Efficient Multiple People Tracking Using Minimum Cost Arborescences

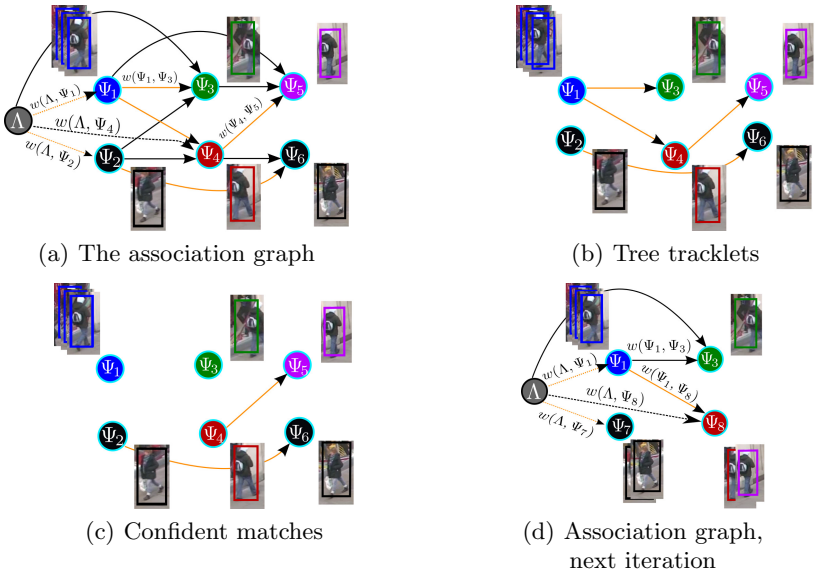
Roberto Henschel[1], Laura Leal-Taixé[2], Bodo Rosenhahn[1]

[1] Institut für Informationsverarbeitung, Leibniz Universität Hannover,
{henschel,rosenhahn}@tnt.uni-hannover.de
[2] Institute of Geodesy and Photogrammetry, ETH Zurich,
leal@geod.baug.ethz.ch

Abstract. We present a new global optimization approach for multiple people tracking based on a hierarchical tracklet framework. A new type of tracklets is introduced, which we call *tree tracklets*. They contain bifurcations to naturally deal with ambiguous tracking situations. Difficult decisions are postponed to a later iteration of the hierarchical framework, when more information is available. We cast the optimization problem as a minimum cost arborescence problem in an acyclic directed graph, where a tracking solution can be obtained in linear time. Experiments on six publicly available datasets show that the method performs well when compared to state-of-the art tracking algorithms.

1 Introduction

A key challenge in many computer vision domains is to automatically detect objects in video sequences and track them with high accuracy over time. For applications such as surveillance, action recognition, animation or human-computer interaction systems, multiple people tracking has emerged as one of the main tasks to be solved. Algorithms in recent literature have shown great performance in semi-crowded environments. In particular, the hierarchical tracklet approach [11] has evolved as an excellent tracking framework, mainly because of its bootstrapping capabilities, and is hence in the spotlight of current research. While latest improvements have been mainly achieved by exploiting the bootstrapping potential of this approach, little has been done to improve the quality of tracklets. Even though tracklets are the input for many tracking methods, usually these are found in a greedy way, and no appropriate method for finding reliable tracklets has been presented so far. In this paper, we present a global association method for tracklet creation within a hierarchical tracking framework. We propose a generalization of tracklets, which we call *tree tracklets*, that fits better into the hierarchical structure and, finally, a tracking method where tracklet association is formulated as a minimum cost arborescence (MCA) problem. The framework thus performs associations at each iteration with improved time complexity of $\mathcal{O}(n)$ in the number of current tracklets n, and its performance is in the range of less efficient state-of-the-art approaches.



(a) The association graph (b) Tree tracklets

(c) Confident matches (d) Association graph, next iteration

Fig. 1. In the association graph (a), each detection/tracklet Ψ_i is connected to all possible tracklets in consecutive frames up to some maximum time gap (for simplicity not all edges are drawn). The nodes of all shown figures are ordered from left to right by increasing frame number w.r.t. the tail of the corresponding tracklet. Each incoming edge of minimal weight is colored orange. The virtual start node Λ is used to model a new trajectory start. Detections are then grouped into two sets (b), which we call tree tracklets and cut into confident matches (c). In the next iteration of our algorithm, a new association graph is built up (d) using the confident matches from the last iteration. Ambiguities in the last iteration are then easily resolved.

1.1 Related Work

The problem of multiple people tracking is related to many computer vision applications. A common way to tackle the problem is to divide it into two subproblems: (i) the detection of all objects and (ii) the correct aggregation of these detections along time to form the final trajectories. People detectors [3, 7] achieve high accuracy if the scene is not too crowded. Wrong detections occur especially when people are partially occluded or when they are standing too close to each other, being interpreted as one person.

The data association framework links detections in time, either on a frame-by-frame basis (online systems) or by taking longer sequences into account (offline systems). Recent advances have been made by formulating the problem of finding the trajectories for all objects and all frames. Solutions are obtained by solving a combinatorial optimization problem. If the number of objects is known a-priori, [12] computes trajectories by solving a min-cost flow problem. The formulation of [2] avoids this restriction; a solution is efficiently obtained by solving a k-shortest-path problem. In [24, 17], the data association is formulated

as a maximum-a-posteriori (MAP) problem; a solution is inferred from a min-cost flow. Instead of computing the complete trajectories at once, [11] introduces a hierarchical approach where, at each iteration, the MAP problem is solved with decreasingly restrictive parameters, using the computed trajectories from the previous iteration to form the current trajectory solution. The Hungarian method [14] is used to compute the associations, therefore, its time complexity is $\mathcal{O}(n^3)$ in the number n of tracklets for each iteration. The hierarchical structure has proven to be a very fruitful approach, since it allows to extract a lot of information out of reliable tracklets. Bootstrapping-like optimization schemes for tracklets have thus become a trend in the tracking field. For instance, the standard MAP formulation of a tracking framework requires probabilities for people entering or leaving the scenery, which are generally not known a-priori. In [11], the distribution of the tails of the tracklets is used to infer these probabilities and [23] constructs an entrance/exit map which is derived from a convex set that is spanned by the tails of long and confident tracklets. Improvements can also be achieved on the motion and appearance affinities. For appearance measurements, [23, 18, 15] train a classifier based on the computed tracklets. Having tracklets at each iteration, [23] uses a quadratic function to extend the motion model to non-linear movements based on the information of current tracklets. Motion dynamics based on the rank of the Hankel matrix are considered in [4], allowing for identifications of objects by their movement characteristics. If several people walk close to each other, their motion will not be independent of each other. [21] infers group behavior of detected people from the tracklets, while [17] uses a physics-based social force model to predict pedestrians' motion and [16] learns this motion context directly from image features in order to perform tracking in image space. Finally, [10] extends the hierarchical framework to take splits and merges of trajectories into account, which commonly happen when people are walking close to each other and then split, or vice versa.

Nonetheless, few works are dedicated to improving the method to find tracklets or the hierarchical data association method itself. In this paper, we propose an alternative association method with superior time complexity in comparison to the Hungarian method together with the modification of the tracklet model to *tree tracklets* which are constructed especially for a hierarchical framework*.

1.2 Contributions

The main contribution of this paper is three-fold:

- A tracklet-based hierarchical tracking system where at each iteration the data association is formulated as a minimum cost arborescence problem.
- A better time complexity than current global tracking methods. The time complexity is $\mathcal{O}(n)$ at each iteration, where n is the number of tracklets.
- A new trajectory model, namely *tree tracklets*, for hierarchical tracking frameworks, which handles false alarms and occlusions in a more natural way.

* The code is publicly available: http://www.tnt.uni-hannover.de/project/MPT/

2 Tracklet Creation

A hierarchical tracking framework should iteratively connect tracklets by taking decisions at each iteration only in very confident cases, so that difficult selections can be postponed for a later iteration when more information is available thereby reducing error propagation. The modified MAP formulation that we introduce is modeled exactly for that purpose: it returns a generalization of the common tracklets [11] as optimum value, which we call *tree tracklets*. Such tracklets are more robust against error propagation and are obtained by solving a minimum cost arborescence (MCA) problem (see Sect. 2.1).

Having the corresponding ordinary tracklets, false detections are removed and restrictive parameters like the size of the time window are progressively weakened, as associations become more and more confident (see Sect. 3 for more details). Note that such a hierarchical approach has already been formulated in [11] for tracklets using the Hungarian method. However, we reformulate the approach to work on *tree tracklets* and use the MCA formulation instead, resulting in more robust tracklets and faster computation.

Finally, apart from serving as a complete tracking system, the tracklets from any of the intermediate steps can be used as initialization for other tracking frameworks.

2.1 Tracklet Creation with Arborescences

We introduce the notation used in this paper. Let $\mathcal{R} = \{\mathbf{r}_i\}$ be a set of object detections obtained from a detector (*e.g.* [8, 7]) on a video sequence, transferred into 3D. Thereby, $\mathbf{r}_i := (\mathbf{p}_i, \mathbf{s}_i, t_i)$, where $\mathbf{p}_i = (x_i, y_i, z_i)$ denotes its position in 3D, $\mathbf{s}_i = (w_i, h_i)$ the size of the bounding box and t_i the time stamp of the detection response, respectively. Let V be an arbitrary set together with time functions $\mathrm{fr}^{\mathrm{head}}, \mathrm{fr}^{\mathrm{tail}} : V \to \mathbb{Z}$. We fix a time window ω of frames that are allowed between two detections to be connected and define the graph $G(V) := G(V, E_\omega)$, where $E_\omega := \{(v, w) \in V \times V \mid \mathrm{fr}^{\mathrm{tail}}(v) < \mathrm{fr}^{\mathrm{head}}(w) \leq \mathrm{fr}^{\mathrm{tail}}(v) + \omega\}$.

Now for $V = \mathcal{R}$ and $\mathrm{fr}^{\mathrm{head}}(\mathbf{r}) = \mathrm{fr}^{\mathrm{tail}}(\mathbf{r})$ defined as the timestamp of $\mathbf{r} \in \mathcal{R}$, we call a weakly connected subgraph $\mathbf{I}$ of $G(V)$ a *tracklet*, if $\mathbf{I}$ is a directed path. For example, Fig. 1(c) consists of 4 tracklets (of maximal length). For a tracklet $\mathbf{I}$, we define its head $\mathbf{r}^{\mathrm{head}} = (\mathbf{p}^{\mathrm{head}}, \mathbf{s}^{\mathrm{head}}, t^{\mathrm{head}}) \in V(\mathbf{I})$ to be the unique detection in $\mathbf{I}$ with lowest time stamp. Accordingly, we define the detection $\mathbf{r}^{\mathrm{tail}} \in V(\mathbf{I})$ with highest timestamp to be the tail.

We obtain the (next) tracklets iteratively: We set $\underline{\mathbf{I}}^0 := \mathcal{R}$. Now let $\underline{\mathbf{I}}^n$ denote the current set of computed tracklets in the n-th iteration and let $V := \underline{\mathbf{I}}^n$. For $\mathbf{I} \in V$, we define $\mathrm{fr}^{\mathrm{tail}}(\mathbf{I})$ as the timestamp of $\mathbf{I}$'s tail, and $\mathrm{fr}^{\mathrm{head}}(\mathbf{I})$ of $\mathbf{I}$'s head, respectively, and say a weakly connected subgraph Ψ of $G(\underline{\mathbf{I}}^n)$ is a *tree tracklet*, if each node of Ψ has indegree ≤ 1 in Ψ.

Note that tree tracklets allow a tracklet to be connected to several tracklets in successive frames, maintaining ambiguities until enough information has been collected in following iterations (see Fig. 1(b)). We call a set $\Phi := \{\Psi_1, \ldots, \Psi_s\}$ of (tree) tracklets a (tree) tracklet hypothesis if all (tree) tracklets are pairwise

disjoint, and say Φ is covering, if all nodes can be reached by some (tree) tracklet of Φ. By using covering tracklets, we will force the framework to explain all detections as best as possible. Finally, we denote by Φ_ω the set of all covering tree tracklet hypotheses. In particular, we solve multiple people tracking by searching for that tracklet hypothesis that fits best to all detections.

We solve the problem of tracklet association using the MAP method:

$$\Phi^* = \underset{\Phi\in\Phi_\omega}{\arg\max}\, P(\Phi \mid \underline{\mathbf{I}}^n) = \underset{\Phi\in\Phi_\omega}{\arg\max}\, P(\underline{\mathbf{I}}^n \mid \Phi)P(\Phi) = \underset{\Phi\in\Phi_\omega}{\arg\max} \prod_{\Psi_k\in\Phi} P(\Psi_k) \ . \quad (1)$$

We have $P(\underline{\mathbf{I}}^n \mid \Phi) = 1$, since Φ is covering. Furthermore, we assume that the tree tracklets Ψ_k are independent of each other. Note that if we restrict (1) to find the maximum only over tracklets, the optimization can be seen as a special case of the ordinary MAP problem used in other frameworks (see [17, 24]).

For $\Phi \in \Phi_\omega$ and $\Psi_k \in \Phi$, let $\mathbf{I}_{k_1}$ be the first appearing tracklet of Ψ_k, where $V(\Psi_k) = \{\mathbf{I}_{k_1}, \ldots, \mathbf{I}_{k_m}\}$. We define, modeled as a Markov chain, the probability

$$P(\Psi_k) := P_{\text{init}}(\mathbf{I}_{k_1}) \prod_{s=1}^{|V(\Psi_k)|} \prod_{\mathbf{I}\in\mathbf{I}_{k_s}^+} P_{\text{link}}(\mathbf{I} \mid \mathbf{I}_{k_s}) \ , \quad (2)$$

where $P_{\text{link}}(\mathbf{I}_j|\mathbf{I}_i)$ denotes the probability that the tracklet $\mathbf{I}_i$ belongs to the same object as $\mathbf{I}_j$. Thereby, $\mathbf{I}_{k_s}^+$ is the set of outgoing nodes from $\mathbf{I}_{k_s}$ in Ψ_k. The probability that a new object enters the scenery at the head of $\mathbf{I}_i$ is defined by $P_{\text{init}}(\mathbf{I}_i)$. We provide the concrete calculations of these probabilities in Sect. 3.

Inserting (2) in (1) and applying the negative logarithm, we obtain

$$\Phi^* = \underset{\Phi\in\Phi_\omega}{\arg\min} \sum_{\Psi_k\in\Phi} \Big(-\log P_{\text{init}}(\mathbf{I}_{k_{i_1}}) + \sum_{s=1}^{|V(\Psi_k)|} \sum_{\mathbf{I}\in\mathbf{I}_{k_s}^+} -\log(P_{\text{link}}(\mathbf{I}|\mathbf{I}_{k_s}))\Big) \ . \quad (3)$$

Next we show the relation of Eq. (3) to the MCA problem.

Given a covering tree tracklet hypothesis $\Phi = \{\Psi_1, \cdots, \Psi_s\}$, we construct a graph $G := G(\Phi)$ by adding a virtual node Λ and link it to the first appearing node I_{k_1} of every $\Psi_k \in \Phi$. Then, G is an aborescence, that is a rooted (in this case at Λ) directed graph such that there is a directed path from Λ to each node [9] (see also Fig. 1(a)). Assigning weights to each edge, the *minimum costs arborescence* (MCA) problem [9, 13] is to find the arborescence $\hat{\Phi}$ of G s.t. the total edge weight $w(\hat{\Phi}) := \sum_{e\in E(\hat{\Phi})} w(e)$ is minimal under all possible arborescences of G rooted at Λ (denoted by A_G). If we define the weights $w(\Lambda, \mathbf{I}) := -\log(P_{\text{init}}(\mathbf{I}))$ and $w(\mathbf{I}_i, \mathbf{I}_j) := -\log P_{\text{link}}(\mathbf{I}_j|\mathbf{I}_i)$ for all $\mathbf{I}, \mathbf{I}_i, \mathbf{I}_j \in \underline{\mathbf{I}}^n$ and rewrite (3) in terms of the defined graph G, we obtain

$$\hat{\Phi} = \underset{(V,E)\in A_G}{\arg\min} \sum_{(\Lambda,\mathbf{I}_i)\in E} w(\Lambda, \mathbf{I}_i) + \sum_{\substack{(\mathbf{I}_i,\mathbf{I}_j)\in E,\\ \mathbf{I}_i\neq\Lambda}} w(\mathbf{I}_i, \mathbf{I}_j) \ . \quad (4)$$

Hence, a solution of (3) is obtained by solving the MCA problem. Then, Φ^* is the set of weakly connected components of $\hat{\Phi} - \{\Lambda\}$. A solution for the MCA

problem can be computed in polynomial time by Edmond's algorithm [5]. In our case though, since G does not have any directed cycle, it can be shown that it is sufficient to select for each node the incoming edge of minimal weight (see for example [13]), so if $(V^*, E^*) := G(\Phi)$, then

$$\hat{\Phi} = \left(V^*, \{(u,v) \in E^* \mid w(u,v) = \min\{w(u',v) \mid (u',v) \in E^*\}\}\right) \ . \tag{5}$$

Regarding the time complexity of this procedure, we can construct the solution $\hat{\Phi}$ simultaneously while building up the graph G, using (5), and adding only computation costs of $\mathcal{O}(1)$ to the cost of constructing G. From that we obtain Φ^*, having computation costs of $\mathcal{O}(n)$ in the number of current tracklets n. Furthermore, we can update the association information after each frame that has been processed.

2.2 Using Bifurcations to Detect Ambiguities

Since tree tracklets can contain bifurcations, we explain how to deal with them. Note that when we say bifuraction it can be a split into two or more branches. Now bifurcations can be used to spot missing detections caused by splits and merges: If persons walk close together, the detector might create only one box around them, resulting in one trajectory for the group. Once a person leaves the group, the corresponding tracklet will also contain this split in form of a bifurcation. Furthermore, they help to spot and remove false detections. Figure 1 illustrates such a situation. The green box is a false detection and the red box is a correct detection within the same frame. The false detection causes an ambiguity. However, only the most likely detection is assigned to the detection in the next frame (Figure 1 (b)). The tree tracklet structure can thus automatically isolate the false detection and construct the right trajectory in the next iteration. Hence, we handle tree tracklets as explained in Algorithm 1 and obtain the next set of tracklets $\underline{\mathbf{I}}^{n+1}$. Note that a currently computed tracklet Ψ becomes one node in the next iteration. For example in Fig. 1(c), the tracklets Ψ_2 and Ψ_6 are connected. Therefore, they are grouped together to the new node Ψ_7 in the next iteration (Fig. 1(d)). The ambiguity between the nodes Ψ_1, Ψ_3 and Ψ_4 is postponed to the next iteration. However, the tracklet Ψ_4 is connected to another tracklet, resulting in new information that can be used in the next iteration.

3 Implementation Details

We provide the definitions of the functions P_{link} and P_{init} that we use in our experiments. For an arbitrary video sequence, information about the scenery is not available a-priori. Hence, we set the entrance probability to be a constant value $\theta \in [0,1]$, so $P_{\text{init}}(\mathbf{I}_i) := \theta$ for all tracklets $\mathbf{I}_i \in \underline{\mathbf{I}}^n$.

Given tracklets $\mathbf{I}_i, \mathbf{I}_j \in \underline{\mathbf{I}}^n$, we define the time difference $\triangle_{i,j} = |t_i^{\text{tail}} - t_j^{\text{head}}|$ and the forward directed velocity vectors in $\mathbf{v}_i, \mathbf{v}_j$ of $\mathbf{I}_i$ and $\mathbf{I}_j$ at $\mathbf{p}_i^{\text{head}}$ and $\mathbf{p}_j^{\text{tail}}$, respectively. Let $\delta_{i,j}^{\text{f}} := \mathbf{p}_i^{\text{tail}} + \triangle_{i,j}\mathbf{v}_i - \mathbf{p}_j^{\text{head}}$ and $\delta_{i,j}^{\text{b}} := \mathbf{p}_j^{\text{head}} - \triangle_{i,j}\mathbf{v}_j - \mathbf{p}_i^{\text{tail}}$ be the error of a linear extrapolation from $\mathbf{I}_i$ to $\mathbf{I}_j$ and vice versa.

Algorithm 1 Computing tree tracklets

Input: MCA $\hat{\Phi}$, rooted at Λ
Output: Tracklets $\underline{\mathbf{I}}^{n+1} := \{\Psi_1, \cdots, \Psi_s\}$
1: $C := \{\Lambda\}$, $c(n) := 0$, for all nodes n of $\hat{\Phi}$.
2: **for** $n \in C$ **do**
3: **if** n has siblings **then**
4: $c(n) :=$ new unique number
5: **else**
6: $c(n) := c(m)$, where m is the parent node
7: **end if**
8: $C := C \cup C_{\text{new}} \setminus \{n\}$, C_{new} being the set of children of n
9: **end for**
10: **for** each assigned value k of the function c **do**
11: Ψ_k is the subgraph of $\hat{\Phi}$ induced by all nodes $n \neq \Lambda$ with $c(n) = k$.
12: **end for**

We define the link probability as $P_{\text{link}}(\mathbf{I}_j \mid \mathbf{I}_i) = A_t(\mathbf{I}_j \mid \mathbf{I}_i) A_m(\mathbf{I}_j \mid \mathbf{I}_i)$ where the term $A_t(\mathbf{I}_j \mid \mathbf{I}_i)$ (as defined in [11]) is used to avoid too big time jumps, and the motion affinity (based on [11]) is defined as:

$$A_m(\mathbf{I}_i \mid \mathbf{I}_j) = \frac{\mathcal{N}(\delta^{\text{f}}_{i,j}, \mathbf{0}, \triangle_{i,j}\Sigma)\mathcal{N}(\delta^{\text{b}}_{i,j}, \mathbf{0}, \triangle_{i,j}\Sigma)}{\mathcal{N}(\mathbf{0}, \mathbf{0}, \triangle_{i,j}\Sigma)^2} \quad . \tag{6}$$

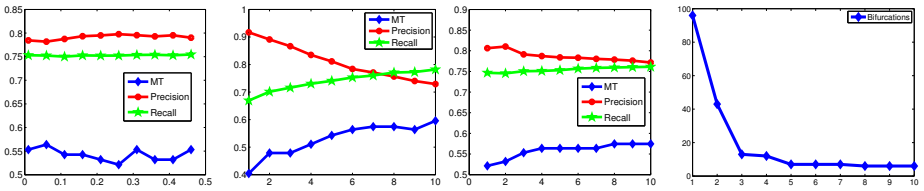
Here, $\mathcal{N}(d, \mathbf{0}, \Sigma)$ denotes the multivariate normal probability distribution of the linear extrapolation error, with mean $\mathbf{0}$ and covariance Σ. To obtain Σ, we run one iteration of the algorithm using only distance information between detections as in [17]. We then use the resulting tracklets to learn an error distribution of the extrapolation using velocity vectors for each triplet of connected nodes. These tracklets are then discarded since they are only used for initialization.

4 Experiments

We evaluate the proposed tracking framework on several publicly available datasets: Bahnhof, Sunnyday, Jelmoli and Linthescher [6] and TownCenter [1]. Detections for Bahnhof and Sunnyday are obtained from [22], for Linthescher and Jelmoli we use [7]. For TownCenter, starting from the first frame, we take every tenth frame and use the detections provided with the dataset [1]. For the evaluation, we use the following metrics [18]: Recall (correctly matched detections / ground truth detections), Precision (correctly matched detections / detections of resulting tracks), MT (mostly tracked trajectories, over 80% tracked), ML (mostly lost trajectories, less than 20% tracked), PT (partially tracked trajectories, tracked $> 20\%$ and $< 80\%$), Frg (track fragments) and Ids (number of identity switches).

4.1 Influence of the Parameters

We first analyze the effect of the parameters of Sect. 3 on the Bahnhof dataset.



(a) Entrance probability θ (b) Maximum frame jumps in the 1st iteration (c) Number of iterations (d) Number of bifurcations at each iteration

Fig. 2. (a,b,c) Parameter analysis on the Bahnhof dataset using MT (mostly tracked), Recall and Precision metrics. (d) Bifurcation evolution at each iteration on the Town-Center sequence.

Entrance probability θ. The decision wether two tracklets are being connected by the algorithm has a dependence on the choice of the entrance probability θ, which controls wether the algorithm connects fewer tracklets with more confidence or more tracklets with less confidence. As we can see in Fig. 2(a), our algorithm is robust against changes of this parameter, but it works better for values < 0.1, which is why we use $\theta = 0.09$ at the first iteration and decrease to 0.05 in the last.

Time window ω. On a dataset with a too small time window, detections/tracklets with a big temporal distance cannot be connected, resulting in shorter tracklets. On the contrary, a too big time window can, depending on the time costs, result in false assignments. Figure 2(b) shows that our algorithm runs on a wide range of time windows producing good MT and Recall values, but as expected, this is at the expense of a decrease in Precision. We find a good compromise with $\omega = 5$, where we are still able to recover reliable tracks in the first iteration and only deal with longer tracks in successive iterations. ω is then increased by one at each iteration.

Number of iterations K. Figure 2(c) plots the MT, Precision and Recall values for the numbers of iterations of our algorithm. As the algorithm proceeds, more and more tracklets are being connected, resulting in an increasing MT value with only small improvements after 4 iterations. For all our experiments, we use $K = 5$ iterations. To remove false detections, we delete tracklets after the second iteration, if their length is smaller than 3 and after the fourth iteration, if their length is smaller than 4. This is why we see a small decrease in Precision at iterations 3 and 5. However, the slightly increasing Recall value shows that our simple tracklet removing approach is already sufficient to remove false detections without producing more missing detections.

Bifurcation handling. Finally, Fig. 2(d) shows the number of bifurcations in the Town center dataset after each iteration. We see how the algorithm improves the results by continuously solving ambiguities. The figure shows nearly an inverse proportional relation between the number of iterations and number of bifurcations, until it approximately converges after 5 iterations.

Fig. 3. Example of tracked pedestrians from the Bahnhof sequence.

4.2 Runtime

Our tracking system is implemented as a non-optimized Matlab code. Using the parameters given in Sect. 3, trajectories on a sequence of 999 frames with 6536 detections and 5 iterations are computed in 8 seconds on a 3.5 GHz machine (see Table 1). Note that we compute the trajectories for the complete sequence at once. Other papers (*e.g.* [17]) separate the sequence into batches to get the results in reasonable time. Our algorithm performs an iteration in linear time, *i.e.* each iteration has a worst-case complexity of $\mathcal{O}(r)$ in the current number of tracklets r, given the cost graph that models the probabilities. In addition to that, the size of our graph decreases after every iteration, since more and more detections are grouped together. Therefore, the total computational complexity is in $\mathcal{O}(Kd)$, where K is the number of iterations and d is the number of detections. In most cases however, the algorithm will have a better runtime complexity, since the number of tracklets in each iteration decreases in most cases exponentially.

Method	Total runtime	Graph creation	Solver
Simplex [24]	16	9	7
Proposed	**8**	**6**	**2**

Table 1. Runtime in seconds of the algorithm on the Bahnhof dataset (999 frames).

Most other globally optimizing tracking frameworks solve a min-cost flow problem [24] on a graph that is similar to the graph used in our model, having entrance/exit, detection and transition edges, respectively. Their algorithm has a worst-case complexity of $\mathcal{O}(n^2 m \log^2(n))$, where n and m denote the number of nodes and edges of the graph, respectively. See also Table 1. Improvements have been made by [20], who solve a k-shortest path problem and obtain a worst-case time complexity of $\mathcal{O}(kN \log N)$, wehre N denotes the number of frames and k denotes the optimal number of trajectories. Note that typically $K << N$.

4.3 Tracking Performance

Finally, we compare our algorithm with state-of-the-art tracking systems. Since we implemented our tracking system using 3D detections, we compare only to methods that are based on 3D detections, ensuring a fair comparison. Hence, we compare to the following state-of-the-art trackers:

– Zhang *et al.* [24] propose a tracking framework based on linear programming,

– Pellegrini *et al.* [19] take avoidance behavior of humans into account, and
– Leal-Taixé *et al.* [17] combine linear programming with a social force model.

Overall, the results of the proposed method are competitive with state-of-the-art tracking approaches, while being computationally more efficient. Note the results on the TownCenter dataset, where our method achieves 6% more Recall than the best method and has 13% more Mostly Tracked (MT) trajectories, while keeping a low number of identity switches. Note that our algorithm neither considers any special social modeling as in [19, 17] nor any type of appearance model. In Fig. 3 we show an example of the trajectories obtained using the proposed method on the Bahnhof sequence.

Dataset	Method	Rec.	Prec.	MT	PT	ML	Frg	Ids
Bahnhof	[24]	67.6	70.9	43.7	46.8	9.6	176	**39**
	[17]	73.3	75.4	51.1	41.5	7.4	**155**	107
	[19]	71.6	**84.9**	46.8	48.9	**4.3**	173	62
	Proposed	**75.3**	78.4	**56.4**	**38.3**	5.3	166	76
Sunnyday	[24]	76.8	70.6	**73.3**	**16.7**	10.0	39	**9**
	[17]	78.1	75.3	**73.3**	**16.7**	10.0	**29**	24
	[19]	75.5	**80.5**	66.6	23.3	10.1	33	15
	Proposed	**79.4**	75.9	**73.3**	20.0	**6.7**	38	12
Linthescher	[24]	56.7	59.7	18.8	**34.6**	46.7	163	**42**
	[17]	61.1	64.6	**23.1**	37.0	39.9	**149**	107
	[19]	59.7	**75.2**	20.2	40.4	39.4	168	44
	Proposed	**61.7**	65.4	20.1	42.8	**37.0**	223	107
Jelmoli	[24]	53.3	62.3	14.9	**46.8**	38.3	**30**	**4**
	[17]	**55.4**	70.6	14.9	51.1	**34.0**	36	25
	[19]	53.5	**76.7**	**17.0**	**46.8**	36.2	48	15
	Proposed	52.3	68.5	12.8	51.1	36.2	40	17
Town center	[24]	77.2	79.9	53.3	39.6	**7.1**	135	233
	[17]	77.9	**88.7**	53.3	37.8	8.9	56	68
	[19]	74.5	77.5	44.0	48.4	7.6	**53**	42
	Proposed	**83.8**	81.2	**66.7**	**23.1**	10.2	70	**41**

Table 2. Tracking results on different datasets. MT = mostly tracked. PT = partially tracked. ML = mostly lost. Frg = fragmented tracks. Ids = identity switches. Values for [24, 19, 17] are taken from [16].

5 Conclusion

In this work we introduced a new type of tracklets, namely *tree tracklets*, which contain bifurcations to naturally deal with ambiguous tracking situations. These are then used in a hierarchical tracking framework, which we formulate as an optimization problem than can be solved in linear time. Experiments show good running time as well as performance compared to state-of-the art tracking systems on six publicly available datasets.

References

1. Benfold, B., Reid, I.: Stable multi-target tracking in real-time surveillance video. In: CVPR (2011)
2. Berclaz, J., Fleuret, F., Türetken, E., Fua, P.: Multiple object tracking using k-shortest paths optimization. TPAMI (2011)
3. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
4. Dicle, C., Sznaier, M., Camps, O.: The way they move: Tracking targets with similar appearance. In: ICCV (2013)
5. Edmonds, J.: Optimum branchings. Journal of Research of the National Bureau of Standards Section B - Mathematical Sciences (4) (1967)
6. Ess, A., Leibe, B., Schindler, K., van Gool, L.: A mobile vision system for robust multi-person tracking. In: CVPR (2008)
7. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. TPAMI (2010)
8. Gall, J., Yao, A., Razavi, N., van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking and action recognition. TPAMI (2011)
9. Gibbons, A.: Algorithmic graph theory. Cambridge University Press (1985)
10. Henriques, J.F., Caseiro, R., Batista, J.: Globally optimal solution to multi-object tracking with merged measurements. In: ICCV (2011)
11. Huang, C., Wu, B., Nevatia, R.: Robust object tracking by hierarchical association of detection responses. In: ECCV (2008)
12. Jiang, H., Fels, S., Little, J.: A linear programming approach for multiple object tracking. In: CVPR (2007)
13. Kamiyama, N.: Arborescence problems in directed graphs: Theorems and algorithms. Graduate School of Information Sciences, Tohoku University (2014)
14. Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly 2(1-2), 83–97 (1955)
15. Kuo, C.H., Huang, C., Nevatia, R.: Multi-target tracking by on-line learned discriminative appearance models. In: CVPR (2010)
16. Leal-Taixé, L., Fenzi, M., Kuznetsova, A., Rosenhahn, B., Savarese, S.: Learning an image-based motion context for multiple people tracking. In: CVPR (2014)
17. Leal-Taixé, L., Pons-Moll, G., Rosenhahn, B.: Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. ICCV Workshops. 1st Workshop on Modeling, Simulation and Visual Analysis of Large Crowds (2011)
18. Li, Y., Huang, C., Nevatia, R.: Learning to associate: hybrid boosted multi-target tracker for crowded scene. In: CVPR (2009)
19. Pellegrini, S., Ess, A., Schindler, K., van Gool, L.: You'll never walk alone: modeling social behavior for multi-target tracking. In: ICCV (2009)
20. Pirsiavash, H., Ramanan, D., Fowlkes, C.: Globally-optimal greedy algorithms for tracking a variable number of objects. In: CVPR (2011)
21. Qin, Z., Shelton, C.R.: Improving multi-target tracking via social grouping. In: CVPR (2012)
22. Yang, B., Nevatia, R.: An online learned crf model for multi-target tracking. In: CVPR (2012)
23. Yang, B., Nevatia, R.: Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In: CVPR (2012)
24. Zhang, L., Li, Y., Nevatia, R.: Global data association for multi-object tracking using network flows. In: CVPR (2008)