Integrated Pedestrian and Direction Classification using a Random Decision Forest

Junli Tao and Reinhard Klette
University of Auckland, Auckland, New Zealand
jtao076@aucklanduni.ac.nz, rklette@auckland.ac.nz

Abstract

For analysing the behaviour of pedestrians in a scene, it is common practice that pedestrian localization, classification, and tracking are conducted consecutively. The direction of a pedestrian, being part of the pose, implies the future path. This paper proposes novel Random Decision Forests (RDFs) to simultaneously classify pedestrians and their directions, without adding an extra module for direction classification to the pedestrian classification module. The proposed algorithm is trained and tested on the TUD multi-view pedestrian and Daimler Mono Pedestrian Benchmark data-sets. The proposed integrated RDF classifiers perform comparable to pedestrian or direction trained separated RDF classifiers. The integrated RDFs yield results comparable to those of state-of-the-art and baseline methods aiming for pedestrian classification or body direction classification, respectively.

1. Introduction

In a *driver assistance system* (DAS) context, pedestrians, as the most vulnerable road users, require special attention for being protected. Algorithms [7, 8, 13, 16, 28] are proposed to classify pedestrians, instead of taking them just as "obstacles" (such as cars, buildings, or rubbish bins) which block a direction of driving. Due to clustered backgrounds, changes in illumination, variable clothing, different poses, or changes in localization, pedestrian *detection* (i.e. localization and classification) is still a challenging task if aiming at close to 100% accuracy.

In order to further analyse their behaviour, *e.g.* when crossing a road, detected pedestrians are tracked consequently. Tracking-by-detection algorithms [3, 20, 21, 26, 29, 30] work by associating temporally detected pedestrians to derived trajectories. Due to noisy classification results and irregular movements, detection and motion-based tracking methods produce even more noisy trajectories, especially when a pedestrian is standing or slowly moving only.

Knowledge about previous trajectories or motion is normally utilized to predict possible locations of pedestrians in the current frame. A pedestrian may change the walking path abruptly, especially when moving slowly, or when starting to walk again after standing. In these cases, the model about previous motion requires time to converge to the new motion, and information about body direction helps to converge faster.

Body direction implies a direction of a future path, which may be totally uncorrelated to the previous trajectory. Discrete body-direction classification algorithms are proposed in [1, 4, 18, 23, 32], assigning either one of eight (namely *N*, *NE*, *E*, *SE*, *S*, *SW*, *W*, or *NW*) or of four directions (namely *N*, *E*, *S*, *W*) to each pedestrian; see Fig. 1. [16] points out that adopting such a classification of directions reduces the effect of noisy tracking results and improves path prediction.

Pedestrian and direction classification are commonly handled separately in different processing modules. Different classifiers are trained for pedestrian and direction classification, respectively. For direction classification, again multiple classifiers are trained for each direction separately. In [11, 18], both tasks are considered together by adopting

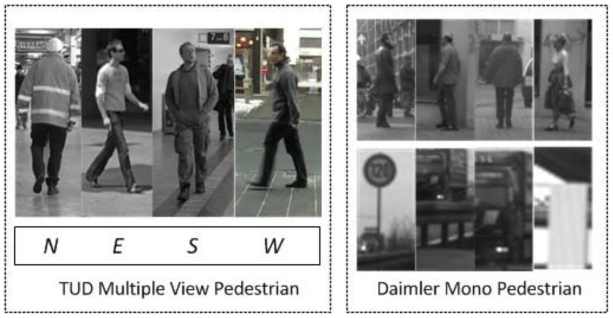

Figure 1. Direction-labelled pedestrian (*N*, *E*, *S*, or *W*), pedestrian, or non-pedestrian bounding boxes. Direction-labelled pedestrians are from the *TUD Multiple View Pedestrian* database, and pedestrian or non-pedestrian boxes are from the *Daimler Mono Pedestrian* database.

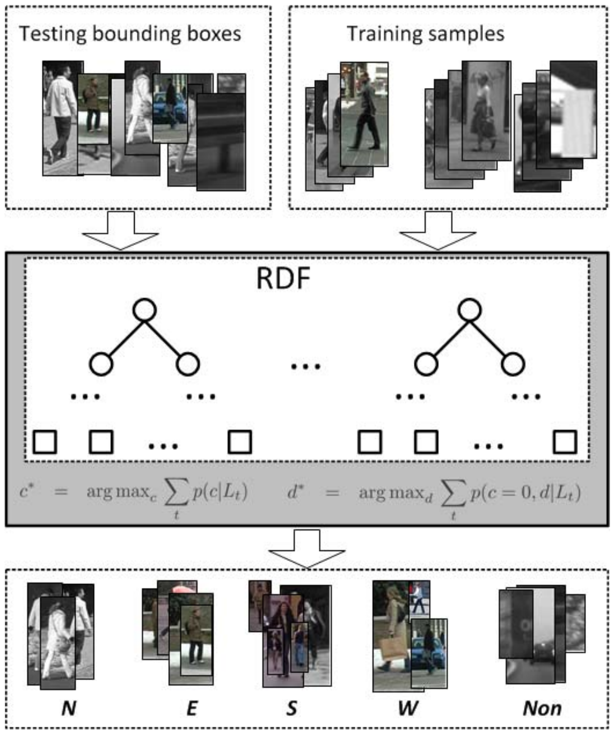
Testing bounding boxes
Training samples
RDF
$c^* = \arg\max_c \sum_t p(c|L_t)$
$d^* = \arg\max_d \sum_t p(c=0,d|L_t)$
N
E
S
W
Non

Figure 2. Pedestrians and their body directions are classified simultaneously in one RDF classifier. During training, non-pedestrians, pedestrians, and direction-labelled pedestrians are employed all together. The structural flexibility of RDF enables auto-adjusting of the focus for both classification tasks. Going through the trained RDF, test bounding boxes are classified to be either a non-pedestrian (class *Non*), or a pedestrian with direction (*N*, *E*, *S*, or *W*).

cascade classifiers or a Bayesian model. This paper proposes novel RDF structures to address both classification tasks in one RDF classifier, as briefly sketched in Fig. 2. The experiments prove that the proposed combined RDFs is able to handle both classification tasks successfully without degrading the performance of each task.

The paper is structured as follows. In Section 2 we briefly review related work. Section 3 provides the proposed algorithm. Experiments and their interpretation are given in Section 4. Section 5 concludes.

2. Related Work

Research on pedestrian classification or detection is conducted at many research institutions worldwide. The derivation of global or holistic descriptors for a rectangular *bounding box* with a trained classifier is a standard approach in the field [7, 8, 9, 10, 17, 27, 28, 24]. Positive bounding boxes contain at least one pedestrian, and negative boxes contain background only; this classification is performed by using holistic features. Then, one or more classifiers are trained to separate pedestrians from the background. Finally, test bounding boxes described by the same feature pass through the classifier. Likelihoods are obtained, denoting whether there is a pedestrian in a given bounding box.

The *histogram of oriented gradients* (HoG) proved to be an efficient description for pedestrian classification [7]. Other features, such as *Haar-like* [17], *histogram of oriented flow* (HoF) [8], or *linear binary patterns* (LBP) [27] are adopted with HoG to further improve the performance. The employed classifier is another important decision. *Support vector machines* (SVM) or AdaBoost are used to deal with high-dimensional feature vectors. See [28] for a comparison of different descriptors and classifiers used for pedestrian detection and classification.

Instead of using a feature vector to describe the whole bounding box, body-part-based features are employed in [19, 12]. Gall *et al.* [13, 14, 15] train a classifier with simple features of randomly extracted patches from training samples. In [19], SIFT features are used to extract patches in training samples containing body parts, *e.g.* leg, arm, or torso, followed by clustering such patches. Visually similar patches are grouped together. When applying the trained classifier, selected boxes are matched with representations of those clusters.

A pictorial structure-based method is proposed in [12] for recognizing objects and their pose. A prior model is calculated based on a database of images of connected body parts. The method aims at detecting an optimal match of a model with a given image. Thus, a pose is recognized when corresponding to a model.

Methods for direction classification adopt features and classifiers already used for pedestrian classification. The HoG proved to perform better than a classic covariance descriptor [2]. Multiple classifiers, *e.g.* by using SVM or AdaBoost, are adopted for direction recognition, as at least four directions (classes) are required to be separated. Each classifier is trained for recognizing one direction [1, 2, 4, 18, 23, 32]. Maximum probability decides then which direction is selected.

[23] presents a method based on silhouettes. Used shape descriptors limit the range of direction to the interval $[0°, 180°]$. [25] proposes to estimate pedestrian direction more robustly by selecting a recognition result based on multiple still images, rather than just based on a single image. In [2], multiple random-tree classifiers are trained and compared with trained SVM classifiers. Outputs are integrated using a *mixture of approximated wrapped Gaussians* (MAWG). Using calculated probabilities of multiple outputs obtained from all participating classifiers, the final direction is obtained by maximizing the mixed probability of the WAWG. Direction recognition is difficult as head pose, torso, and body might point into different directions. But they are related to each other. For instance, in [4, 5], body direction is estimated by considering loca-

tion and head pose, and assuming that tracks are available.

Very few researchers consider combining the pedestrian detection and direction recognition tasks. [18] presents a three stage process. Stages 1 and 2 adopt different HoG (with blocks either overlapping or not) to reject non-pedestrian boxes. In Stage 3, four SVM classifiers are trained for the four directions separately using pedestrian samples only. In [11], a unified Bayesian model, using shape and motion cues, is proposed to classify pedestrians and to recognize one of four directions.

RDF techniques have been adopted in various applications already, including keypoint recognition [22], object detection [13], tracking [14], or action recognition [31]. In [15], multiple objects (cars or pedestrians) are classified using one classifier trained with samples of both types of objects. In this paper we propose to integrate pedestrian and direction classification tasks into *one* RDF classifier.

An RDF has a flexible structure, and details are given in Section 3.1. Each node of one of the trees is a weak classifier, and different target functions adopted enable the splitting process to achieve different goals. We proposes novel structured RDFs, which are actually able to focus on different tasks simultaneously. For the adaptive structured RDF, initially, the splitting nodes focus on classifying pedestrians in distinction to the background. When samples arrive at the current node which contain mostly pedestrians, direction classification comes into the process. For our randomly structured RDF, the target for a split node is randomly selected.

3. Proposed Algorithm

The first subsection starts with a general outline, also recalling the RDF framework. Details of the algorithm are provided in the following subsections.

3.1. Random Decision Forests

An RDF consists of a set of randomly trained decision trees T_t, for $t \in \{1, \ldots, N\}$. Each tree is considered as being a weak classifier. Furthermore, such a weak classifier consists of a set of split functions hierarchically arranged into a tree structure. Thus, classification problems are split by answering simple questions of the split functions.

There are two types of nodes in a decision tree, internal (or split) or terminal (or leaf) nodes. Each split node has a split function and two outgoing edges which are connected to two nodes (split or leaf nodes); see Fig. 3. A split function $h_\phi(\cdot)$ decides which node (left or right) comes next:

$$\mathcal{I}_L(\phi) = \{I \in \mathcal{I} | h_\phi(I) = 0\} \tag{1}$$

$$\mathcal{I}_R(\phi) = \{I \in \mathcal{I} | h_\phi(I) = 1\} \tag{2}$$

where $\mathcal{I}$ denotes the set of inputs; $h_\phi(\cdot)$ and its parameters ϕ are defined later.

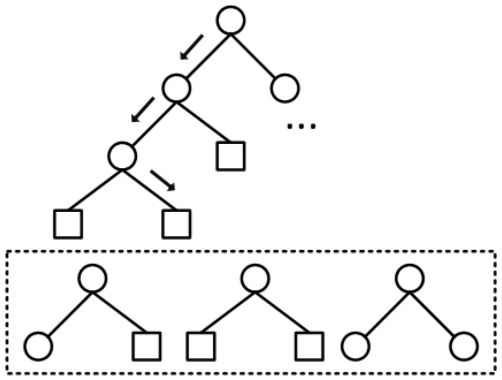

Figure 3. *Top*: A decision tree; circles denote split nodes, rectangles denote leaf nodes. The arrows denote the path a bounding box might define until reaching a leaf node. *Bottom*: Three possible split cases.

During training, a set $\mathcal{I}^{tr} \subset \mathcal{I}$ of labelled pedestrian and background samples are employed for expanding the trees. Starting at a root node, a decision tree is trained by growing subsequently internal nodes, i.e. by selecting suitable functions $h_\phi(\cdot)$ with respect to a predefined target function.

Samples $I \in \mathcal{I}^{tr}$ split along the internal nodes and end up in a leaf node where a stop criterion is reached. The distribution of classes in a leaf node L is then obtained with respect to the samples $I \in \mathcal{I}^{tr}$ reaching this leaf node. The leaf node assigns pedestrian or non-pedestrian class probabilities $p(c|L)$, for $c \in \{0, 1\}$, and probabilities $p(c = 0, d|L_t)$, for $d \in \{N, E, S, W\}$, for body directions.

Each tree grows randomly and independently of the others. Randomness is important when training a tree. This ensures a variety in the forest, or, in other words, trees are less correlated this way with each-other. For a forest, it would be meaningless to assemble "similar" trees. Details about introducing randomness into the training procedure are given in Section 3.5.

Let $\mathcal{I}^{ts} \subset \mathcal{I}$ be a set of input bounding boxes used for testing. Any $I^{ts} \in \mathcal{I}^{ts}$ is passed through the N trees T_t of a trained forest. In each tree T_t, I^{ts} ends up in a leaf L_t. Thus, N distributions are assigned to one test box. By

$$c^* = \arg\max_c \sum_{t=1}^{N} p(c|L_t) \tag{3}$$

$$d^* = \arg\max_d \sum_{t=1}^{N} p(c = 0, d|L_t) \tag{4}$$

we define that a maximum-likelihood decision is adopted to classify a test box and, if $c^* = 0$, their direction.

3.2. Features

Because HoG features lead to a strong performance in solving both pedestrian and body direction classification tasks [7, 2], similar features are adopted here. Bounding boxes used for training or in testing are of identical size (in our implementation of size 64×192). Three levels of cell sizes, namely 8×8, 16×16, and 32×32, are adopted to generate the feature vector, with a block size of 2×2 cells and a block stride of one cell for each direction. An HoG is calculated for each cell using 9 bins, which leads to a 7,165-dimensional feature vector.

But instead of using the vectors as a whole to train the classifier, only a few components (possibly just one) of the feature vectors are used in one split node to separate training samples into two kinds of subsequent nodes. A feature vector is resized into an HoG matrix B of size 796×9, where each row corresponds to a normalized cell, and each column to a bin.

3.3. Split Functions

As discussed above, split functions are simplified questions aiming at solving the classification problem step by step. They are relatively simple but essential.

Theoretically, $h_\phi(I)$ could be any function that produces a binary value in $\{0, 1\}$. In this case, one option is to compare two feature values, defined by

$$h_\phi(I) = \begin{cases} 0 & \text{if} \quad B(p,i) - B(q,j) > \tau \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

where the parameters $\phi = \{p, q, i, j, \tau\}$ denote two block numbers, the bin numbers, and the threshold.

More parameters increase the chance of over-fitting [6]. A second option is given by using one feature-value only, defined by

$$h_\phi(I) = \begin{cases} 0 & \text{if} \quad B(p,i) > \tau \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

In order to split the training samples properly, parameters $\phi = \{p, q, i, j, \tau\}$ or $\phi = \{p, i, \tau\}$ for each internal node are learned with respect to maximizing a predefined target function. Having different target functions, split nodes are generated differently. Thus it is essential to define an appropriate objective function for obtaining "good" split functions.

3.4. Goodness of Splitting

In order to handle the pedestrian and body direction recognition simultaneously, the target function $o_{cd}(\phi, \mathcal{I})$ is defined by two terms, a pedestrian classification term $o_c(\phi, \mathcal{I})$ and a direction classification term $o_d(\phi, \mathcal{I})$. An option is to identify $o_{cd}(\phi, \mathcal{I})$ randomly either with $o_c(\phi, \mathcal{I})$ or with $o_d(\phi, \mathcal{I})$. Another option is a weighted sum

$$\begin{aligned} o_{cd}(\phi, \mathcal{I}) &= o_c(\phi, \mathcal{I}) + w(\mathcal{I}) \cdot o_d(\phi, \mathcal{I}) && (7) \\ w(\mathcal{I}) &= \gamma \cdot \max\{p(c = 0|\mathcal{I}) - \eta, 0\} && (8) \end{aligned}$$

The weight $w(\mathcal{I})$ changes with respect to $p(c_0|\mathcal{I})$, the probability of a pedestrian in the current node. Direction classification is conducted only if the probability of being a pedestrian is larger than a threshold η. The higher the probability of a pedestrian is, the more we consider $o_d(\phi, \mathcal{I})$. At the beginning, pedestrian classification has the priority. A simplified demonstration is shown in Fig. 4; here we assume that γ and η are constant variables.

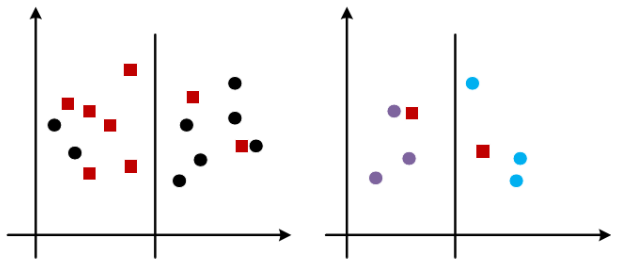

Figure 4. Auto adjusting the split focus from a parent split node (left) to one of the two child nodes (right). *Left:* Red rectangles identify non-pedestrian samples, and black circles are for pedestrians (direction-labelled or unlabelled). As $p(c = 0|\mathcal{I}) < \eta$, splitting focuses on separating samples of pedestrians from those of non-pedestrians. *Right:* Darker (cyan) and lighter (purple) circles denote pedestrians with different directions, say, *N* and *S*. As $p(c = 0|\mathcal{I}) > \eta$, splitting focuses now on classifying directions.

In order to measure the goodness of splitting, for pedestrians and non-pedestrians with $o_c(\phi, \mathcal{I})$, and direction classes with $o_d(\phi, \mathcal{I})$, Shannon entropy-based measures $o_c(\phi, \mathcal{I})$ and $o_d(\phi, \mathcal{I})$ are defined as follows:

$$\begin{aligned} o_c(\phi, \mathcal{I}) &= E_c(\mathcal{I}) - \sum_{k \in \{L,R\}} \omega_k E_c(\mathcal{I}_k(\phi)) && (9) \\ o_d(\phi, \mathcal{I}) &= E_d(\mathcal{I}) - \sum_{k \in \{L,R\}} \omega_k E_d(\mathcal{I}_k(\phi)) && (10) \\ E_c(\mathcal{I}) &= -\sum_c p(c|\mathcal{I}) \log(p(c|\mathcal{I})) && (11) \\ E_d(\mathcal{I}) &= \\ & -\sum_d p(c = 0, d|\mathcal{I}) \log(p(c = 0, d|\mathcal{I})) && (12) \\ \omega_k &= |\mathcal{I}_k(\phi)| / |\mathcal{I}(\phi)| && (13) \end{aligned}$$

where $E_c(\mathcal{I})$ and $E_d(\mathcal{I})$ denote entropies of pedestrian and direction classes, respectively, and ω is the weight for balancing the bias caused by the different numbers of samples going to the left or right child node.

3.5. Implementation

Training: as the dataset adopted in [11] is not publicly available, we combine several datasets (TUD multiview pedestrian, the Daimler Mono Pedestrian *Classification* Benchmark, and Daimler Mono Pedestrian *Detection* Benchmark data-sets) to train and test the proposed method, named the *Pedview* dataset. For training, 15,000 nonpedestrian samples from the *Classification* Benchmark dataset, 6,000 pedestrian samples from the *Detection* Benchmark dataset, and 4,935 direction-labelled pedestrian samples from the TUD multi-view pedestrian dataset are employed. Of those, 8,000 samples (3,000 from the TUD dataset) are randomly chosen for training a single tree.

Learn split node: Starting with the first split node (the root node), samples are split to the left or right child node according to the value of the split function. As the split function is supposed to split different classes, the suitable parameters ϕ are learned from the samples. For ensuring a variety of trees and time efficiency, parameters in ϕ_s, for $s = 1, 2, \ldots, 1000$, except τ, are selected randomly. For parameters p_s and i_s, a range $[\tau_{min}, \tau_{max}]$ of values $B(p_s, i_s)$ is calculated. Then, out of ten randomly selected values $\tau \in [\tau_{min}, \tau_{max}]$ we choose that τ_s which maximizes Equ. (7) for $\{p_s, i_s\}$. The intention is to have those parameters $\phi = \{p^*, i^*, \tau^*\}$ for the current node which maximize Equ. (7) for all the parameters $\phi_s = \{p_s, i_s, \tau_s\}$, for $s = 1, \ldots, 1000$.

Stop growing: If a stop criteria is reached, the node stops splitting, otherwise, split functions are again learned for both child nodes using the samples reaching this node. The depth in the tree and the number of samples reaching a node are considered as stop criterion, as a *deep tree* (i.e. a tree of large depth) leads to an over-fitting problem and, if only a few samples reach a leaf node, then there might also be a bias in the distribution. Thus, if a defined maximum depth is reached, or the number of samples (so far) at a node is below a threshold then we stop with splitting. Specific settings for each forest are given in Section 4.1.

Compensate sample bias: In order to reduce a bias defined by the different cardinalities of samples across classes during training, we compensate by using the following factors r_c and r_d at a given leaf node L:

$$p(c|L) = |\mathcal{I}_c^L| \cdot r_c / \sum_c (|\mathcal{I}_c^L| \cdot r_c) \tag{14}$$

$$r_c = |\mathcal{I}^{tr}| / |\mathcal{I}_c^{tr}| \tag{15}$$

$$p(d|L) = |\mathcal{I}_d^L| \cdot r_d / \sum_d (|\mathcal{I}_d^L| \cdot r_d) \tag{16}$$

$$r_d = |\mathcal{I}^{ld}| / |\mathcal{I}_c^{ld}| \tag{17}$$

$$p(c = 0, d|L) = p(c = 0|L) \cdot p(d|L) \tag{18}$$

where $|\mathcal{I}^{tr}| = 8,000$ for each tree; set $\mathcal{I}^{ld} \subset \mathcal{I}^{tr}$ contains all the direction-labelled samples in $\mathcal{I}^{tr}$, and set $\mathcal{I}^L \subset \mathcal{I}^{tr}$ contains all the samples arriving at leaf node L. Subscripts c or d select training samples in class c, or with direction d.

The larger the number N of trees, the better the performance of the classifier. We choose $N = 120$ for all experiments.

Testing: 10,000 nonpedestrian bounding boxes from the *Classification* Benchmark dataset, 3,000 pedestrian bounding boxes from the *Detection* Benchmark dataset, and 248 direction-labelled pedestrian bounding boxes from the TUD multi-view pedestrian dataset are adopted.

Each test input is passed through each tree T_t in the forest. First, the feature matrix B is calculated. For any current node, the value of the split function is computed, and then the input is passed on to the corresponding (left or right) child node until it reaches a leaf node L_t. Equation (4) specifies the final output.

Algorithms 1 and 2 specify our training and RDF testing procedures, respectively.

Algorithm 1 (Training)
Input: Randomly selected set $\mathcal{I}$ of 7,500 samples with corresponding HoG matrixes B.
Output: Trained trees T_t, for $t = 1, 2, \ldots, N$

```
1: Let $T_t = \emptyset$, $num = |\mathcal{I}|$, $dep = 0$, stop criterion $t_{num} = 20$, $t_{dep} = 15$, temporal data store variables $temp_{ocd1} = 0$, $temp_{ocd2} = 0$.
2: if $num < t_{num} || dep > t_{dep}$ then
3:    calculate $p(c|L), p(c = 0, d|L)$ with $\mathcal{I}$, according to Eqs. (14) and (18);
4:    add leaf $L$ to the tree: $T_t = T_t \cup L$
5:    return $T_t$.
6: else
7:    $dep = dep + 1$;
8:    for $s = 1, \ldots, 1000$ do
9:       randomly select a parameter set $\{p_s, i_s\}$;
10:      find range $[\tau_{min}, \tau_{max}]$ of $p_s$;
11:      for $h = 1, \ldots, 10$ do
12:         randomly select $\tau_h \in [\tau_{min}, \tau_{max}]$;
13:         split $\mathcal{I}$ into $\mathcal{I}_{Lh}, \mathcal{I}_{Rh}$ according to Equ. (6);
14:         calculate $o_{cd}(\{p_s, i_s, \tau_h\}, \mathcal{I})$ with Equ. (7);
15:         if $o_{cd}(\{p_s, i_s, \tau_h\}, \mathcal{I}) > temp_{ocd2}$ then
16:            $temp_{ocd2} = o_{cd}(\phi_s, \mathcal{I})$;
17:            $\tau_s = \tau_h$, $\phi_s = \{p_s, i_s, \tau_s\}$;
18:         end if
19:      end for
20:      if $temp_{ocd2} > temp_{ocd1}$ then
21:         $\phi^* = \phi_s$;
22:      end if
23:   end for
24:   expand tree by new split node: $T_t = T_t \cup \phi^*$;
25:   split $\mathcal{I}$ into $\mathcal{I}_L$ and $\mathcal{I}_R$;
26:   $num = |\mathcal{I}_L|$, $\mathcal{I} = \mathcal{I}_L$, and go to line 2;
27:   $num = |\mathcal{I}_R|$, $\mathcal{I} = \mathcal{I}_R$, and go to line 2;
28: end if
```

4. Experiments

In this section, we aim (1) to prove that the proposed task-combining forests perform comparable to single-task

Algorithm 2 (Testing)
Input: Test bounding box I, corresponding HoG matrix B, trained trees T_t, with $t = 1, 2, \dots, N$.
Output: Class label c^* and, if $c^* = 0$, also d^*.

```
1: for $t = 1, \dots, N$ do
2:     pass $I$ through $T_t$ until reaching a leaf node $L_t$, obtain distribution $p(c|L_t), p(c = 0, d|L_t)$;
3: end for
4: obtain $c^*$ and $d^*$ with Equ. (4);
5: return $c^*, d^*$.
```

aimed methods, and (2) compare different settings to structure a task-combining forest. The experimental design is described in detail in Section 4.1, and the experimental results are illustrated in Section 4.2.

4.1. Experimental Design

Nine experiments reported in this section have been designed as follows:

(1) PedD1 - RDF trained with pedestrian and non-pedestrian samples using split function Equ. (6), maximum depth to be 15, minimum number of samples to be 20,

(2) PedD2 - RDF trained with pedestrian and non-pedestrian samples using split function Equ. (5), maximum depth to be 15, minimum number of samples to be 20,

(3) FourD1 - RDF trained with four-direction-labelled pedestrian samples using split function Equ. (6), maximum depth to be 10, minimum number of samples to be 10,

(4) FourD2 - RDF trained with four-direction-labelled pedestrian samples using split function Equ. (5), maximum depth to be 10, minimum number of samples to be 10,

(5) FourPedWD1 - RDF trained with four-direction-labelled samples, pedestrian samples and non-pedestrian samples using split function Equ. (6) and weighted-combining target function Equ. (7), maximum depth to be 15, minimum number of samples to be 20,

(6) FourPedWD2 - RDF trained with four-direction-labelled samples, pedestrian samples and non-pedestrian samples using split function Equ. (5) and weighted-combining target function Equ. (7), maximum depth to be 15, minimum number of samples to be 20,

(7) FourPedRD1 - RDF trained with four-direction-labelled samples, pedestrian samples and non-pedestrian samples using split function Equ. (6), and target function randomly selected to be o_c or o_d, maximum depth to be 15, minimum number of samples to be 20,

(8) FourPedRD2 - RDF trained with four-direction-labelled samples, pedestrian samples and non-pedestrian samples using split function Equ. (5), and target function randomly selected to be o_c or o_d, maximum depth to be 15, minimum number of samples to be 20,

(9) PedSVM - Linear SVM trained with pedestrian and non-pedestrian samples,

Table 2. Confusion matrix FourD1

	N	*E*	*S*	*W*
N	0.91	0.00	0.09	0.00
E	0.00	0.73	0.18	0.09
S	0.45	0.00	0.53	0.03
W	0.12	0.18	0.12	0.56

Table 3. Confusion matrix FourD2

	N	*E*	*S*	*W*
N	0.90	0.00	0.82	0.02
E	0.10	0.67	0.05	0.19
S	0.48	0.10	0.36	0.07
W	0.06	0.22	0.00	0.72

The four directions are *N*, *E*, *S*, and *W*. The forests trained with four direction labelled samples are tested with eight directions, *N*, *NE*, *E*, *SE*, *S*, *SW*, *W*, and *NW*.

4.2. Experimental Results

Direction classification performance. See Table 1 for the direction classification accuracy obtained by six forests (FourD1, FourD2, FourPedWD1, FourPedWD2, FourPedRD1, and FourPedRD2), and three state-of-the-art direction classification methods ([1]-Max, [2]-AWG, and [4]). The average performance of four and eight direction cases are illustrated in column *overall(Four)* and *overall(Eight)* respectively. The classifications of in-between directions, *NE*, *SE*, *SW*, and *NW*, are taken as correct if they are classified to be their adjacent directions. When test FourPedWD1, FourPedWD2, FourPedRD1, and FourPedRD2, classification results are obtained with a detection rate of 0.95. In order to obtain more robust results, the direction classification results are filtered using a constant threshold.

As shown in Table 1, the integrated forests yield comparable results to direction forests and three state-of-the-art methods, which proves that the proposed integration RDF structure is efficient for combining direction classification with pedestrian classification.

One value-based split function performs better than two value-based split functions in direction forests and weight-integrated forests. But the performance of the two value-based function is significantly improved by the random integrated structure. Both random integrated forests outperform the weight integrated forests. Thus, based on this *Pedview* dataset, we conclude that the introduction of appropriate randomness leads to better performance, especially when more parameters are employed in split function.

The confusion matrixes are illustrated in Table 2 to 7. As shown in Table 1, the performance of *S* direction classification is relatively low over the forests. The confusion matrices show that *S* is mostly classified to be *N*, The *W* direction is mostly confused with the *E* direction.

Pedestrian classification. Precision-recall curves of

Table 1. Direction classification performance

	N	*NE*	*E*	*SE*	*S*	*SW*	*W*	*NW*	overall(Four)	overall(Eight)
FourD1	0.91	0.95	0.73	0.48	0.53	0.56	0.59	0.77	0.69	0.69
FourD2	0.90	0.91	0.67	0.41	0.36	0.40	0.72	0.61	0.66	0.62
FourPedWD1	0.92	0.92	0.79	0.48	0.23	0.49	0.63	0.84	0.64	0.66
FourPedWD2	0.90	0.86	0.64	0.55	0.25	0.40	0.65	0.78	0.61	0.63
FourPedRD1	0.95	0.95	0.78	0.41	0.32	0.47	0.60	0.86	0.66	0.67
FourPedRD2	0.91	0.83	0.85	0.52	0.37	0.44	0.69	0.72	0.71	0.67
[2]-AWG	0.76	0.57	0.95	0.36	0.64	0.55	0.86	0.52	0.80	0.65
[4]-Tracking	0.71	0.37	0.65	0.36	0.41	0.59	0.70	0.53	0.62	0.54
[1]-Max	0.46	0.35	0.54	0.08	0.4	0.08	0.38	0.23	0.45	0.32

Table 4. Confusion matrix FourPedWD1

	N	*E*	*S*	*W*
N	0.92	0.02	0.07	0.00
E	0.16	0.79	0.05	0.00
S	0.68	0.04	0.23	0.04
W	0.06	0.31	0.00	0.62

Table 5. Confusion matrix FourPedWD2

	N	*E*	*S*	*W*
N	0.90	0.03	0.07	0.00
E	0.14	0.64	0.09	0.14
S	0.65	0.05	0.26	0.05
W	0.00	0.35	0.00	0.64

Table 6. Confusion matrix FourPedRD1

	N	*E*	*S*	*W*
N	0.95	0.00	0.05	0.00
E	0.06	0.78	0.11	0.06
S	0.61	0.05	0.32	0.02
W	0.00	0.33	0.07	0.60

Table 7. Confusion matrix FourPedRD2

	N	*E*	*S*	*W*
N	0.91	0.02	0.05	0.02
E	0.10	0.85	0.00	0.05
S	0.51	0.07	0.37	0.05
W	0.06	0.25	0.00	0.69

pedestrian and integrated forests are illustrated in Figure 5. With an increase of detection rate (recall), the precision decreases accordingly. When the detection rate is over 0.9, the precision drops dramatically. Thus, the forests are compared with a base-line method PedSVM in Table 8, with a detection rate of 0.9.

The integrated forests perform similar to pedestrian forests and base-line method, which proves that the integrated RDF structure is efficient for combining pedestrian classification with direction classification, without degrading in performance.

Processing time. The processing time for one bounding box when passed through a tree ($depth < 15$) is less than 5 ms on a PC defined by Core i7 1.9 Hz, and 3.84GB. As the processing of trees are independent calculations, *i.e.* the bounding box goes through each tree without interfering with the other trees, parallel computing is possible.

Table 8. Pedestrian classification performance

	Precision	False Positive
PedSVM	0.988	0.004
PedD1	0.985	0.005
PedD2	0.991	0.003
FourPedWD1	0.987	0.004
FourPedWD2	0.984	0.005
FourPedRD1	0.988	0.004
FourPedRD2	0.986	0.004

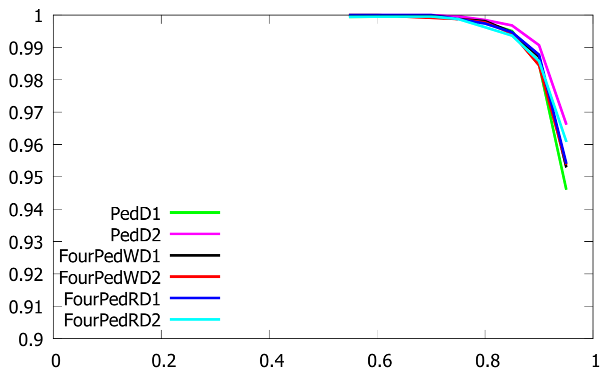

Figure 5. Precision-recall graph for pedestrian classification.

5. Conclusions

This paper proposes to integrate pedestrian detection and direction classification into a single RDF classifier. The paper discussed how to specify and implement such an RDF, and illustrated its efficiency by reporting about experiments using three data sets available online. The proposed integrated classifier performs comparable to separately trained RDFs for pedestrian detection or direction classification tasks, and to state-of-the-art methods. There is still a need to further improve the performance for decisions between *S* and *N*, *W* and *E* directions. Expanding available sets of direction-labelled samples on the net would be beneficial for more extensive testing.

References

[1] M. Andriluka, S. Roth, and B. Schiele. Monocular 3d pose estimation and tracking by detection. *CVPR*, 623–630, 2010. 1, 2, 6, 7

[2] D. Baltieri, R. Vezzani, and R. Cucchiara. People orientation recognition by mixtures of wrapped distributions on random trees. *ECCV*, 270–283, 2012. 2, 4, 6, 7

[3] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE PAMI*, 33:1820–1833, 2011. 1

[4] C. Chen, A. Heili, and J. Odobez. Combined estimation of location and body pose in surveillance video. *AVSS*, 5–10, 2011. 1, 2, 6, 7

[5] C. Chen, A. Heili, and J.-M. Odobez. A joint estimation of head and body orientation cues in surveillance video. *ICCV Workshops*, 860–867, 2011. 2

[6] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations Trends Computer Graphics Vision*, 7:81–227, 2011. 4

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, vol. 1, 886–893, 2005. 1, 2, 4

[8] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. *ECCV*, 428–441, 2006. 1, 2

[9] P. Dollár, C. Wojek, B. Schiele, and P. Perona, Pedestrian detection: A benchmark. *CVPR*, 304–311, 2009. 2

[10] P. Dollár, C. Wojek, B. Schiele, and P. Perona, Pedestrian detection: An evaluation of the state of the art. *IEEE PAMI*, 34:743–761, 2012. 2

[11] M. Enzweiler and D.M. Gavrila. Integrated pedestrian classification and orientation estimation. *CVPR*, 982–989, 2010. 1, 3, 4

[12] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61:55–79, 2005. 2

[13] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. *CVPR*, 1022–1029, 2009. 1, 2, 3

[14] J. Gall, N. Razavi, and L. Van Gool. On-line adaption of class-specific codebooks for instance tracking. *BMVC*, 2010. 2, 3

[15] J. Gall, N. Razavi, and L. Van Gool. An introduction to random forests for multi-class object detection. *Outdoor Large-Scale Real-World Scene Analysis*, 243–263, 2012. 2, 3

[16] T. Gandhi and M. M. Trivedi. Image based estimation of pedestrian orientation for improving path prediction. *IEEE IV*, 506–511, 2008. 1

[17] P. Geismann and G. Schneider. A two-staged approach to vision-based pedestrian recognition using Haar and hog features. *IEEE IV*, 554–559, 2008. 2

[18] K. Goto, K. Kidono, Y. Kimura, and T. Naito. Pedestrian detection and direction estimation by cascade detector with multi-classifiers utilizing feature interaction descriptor. *IEEE IV*, 224–229, 2011. 1, 2, 3

[19] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 77:259–289, 2008. 2

[20] B. Leibe, K. Schindler, and L. Van Gool. Coupled detection and trajectory estimation for multi-object tracking. *ICCV*, 1–8, 2007. 1

[21] D. Mitzel, E. Horbert, A. Ess, and B. Leibe. Multi-person tracking with sparse detection and continuous segmentation. *ECCV*, 397–410, 2010. 1

[22] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE PAMI*, 32:448–461, 2010. 3

[23] S. Piérard and M. Van Droogenbroeck. Estimation of human orientation based on silhouettes and machine learning principles. *ICPRAM*, 2012. 1, 2

[24] L. Oliveira, U. Nunes and P. Peixoto. On exploration of classifier ensemble synergism in pedestrian detection. *IEEE ITS*, 11:16–27, 2010. 2

[25] H. Shimizu and T. Poggio, Direction estimation of pedestrian from multiple still images. *IEEE IV*, 596–600, 2004. 2

[26] J. Tao and R. Klette, Tracking of 2d or 3d irregular movement by a family of unscented Kalman filters. *JICCE*, 10:307–314, 2012. 1

[27] X. Wang, T. X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. *ICCV*, 32–39, 2009. 2

[28] C. Wojek, S. Walk, and B. Schiele. Multi-cue onboard pedestrian detection. *CVPR*, 794–801, 2009. 1, 2

[29] J. Xing, H. Ai, and S. Lao. Multi-object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses. *CVPR*, 1200–1207, 2009. 1

[30] B. Yang and R. Nevatia. Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. *CVPR*, 1918–1925, 2012. 1

[31] A. Yao, J. Gall, and L. Van Gool. A Hough transform-based voting framework for action recognition. *CVPR*, 2061–2068, 2010. 3

[32] G. Zhao, M. Takafumi, K. Shoji, and M. Kenji. Video based estimation of pedestrian walking direction for pedestrian protection system. *J. Electronics (China)*, 29:72–81, 2012. 1, 2