Multiple Target Tracking Using Frame Triplets

Asad A. Butt and Robert T. Collins

Dept. of Computer Science and Engineering,
The Pennsylvania State University, University Park, PA. 16802
{asad,rcollins}@cse.psu.edu

Abstract. This paper addresses the problem of multi-frame, multi-target video tracking. Unlike recent approaches that use only unary and pairwise costs, we propose a solution based on three-frame tracklets to leverage constant-velocity motion constraints while keeping computation time low. Tracklets are solved for within a sliding window of frame triplets, each having a two frame overlap with neighboring triplets. Any inconsistencies in these local tracklet solutions are resolved by considering a larger temporal window, and the remaining tracklets are then merged globally using a min-cost network flow formulation. The result is a set of high-quality trajectories capable of spanning gaps caused by missed detections and long-term occlusions. Our experimental results show good performance in complex scenes.

1 Introduction

We address the problem of tracking multiple targets through a video sequence. The problem is significantly harder than single target tracking for many reasons, including matching ambiguity between similar nearby targets, and the potential for interaction and occlusions between targets. Recent detect-then-track methods divide multi-target tracking into two separate subproblems [1]:

1. Detect all the objects of interest within each frame. It is acceptable if this detection step produces false positives.
2. Perform data association to assign a unique label to all observations in different frames that correspond to the same object of interest, thus identifying individual object trajectories.

In this paper we focus on the second, data association step. Early methods for multi-frame data association took a greedy approach, solving a series of bipartite assignment problems to match an evolving set of trajectories to target observations in each new frame [3]. This greedy approach does not work well for closely-spaced, interacting targets [4]. Recent methods have attempted to bring more global information to bear by creating a graph of all the observations within a sequence [5–7], or by using a hierarchical solution method [8]. However, these graph-based methods only consider pairwise connections between observations, and thus are not able to capture higher-order constraints such as constant velocity that describe object motion across three or more frames. Unfortunately,

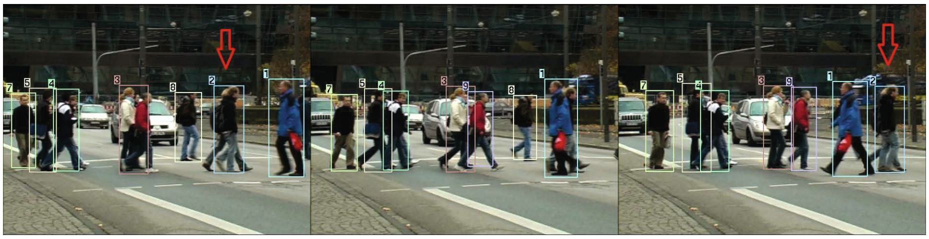
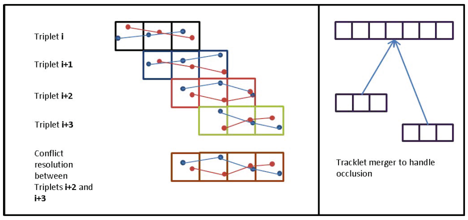


Fig. 1. Result of our algorithm shown on frames from the TUD crossing sequence [2]. From left to right, the person labeled 2 is completely occluded by 1, but the track is recovered.

graph methods using higher-order cliques generally fall victim to exponential increase in size of the search space as the number of frames increases.

In this paper, we seek a tradeoff between using higher-order cliques and maintaining computational efficiency. Specifically, we solve the data association problem over three frames at a time to obtain tracklets, which are fragments of complete trajectories. The three frame problem keeps the size of the search space in check while allowing us to use higher-order motion constraints such as constant velocity. Such motion constraints greatly aid tracking during periods when objects disappear for a short time due to missed detections or occlusion (Fig. 1). It is important to note that our constant velocity measure is applied over a sliding window of three frames, and hence, can be described as piece-wise constant velocity — targets do not have to move in a straight line with constant speed through the entire sequence, and can slow down, speed up, stop, or change direction. Our approach starts by finding tracklets within a sliding window of frame triplets, where each triplet has a two frame overlap with the preceding and following triplets (Fig. 2). This overlap allows us to detect and correct conflicts arising between local independent tracklet solutions. Tracklets are then merged globally using a min-cost flow approach. Although our solution provides a framework where both appearance and motion constraints can be combined for data association, our three-frame motion constraints are powerful enough to work well by themselves when appearance information is not available or is not discriminative.

Fig. 2. Overview of our algorithm. Tracklets are extracted within a sliding window of three frames. Inconsistencies between overlapping tracklets are resolved by increasing the size of the window to four frames. Tracklets are then linked using min-cost flow to recover complete trajectories.


Triplet i
Triplet i+1
Triplet i+2
Triplet i+3
Conflict resolution between Triplets i+2 and i+3
Tracklet merger to handle occlusion

2 Related Work

Recursive filtering approaches for single target tracking, such as the Kalman filter or particle filter, are well-studied in the tracking literature[9] but do not perform well in multi-target settings. Such trackers have a tendency to "jump" between similar targets that pass near each other, leading to identity swap errors. In Khan et al. [10], multiple independent single target trackers are temporarily coupled together with links in a graphical model as targets come close to each other. The joint probabilistic data association filter (JPDAF) attempts to bypass the problem entirely, extending trajectories based on weighted averages of nearby target observations in new frames [9].

Data association refers to the problem of making explicit assignments between observations in two or more frames. Because data association methods represent and reason over the combinatorial space of assignments, they fall squarely in the realm of discrete combinatorial optimization problems. Considering that there are N^F different trajectories passing through N points observed over F frames, it is easy to see that multi-frame data association is NP-hard due to the combinatorial explosion in size of the solution space as problem sizes grow. Although enumeration-based methods such as multi-hypothesis tracking (MHT) attempt to limit this growth by heuristic pruning, they remain cumbersome. Approximate solution methods include greedy bipartite data association on a frame-by-frame basis [3]. The two-frame bipartite assignment problem (aka linear assignment problem) can be solved exactly in polynomial time, leading to fast algorithms. However, greedy two-frame solutions do not work well for cases with target interaction or occlusion [4].

Many approaches progressively associate tracklets to obtain the final trajectories. Li et al. [11] use ranking and classification algorithms to learn parameters for tracklet association. Yang et al. [12] create a CRF of the tracklets, and use the RankBoost algorithm to select features for the cost. In contrast, Song et al. [13] combine tracklets using a graph evolution scheme, where the features along association paths are used to stochastically adapt the affinities.

Recent approaches formulate the multi-frame data association problem globally as a polynomial-time network flow problem [5, 6, 14]. Zhang et al. [5] create a network with two nodes for each detection, a link between these nodes weighted by the probability that the detection is part of the solution, a link between detections in adjacent frames weighted by the cost of them being part of the same track, and a flow conservation constraint to ensure that no two trajectories share any observation. The best solution for K trajectories is found using a min-cost flow algorithm, while an outer loop searches for the optimal value K of number of objects. Berclaz et al. [14] and Pirsiavash et al. [6] show that the best set of paths and number of paths K in a min-cost flow network can be computed efficiently using a successive shortest paths algorithm, yielding the same solution as [5] but much more quickly. Although these approaches address the multi-frame data association problem in polynomial time, they do so by restricting the form of cost functions used to measure the quality of a trajectory to be a summation over only unary and pairwise edge weights. This limits the terms that can be incorporated into the cost function to information that can be computed between pairs of observations in adjacent frames, such as distance. Higher-order constraints such as constant velocity cannot be represented.

A recent paper by Collins [15] formulates multi-frame data association as an NP-hard integer linear program and presents an ICM-like approximate algorithm for iteratively improving an initial feasible solution. Although the method can handle arbitrary higher-order cost functions, it is unclear how to simplify the approach to take advantage of cost functions with bounded order, such as constant velocity constraints computed over cliques of size three. Another recent paper by Brendel et al. [8] formulates multi-target data association as an integer quadratic program, specifically as a maximum weight independent set (MWIS) problem, and presents a polynomial time approximate solution method. The method starts by solving for two-frame tracklets independently, then linking these using a learned distance measure. Long term occlusions are handled by using MWIS hierarchically to merge smaller tracklets into longer ones.

3 Tracklet Generation

At the heart of our approach lies a method for generating good tracklets over observations in a small window of frames. Our algorithm uses sliding windows of three frames, i.e. frame triplets, however the approach can be generalized to handle larger sets of frames. The main tasks include constructing tracklet candidates (Section 3.1), defining a utility function to measure quality of each tracklet (Section 3.2), and solving for the set of candidate tracklets that yields the best total utility score (Section 3.3). Following independent tracklet generation in overlapping subwindows throughout the sequence, compatible overlapping tracklets are merged and any local incompatibilities between tracklets are resolved (Section 4.1). Tracklets are then linked in a global optimization phase based on min-cost network flow (Section 4.2).

3.1 Problem Formulation

Fig. 3 shows an illustrative example of three frames, with two observations in each frame. The goal is to link these observations into a set of tracklets subject to one-to-one matching constraints that disallow using any observation more than once. As shown in the figure, the problem is formulated as a tripartite multigraph, with candidate tracklets represented by hyperedges connecting one observation in each frame. Note the difference between this formulation and a typical network flow graph – here, edges connecting A and A' are different for the tracklets (A, A', A'') and (A, A', B''). A utility function defined over hyperedges measures the quality of each tracklet, and by extension, the quality of a set of tracklets. The goal is to choose a set of tracklets that maximizes sum of utility scores subject to the one-to-one matching constraints. Of the eight candidate hyperedges (tracklets) in this example, at most two can be selected without violating the matching constraints.

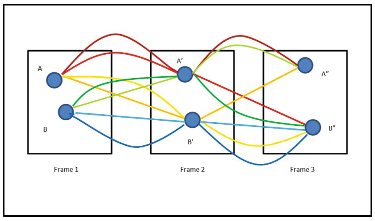


Fig. 3. A three frame multigraph. Each frame has two observations corresponding to two different targets. If all possible matches are potential correspondences there are eight possible tracklets, each shown in a different color.

More formally, let observations $\mathbf{z}_i = \{z_1(i), z_2(i), ..., z_k(i)\}$ be all detections in frame i. An ordered triplet of frames starting at time i thus contains the set of hyperedges $\Gamma_i = \{\mathbf{z_i} \times \mathbf{z_{i+1}} \times \mathbf{z_{i+2}}\}$. A tracklet $\tau_{jkl} \in \Gamma_i$ is a 3-tuple $\big(z_j(i), z_k(i+1), z_l(i+2)\big)$ such that observations in the tuple potentially represent the same target in all three frames. It is possible that not all hyperedges in Γ_i represent valid tracklets, e.g. due to violations of distance-based gating thresholds. Let the set of valid tracklets in Γ_i be $\Omega_i = \{\tau_{\mathbf{a}}, \tau_{\mathbf{b}}, ...\tau_{\mathbf{k}}\}$. Since most tracklets will share observations with other tracklets, only certain subsets of Ω_i represent feasible combinations of tracklets in a given frame triplet.

To reason about candidate tracklets in Γ_i and their feasible sets, we create a new graph $G = (V, E)$, where each $v \in V$ is a tracklet from Ω_i, and E contains an edge between vertices v_j and v_k if the corresponding tracklets share observations and thus cannot both be present together in a feasible solution. Each vertex $v_j \in V$ is assigned a weight U_j, a utility value measuring quality of that tracklet, as described in the next section. Subsets of tracklets in V are represented by an indicator vector of binary decision variables, $\mathbf{x} = (x_j) \in \{0,1\}^n$, where $x_j = 1$ when tracklet j is in the subset, and 0 otherwise.

The optimal set of tracklets in Ω_i is one that maximizes the sum of utility scores while maintaining feasibility with regard to the one-to-one matching constraints. This optimal set is the binary solution vector $\mathbf{x}^*$ given by

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \mathbf{x}^T M \mathbf{x}, \tag{1}$$

$$s.t.\ \forall j \in V, x_j \in \{0,1\},\ \text{and} \tag{2}$$

$$\forall (j,k) \in E, x_j \cdot x_k = 0 \tag{3}$$

where M is an $N \times N$ utility matrix defined over $N = \|V\|$ candidate tracklets. There are a finite number of feasible solutions to problem Eq. 1. Although we could solve this as a quadratic integer program, we use a more efficient approximate solution method, described in Section 3.3.

3.2 Utility Matrix

Quality of each tracklet τ is measured by a utility score $U(\tau)$ that is a function of the three observations forming the tracklet. This function can take into account the detector confidence score of each observation, similarity of the appearance of the observations, and geometric compatibility of the locations of the observations. In this section we focus on a geometric utility score that prefers targets that move with constant velocity across the three frames.

We calculate an error term d for each tracklet τ based on a constant velocity motion model. Using the Taylor series approximation, we know that

$$F''(x) = \frac{F(x-h) - 2F(x) + F(x+h)}{h^2} + O(h^2) \ . \tag{4}$$

For the constant velocity (zero acceleration) model, $F''(x) = 0$. Hence, for a tracklet τ containing observations $\{\mathbf{z}_j(i), \mathbf{z}_k(i+1), \mathbf{z}_l(i+2)\}$, each of which is defined by a 2×1 location vector, the error measure will be

$$d(\tau) = |\mathbf{z}_j(i) - 2\mathbf{z}_k(i+1) + \mathbf{z}_l(i+2)|. \tag{5}$$

To reward tracklets moving with constant velocity, we want a utility score that increases as the error measure in Eq. 5 decreases. We set

$$U(\tau) = \beta \exp^{-\alpha d(\tau)} \tag{6}$$

where $\alpha = 0.01$ and β is set to the number N of potential tracklets.

The utility matrix $M(G)$ of graph G for use in Eq. 1 is a variant of the adjacency matrix of G, with diagonal values being utility scores $U(\tau_j)$, for j=1 to N, and off-diagonal entry (i,j) set to 0 if (τ_i, τ_j) is an infeasible pair of tracklets, and set to 1 if they can both exist in a feasible solution. Table (1) shows an example utility matrix $M(G)$ for the three-frame example shown in Fig. 3, where tuples (i,j,k) represent the eight candidate tracklets.

Table 1. Utility matrix M(G) for Fig. 3. The diagonal values give the utility values for each tracklet. The 0's indicate infeasible pairs of tracklets that share some observation(s). The 1's represent pairs of tracklets that can co-exist.

	(A,A',A")	(A,A',B")	(A,B',A")	(A,B',B")	(B,A',A")	(B,A',B")	(B,B',A")	(B,B',B")
(A,A',A")	$U_{AA'A''}$	0	0	0	0	0	0	1
(A,A',B")	0	$U_{AA'B''}$	0	0	0	0	1	0
(A,B',A")	0	0	$U_{AB'A''}$	0	0	1	0	0
(A,B',B")	0	0	0	$U_{AB'B''}$	1	0	0	0
(B,A',A")	0	0	0	1	$U_{BA'A''}$	0	0	0
(B,A',B")	0	0	1	0	0	$U_{BA'B''}$	0	0
(B,B',A")	0	1	0	0	0	0	$U_{BB'A''}$	0
(B,B',B")	1	0	0	0	0	0	0	$U_{BB'B''}$

As in the quadratic MWIS formulation of [8], our algorithm also has the advantage of being able to easily incorporate other pairwise soft constraints directly into the problem formulation. For example, instead of setting $M(i,j)$ to 1 for feasible pairs of tracklets, we could set $M(i,j) = r$, where $r \in [0,1]$ is a measure of the similarity in the velocity of tracklets τ_i and τ_j. In this way we could encourage solutions where nearby targets move in the same direction [8].

3.3 An Approximate Spectral Solution

We would like to solve Eq. 1 to find the set of tracklets having highest total utility while not violating any one-to-one matching constraints. As currently written, this integer quadratic program is NP-hard. Instead of solving it directly, we use a spectral technique to solve a relaxed version of the problem more efficiently [16]. We relax the integrality constraint in Eq. 2 and denote the relaxed version of vector $\mathbf{x}$ by $\mathbf{y}$. We also relax the one-to-one matching constraints in Eq. 3. Note, however, that we have encoded a soft version of the matching constraints in the off-diagonal terms of M, which helps to encourage feasible solutions to form, even in the relaxed problem. By Raleigh's ratio theorem, the vector $\mathbf{y}^*$ that maximizes $\mathbf{y}^T M \mathbf{y}$ is the principal eigenvector of M. By the Perron-Frobenius theorem, $0 \leq y_i^* \leq 1$, for $i = 1, ..., N$. To recover a binary feasible solution we perform a greedy rounding step while strictly enforcing the one-to-one matching constraints, similar to [16]. While there are better methods of discretizing the relaxed solution, such as the integer projected fixed point method of [17], our solution does not rely heavily on the discretization step.

3.4 Handling Missed Detections and Tracklet Birth/Death

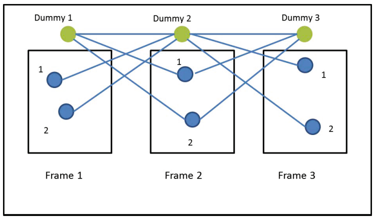


Fig. 4. Dummy observations are added to each frame to handle missed detections and target birth/death. These observations are connected to all real observations in the previous and next frames. For clarity, links between real observations are not shown.

A practical tracking algorithm must handle missed detections, short term occlusions, and variable numbers of targets entering or exiting each frame. For each frame in triplet Γ_i we add a virtual dummy observation, with a link to each real observation in the next frame (if it exists), and with each observation in the previous frame (if it exists), as shown in Fig. 4. These virtual nodes expand the number of potential tracklets included in our three-frame multigraph. The utility values of partial tracklets containing two real observations are calculated based on Euclidean distance between the observations in a way that favors shorter distances traveled. Utility scores for tracklets that contain only one real observation are computed by taking the minimum score from candidate tracklets that contain that observation, or 0.001 if there are no other such tracklets. The reasoning behind this is that we still would like a one-observation tracklet to be selected if there are no better alternatives, but having a very low score, almost any reasonable alternative is better.

4 Tracklet Linking

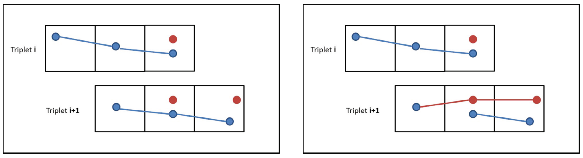
Tracklets obtained independently for each sliding window of three frames must be linked to obtain a complete set of trajectories over the entire sequence. We perform this task in two steps.

1. In a single pass through the sequence, conflicts between overlapping tracklets are resolved, while compatible tracklets representing unambiguous continuations are merged (Section 4.1). This pass through the data yields tracklets of variable length, but trajectories may still be fragmented due to occlusions that persist for several frames.
2. A global merging of the remaining fragmented trajectories is performed using a min-cost network flow formulation with tracklets as nodes (Section 4.2). An appearance-based similarity measure is used for encouraging linking of tracklets through occlusions.

These two steps are described in more detail below.

4.1 Merging and Conflict Resolution

As discussed in the last section, we solve for tracklets within overlapping sliding windows of three frames. Solutions for adjacent frame triplets Γ_i and Γ_{i+1}, for example, will have frames i+1 and i+2 in common. Two trackets $\tau_i \in \{\mathbf{z}_i \times \mathbf{z}_{i+1} \times \mathbf{z}_{i+2}\}$ and $\tau_{i+1} \in \{\mathbf{z}_{i+1} \times \mathbf{z}_{i+2} \times \mathbf{z}_{i+3}\}$ may thus share two observations, one observation, or no observations. Pairs of tracklets with no observations in common are unrelated, and of no interest here. Tracklets with one or two observations in common are candidates for either linking, or conflict resolution, as illustrated in Fig. 2 and described below.



(a) No ambiguity **(b)** Conflict occurs

Fig. 5. Possible scenarios when linking tracklets from consecutive triplets. Linking is easy in (a), because the tracklets agree on the shared observations. However, due to a conflict between the two tracklets in (b), there is ambiguity as to which observation should follow the second observation in triplet 1.

Let $\tau_i = (a, b, c)$ and $\tau_{i+1} = (b', c', d')$ and consider the following scenarios:

1. If both $(b{=}b')$ and $(c{=}c')$, the two trackets τ_i and τ_{i+1} agree on their two shared observations, and we have the case shown in Fig. 5a. The two tracklets can be immediately linked since the overlapping windows agree on which observations belong to this partial trajectory.
2. If $(b{=}b')$ but $(c \neq c')$, or if $(b \neq b')$ but $(c{=}c')$, we have a conflict, one example of which is illustrated in Fig. 5b. For whatever reason – noise, occlusion, entry and exit, or proximity of targets – the two tracklets τ_i and τ_{i+1} disagree on the continuation of the trajectory, causing a conflict that must be resolved before linking can occur.

Conflicts between tracklets in adjacent frame triplets Γ_i and Γ_{i+1} are resolved by expanding the tracklet generation procedure in Section 3 to operate over 4 frames, i to $i+3$, using only the observations from tracklets that have conflicts. Because the observations from compatible tracklets that have already been linked are no longer considered, the number of observations in this expanded 4-frame assignment problem is typically quite small, and thus the combinatorics involved in expanding from three frames to four is kept in check. The error measure (and thus utility) is based on constant acceleration computed over a 4 tuple τ_{jklm}

$$d(\tau) = |\mathbf{z}_j(i+3) - 3\mathbf{z}_k(i+2) + 3\mathbf{z}_l(i+1) - \mathbf{z}_m(i)| \ . \tag{7}$$

The complete procedure for linking compatible overlapping tracklets and resolving local conflicts is performed as a single pass through the ordered sequence of frame triplet solutions. This process maintains a gradually growing set of tracks T. We initialize the set of tracks T as the tracklets from the first frame triplet Γ_1 of the sequence. Let Γ_i be the frame triplet at the next time instance i, with i initialized as 2. Proceed as follows through the sequence of frame triplets:

1. For each tracklet $\tau = \big(z(i), z(i+1), z(i+2)\big)$ in Γ_i, find a track T_j such that $T_j(i) = z(i)$ or $T_j(i+1) = z(i+1)$.
2. If T_j exists, and $T_j(i+1) = z(i+1)$, then extend track T_j by making the assignment $T_j(i+2) = z(i+2)$.
3. If T_j exists, but $T_j(i) \neq z(i)$ or $T_j(i+1) \neq z(i+1)$, add $\{z(k), T_j(k)\}$ to C, where $k = (i-1), ..., (i+2)$.
4. If no such T_j exists, use the triplet τ to initialize a new track in T.
5. Repeat Step 1 until all tracklets in Γ_i have been visited.
6. For all conflicting observations in C, solve for the optimal four tuple tracklets in frames i through $(i+3)$, and use these to update tracks in T.
7. If the end of the sequence has been reached, return T. Else set $i = i+1$ and repeat Step 1.

4.2 Min-cost Flow for Occlusion Handling

The linking step above results in a feasible (no conflicts) set of variable-length tracklets. However, there still remains the problem of recovering full trajectories in scenes with longer term occlusions. For this purpose we use the min-cost network flow formulation of [5] to link tracklets across occlusion gaps. A flow network is constructed with tracklets as nodes and with a directed edge placed from node T_i to node T_j when tracklet T_j begins after T_i has ended, as shown in Fig. 6. The velocity of T_i and the number of frames in the gap between the two tracklets is used to estimate a search region around the last observation of T_i such that the first observation of T_j lies within it. A source node s and a sink node t are also added, with an edge from s to each of the tracklet nodes, to handle object entry into the scene at any time, as well as an edge from each tracklet to t, to handle object exit. Fig. 6a shows an example with four tracklets, and Fig. 6b shows the corresponding flow network. Since $T1$ does not overlap any other tracklet, it has a transition edge to each of the other nodes. On the other

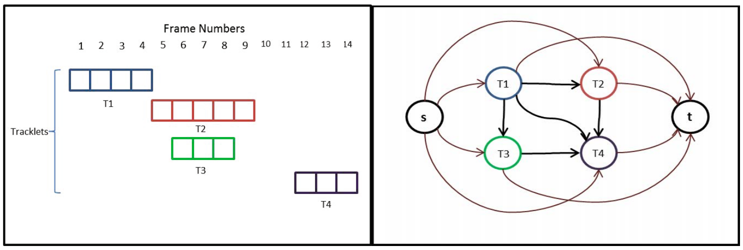


Fig. 6. Tracklets are merged to recover complete trajectories spanning occlusions. (a) An illustrative example consisting of four trajectory fragments. (b) The corresponding min-cost flow network.

hand, T2 overlaps T3, so the only tracklet following $T2$ is $T4$. If $f_{s,i}$ is the flow from s to i and $f_{i,t}$ is the flow from i to t, then the flow conservation constraint $\sum_i f_{si} = \sum_i f_{it}$ ensures that each tracklet can only be linked to one preceding and one following tracklet. The costs on arcs from s are set to a small positive number γ. The costs on arcs to t are set to $-\gamma - \epsilon$. This is done so that some flow always passes through the tracklet nodes. The costs on the arcs between the other nodes are based on the appearance similarity ϕ of the tracklet targets, as measured by similarity of their color histograms. A threshold value θ is used such that $\theta - \phi$ is positive for low similarity value, and $\theta - \phi$ is negative when the tracklets are more similar. The push-relabel method proposed by Goldberg [18] is used to solve the resulting min-cost flow problem.

5 Computational Complexity

Given on average n observations in each frame, the worst case scenario would be the formation of n^3 potential tracklets for a frame triplet. In most cases, however, an observation in one frame may only be matched to k observations in the next, where $k << n$. The number of tracklets in a triplet of frames is then nk^2. The spectral method of [16] has complexity of $O(m^{3/2})$, where m is the number of tracklets. Hence, it takes $O((nk^2)^{3/2})$ to generate tracklets in each frame triplet. In a sequence of F frames there are $F - 2$ overlapping triplets. It thus takes $O((Fnk^2)^{3/2})$ time to generate all three-frame tracklets in the sequence. The global merging step using min-cost flow has a complexity of $O(v^2 u \log v)$, where v is the total number of tracklets, and u is the number of arcs in the flow network (arcs connecting each tracklet to later tracklets that could be continuations).

6 Experiments

6.1 Importance of Constant Velocity

To show the utility of constant velocity, we use two groundtruthed datasets of trajectories from pedestrians walking in a student union building [15]. One sequence is relatively "sparse", with an average of 5 observed people per frame, while the other sequence is more densely populated with roughly 20 people observed per frame. Each sequence is 15 minutes long, and human coders have annotated the data to generate ground truth trajectories for all pedestrians.

As a baseline method, we implemented a greedy Hungarian algorithm that gradually extends a set of trajectories by incrementally associating them to observations in each new frame. We used two versions of this algorithm - using constant velocity to predict an object's position in the next frame, and using the current position as the prediction. Results from a sample sparse sequence are shown in Fig. 7. We calculate the mismatches [19] that occur in the estimated trajectories and sum them over 31 sequences. The number of mismatches for the Hungarian algorithm without constant velocity prediction is 53, for the Hungarian algorithm with constant velocity is 14, and for our approach is 11. These results clearly show the advantage of using constant velocity, and that our approach is superior to the greedy Hungarian baseline.

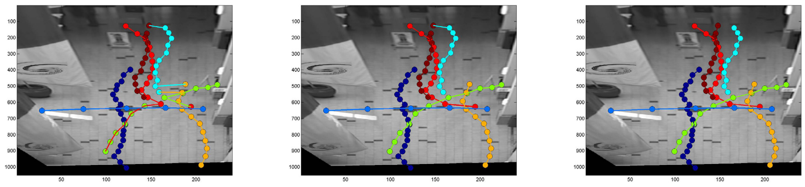


Fig. 7. A sample sequence. Dots associated with the same ground truth trajectory have the same color. (a) Result of the Hungarian method without motion prediction. (b) Hungarian method using constant velocity prediction. (c) Results of our method.

6.2 Comparison on Public Datasets

We also ran our algorithm on the publicly available TUD Crossing sequence [1] and ETHMS dataset [2]. We use the same ETHMS sequence as [5] with 999 frames. The results are compared with the dynamic programming (DP) solution in [6] and the successive shortest path (SSP) - or the equivalent push-relabel algorithm - described in [5, 6]. We use the implementation of the authors of [6] for the DP algorithm, and the push-relabel code by Goldberg [18]. We use an "out-of-the-box" pre-trained pedestrian detector described in [20]. The cost functions used for DP and SSP are based on the detector confidence score, as in [6]. For our algorithm we use the constant velocity measure described by Eq. 5. For these experiments we do not do the appearance-based min-cost flow post processing. Our implementation of the frame-triplet code is in MATLAB, and takes 2.03 seconds to run on the 201 frames of the TUD sequence.

The DP algorithm handles k-frame occlusions by creating links between detections that are k frames apart. Fig. 8 shows the DP implementation with $k = 2$ for comparison with our algorithm where dummy observations handle short-term occlusions. The detection rate is plotted against false positives per frame for the two sequences. The goal is to keep the false positives to a minimum while the detection rate is increased. For the TUD sequence, our algorithm outperforms DP. The graph for ETHMS shows that SSP outperforms the other algorithms initially. However, it only has a maximum detection rate of slightly over 50%. While the DP algorithm performs a little better than ours on this sequence, our algorithm's performance remains competitive with regards to this measure.

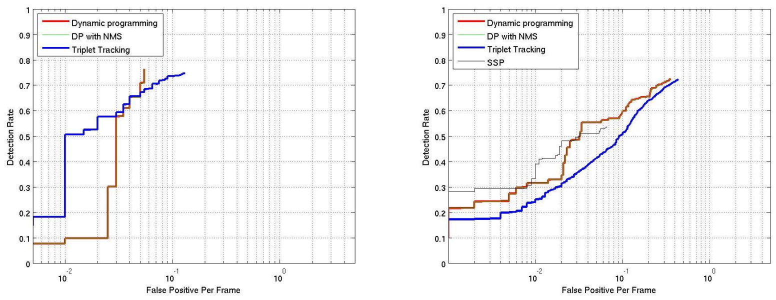


Fig. 8. Detection rate versus FPPI on the TUD sequence [1] (**left**) and ETHMS dataset [2] (**right**). DP performs slightly better on the ETHMS dataset, whereas our method performs better on the TUD sequence for this metric.

The strength of our algorithm becomes clear when comparing the number of mismatches (Table 2). A mismatch can occur when objects that are close to each other swap track labels, or when a track is lost and reinitialized with a different label. We count the number of mismatches as described in [19]. Our algorithm easily outperforms DP with $k = 1$ and $k = 2$ for both sequences. The low number of mismatches for SSP is deceptive since this happens because it fails to detect people in cluttered areas, resulting in much fewer detections than the other algorithms.

Table 2. The numbers reported here are (Number of mismatches/Total number of detections). The results are shown for the TUD sequence, and the first 350 frames of the ETHMS sequence. DP was run along with non-maximum suppression (NMS).

Sequence	**DP (k=1)**	**DP (k=2)**	**Our Algo**	**SSP**
ETHMS	37/1387	39/1427	14/1496	11/1057
TUD	32/768	29/782	23/800	-

A visual comparison of our algorithm and DP is shown in Fig. 9. By inspecting the ID labels of various people we see that our algorithm does a better job of maintaining consistent long-term labels.

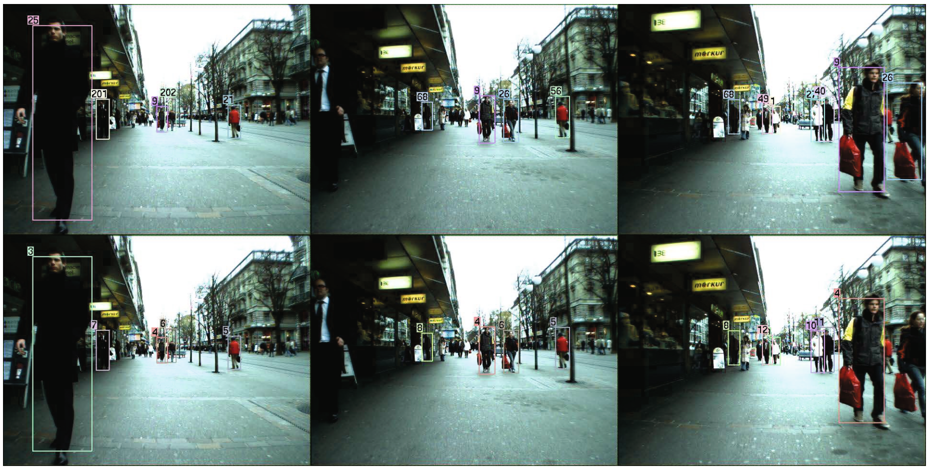


Fig. 9. The top row shows the result of running the DP (with NMS) tracker of [6] on part of the ETHMS sequence. The bottom row shows our result. Our algorithm does a better job at maintaining ID labels over time.

7 Conclusion

We have proposed a multi-frame, multi-target tracking approach based on computing three-frame tracklets subsequently linking them. Our algorithm focuses on geometric relationships between observations, and does not require a strong appearance model. Within a sliding window of frame triplets, we form all potential 3-tuples of tracklets and compute a utility value for each tracklet based on its conformance to a constant velocity motion model. The optimal set of disjoint tracklets is solved for using a spectral method. The linking stage of our algorithm allows us to correct any tracklet conflicts by solving over a larger number of frames while merging compatible tracklets representing unambiguous continuations. This pass through the data yields tracklets of variable length, but trajectories may still be fragmented due to occlusions that persist for several frames. A final stage applies the min-cost network flow formulation to link tracklets across long-term occlusion gaps, guided by target appearance information. We compared our results with algorithms from [5, 6], and showed that our algorithm outperforms them with respect to track label mismatches, while being competitive with respect to detection rate and false positives.

Acknowledgements. This work was partially funded by NSF grant IIS-1218729.

References

1. Andriluka, M., Roth, S., Schiele, B.: People-tracking-by-detection and people-detection-by-tracking. In: IEEE Conf. on Computer Vision and Pattern Recognition (2008)

2. Ess, A., Leibe, B., Schindler, K., van Gool, L.: A mobile vision system for robust multi-person tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008). IEEE Press (2008)
3. Veenman, C.J., Reinders, M.J.T., Backer, E.: Resolving motion correspondence for densely moving points. IEEE Transactions on Pattern Analysis and Machine Intelligence 23, 54–72 (2001)
4. Wu, B., Nevatia, R.: Tracking of multiple, partially occluded humans based on static body part detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 951–958 (2006)
5. Zhang, L., Li, Y., Nevatia, R.: Global data association for multi-object tracking using network flows. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008, pp. 1–8 (2008)
6. Pirsiavash, H., Ramanan, D., Fowlkes, C.C.: Globally-optimal greedy algorithms for tracking a variable number of objects. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2011)
7. Shafique, K., Shah, M.: A noniterative greedy algorithm for multiframe point correspondence. IEEE Trans. on Pattern Analysis and Machine Intelligence 27, 51–65 (2005)
8. Brendel, W., Amer, M., Todorovic, S.: Multiobject tracking as maximum weight independent set. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 1273–1280 (2011)
9. Blackman, S., Popoli, R.: Design and Analysis of Modern Tracking Sys. Artech House, Norwood (1999)
10. Khan, Z., Balch, T., Dellaert, F.: Mcmc-based particle filtering for tracking a variable number of interacting targets. IEEE Transactions on Pattern Analysis and Machine Intelligence 27, 1805–1819 (2005)
11. Li, Y., Huang, C., Nevatia, R.: Learning to associate: Hybridboosted multi-target tracker for crowded scene. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 2953–2960 (2009)
12. Yang, B., Huang, C., Nevatia, R.: Learning affinities and dependencies for multi-target tracking using a crf model. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 1233–1240 (2011)
13. Song, B., Jeng, T., Staudt, E., Roy-Chowdhury, A.: A Stochastic Graph Evolution Framework for Robust Multi-Target Tracking (2010)
14. Berclaz, J., Fleuret, F., Turetken, E., Fua, P.: Multiple object tracking using k-shortest paths optimization. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, 1806–1819 (2011)
15. Collins, R.: Multitarget data association with higher-order motion models. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2012. IEEE Press (2012)
16. Leordeanu, M., Hebert, M.: A spectral technique for correspondence problems using pairwise constraints. In: International Conference on Computer Vision (2005)
17. Leordeanu, M., Hebert, M., Sukthankar, R.: An integer projected fixed point method for graph matching and map inference. In: Proceedings Neural Information Processing Systems. Springer (2009)
18. Goldberg, A.V.: An efficient implementation of a scaling minimum-cost flow algorithm. Journal of Algorithms 22, 1–29 (1992)
19. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: the clear mot metrics. J. Image Video Process. 2008, 1:1–1:10 (2008)
20. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence 32, 1627–1645 (2010)