Learning an image-based motion context for multiple people tracking

Laura Leal-Taixé[1,2], Michele Fenzi[2], Alina Kuznetsova[2], Bodo Rosenhahn[2], and Silvio Savarese[3]

[1]Photogrammetry and Remote Sensing, ETH Zürich
[2]Institute for Information Processing, Leibniz University Hannover
[3]Computational Vision and Geometry Lab, Stanford University

Abstract

We present a novel method for multiple people tracking that leverages a generalized model for capturing interactions among individuals. At the core of our model lies a learned dictionary of interaction feature strings which capture relationships between the motions of targets. These feature strings, created from low-level image features, lead to a much richer representation of the physical interactions between targets compared to hand-specified social force models that previous works have introduced for tracking. One disadvantage of using social forces is that all pedestrians must be detected in order for the forces to be applied, while our method is able to encode the effect of undetected targets, making the tracker more robust to partial occlusions. The interaction feature strings are used in a Random Forest framework to track targets according to the features surrounding them. Results on six publicly available sequences show that our method outperforms state-of-the-art approaches in multiple people tracking.

1. Introduction

Many computer vision tasks are related to the problem of understanding the semantic content of a scene from a video sequence. Humans are often the center of attention of a scene, therefore, the ability to detect and track multiple people from a video has emerged as one of the top tasks to address in our field. A common approach to multiple people tracking follows the idea of estimating the hypotheses of the locations of people using a detector for each frame. Those hypotheses are then associated in time, so as to form consistent tracks for each individual. One of the problems in tracking-by-detection methods is that they are highly dependent on detection results. Methods that use a physical model to estimate pedestrians' motion [19, 25] are completely unaware of the effect of undetected pedestrians, which reduces its effectiveness in semi-crowded environments, where it is very common to observe occlusions and it is very hard to estimate a pedestrian's trajectory.

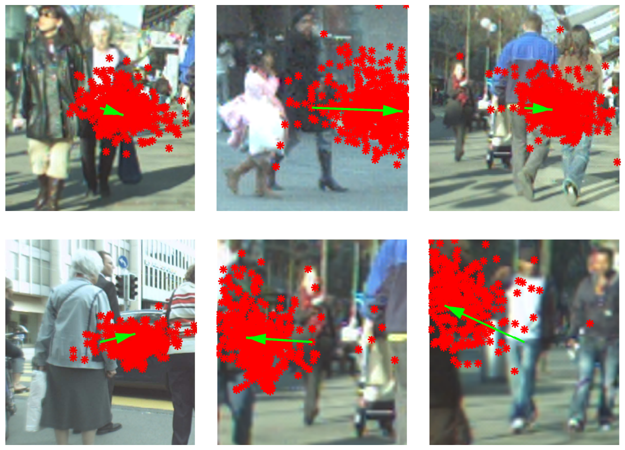


Figure 1: We estimate the velocity of the pedestrian in image coordinates by learning a mapping from image features to pedestrian velocity using a Hough Random Forest (RF) framework. The green arrow indicates the ground truth velocity of the pedestrian, and in red we plot the votes made by the corresponding leafs of the learned RF.

In this paper, we propose to construct a model that estimates how a pedestrian moves according to the motion and appearance features around him/her. An advantage of our approach is that we relax the dependency of tracking on detections, since now we can compute the motion of a pedestrian taking into account his/hers environment, even if other pedestrians are not explicitly detected. The proposed approach is based solely on image features and efficient classification techniques, which means it can be potentially implemented in real-time and therefore used for tasks such as pedestrian intention detection in autonomous car navigation.

We acknowledge the support of an ONR grant 00014-13-1-0761 and the DARPA UPSIDE grant A13-0895-S002.

1.1. Related work

Multiple people tracking is a key problem for many computer vision tasks, such as surveillance, animation or activity recognition. Tracking is commonly divided in two steps: object detection and data association. First, we detect objects in each frame of the sequence and second, the detections are matched to form complete trajectories. In order to deal with crowded environments, where occlusions and false detections are common, researchers have focused on creating reliable detectors that can work even if parts of the target are occluded [12, 11, 30, 29] as well as on obtaining a more robust data association. The data association problem is usually solved on a frame-by-frame basis [17] or one track at a time [2], but recent works show that it is more reliable to jointly solve the matching problem for all tracks and all frames, either in discrete space using Linear Programming (LP) [14, 34, 26, 3] or in continuous space [1].

Most tracking systems work with the assumption that the motion model for each target is independent, but in reality, a pedestrian follows a series of social rules, *i.e.* is subject to social forces according to other moving targets around him/her. These have been defined in what is called the social force model (SFM) [13, 15] which has been used for abnormal crowd behavior detection [22], crowd simulation [24] and has only recently been applied to multiple people tracking [27, 25, 31, 19, 6, 1] and sports analysis [18]. The problem with these methods is that they are limited to a few hand-designed force terms, such as *collision avoidance* or *group attraction*. Furthermore, its inclusion in an LP framework either requires an iterative approach as in [19] or complex slower solvers as in [20, 5, 7]. At the same time, these models depend heavily on detections, since the forces will only be computed among detected pedestrians. In the case of crowded scenes, partially occluded pedestrians will never have an effect on the trajectory of detected pedestrians.

In this paper, we aim at performing multiple people tracking from uncalibrated monocular images. For this goal, we introduce the *interaction feature strings*, which encode information about a pedestrian's velocity depending on his/her environment. These feature strings are created using only image features (and therefore requiring no 3D information of the scene), and then used in a Random Forest (RF) framework, which we train to estimate the velocity of a pedestrian at a certain frame. Random Forests [4] have gained popularity these last years for applications like pose estimation [28] or object detection [12], but, to the best of our knowledge, have not been used so far in tracking for predicting an object's future position. A clear advantage of our method is that it relaxes the dependency on detections, since the effect of a partially occluded (and potentially not detected) pedestrian can still be encoded in the interaction feature string, while it will be ignored by common tracking-by-detection methods [19, 16, 25].

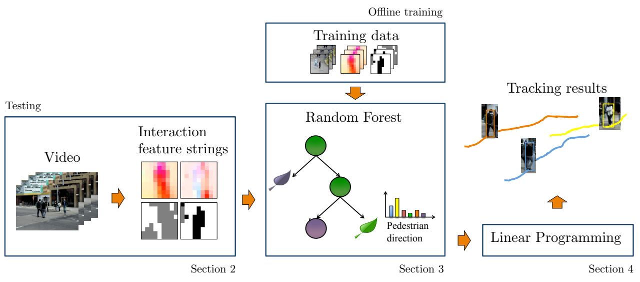


Figure 2: Diagram of the proposed approach. From a new video, we compute interaction feature strings (Section 2) and use them to train a Random Forest (Section 3), which will output the estimated pedestrian velocity. This information is used directly for tracking in a Linear Programming framework (Section 4).

1.2. Contributions

The contribution of this paper is threefold:

- We propose a way to effectively estimate a pedestrian's velocity in image coordinates which works even under camera motion.
- We introduce the *interaction feature strings*, used to encode pedestrian interactions from low-level image features. These lead to a much richer representation of the physical interactions between targets when compared to the previously used hand-designed, physics-based terms of the Social Force Model.
- We are able to encode the effect of undetected pedestrians, therefore relaxing the dependency of most tracking methods on detections.

The paper is organized as follows: in Section 2 we present how to compute interaction feature strings using low-level features. Section 3 introduces the Random Forest framework used to estimate pedestrians' velocities, while in Section 4 we give a brief description of our tracking framework. The last two sections are devoted to experimental results and conclusions.

2. Interaction feature string

Our method is based on what we call *interaction feature strings*, which encode image features that represent a particular scene configuration. A scene $\mathcal{I}(\mathbf{p}_i^t)$ is defined as a patch centered around a detected pedestrian i at time t and position $(x, y) \in \mathbb{R}^2$ in pixel coordinates. It has a size of $[h_i s_h, h_i s_w]$, where h_i represents the pedestrian's height, s_h and s_w are scaling factors. The patches are scaled according to the pedestrian's height to obtain a scale-invariant representation that allows us to deal with scenes both closer and further away from the camera. We set the scaling factors to $s_h = 1.1$ and $s_w = 1$ for all experiments. An example of a scene in shown in Figures 3(a), 3(f), 3(k), 3(p).

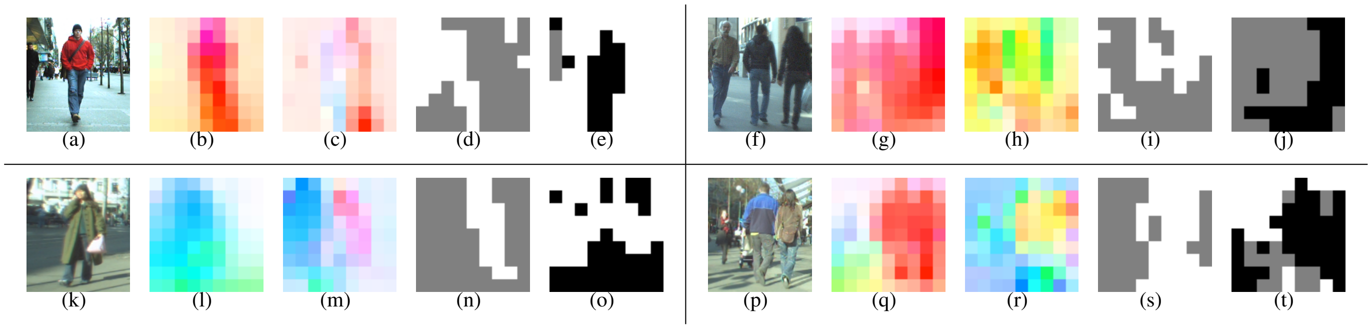


Figure 3: Examples of features computed for four scenes, with $N_B = 9 \times 9$ and $N_{FR} = 7$. (a,f,k,p) Original image scene centered around a pedestrian. (b,g,l,q) Mean Optical Flow (MOF). (c,h,m,r) Difference of Optical Flows (DOF). (d,i,n,s) Ternary Optical Flow (TOF). (e,j,o,t) Ternary Angular Optical Flow (TAOF). For TOF and TOAF, the value 1 is represented in white, -1 in grey and 0 in black.

We divide the scene in N_B blocks and compute a set of features per block. For different types of interactions between pedestrians, different blocks will contribute to describing the scene. For example, in a scene where a pedestrian walks alone, as in Figure 3(a), the central blocks will contain most of the relevant information. If there are more pedestrians involved in a scene, the outer blocks will become more and more relevant.

For each of the blocks, we compute several descriptive features $\mathbf{F}_b(\mathbf{p}_i^t) = \left(F_b^1(\mathbf{p}_i^t), F_b^2(\mathbf{p}_i^t) \ldots F_b^{N_f}(\mathbf{p}_i^t)\right) \in \mathbb{R}^{N_f}$, where N_f is the total number of feature channels per block and b is the block index. We concatenate the block features into one interaction feature string $\mathcal{F}(\mathbf{p}_i^t) = \{\mathbf{F}_b(\mathbf{p}_i^t)\}_{b=1}^{N_B}$. Note that, since we are interested in motion features, we use N_{FR} frames ahead of the scene we are analyzing. We analyze the effect of the choice of N_{FR} as well as N_B in the experimental section.

The features we use include:

- Mean Optical Flow (MOF): We take the Optical Flow of a scene $\mathcal{I}(\mathbf{p}_i^t)$ in several consecutive frames $\{t, \ldots, t+N_{FR}\}$, let us call them $\mathbf{OF}_t, \ldots, \mathbf{OF}_{t+N_{FR}}$. Note that all Optical Flows are computed on the scene centered around $\mathbf{p}_i^t$, *i.e.* pedestrian i at time t, regardless of the timestamp of the image. We then average the Optical Flows in time, divide the scene in N_B blocks, and take the mean of the Optical Flows of the pixels inside each block as feature channel. This yields $2 \times N_B$ feature channels per scene, one for each image axis and block.

- Difference of Optical Flows (DOF): We take the Optical Flows as before $\mathbf{OF}_t, \ldots, \mathbf{OF}_{t+N_{FR}}$ and make the difference between consecutive Optical Flows, *i.e.* $[\mathbf{OF}_t - \mathbf{OF}_{t-1}, \ldots, \mathbf{OF}_{t-N_{FR}+1} - \mathbf{OF}_{t-N_{FR}}]$. We then divide the scene in N_B blocks, and take the mean Difference of Optical Flows of the pixels inside each block. We have again $2 \times N_B$ feature channels per scene.

- Histogram of Optical Flows (HOF): We take the Optical Flows $\mathbf{OF}_t, \ldots, \mathbf{OF}_{t+N_{FR}}$ and average them. We then divide the scene in N_B blocks, and take the histogram of the angles of the Optical Flows of the pixels inside each block. We use 8 bins for the histograms, therefore we have $8 \times N_B$ feature channels per scene.

- Ternary Optical Flow (TOF): We compute $\mathbf{OF}_t$ and $\mathbf{OF}_{t+N_{FR}}$, and take the mean of the flows inside each of the N_B blocks. We then compare the norm of the two OF descriptors and create a new descriptor TOF. Each bin b of the new descriptor, which corresponds to a block in our scene, will have the following values:

$$\mathrm{TOF}(b) = \begin{cases} 0 & \text{if } \|\mathbf{OF}_t(b)\| = \|\mathbf{OF}_{t+N_{FR}}(b)\| \\ 1 & \text{if } \|\mathbf{OF}_t(b)\| > \|\mathbf{OF}_{t+N_{FR}}(b)\| \\ -1 & \text{if } \|\mathbf{OF}_t(b)\| < \|\mathbf{OF}_{t+N_{FR}}(b)\| \end{cases}$$

  We have N_B feature channels per scene. The idea of TOF is to capture the general trend of the motion of the scene, in a similar way as is done in [33] for action recognition. The TOF is useful to disambiguate between pedestrians walking in different directions, see Figures 3(a)-3(e) and 3(k)-3(o).

- Ternary Angular Optical Flow (TAOF): As before, we have $\mathbf{OF}_t$ and $\mathbf{OF}_{t+N_{FR}}$, and take the mean of the flows inside each of the N_B blocks. We then compare the angle of the two OF descriptors and create a new descriptor TAOF. Each bin b of the new descriptor will have the following values:

$$\mathrm{TAOF}(b) = \begin{cases} 1 & \text{if } \angle(\mathbf{OF}_t(b), \mathbf{OF}_{t+N_{FR}}(b)) \geq \frac{\pi}{4} \\ -1 & \text{if } \angle(\mathbf{OF}_t(b), \mathbf{OF}_{t+N_{FR}}(b)) \leq -\frac{\pi}{4} \\ 0 & \text{otherwise} \end{cases}$$

The TAOF is useful to disambiguate between different interactions, such as pedestrians walking together *vs.* pedestrians walking in opposite directions, see Figures 3(f)-3(j) and 3(p)-3(t).

An example of the features for four different scenes is shown in Figure 3. Note that we do not compensate for camera motion, and therefore it is included in our feature strings and will also be taken into account in the pedestrian velocity estimation. This is an important property of our feature strings, since we are tracking in image coordinates. Once we have a descriptive set of features for a scene, the goal is to train a Random Forest to be able to estimate the velocity of a pedestrian.

3. Hough Random Forests

Once we have the set of interaction feature strings obtained from training data, we need a framework that allows us to estimate pedestrians' velocities. We make use of a powerful discriminative learning technique, Random Forests. Given a scene, we compute the interaction feature string out of low-level features and then use it as input for the Random Forest to produce the most likely pedestrian velocity. This information is then used in a probabilistic tracking framework, such as Linear Programming, as we detail in Eqs. (5) and (8), to obtain the final set of trajectories.

A Random Forest consists of a set of random trees [4] that are trained to learn a mapping from a feature string $\mathcal{F}(\mathbf{p}_i^t)$ to a hypothesis space $\mathcal{H}$. In our case, hypothesis $\mathbf{h}$ represents the velocity of a pedestrian $\mathbf{v} = [v_x, v_y]$. Since $\mathbf{v}$ is a continuous variable, we make use of Hough Random Forests [12]. The mapping is learned as explained in Section 3.1 and is used to solve the data association problem in multiple object tracking as detailed in Section 4.

3.1. Training

For each scene $\mathcal{I}(\mathbf{p}_i^t)$ at a given pedestrian position $\mathbf{p}_i^t$, we compute an interaction feature string $\mathcal{F}(\mathbf{p}_i^t)$. We also assume we have the annotated ground truth velocity $\mathbf{v}_i^t$ of the pedestrian. Each tree T in the forest is therefore trained with the set $\mathcal{S} = \{\mathcal{F}(\mathbf{p}_i^t), \mathbf{v}_i^t\}$. Each leaf L of the tree stores the probability distribution of pedestrian velocities $p(\mathbf{v}|L)$.

Binary tests. Following the standard random forest framework [8], we start training at the root and choosing a binary test to separate the training set $\mathcal{S}$ into the two child nodes, left and right, such that $\mathcal{S}_L \cap \mathcal{S}_R = \emptyset$. At each child node we follow the same procedure until a termination criterion is met; in our case we define the maximum tree depth D_T. The final node, called a leaf node, should contain the training examples with similar features. We consider two types of binary tests or weak classifiers:

1. A decision stump classifier:

$$B_1 = \begin{cases} 0, & \text{if } \mathcal{F}_b^j(\mathbf{p}_i^t) > \tau_1 \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

2. A classifier that compares different features in different blocks:

$$B_2 = \begin{cases} 0, & \text{if } F_{b_1}^{j_1}(\mathbf{p}_i^t) > F_{b_2}^{j_2}(\mathbf{p}_i^t) + \tau_2 \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

For each type of weak classifier, we perform a set of tests, with randomly chosen parameters $\{b, j, b_1, b_2, j_1, j_2, \tau_1, \tau_2\}$. From this pool of binary tests we choose the best one according to the split criterion explained next.

Split criterion. A good classifier should split the parent set in such a way as to minimize the uncertainty of the child sets. We use a similar measure as [12], namely the offset uncertainty, defined as:

$$H(\mathcal{S}) = \sum_{\mathbf{v} \in \mathcal{S}} \|\mathbf{v} - \mathbf{v}_A\|^2 \tag{3}$$

where $\mathbf{v}_A$ is the mean of all velocities in set $\mathcal{S}$. At any given node, the set is then evaluated with our pool of binary tests and the one that minimizes the uncertainty is chosen:

$$\underset{s,b,j,b_1,b_2,j_1,j_2,\tau_1,\tau_2}{\arg\min} \left(H(\mathcal{S}_L | B_s = 0) + H(\mathcal{S}_R | B_s = 1) \right) \tag{4}$$

3.2. Testing

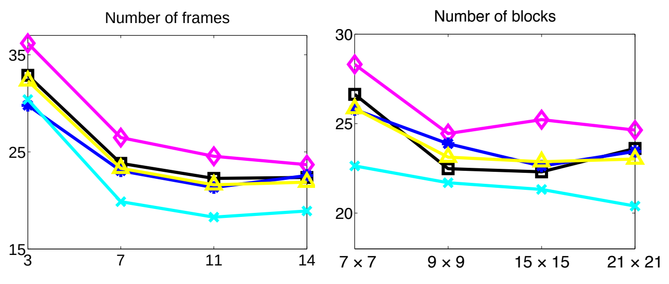
At testing time, we feed each tree T with the feature string of the scene and aim to estimate the pedestrian velocity. Based on the input $\mathcal{F}(\mathbf{p}_i^t)$, the test example reaches a certain leaf L. We use a voting scheme to estimate the final velocity of the pedestrian $\mathbf{v}_{\text{RF}}$. The votes are collected from all the N_T trees of the forest.

4. Tracking with Linear Programming

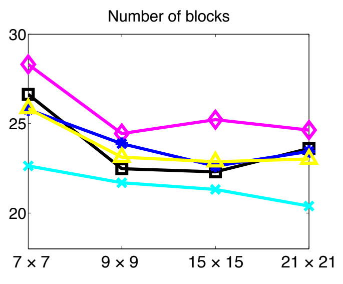
In this section, we present the tracking framework where we incorporate the velocity $\mathbf{v}_{\text{RF}}$ learned by the Random Forest in order to solve the data association problem and improve pedestrian tracking.

Let $\mathcal{O} = \{\mathbf{p}_i^t\}$ be a set of object detections with $\mathbf{p}_i^t = (x, y, t)$, where (x, y) is the 2D image position and t is the time stamp. A trajectory is defined as a list of ordered object detections $T_k = \{\mathbf{p}_{k_1}^{t_1}, \mathbf{p}_{k_2}^{t_2}, \cdots, \mathbf{p}_{k_N}^{t_N}\}$, and the goal of multiple object tracking is to find the set of trajectories $\mathcal{T}* = \{T_k\}$ that best explains the detections. This can be formulated as a minimization with the following objective function:

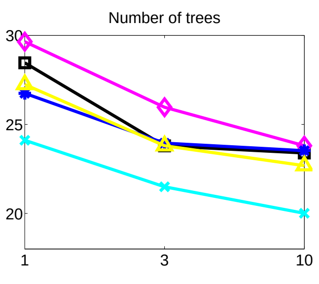
$$\mathcal{T}* = \underset{\mathcal{T}}{\mathbf{argmin}} \sum_i C_{\text{in}}(i) f_{\text{in}}(i) + \sum_i C_{\text{out}}(i) f_{\text{out}}(i) + \sum_i C_{\text{det}}(i) f(i) + \sum_{i,j} C_{\text{t}}(i,j) f(i,j) \tag{5}$$



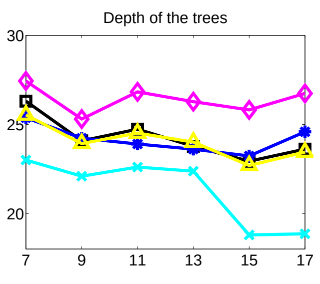
(a)



(b)



(c)



(d)

Figure 4: Pedestrian velocity estimation error (degrees) wrt. several parameters: (a) Number of frames N_{FR} used to compute the features. (b) Number of blocks N_B used to represent a scene. (c) Number of trees N_T of the Random Forest. (d) Depth D_T of those trees. Results plotted for each dataset: BAHNHOF (black squares), SUNNYDAY (pink diamonds), JELMOLI (cyan crosses), LINTHESCHER (blue stars). Mean results for all datasets (yellow triangles).

subject to edge capacity constraints, flow conservation at the nodes and exclusion constraints. This formulation corresponds to a Linear Program. We refer the interested reader to [19, 34] for more details on the formulation.

The costs C_{in} and C_{out} define how probable it is for a trajectory to start or end. The detection cost C_{det} is directly linked to the probability of the detection according to the detector used.

The cost of a link edge encodes the velocity estimation, and is expressed as:

$$C_{\text{t}}(i,j) = w_{\text{RF}} \cdot C_{\text{RF}}(i,j) + (1 - w_{\text{RF}}) \cdot C_{\text{d}}(i,j) \tag{6}$$

where $C_{\text{d}}(i,j)$ is a term based on the distance between detections $\mathbf{p}_i^t$ and $\mathbf{p}_j^{t+\Delta t}$:

$$C_{\text{d}}(i,j) = -\log\left(1 - \frac{\|(\mathbf{p}_j^{t+\Delta t} - \mathbf{p}_i^t\|}{V_{\max}\Delta t}\right) \tag{7}$$

where $V_{\max}$ is the maximum speed of a pedestrian in pixels. The term $C_{\text{RF}}(i,j)$ includes the information given by the Random Forest. It evaluates how close the detection $\mathbf{p}_j^{t+\Delta t}$ is to the prediction of the position of $\mathbf{p}_i^t$ given the estimated velocity $\mathbf{v}_{\text{RF}}$:

$$C_{\text{RF}}(i,j) = -\log\left(1 - \frac{\|(\mathbf{p}_j^{t+\Delta t} - (\mathbf{p}_i^t + \mathbf{v}_{\text{RF}}))\|}{V_{\max}\Delta t}\right) \tag{8}$$

The confidence weight w_{RF} determines how much we trust the Random Forest estimation. We fix this value at $w_{\text{RF}} = 0.9$. The Linear Program in Eq. (5) can be efficiently solved using Simplex [19] or k-shortest paths [3].

5. Experimental results

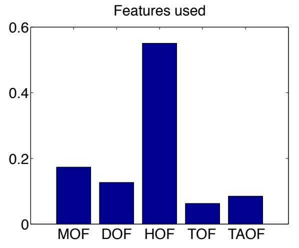
In order to evaluate the multiple people tracking performance of the proposed algorithm, we use four publicly available datasets: BAHNHOF, SUNNYDAY and JELMOLI from [10] and LINTHESCHER from [9]. These datasets are taken from a mobile camera moving around in crowded scenarios. Note that we do not use calibration or odometry data, since we track in image coordinates.

5.1. Pedestrian velocity estimation

This set of experiments aims at showing how well our approach estimates the velocity of a pedestrian. The setup of this experiment is the following: we use one sequence for testing and the other three for training. We measure the error in degrees between the estimated pedestrian velocity and the ground truth.

In the first experiment, we analyze how results are affected by the parameters of our method, namely: (i) the feature parameters described in Section 2, *i.e.* the number of blocks N_B and the number of frames we take to compute the features N_{FR}; (ii) the Random Forest parameters from Section 3, *i.e.* the number of trees N_T and the depth of the trees D_T. Results are presented in Figure 4, for each sequence individually and also the mean for all four sequences, which is shown in yellow triangles.

As we can see in Figure 4(a), taking 3 frames to compute the interaction feature strings is not enough to create a significant motion estimator. If we take 11-14 frames, which corresponds to 0.8-1 second, we obtain much more accurate velocity estimations. In Figure 4(b), we plot the results regarding the number of blocks N_B. Best results are obtained with 15×15 or 21×21 blocks. If we use less blocks, it is possible that the Optical Flow of different pedestrians is mixed into one block, creating features which are not descriptive enough for the level of detail we are interested



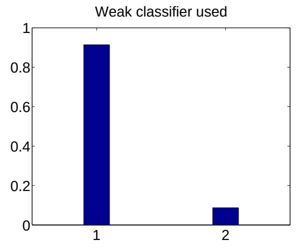


Figure 5: (a) Normalized histogram of the features used at the nodes of the Random Forest. (b) Normalized histogram of the weak classifiers used at the nodes (B_1 and B_2).

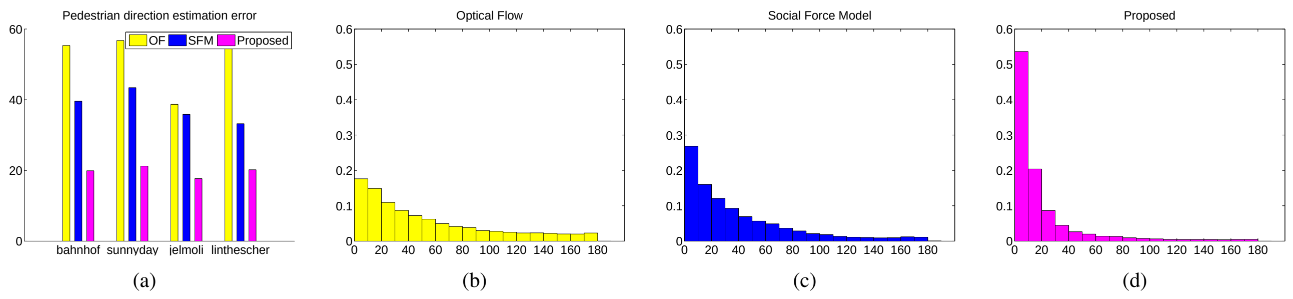


Figure 6: (a) Comparison of pedestrian velocity estimation error (degrees) obtained by Optical Flow prediction, Social Force Model prediction and the proposed approach. (b) Histogram of velocity estimation errors given by Optical Flow. (c) Histogram of velocity estimation errors given by the Social Force Model. (d) Histogram of velocity estimation errors given by the proposed approach. Mean results on the BAHNHOF, SUNNYDAY, JELMOLI and LINTHESCHER datasets.

in. If we look at the parameters of the Random Forest, we see in Figure 4(c) that in general using more trees yields a lower error, as expected. The depth of the trees, on the other hand, reaches an optimum at 15, as we can see in Figure 4(d). After that, the Random Forest suffers from overfitting, specially for datasets with less training data, such as BAHNHOF, SUNNYDAY or LINTHESCHER. For JELMOLI (shown as cyan stars), which has the largest amount of training data, results are greatly improved once we reach depth 15 or more. For tracking, we take the values of $N_{FR} = 11$, $N_B = 15$, $D_T = 15$ and $N_T = 10$.

We also plot in Figure 5(a) a normalized histogram of the features used by each node to split the data. The Histogram of Optical Flows (HOF) is the one that is most used by the Random Forest to split the data, since it contains a lot of detailed information about the scene. Note in Figure 5(b), that the decision stump weak classifier or binary test B_1 (see Section 3.1) is by far the most used, since the Optical Flow of the pedestrian we are tracking is the most important source of information to determine his/her velocity. The other classifier B_2 is used mainly to disambiguate between difficult situations, *e.g.* when two pedestrians cross *vs*. when they walk together.

The second experiment aims at comparing the performance of our method with two baselines: (i) the Social Force Model (SFM) [13] and (ii) the Optical Flow (OF). We compare how well each method can estimate the velocity of a pedestrian. Note that the SFM is computed in 3D real world coordinates and contains three terms, namely constant velocity assumption, collision avoidance and group attraction, as in [19, 31, 25]. The Optical Flow method takes the mean Optical Flow of the central block of our method as velocity of the pedestrian. These baselines will also be used later for tracking. In Figure 6(a), we show the error in degrees of each method. As we can see, the proposed method is able to estimate a pedestrian's velocity 20 degrees more accurately than SFM, and up to 40 degrees more accurately than OF. In Figures 6(b), 6(c) and 6(d) we plot the quantized relative frequency of the velocity estimation errors in degrees. As we can see, our method makes more than 50% of the estimations with less than 10 degrees of error, compared to 20% of OF and 30% of SFM.

5.2. Multiple people tracking: knowledge transfer

In this section, we apply the learned model for multiple people tracking. We compare the results with [32, 23] which are presented on the BAHNHOF and SUNNYDAY datasets. We use the same detections and the metrics described in [21], which measure: Recall (correctly matched detections / total detections in ground truth); Precision (correctly matched detections / total detections in the tracking result); number of Mostly Tracked trajectories (MT, $\geq 80\%$ of the track is correct); Mostly Lost (ML, $\leq$ 20%); Partially Tracked (PT, $>$ 20% and $<$ 80%); track fragmentations (Frg), number of times a ground truth trajectory is fragmented; and total number of identity switches (Ids).

Furthermore, since 3D calibration is available for both sequences, we compare with the following state-of-the-art trackers which use 3D detections:

- [34], a tracking algorithm based on Linear Programming.
- [25], which includes social behavior, using the code provided by the authors.
- [19], which includes social and grouping behavior on a Linear Programming framework, using the code provided by the authors.

We also create two baselines using the same Linear Programming formulation as presented in this paper. For all 3 methods, the same parameters will be used:

- LP + 2D: Linear Programming using only pixel distance between pedestrians to solve the data association problem.
- LP + OF: Linear Programming with pedestrian velocity estimation coming only from Optical Flow.
- Proposed: Linear Programming formulation presented in this paper with pedestrian velocity estimation trained using Random Forests.

Method	Rec.	Prec.	MT	PT	ML	Frg	Ids
Zhang *et al.* [34]	74.6	77.8	55.6	38.1	6.2	178	138
Leal-Taixé *et al.* [19]	74.1	75.3	55.1	36.9	7.9	184	131
Pellegrini *et al.* [25]	72.3	84.1	51.6	42.7	5.6	206	77
Milan *et al.* [23]	77.3	87.2	66.4	25.4	8.2	69	57
Yang & Nevatia [32]	79.0	**90.4**	68.0	24.8	7.2	**19**	**11**
LP + 2D	80.7	83.6	64.1	29.6	6.2	91	70
LP + OF	76.1	80.2	55.9	33.5	10.5	104	75
Proposed	**83.8**	79.7	**72.0**	**23.3**	**4.7**	85	71

Table 1: Results on BAHNHOF and SUNNYDAY datasets. MT = mostly tracked. PT = partially tracked. ML = mostly lost. Frg = fragmented tracks. Ids = identity switches.

We show the comparative results averaged for both datasets in Table 1. As we can see, our method obtains the highest recall rate, outperforming state-of-the-art by almost 5%. Precision is slightly lower, mostly due to double detections which create ghost trajectories easily treated during post-processing. A high recall rate is more meaningful for tracking, as is also shown by the fact that we have a mostly tracked (MT) rate of 72% *vs*. 68% of state-of-the-art. We also see that results with LP+2D are better than with LP+OF, which estimates a pedestrian velocity only with the Optical Flow direction. Even though the proposed method strongly relies on OF information, its integration within our framework increases the rate of mostly tracked (MT) pedestrians by 8% when compared to LP+2D. Note that LP+2D, LP+OF and the proposed method all use the same parameters for the LP. In general, methods working in 3D and with Social Force Models depend highly on the proper calibration of the cameras. Since odometry information is found automatically and is prone to errors, this severely affects the efficacy of [19] and [25].

Next, we show the results on each sequence separately. Aside from the four sequences used before, we also report results on CROSSING and PEDCROSS2 from [9], for which we do not have odometry information, and therefore we cannot report the results of methods that work with 3D coordinates. For the sequences BAHNHOF and SUNNYDAY we use the detections of [32], while for the LINTHESCHER, JELMOLI, CROSSING and PEDCROSS2, we use a part-based model detector [11].

In Table 2 we can see the results on all six sequences. The proposed method outperforms all other methods with higher recall rates and less mostly lost (ML) tracks. For SUNNYDAY and CROSSING, we also obtain few identity switches, 3 and 5 respectively. As we can see, even if Optical Flow features contain a lot of information on the pedestrian velocity, their naive use leads to a poor performance as shown by the results obtained with LP+OF. This shows that the proposed method is able to take the most out of a feature channel (OF) that on its own is not able to provide good velocity estimations.

Finally, we show some examples of velocity estimation in Figure 7, where the ground truth velocity in pixel coordinates is plotted in a green arrow and the one estimated by our method in a red arrow. We can see the high accuracy of the velocity estimations, even when the pedestrian is walking at low speed or along with the camera motion.

Dataset	Method	Rec.	Prec.	MT	PT	ML	Frg	Ids
Bahnhof	[19]	73.3	75.4	51.1	41.5	7.4	155	107
	[25]	71.6	84.9	46.8	48.9	4.3	173	**62**
	LP+2D	79.2	**85.8**	60.6	34.0	5.4	**80**	**62**
	LP+OF	75.3	81.5	53.2	37.2	9.6	90	67
	Proposed	**82.4**	80.6	**70.3**	**25.5**	**4.2**	81	68
Sunnyday	[19]	78.1	75.3	73.3	16.7	10.0	29	24
	[25]	75.5	**80.5**	66.6	23.3	10.1	33	15
	LP+2D	87.5	73.7	80.0	10.0	10.0	11	8
	LP+OF	82.0	71.9	63.3	26.7	10.0	12	6
	Proposed	**90.4**	75.6	**80.0**	**13.3**	**6.7**	**4**	**3**
Linthesch.	[19]	61.1	64.6	23.1	37.0	39.9	149	107
	[25]	59.7	**75.2**	20.2	40.4	39.4	168	44
	LP+2D	66.8	62.4	33.6	29.3	37.1	144	115
	LP+OF	64.8	60.4	27.9	33.2	38.9	**140**	**40**
	Proposed	**67.9**	58.6	**31.7**	**33.2**	**35.1**	172	132
Jelmoli	[19]	55.4	70.6	14.9	51.1	34.0	36	25
	[25]	53.5	**76.7**	17.0	46.8	36.2	48	**15**
	LP+2D	62.8	68.2	27.5	40.5	32.0	30	25
	LP+OF	51.1	66.3	19.1	42.5	38.4	**14**	**15**
	Proposed	**64.7**	64.4	**27.7**	**40.4**	**31.9**	37	32
Crossing	LP+2D	77.9	61.3	38.5	34.6	26.9	12	9
	LP+OF	63.5	**66.9**	26.9	38.5	34.6	8	6
	Proposed	**78.8**	56.6	**42.3**	**30.8**	**26.9**	**8**	**5**
Pedcross2	LP+2D	57.6	62.6	26.0	36.0	38.0	71	65
	LP+OF	43.2	**68.8**	10.7	43.3	46.0	**62**	**42**
	Proposed	**58.2**	60.0	**25.3**	**40.7**	**34.0**	67	72

Table 2: Results on six publicly available sequences. MT = mostly tracked. PT = partially tracked. ML = mostly lost. Frg = fragmented tracks. Ids = identity switches.

6. Conclusions

In this paper, we presented a novel method for multiple people tracking on monocular images. Using only low-level image features, we computed what we call the interaction feature strings, which contain not only a rich representation of the velocity of the pedestrian, but also of the physical interaction between targets. The latter was previously modeled using the Social Force Model, which needed accurate 3D information of the pedestrian's position and was highly dependent on the detection rate. On the contrary, our method implicitly encodes the effect of undetected targets, making the tracker more robust to partial occlusions. The interaction feature strings are then used to train a Random Forest to estimate the velocity of a pedestrian. Results on six publicly available datasets show that our method outperforms state-of-the-art approaches in multiple people tracking, obtaining higher recall rates and lower rates of completely untracked pedestrians.

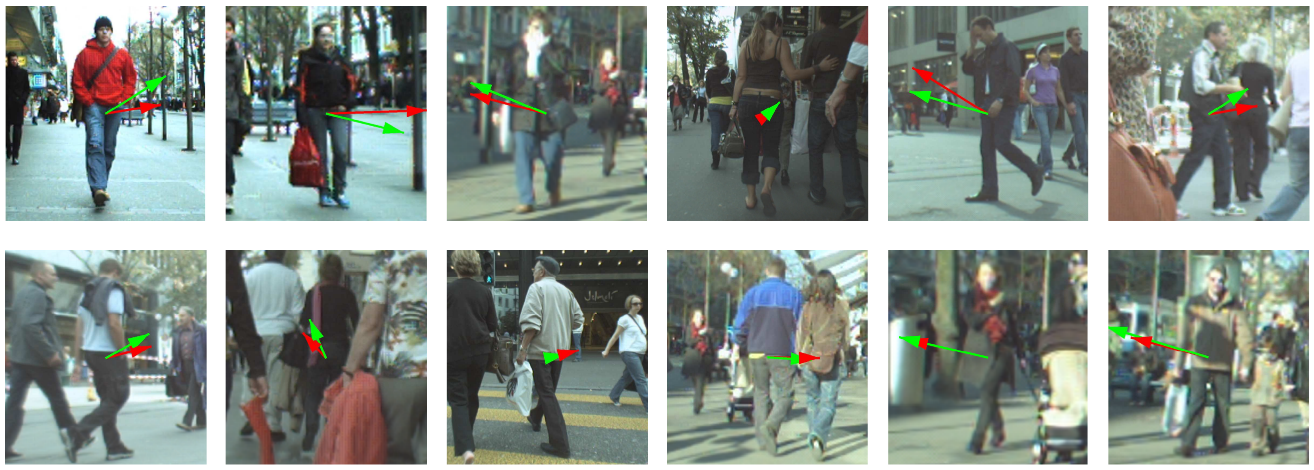


Figure 7: Examples of estimated pedestrian velocities (red arrow) *vs*. ground truth velocities (green arrow). The proposed method is able to estimate both direction and norm accurately, even in cases where pedestrians have very low speed.

References

[1] A. Andriyenko and K. Schindler. Discrete-continuous optimization for multi-target tracking. *CVPR*, 2011. 2

[2] J. Berclaz, F. Fleuret, and P. Fua. Robust people tracking with global trajectory optimization. *CVPR*, 2006. 2

[3] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *TPAMI*, 2011. 2, 5

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 2, 4

[5] A. Butt and R. Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. *CVPR*, 2013. 2

[6] W. Choi and S. Savarese. Multiple target tracking in world coordinate with single, minimally calibrated camera. *ECCV*, 2010. 2

[7] W. Choi and S. Savarese. A unified framework for multi-target tracking and collective activity recognition. *ECCV*, 2012. 2

[8] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. *MSR-TR-2011-114*, 2011. 4

[9] A. Ess, B. Leibe, K. Schindler, and L. van Gool. A mobile vision system for robust multi-person tracking. *CVPR*, 2008. 5, 7

[10] A. Ess, B. Leibe, and L. van Gool. Depth and appearance for mobile scene analysis. *ICCV*, 2007. 5

[11] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 2010. 2, 7

[12] J. Gall, A. Yao, N. Razavi, L. van Gool, and V. Lempitsky. Hough forests for object detection, tracking and action recognition. *TPAMI*, 2011. 2, 4

[13] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51:4282, 1995. 2, 6

[14] H. Jiang, S. Fels, and J. Little. A linear programming approach for multiple object tracking. *CVPR*, 2007. 2

[15] A. Johansson, D. Helbing, and P. Shukla. Specification of a microscopic pedestrian model by evolutionary adjustment to video tracking data. *Advances in complex systems*, 2007. 2

[16] S. Khamis, V. Morariu, and L. Davis. A flow model for joint action recognition and identity maintenance. *CVPR*, 2012. 2

[17] Z. Khan, T. Balch, and F. Dellaert. Mcmc-based particle filtering for tracking a variable number of interacting targets. *TPAMI*, 2005. 2

[18] T. Lan, L. Sigal, and G. Mori. Social roles in hierarchical models for human activity recognition. *CVPR*, 2012. 2

[19] L. Leal-Taixé, G. Pons-Moll, and B. Rosenhahn. Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. *ICCV. 1st Workshop on Modeling, Simulation and Visual Analysis of Large Crowds*, 2011. 1, 2, 5, 6, 7

[20] L. Leal-Taixé, G. Pons-Moll, and B. Rosenhahn. Branch-and-price global optimization for multi-view multi-object tracking. *CVPR*, 2012. 2

[21] Y. Li, C. Huang, and R. Nevatia. Learning to associate: hybrid boosted multi-target tracker for crowded scene. *CVPR*, 2009. 6

[22] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. *CVPR*, 2009. 2

[23] A. Milan, K. Schindler, and S. Roth. Detection- and trajectory-level exclusion in multiple object tracking. *CVPR*, 2013. 6, 7

[24] N. Pelechano, J. Allbeck, and N. Badler. Controlling individual agents in high-density crowd simulation. *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2007. 2

[25] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: modeling social behavior for multi-target tracking. *ICCV*, 2009. 1, 2, 6, 7

[26] H. Pirsiavash, D. Ramanan, and C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. *CVPR*, 2011. 2

[27] P. Scovanner and M. Tappen. Learning pedestrian dynamics from the real world. *ICCV*, 2009. 2

[28] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from a single depth image. *CVPR*, 2011. 2

[29] S. Walk, N. Majer, K. Schindler, and B. Schiele. New features and insights for pedestrian detection. *CVPR*, 2010. 2

[30] C. Wojek, S. Walk, S. Roth, and B. Schiele. Monocular 3d scene understanding with explicit occlusion reasoning. *CVPR*, 2011. 2

[31] K. Yamaguchi, A. Berg, L. Ortiz, and T. Berg. Who are you with and where are you going? *CVPR*, 2011. 2, 6

[32] B. Yang and R. Nevatia. An online learned crf model for multi-target tracking. *CVPR*, 2012. 6, 7

[33] L. Yeffet and L. Wolf. Local trinary patterns for human action recognition. *ICCV*, 2009. 3

[34] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. *CVPR*, 2008. 2, 5, 6, 7