high confidence $\mathrm{Conf}(B_{gh})$, which is defined in the follows:



(a) (b)

**Fig. 1.** The formation of a positive bag. (a) The image frame. (b) The detected foreground and its bag formulation. The blue rectangle is the foreground blob, while the red rectangles correspond to the instances inside the corresponding bag. This figure is best viewed in color mode.

$$Conf(B_{gh}) = \prod_{k} Pr(B_{gh}|B_k^+),\ 1 \leq k \leq n,\ k \neq g, \tag{1}$$

where $Pr(B_{gh}|B_k^+)$ is estimated based on the Noisy-OR model [14]:

$$Pr(B_{gh}|B_k^+) \propto \{1 - \prod_{j}[1 - Pr(B_{gh}|B_{kj}^+)]\}. \tag{2}$$

We can design different estimations for $Pr(B_{gh}|B_{kj}^+)$ based on different data. For the task of object detection, we intend to evaluate the similarity between two object blob silhouettes, and $Pr(B_{gh}|B_{kj}^+)$ is estimated as $Pr(B_{gh}|B_{kj}^+) \propto \exp\{-D(B_{gh}, B_{kj}^+)\}$, where $D(B_{gh}, B_{kj}^+)$ measures the distance between the silhouettes $B_{gh}$ and $B_{kj}^+$. In this paper, we design the distance based on distance transform. To make sure our method is as general as possible, only positive bags are required for the computation of the evidence. In the situations when the negative bags are also available, we can use the Evidence Confidence metric proposed in [15].

The aforementioned instance selection scheme is a batch process. To enable the on-line selection, we propose an on-line algorithm for the above selection algorithm. We choose to realize the on-line learning by selecting the instances from every $R$ frames, where $R$ is a pre-defined value to determine the size of the interval. The proposed on-line learning algorithm is presented in Algorithm 1.

### 2.2 On-line Extremely Randomized Trees

Given a set of selected instances, we attach shape context descriptors to the sampled points from the corresponding silhouettes. The obtained descriptors are used to construct a codebook of shapes for object silhouettes. The codebooks for the Generalized Hough Transform are usually generated using unsupervised $k$-means clustering [5], [16]. We call them generative codebooks as there is no discrimination involved. Recently discriminative codebook generation methods are proposed [12], [6]. The generated codebooks are considered as discriminative codebooks as

**Algorithm 1.** Automatic On-line Instance Selection

INPUTS:
$\mathbb{F}_t$ - The extracted foreground from frame $t$ to frame $t + R$, where $R$ is a pre-defined value
$K$ - The number of instances to select from all the instances
OUTPUTS:
A set of instances $B_{ij}$ for training the randomized trees
1: Form the positive bags $\mathbb{B} = \{B_1^+, B_2^+, ..., B_n^+\}$ based on $\mathbb{F}_t$
2: Compute the confidence for all the instances $B_{ij}$
3: Select the top $K$ instances $B_{ij}$ with the highest confidence

they are trained in a supervised way. The supervision enables the codebook entries to cast more reliable probabilistic votes. In [12], a Random Hough Forest is constructed using both positive and negative image patches, with an objective function that measures the class and offset uncertainty. On the other hand, a set of Extremely Randomized Trees are constructed in [6], and the trees are grown using an objective function that combines the discrimination and regression. The discriminative codebooks are shown to outperform the generative codebook in the experiments. As a result, we use the discriminative codebook in our paper.

We choose the Extremely Randomized Tress [17] as our discriminative codebook. The randomized trees algorithm [17] constructs an ensemble of decision or regression trees. And each tree is grown by splitting each node into two child nodes, using the random split that achieves the best decision or regression performance based on the whole training set. The randomized trees algorithm is firstly proposed for classification, and Okada employs it as the codebook for the Hough voting [6]. Each primitive image feature passes through each randomized tree until it reaches one of the leaf node. The leaf node contains information about the discrimination of the image feature(whether it belongs to an object or not), and possible object locations are collected during the training. The response of one image feature is an ensemble of the responses from all the trees. Using the responses, each feature can cast probabilistic votes for object hypotheses. The randomized tree construction algorithms in [17], [6] are all based on the whole training set. It is not appropriate to use them under our problem settings, as we want to be able to update the trees in an on-line fashion. Inspired by the on-line random forest algorithm in [13], we propose an on-line learning algorithm for constructing the randomized trees here. It is noted that the randomized trees are different from random forests as there is no bootstrapping involved in the randomized trees [17].

We build each randomized tree as a decision tree, which contains the decision nodes and the leaf nodes. Unlike the leaf node, each decision node retains no object location but only a split condition $s = \{f_d, \theta_d\}$, where $f_d$ and $\theta_d$ are a randomly chosen attribute from the image feature vector, and its threshold respectively. The split $s$ is the best split chosen from a set of random splits

$S = \{s_1, s_2, ..., s_N\}$ based on some quality measure. In our paper, the information gain is chosen as the quality measure. Denote $M$ as the set of image features in the current node. Let $M_L$ and $M_R$ be the images features in the left child node and right child node respectively, according to the split $s$. The information gain of split $s$ is $IG_s(M) = \frac{|M_L|}{|M|}E(M_L) + \frac{|M_R|}{|M|}E(M_R) - E(M)$, where $E(M) = -\sum_{i=1}^{C} p_i \log(p_i)$ is the entropy for $C$ classes.

When in off-line mode, all the data is available, and therefore a robust estimate can be made at each decision node. In the on-line mode, however, the data is gathered over time, and hence, when to split depends on the following factors: i) whether there is enough data for the robust estimate of statistics; ii) whether the split is good enough in terms of the quality measure. Based on the these factors, two hyper-parameters are introduced for the on-line learning of a random tree: i) the minimum number of training data (i.e., shape context descriptors) $\gamma$ to gather before making a split; ii) the minimum information gain $\delta$ for a node to split. And therefore, a node can be split into two child nodes only if $|M| > \gamma$ and $\exists s \in S,\ IG_s(M) > \delta$. The values of $\gamma$ and $\delta$ are set in a similarly way to method mentioned in [13].

The on-line learning algorithm for the randomized trees construction is presented in Algorithm 2. The input to algorithm is either a positive or negative training sample $\langle\ x,\ y\ \rangle$, which contains a feature descriptor $x$ and its label $y \in \{1, 0\}$. In this paper we use the shape context as the feature descriptor. The positive feature descriptors describe the sampled points from the selected instance $B_{ij}$ from Algorithm 1, whereas the negative descriptors describe the

**Algorithm 2.** On-line Extremely Randomized Trees

INPUTS:
$\langle\ x,\ y\ \rangle$ - a training sample from a sampled keypoint
$\gamma$ - The minimum number of training data to gather before making a split
$\delta$ - The minimum information gain for a node to split
$T$=$\{t_1, t_2, ..., t_n\}$ - A set of Extremely Randomized Trees
OUTPUTS:
$T'$=$\{t'_1, t'_2, ..., t'_n\}$ - The updated Extremely Randomized Trees

```
for Each Extremely Randomized Tree t_i do
    l_j ← locateLeaf(x, t_i)
    l_j ← appendData( l_j, ⟨x, y⟩)
    if |l_j| > γ then
        S ← createSplts( l_j)
        if ∃s ∈ S, IG_s(l_j) > δ then
            createLeftChild( l_j, s)
            createRightChild( l_j, s)
        end if
    end if
end for
```

sample points from the background edges. Positive samples also retain the offsets to the centroids of an object, so that the constructed randomized tree can be used for probabilistic voting, which is detailed in Section 2.3. When updating a tree, a training sample firstly passes each randomized tree until it reaches the leaf node. After appending the new feature to the leaf node, we calculate whether it is necessary to split the current leaf. In the case of a split, the data retained in the old leaf node will be propagated to its child nodes, and the old leaf node becomes a decision node.

### 2.3 Object Detection

We begin the moving object detection with identifying moving edges between adjacent frames. We apply Canny edge detection [18] to obtain the edge map for each frame. Moving edges are then extracted by comparing edges between adjacent frames. We then sample keypoints from the moving edges, and attach shape context descriptor to each sampled keypoints. Let $F = \{f_1, f_2, ...f_n\}$ be the shape context descriptors obtained from the current frame, $F$ will be then fed into the randomized trees $T = \{t_1, t_2, ..., t_n\}$ to cast probabilistic votes for an object $o$ and its location $x$. The probabilistic vote $p(o, x|f_i, T)$ from feature $f_i$ can be decomposed as $p(o|f_i, T)p(x|o, f_i, T)$. The first term $p(o|f_i, T)$ is a probabilistic output from the ensemble of trees. Denote $M_{f_i,t_j}$ as the set of training features belong to the leaf node to which $f_i$ reaches in tree $t_j$. Let the number of training features in $M_{f_i,t_j}$ be $N_{f_i,t_j} = |M_{f_i,t_j}|$, and that of the positive features be $N^p_{f_i,t_j} = |M^p_{f_i,t_j}|$. The purity of the leaf node can be defined as $\gamma_{f_i,t_j} = \frac{N^p_{f_i,t_j}}{N_{f_i,t_j}}$. We only consider the trees with leaf nodes whose purity is higher than a predefined threshold. Assume the number of such trees to be $N^o_{f_i}$, and $p(o|f_i, T)$ is defined as $p(o|f_i, T) = \frac{N^o_{f_i}}{N_T}$, where $N_T$ is the number of randomized trees.

---

**Algorithm 3.** Moving Object Detection with On-line Generalized Hough Transform

AUTOMATIC LABELING AND ON-LINE LEARNING
**for** every $R$ frames **do**
  Perform background subtraction, and group foreground pixels into blobs
  Use the Algorithm 1 to select instances from the foreground blobs
  Attach descriptors to sample edge points from instances and background
  Use the descriptors to update the randomized trees based on Algorithm 2
**end for**
ON-LINE MOVING OBJECT DETECTION
**for** Each frame **do**
  Identify moving edges
  Attach descriptors to the sample edge points from the moving edges
  Use the randomized trees to cast probabilistic votes based on the descriptors
**end for**

---

The second term $p(x|o, f_i, T)$ describes the distribution of possible object centroid location in regard to $f_i$ supposing $f_i$ being part of the object. The distribution is estimated using a non-parametric density estimation using all the trees:

$$p(x|o, f_i, T) \propto \sum_{j=1}^{N_T} \{\gamma_{f_i,t_j} \sum_{k \in M^p_{f_i,t_j}} K(\frac{x - x^p_k(f_i)}{b(x^p_k)})\}, \quad (3)$$

where $K(.)$ is a window function, $b(.)$ is its bandwidth, and $x^p_k(f_i)$ corresponds to the object centroid location relative to the feature $f_i$ based on the positive training feature $x^p_k$.

The complete unsupervised moving object detection algorithm is summarized in Algorithm 3. The proposed algorithm updates the randomized trees using the collected training samples from every $R$ frames, and then the updated trees are used to cast probabilistic votes for object hypotheses based on the moving edge detection results.

## 3 Experiments

**Experimental Setup.** We evaluate the performance of the proposed framework on moving object detection using three video datasets. The first two of them, including the PETS2006 benchmark set[1] and the i-LIDS dataset[2], are indoor video surveillance on pedestrian activities. The third dataset contains outdoor traffic surveillance video captured in a highway during daytime.

**Evaluation Metric.** We follow the evaluation criteria employed in [5] that covers three categories, and they are *relative distance*, *cover*, and *overlap*. The *relative distance* measures the distance between the center of a bounding box and that of the ground truth. The *cover* and *overlap* measure how much area of the ground truth bounding box is covered by the detection hypothesis, and vice versa. A hypothesis is classified as a true positive if the *relative distance* $\leq 0.5$ and both *cover* and *overlap* are above 50%.

### 3.1 The PETS2006 Dataset

We evaluate the two components of the proposed framework using the PETS2006 dataset. We extract four sequences from the dataset, and use one sequence for training, and the rest three for testing. The number of moving objects in the testing sequences are 842, 312 and 413 respectively.

**The Automatic On-line Instance Selection.** To evaluate our automatic labeler, we collect two training sets from the training sequence using manual selection and the proposed labeler respectively. We then use them to train two sets of batch Extremely Randomized Trees [6] respectively. The obtained randomized trees are tested on the testing sequences based on the detection algorithm

[1] http://www.cvg.rdg.ac.uk/PETS2006/data.html
[2] http://www.elec.qmul.ac.uk/staffinfo/andrea/avss2007.html

detailed on Section 2.3. The precision recall curves of both sets of trees are depicted in Figure 2a. As shown in the figure, the randomized trees trained on the automatic labeled set achieve comparable performance with the trees trained on the hand labeled set. It is noted that, the former even outperforms the latter on the third testing sequence. This might be due to the selection bias of the manual labeling. Sample detection results can be found in Figure 3.

**The On-line Extremely Randomized Trees.** We compare the proposed on-line learning algorithm for the randomized trees with the corresponding batch learning algorithm [6]. Given the same training set, we construct two sets of randomized trees using the on-line and batch learning algorithm respectively. These two sets of randomized trees are then tested on the testing sequences. Figure 2b depicts the precision recall curves of both sets of randomized trees on the testing sequences. It can be seen from the curves that, the on-line learning algorithm reaches comparable or even better precision than the batch learning algorithm at the same recall value. These indicate that the proposed on-line learning algorithm for the randomized trees adapts to the incoming data better than the batch learning algorithm.

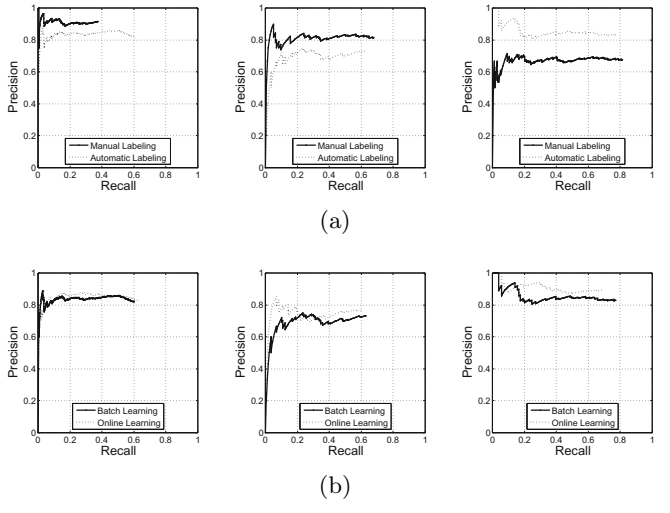



**Fig. 2.** The precision recall curves on the PETS2006 dataset: (a) the curves correspond to manual and automatic labeling, (b) the curves correspond to batch and on-line learning

### 3.2 The i-LIDS Dataset

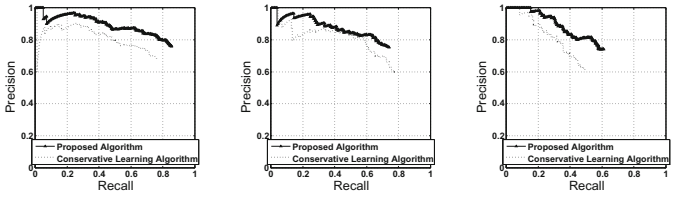
As our second experiment, we compare the proposed algorithm with the unsupervised on-line conservative learning algorithm in [11]. The labelers of both algorithms are based on the simple background subtraction results from the training video. In [11], a reconstructive model based on appearance and shape is employed to verify the foreground blobs. In this experiment, only the shape

(a) Sequence 1 (b) Sequence 2 (c) Sequence 3

**Fig. 3.** Sample detection results of randomized trees based on manual and automatic labeling. For each pair of results, the manual labeling-based detection is shown on the left, and the automatic labeling-based detection is shown on the right.

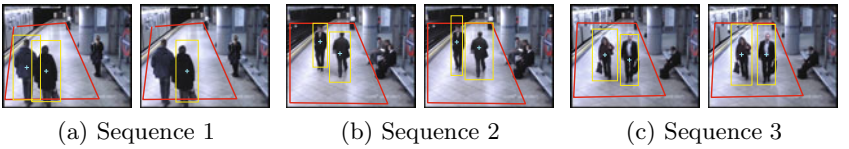
information is used for a fair comparison of both frameworks. The conservative labeler only considers the foreground blobs whose aspect ratio are within the predefined limits. For instance, Figure 1 depicts one frame in the training set, and also the corresponding foreground blob. The aspect ratio of the blob exceeds the predefined limit, and hence is not considered by the conservative learning. The conservative learning algorithm might fail to capture the multi-modal nature of the data due to its conservativeness. On the other hand, our proposed labeler does not have such requirement, and also accepts this blob for instance selection. As a result, the proposed algorithm would have less selection bias. For the object detection, we use the proposed on-line randomized trees for object detector for both algorithms.

We extract three sequences from the i-LIDS dataset, and each of them contains 250, 202, and 262 objects respectively. We compare both algorithms on these sequences, and their performances are shown in the precision recall Curves




(a) Sequence 1 (b) Sequence 2 (c) Sequence 3

**Fig. 4.** The performance of the proposed algorithm and the conservative learning algorithm on the i-LIDS dataset



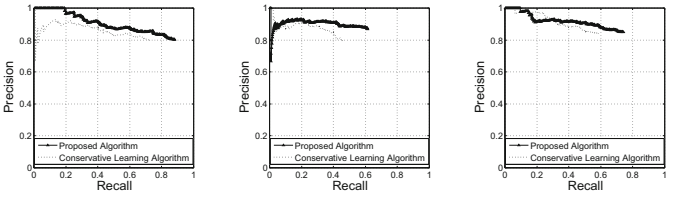
(a) Sequence 1 (b) Sequence 2 (c) Sequence 3

**Fig. 5.** Sample detection results on the i-LIDS dataset. For each pair of results, the detection obtained by the proposed algorithm is shown in the left, and that obtained by the conservative learning is shown on the right.

in Figure 4. It can been observed from the figure that, the proposed algorithm outperforms the conservative learning framework. Our framework reaches higher precision in all testing sequences. This indicates that the proposed framework captures the multi-modal nature of pedestrian silhouettes better than the conservative framework. Sample detection results can be found in Figure 7.

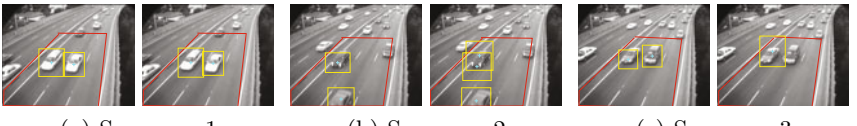
### 3.3 The Traffic Dataset

As our last experiment, we compare the proposed algorithm with the unsupervised on-line conservative learning algorithm in [11] for vehicle detection. Similarly, only shape information is used here, and we also use the on-line randomized trees for object detection. The performance of both learning algorithms are shown in Figure 6. It is seen in the figure that, the proposed algorithm slightly outperforms the conservative learning algorithm. This result indicates that the silhouettes of the vehicles might follow a unimodal distribution, since most vehicles in the videos are vans and trucks. Sample detection results can be found in Figure 7.




(a) Sequence 1 (b) Sequence 2 (c) Sequence 3

**Fig. 6.** The performance of the proposed algorithm and the conservative learning algorithm on the traffic set



(a) Sequence 1 (b) Sequence 2 (c) Sequence 3

**Fig. 7.** Sample detection results on the traffic dataset. For each pair of results, the detection made by the proposed algorithm is shown in the left, and that made by the conservative learning algorithm is shown on the right.

## 4 Conclusions

We have presented a novel algorithm for on-line unsupervised learning of object detection system. The basic idea is to start with a simple motion detection system, and then select the optimal foreground blobs based on the Multiple Instance Learning. Subsequently the selected blobs are used to construct a set

of Extremely Randomized Trees in an on-line manner. We have evaluated the algorithm on three video datasets. The experimental results demonstrate that our algorithm outperforms the on-line conservative learning algorithm.

**Acknowledgements.** National ICT Australia (NICTA) is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Center of Excellence program. Dr. Wei Wang is supported by ARC Discovery Grant DP0987273.

## References

1. Dalai, N., Triggs, B., Rhone-Alps, I., Montbonnot, F.: Histograms of oriented gradients for human detection. In: CVPR, vol. 1 (2005)
2. Munder, S., Gavrila, D.: An experimental study on pedestrian classification. TPAMI (2006)
3. Andriluka, M., Roth, S., Schiele, B.: People-tracking-by-detection and people-detection-by-tracking. In: CVPR (2008)
4. Fergus, R., Perona, P., Zisserman, A.: Weakly supervised scale-invariant learning of models for visual recognition. IJCV (2007)
5. Leibe, B., Leonardis, A., Schiele, B.: Robust Object Detection with Interleaved Categorization and Segmentation. IJCV (2008)
6. Okada, R.: Discriminative Generalized Hough Transform for Object Detection. In: ICCV (2009)
7. Balcan, M., Blum, A., Yang, K.: Co-training and expansion: Towards bridging theory and practice. In: NIPS (2005)
8. Javed, O., Ali, S., Shah, M.: Online detection and classification of moving objects using progressively improving detectors. In: CVPR (2005)
9. Wu, B., Nevatia, R.: Improving part based object detection by unsupervised, online boosting. In: CVPR (2007)
10. Nair, V., Clark, J.: An unsupervised, online learning framework for moving object detection. In: CVPR (2004)
11. Roth, P., Grabner, H., Skocaj, D., Bischof, H., Leonardis, A.: On-line conservative learning for person detection. In: VS-PETS (2005)
12. Gall, J., Lempitsky, V.: Class-Specific Hough Forests for Object Detection. In: CVPR (2009)
13. Saffari, A., Leistner, C., Santner, J., Godec, M., Bischof, H.: On-line Random Forests. In: The 3rd On-line learning for Computer Vision Workshop (2009)
14. Maron, O., Lozano-Pérez, T.: A framework for multiple-instance learning. In: NIPS (1998)
15. Li, W., Yeung, D.: Localized content-based image retrieval through evidence region identification. In: CVPR (2009)
16. Maji, S., Malik, J.: Object detection using a max-margin hough transform. In: ICCV (2009)
17. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning (2006)
18. Canny, J.: A computational approach to edge detection. TPAMI (1986)