Efficient Track Linking Methods for Track Graphs Using Network-flow and Set-cover Techniques

Zheng Wu[1], Thomas H. Kunz[2], and Margrit Betke[1*]
Departments of [1]Computer Science and [2]Biology, Boston University, Boston, MA 02215

Abstract

This paper proposes novel algorithms that use network-flow and set-cover techniques to perform occlusion reasoning for a large number of small, moving objects in single or multiple views. We designed a track-linking framework for reasoning about short-term and long-term occlusions. We introduce a two-stage network-flow process to automatically construct a "track graph" that describes the track merging and splitting events caused by occlusion. To explain short-term occlusions, when local information is sufficient to distinguish objects, the process links trajectory segments through a series of optimal bipartite-graph matches. To resolve long-term occlusions, when global information is needed to characterize objects, the linking process computes a logarithmic approximation solution to the set cover problem. If multiple views are available, our method builds a track graph, independently for each view, and then simultaneously links track segments from each graph, solving a joint set cover problem for which a logarithmic approximation also exists. Through experiments on different datasets, we show that our proposed linear and integer optimization techniques make the track graph a particularly useful tool for tracking large groups of individuals in images.

1. Introduction

The interpretation of the motion of large groups of individuals is a challenging problem in computer vision. The fundamental difficulty in solving this problem lies in the data association step of the tracking process. When correspondences between objects and observations are not determined correctly, the state estimation step of the tracking process, typically recursive Bayesian filtering, also fails. In visual tracking, the occlusion events, especially long-term occlusions, are responsible for causing problems during the data association phase. They typically create "track-lost" or "track-switch" errors. When occlusion first occurs, individual tracks, sometimes multiple tracks, become merged. When the occlusion event is over, the earlier merged tracks are supposed to separate into appropriate individual tracks (Fig. 1). To maintain the object identity through merging and splitting events, the "track linking method," considers each individual track as a "tracklet" and links these tracklets together segment by segment.

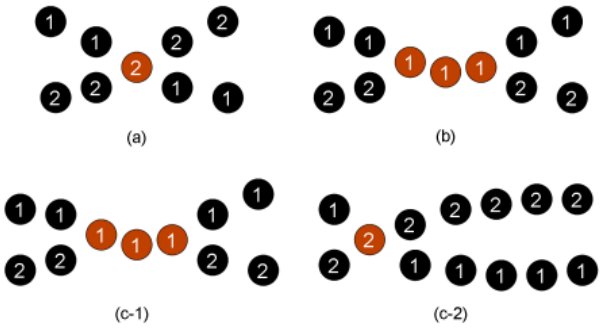

Figure 1. Three different occlusion scenarios: (a) short-term occlusion, (b) long-term occlusion, and (c) occlusion in two camera views. Red nodes represent merged measurements; numbers are labels for objects. Short-term occlusion (a) is usually easy to resolve if objects have strong motion patterns. Long-term occlusion (b) is more difficult to explain since motion information about the objects (i.e. linear dynamics) typically only characterizes them for a short time period. If multiple views are available (here two), a long-term occlusion in one view (c-1) may be resolved by analyzing the status of the objects in another view where the occlusion does not occur or only occurs for a short time (c-2). Throughout this paper, we do not assume objects are significantly distinctive in appearance or motion characteristics. Such an assumption would simplify the problem of occlusion reasoning, but cannot be made for our data.

Track linking, as a batch process, is a generalization of traditional measurement-to-measurement association: here, the matching involves trajectory segments (tracklets). Occlusion ambiguity is resolved by optimizing some cost function that considers the smoothness of object motion and appearance over several frames. With this approach, tracklets may be stitched together and full trajectories may be recovered. Previous approaches typically adopted a *local* linking

*This material is based upon work partially supported by the National Science Foundation under IIS HCI Grant 0910908 and ODDR&E MURI10 Program Grant Number N00014-10-1-0952 through the University of Washington.

strategy that computed the pairwise cost between tracklets in a recursive way [6, 13, 19]. Nillius et al. [11] introduced the "track graph" and suggested use of a Bayesian network inference algorithm on this graph as a *global* linking strategy, which allowed matching of several tracklets at the same time. This method is computationally expensive for large graphs. We instead propose a two-stage network-flow algorithm that builds the track graph and allows us to provide both *local* and *global* solutions for track-linking. We demonstrate how to resolve occlusion efficiently by interpreting it as a linear network optimization problem or set cover problem. Also, we show that our approach can be easily extended to the multi-view multi-object tracking scenario. We experimentally compared the performance of our methods to three other methods, the global linking approach by Nillius et al. [11] and two traditional approaches [2]. We used both synthetic and real video data with various levels of object density. Our experiments reveal that the relative performance of the methods depended on the level of object density. Our global linking method outperformed the other methods in dense object scenarios, where occlusion reasoning becomes particularly difficult. Our paper makes the following algorithmic and experimental contributions:

- For the *local linking scheme,* our method automatically constructs the track graph and resolves it by computing the minimum flow and a series of bipartite graph matches, each of which can be solved in polynomial time.
- For the *global linking scheme,* we convert the problem of track graph reasoning to the standard set cover problem, for which a logarithmic approximation solution exists.
- For *track linking in multiple views,* our method constructs the track graph for each view independently and resolves the track graphs jointly by solving a joint-set-cover problem. We provide a greedy logarithmic approximation method for this problem. To the best of our knowledge, we are the first to formulate the track linking problem in both the *temporal (across-time)* and *spatial (across-camera)* domains.
- Our experiments reveal that the choice of tracking algorithm should depend on the level of object density in the video. Our track-linking framework has the flexibility to handle both easy (sparse) and challenging (dense) datasets.

2. Related Work

Traditional data association approaches work on the measurement level where observations (e.g., a bounding box) are returned from the object detection module. The radar literature [2] describes mature algorithms for tracking multiple targets within a dynamic system, such as Multiple Hypothesis Tracking (MHT) and Joint Probabilistic Data Association (JPDA). MHT enumerates all possible combinations through time by building a hypothesis tree and selecting the most likely combination as its optimal solution. JPDA only analyzes the correspondence between two frames. It does not pursue the best solution but instead computes the expected track state over all hypotheses.

The probabilistic association methods have their integer optimization counterparts. When only pairwise correspondences are considered, finding the best measurement-to-measurement match can be modeled as linear network optimization problem [22, 7, 1], for which an optimal solution can be efficiently obtained in polynomial time. By contrast, the discrete optimization version of MHT, known as the multidimensional assignment problem [14], is NP-hard. An approximate solution can be obtained by semi-definite programming [15], Lagrange relaxation [5], or randomized greedy search [17].

Sampling-based algorithms form another category of data association methods. They have gained popularity recently, partially because of advances in Monte Carlo theory. Oh et al. [12] proposed a general framework to sample the data association hypothesis using a Markov Chain Monte Carlo (MCMC) approach. They showed that their batch processing method is able to track a large number of simulated objects. Khan et al. [8] introduced a probabilistic model to associate merged and split measurements using a MCMC-based particle filter. Yu and Medioni [21] also extended the framework by Oh et al. [12] to identify the best spatial and temporal association of regions with a Data-Driven MCMC sampling approach. As for most sampling methods, tuning the parameters that achieve relatively fast convergence is always a nontrivial task.

A challenge of measurement-level data-association methods is that the problem size is often too large for batch processing. A compromise between accuracy and batch size is often made using a sliding window [18]. The choice of the length of such sliding window is typically made ad hoc, based on the particular data set. It is natural to extend the framework of temporal data-association to the tracklet level, where a matched unit is a pair of trajectory segments [9]. Instead of organizing temporal data-association hierarchically, in which, at each level, *local* links between track fragments are produced [6, 13, 19], Nillius et al. [11] solved the problem *globally* by processing the track graph that represented all object interactions. Since the state space, i.e., the permutation space over the object identities, is large, their method incorporated some heuristics to make it practical, especially when objects interaction was frequent.

In our paper, we make use of the track graph representation by Nillius et al. [11]. Our algorithms for automatically constructing and interpreting the track graph, however, both significantly differ from theirs. We propose a network flow algorithm for creating the track graph and employ both *lo-*

cal and *global* linking strategies for interpreting the track graph that address the different types of occlusion events. Our formulation is based on a linear/integer optimization framework and does not incorporate any heuristics. We also introduce a new strategy for linking tracklets that involves matches across camera views. The strategy can be seen as a "track-to-track fusion scheme." In contrast to the work by Wu et al. [18], which described a track-to-track fusion scheme for multi-object multi-view sequential tracking, the focus of our paper is the development of a batch linking algorithm that uses a track graph representation. We must stress that the density of objects we are interested in is up to one hundred objects per frame. This means that, although the size of the objects is small in the images, they very frequently occlude each other.

3. Track linking

A track graph $G = (V, E)$ is defined over sets of vertices V that represent individual or merged tracks[1] and edges E that represent merging or splitting events. Directed edge $e_{i,j}$ from vertex v_i to v_j represents that track v_i is merged with track v_j if v_j is a merged track, or that v_i is split to track v_j if v_i is a merged track (Fig. 2). The *flow* on the edge indicates how many objects are involved during the merging or splitting event. The vertex that has only incoming edges is called *sink*; the vertex that has only outgoing edges is called *source*. The set of all source vertices is denoted by S, and the set of all sink vertices by T. Each vertex has its *track-capacity* to represent single or multiple objects[2]. For a source vertex, its associated track-capacity is the sum of outgoing flows; for a sink vertex, its associated track-capacity is the sum of incoming flows; for other intermediate vertices, the sum of incoming flows is equal to the sum of outgoing flows (for balance). For tracking in a single view, an isolated vertex that has no incoming or outgoing edges has capacity one. We remove these isolated vertices in preprocessing, as they do not require occlusion reasoning.

3.1. Algorithm to Construct Track Graph

We propose a two-stage algorithm for automatically constructing the track graph. In the first stage, our algorithm generates track fragments and merge/split hypotheses that define the vertices and edges of the track graph. In the second stage, a path-reducing min-flow algorithm determines the track-capacity of each vertex and flow for each edge (e.g., Fig. 2(b)).

Stage I: The algorithm first processes the image sequence forward in time to generate basic tracks and merge hypotheses. It then goes backward to break some tracks, when necessary, and generate split hypotheses, and finally constructs the track graph:

I.1 **Tracking Forward:** A new tracker is initiated when a measurement cannot be associated with an existing tracker. Each existing tracker chooses the measurement nearest to its position estimate, which is computed by a Kalman filter, as its current observation. If a measurement is the nearest neighbor of the position estimates of multiple trackers, each of these trackers terminates itself, and a new tracker is initiated for this measurement. Meanwhile, a track-merge hypothesis H_m is generated and added to the list of hypotheses. An existing tracker also terminates itself if it is not associated with any measurement for a certain number of frames.

I.2 **Tracking Backward:** If a track is not initiated within the zone of the scene, where objects enter (e.g., the image boundary), then it must be a track that is split from a previously merged track. Its position is predicted backward in time to find a nearest measurement. The track that originally included this measurement will be denoted as a merged track. Meanwhile, a track split hypothesis H_s is generated and added to the list of hypotheses.

I.3 **Building Track Graph:** The list of merge/split hypotheses is sorted according to time. A vertex of the track graph is created for each track on this list. For each merge hypothesis H_m that merges track $\mathcal{T}_{i_1}, \mathcal{T}_{i_2}, ...\mathcal{T}_{i_m}$ to track $\mathcal{T}_j$, corresponding edges from vertices $v_{i_1}, v_{i_2}, ..., v_{i_m}$ to v_j are added to the track graph. For each split hypothesis H_s that splits track $\mathcal{T}_i$ into track $\mathcal{T}_{j_1}, \mathcal{T}_{j_2}, ...\mathcal{T}_{j_n}$, the corresponding edges from vertex v_i to $v_{j_1}, v_{j_2}, ..., v_{j_n}$ are added.

Stage II: The number of objects in the track graph is equivalent to the amount of flow passing through the network. Since each track represents at least one object, we have a lower bound on the capacity of the edges in the track graph. This is not sufficient for uniquely determining the actual number of objects and describes the ambiguity caused by occlusion, i.e., an arbitrary number of objects can "hide" in any merged track. For single-view tracking, where we cannot rely on additional information provided by unoccluded views, we require our algorithm to select the smallest number of objects that can explain the scene. We thus converted our problem into a minimum-flow problem where the lower bound on the capacity of each edge is one. We use a polynomial-time algorithm that iteratively searches for a "reducing path" (as opposed to the "augmenting path" in the max-flow Ford-Fulkerson method [4]) and updates the residual network:

[1] A merged track is due to either a close interaction between objects or a projection of moving objects that are physically far apart in 3D space.

[2] Unlike traditional terminology, we here define "capacity" for both vertices and edges.

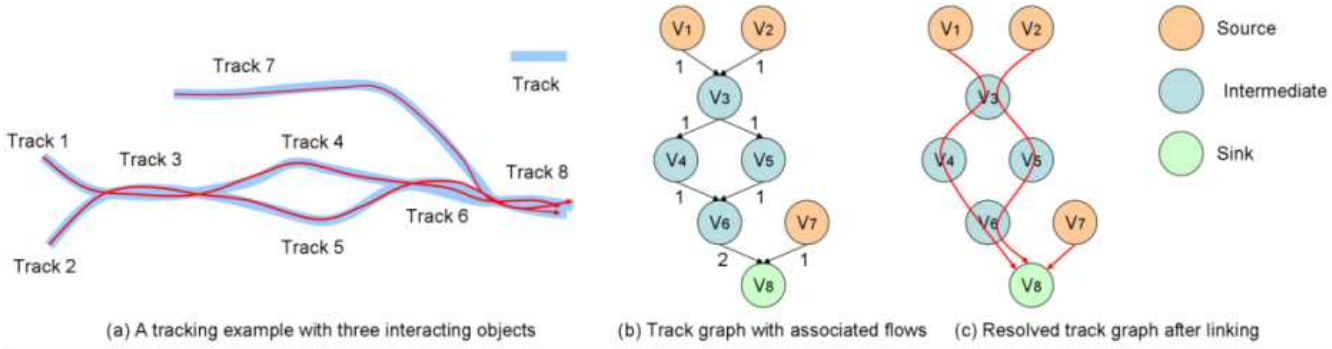

Figure 2. An example of tracking that consists of three interacting objects and eight system-generated tracks (a) and the corresponding track graph (b). The track graph represents two objects that occlude each other for a while, then move apart, then merge again, and finally interact with the third object. The track graph is particularly useful to visualize such frequent track-merging and track-splitting events. Our local or global linking algorithm processes the track graph (b) and produces the resolved graph (c), where each red arrow connects multiple vertices (i.e., tracks) and maintains the identity of the tracked object.

II.1 **Finding a feasible flow:** Starting with the source vertices, keep pushing flow through the graph G until the lower-bound capacity $c(u, v)$ (one in our case) of every edge $e_{u,v}$ is satisfied, which returns a feasible flow f. Determine the residual graph G_f to be the network with capacity $c_f(u, v) = f(u, v) - c(u, v)$.

II.2 **Path-reducing step:** If G_f has a path p from one source node in S to one sink node in T, reduce the edge capacity of G_f along path p by $c_f(p) = \min\{c(u, v)|(u, v) \in p\}$, and subtract $c_f(p)$ units along p from flow f. Repeat this step until no valid path can be found in residual graph G_f. The result flow f is the minimum flow.

3.2. Linking Scheme for Track Graph

We propose local and global linking strategies to processes the track graph. If occlusion occurs for a short period of time or the object features are sufficiently discriminate, we can use a local linking strategy. If occlusion occurs during a long period of time or object features have to be monitored for several frames (e.g., motion smoothness or periodic motion patterns), we need to use a global linking strategy.

3.2.1 Local Linking

If one track can determine its successor without the need to look further we use of the local linking strategy. Local information is thus not passed through the whole graph, and linking can be done efficiently by a series of matchings of bipartite graphs. The vertices on the graph are first sorted according to the initiation time of its corresponding track. The local linking processes each vertex sequentially until all vertices have been matched. By construction, there are only three types of local structures in a track graph, see Fig. 3.

- For a type-(a) structure that represents a merge hypothesis $H_m : \{(\mathcal{T}_{i_1}, \mathcal{T}_{i_2}, ...\mathcal{T}_{i_m}) \vdash \mathcal{T}_k\}$, we extend each individual track with the merged track and smooth the trajectory where the tracks are joined. Then we create the representation $(\mathcal{T}_{i_1}\mathcal{T}_k, \mathcal{T}_{i_2}\mathcal{T}_k, ..., \mathcal{T}_{i_m}\mathcal{T}_k)$.
- For a type-(b) structure that represents a split hypothesis $H_s : \{\mathcal{T}_k \vdash (\mathcal{T}_{j_1}, \mathcal{T}_{j_2}, ...\mathcal{T}_{j_n})\}$, we extend each split track reversely with the merged track and smooth the trajectory at the joint. The tracks are now $(\mathcal{T}_k\mathcal{T}_{j_1}, \mathcal{T}_k\mathcal{T}_{j_2}, ..., \mathcal{T}_k\mathcal{T}_{j_n})$,
- For a type-(c) structure that represents a merge hypothesis immediately followed by a split hypothesis, we first extend each individual track with the merged track as we do for a type-(a) structure, then we search for the best match between two sets of tracks $H_a : \{(\mathcal{T}_{i_1}, \mathcal{T}_{i_2}, ...\mathcal{T}_{i_m}) \vdash (\mathcal{T}_{j_1}, \mathcal{T}_{j_2}, ...\mathcal{T}_{j_n})\}$, which is a bipartite matching problem. The flow $f_{i\to k}$ determines the number of times track $\mathcal{T}_i$ has to be matched, and the flow $f_{k\to j}$ determines the number of times track $\mathcal{T}_j$ has to be matched. The matching cost between a pair of tracks $(\mathcal{T}_i, \mathcal{T}_j)$ depends on the specific application. Once we solve for the best match, we link each track $\mathcal{T}_i$ with its match $\mathcal{T}_j$: $\mathcal{T}_i\mathcal{T}_j$.

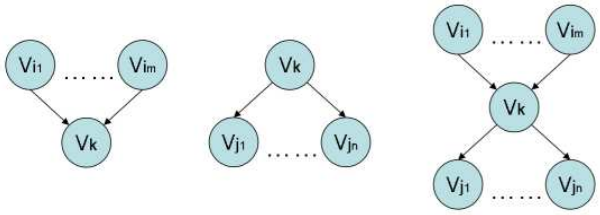

Figure 3. The local structure of track graph. In all three cases, v_k represents a merged track (a) v_k is a sink node; (b) v_k is a source node; (c) v_k is an intermediate node, where the numbers of incoming and outgoing edges might be different, but the flow going through v_k is balanced.

The above procedure repeats until all the hypotheses are processed. Since each linking operation makes a locally

optimal choice, the results of the algorithm are locally optimal.

3.2.2 Global Linking

Global linking may connect several trajectory segments together at the same time. We convert this problem to a generalized set-cover problem as follows.

For a given track graph, we enumerate all possible paths from source set S to sink set T, where each path consists of a sequence $\{v_{i_1} v_{i_1} ... v_{i_p}\}$ of vertices visited. The set of all paths is denoted as P. A weight w_p is associated with a path p that measures the likelihood of the path being a true trajectory or, equivalently, the "cost" of the path. The objective function then is defined as selecting a subset P' of P such that the sum of the costs of all selected paths is minimum. Each vertex $v \in V$ has to be in some path at least t_v times, where t_v is the track-capacity of v computed from the minimum flow during the construction of the graph. Note this is a classic set cover problem when $t_v = 1$ for all vertices. Mathematically, this is equivalent to the following linear/integer programming problem, where x_p is an integer variable to indicate if path p is selected x_p times:

$$\begin{aligned} &\min \sum_p w_p x_p \\ &\text{s. t.} \sum_{p:v\in p} x_p \geq t_v, \forall v \in V \\ &x_p \geq 0 \text{ and } x_p \text{ is integer.} \end{aligned} \tag{1}$$

Given this conversion from a track-linking to a set-cover problem, we use the following deterministic greedy method to solve Eq. 1, which has the same approximation quality as LP relaxation [16], but is much easier to implement:

GREEDY METHOD FOR TRACK SET COVER PROBLEM

Input: A track graph G that has a set V of vertices, representing 2D tracks, with their associated track-capacities t_v, and a set E of edges for merge/split relations.

- Enumerate all possible paths from source S to sink T. Denote such path set as P.
- For each path $p \in P$, compute its weight w_p as the likelihood of p being the true trajectory of some object. Initialize solution set $P' = \emptyset$, $x_p = 0, p \in P$.
- **While** there is some $t_v > 0$,
  1. For each $p \in P$, let U_p be the set of vertices that has not been fully covered on that path, i.e., $t_v > 0$, and define cost $c_p = w_p/|U_p|$.
  2. Choose $p^* = \arg\min_p c_p$. Let $t^* = \min_{v\in U_{p^*}} t_v$. Update $x_{p^*} = x_{p^*} + t^*$. For each $v \in U_{p^*}$, let $t_v = t_v - t^*$. Update $P' = P' \cup p^*$.

Output: A resolved track graph with P' as the set of linked tracks.

The deterministic greedy method to solve the set cover problem has a $O(MN + M^2)$ running time [20], here $M = |V|$ and $N = |P|$. It achieves an approximation ratio of $\mathcal{H}(s)$, where s is the size of the largest set and $\mathcal{H}(n) = \sum_{i=1}^{n} 1/i \approx \log(n)$ is the n-th harmonic number [10].

3.2.3 Linking in Multi-view

The most general scenario of tracklet linking is to link tracklets from multiple views with a global-linking cost. For ease of notation, we here consider only two views, but the method can be extended to an arbitrary number of views. We formulate the multi-view global-linking problem as a *joint*-set-cover problem. Specifically, we generate a track graph for each view independently as $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. For each graph G_i, $i = 1, 2$, we enumerate all valid paths in set P_i. We define a_p and b_q to measure the respective likelihood of path $p \in P_1$ and $q \in P_2$ being true trajectories. Our goal is to choose a subset $P_i' \subseteq P_i$ to achieve a cover on V_i for each view, subject to the additional constraint that enforces that any selected path $p \in P_i'$ has a corresponding path $q \in P_j'$ with matching cost $c_{p,q}$. We seek the solution that achieves the minimum weighted sum. Mathematically, it can be formulated as the following linear/integer programming problem, where $z_{p,q}$ is a binary variable to indicate if a path pair $(p, q), p \in P_1, q \in P_2$ is selected or not:

$$\begin{aligned} \min(&\sum_p a_p \sum_q z_{p,q} + \sum_q b_q \sum_p z_{p,q} + \sum_p \sum_q c_{p,q} z_{p,q}) \\ \text{s. t.} &\sum_{p:u\in p} \sum_q z_{p,q} \geq 1, \forall u \in V_1 \\ &\sum_{q:v\in q} \sum_p z_{p,q} \geq 1, \forall v \in V_2 \\ &z_{p,q} \geq 0 \text{ and } z_{p,q} \text{ is integer} \end{aligned} \tag{2}$$

Proposition 3.1 *The joint-set-cover problem defined in Eq. 2 can be reduced to a standard set cover problem.*

Proof For each pair of sets $p \in P_1, q \in P_2$, we create a joint set $o = p \cup q$ with an associated weight $w = a_p + b_q + c_{p,q}$. The new set of o is denoted as O and the new vertex set as $V = V_1 \cup V_2$. Now we need to find a subset $O' \subseteq O$ that is a cover on V with a minimum weighted sum, which is the standard set cover problem. ∎

Given this proof, we can reuse our greedy approximation algorithm for the multi-view linking strategy. We found that it is not necessary to enforce the multiple assignment constraint (t_v in Eq. 1) in practice, as the across-view matching constraint implicitly determines the number of times each tracklet has to be selected. In case some object does not appear in both views, e.g., set $p \in P_1$ has no matching set $q \in P_2$, we add all pairs (p, q_0) to the joint set O, where

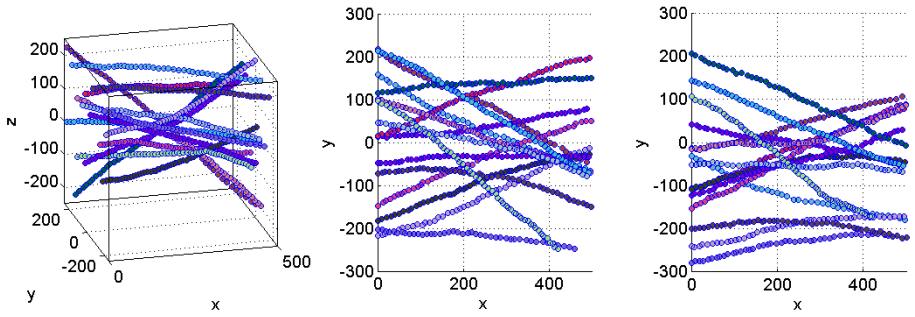

Figure 4. Fifteen sample trajectories in 3D space (left), randomly generated by the simulator, and their images in two views (middle and right) with numerous occlusions. We used matching colors to visualize corresponding trajectories.

$p \in P_1$ and q_0 is a "dummy" placeholder, and assign a large associated matching cost so that these elements have a low priority of being selected.

4. Experiments and Results

We adopted these performance metrics (details in [3]):
Miss Rate (MR): the ratio of misses in the sequence, computed over the number of objects present in all frames,
False Positive Rate (FPR): the ratio of false-positive detections that cannot be matched to any ground-truth trajectories over the number of detections,
Mismatch Rate (MMR): the ratio of the number of track switches over the number of objects present in all frames,
Multiple Object Tracking Accuracy (MOTA): the final score to summarize tracking accuracy = 1-MR-FPR-MMR. We applied smoothing over the label matches between system-generated tracks and ground truth so that MMR estimation became more accurate.

4.1. Simulation Experiment and Results

We randomly generated spheres with 10-unit radii, moving at constant speed in a 500^3-unit 3D space (Fig. 4). The arrival time of each sphere is drawn uniformly from interval $[1, T_{\max}]$ with $T_{\max} = 250$ frames. We created two virtual cameras for viewing the spheres from directions differing by 45^o. The motion model of each sphere is $X^{(t)} = FX^{(t-1)} + W^{(t)}$ and $Z^{(t)} = HX^{(t)} + V^{(t)}$ with 6D state X (3D position and velocity), 2D observation Z (virtual view of sphere), state transition matrix F, projection matrix H, and zero-mean Gaussian noise processes W and V with respective covariance matrices $\mathrm{diag}(1, 1, 1, 0.1, 0.1, 0.1)$, and $\mathrm{diag}(1, 1)$. We generated 6 datasets (D1-D6) with increasing density. Each dataset contained 5 sequences, each with 250 frames per view, resulting in a total of 15,000 test frames. Key statistics of the synthetic data are summarized in Table 1, rows 1–4. Row 4 shows the average number of errors (missed detections, false alarms, and track switches) that correspond to a 0.01 MOTA score.

The track graph representation was constructed by forward-backward nearest-neighbor filtering (Sec. 3.1). All three linking methods used the same set of tracklets from the track graph as input. For the local linking method, the cost of pairing two tracklets was chosen to be the standard deviation of the linear-regression residual over the observed 2D coordinates (assuming that the motion was along a straight line for short periods). For the global linking method, the cost function that measures how likely several tracklets can form a smooth trajectory was evaluated by Kalman smoothing. For the multi-view linking method, the across-view cost function was defined as the reconstruction error according to the (virtual) epipolar geometry. In our implementation of the global linking method by Nillius et al. [11], we followed their recommendation to restrict the dependence between two vertices (here the number of objects involved in an occlusion event and the frequency such events) within a certain time period (here 20 frames).

We measured the performance of each tracking method by the MOTA metric (Table 1, rows 5–8), for which a **small** difference in a score can reveal a **significant** difference in tracking accuracy (see row 4). Not surprisingly, the performance for all methods decreased as the density of objects in the scene increased. Both the global and multi-view linking methods outperformed the local linking strategy. The method by Nillius et al. [11], also global approach, achieved comparable performance but failed to handle very dense scenarios (no reports for D5, D6). It was simply too slow because its state space was too large (even with proposed heuristics [11]). For a vertex with n incoming and n outgoing edges, our global linking method enumerates n^2 paths passing this vertex. In contrast, the method by Nillius et al. [11] must evaluate $n!$ possibilities of matching between incoming and outgoing edges.

The multi-view linking method has a better performance than the global linking method, since it uses additional geometric information, except in the most dense scenario of our simulation (D6), where the size of the proposed joint set cover problem is much larger than each single set cover problems. In this case, the additional benefits that geometric information provided was compromised by the inaccu-

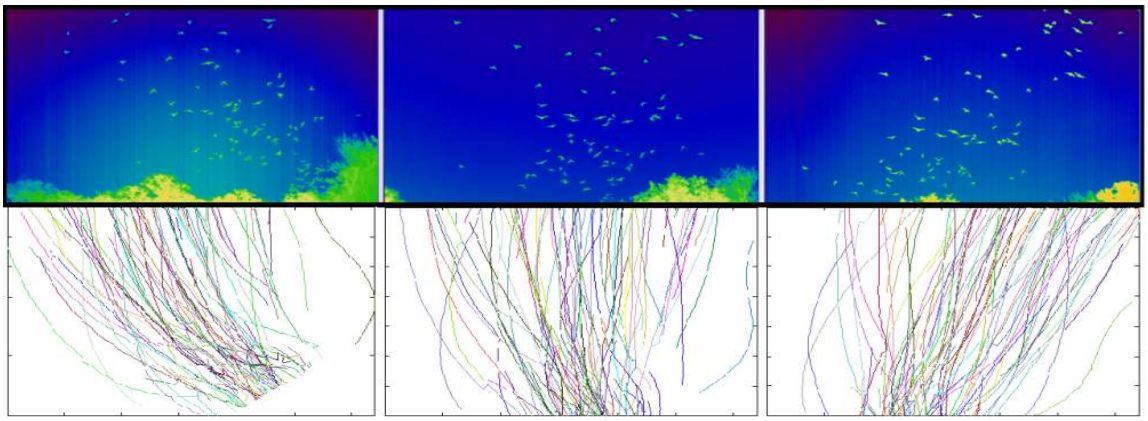

Figure 5. Corresponding infrared video frames from three cameras (top) and system-generated trajectories (bottom) from dataset D3.

racy of the greedy solution. An advantage of the multi-view linking method is that, as a byproduct, it gives the trajectory correspondences between views, which can be further used for 3D path reconstruction.

Table 1. Statistics of Synthetic Dataset and Results for 4 Methods

	D1	D2	D3	D4	D5	D6
Avg. Objs / frame	8.8	17.8	27.1	36.2	45.5	54.6
Max. Objs / frame	16	27	38	51	59	73
Occlusions / frame	0.13	0.60	1.53	2.69	3.87	5.48
# errors, 0.01 MOTA	23	46	70	95	120	145
Nillius et al. [11]	0.92	**0.92**	0.86	0.81	NA	NA
Local Linking	0.90	0.83	0.76	0.68	0.61	0.55
Global Linking	**0.95**	0.91	0.87	**0.85**	**0.81**	**0.80**
Multiview Linking	**0.95**	**0.92**	**0.89**	**0.85**	**0.81**	0.78

4.2. Results on Real Video Data

We tested our track-linking algorithms on real datasets for infrared video analysis of wild animals. We processed videos of the emergence of a colony of Brazilian free-tailed bats from a natural cave in Texas, recorded by three calibrated thermal infrared cameras (Fig. 5). The cameras had been placed at a distance from the cave that allowed capturing of the *entire* group of flying bats from different viewing directions with overlapping fields of views. We did not have sufficient appearance information to distinguish between bats, which look very similar to each other.

We applied background subtraction to detect bats in each image, followed by labeling of connected components. The size of the projection of each bat ranged from 10 to 40 pixels, depending on the distance of the bat to the camera. The position of each bat was located by finding the pixel with the highest intensity value within the connected component. Because of occlusion, a single component might correspond to the overlapping images of multiple bats.

We manually labeled 3 datasets (B1-B3) of different densities, which included about 20, 50, and 100 bats per frame, respectively. Datasets B1 and B3 contained 100 frames for each view; B2 200 frames. For this real dataset, we compared the performance of four track-linking approaches as well as the two classic measurement-level approaches JPDA and MHT (Table 2). We used 5-scanback for MHT and one-scanback for JPDA. We used the same cost functions for the four track-linking methods as for the synthetic data.

Table 2. Tracking Performance for Infrared Video Dataset

	MOTA	MR	FPR	MMR
B1 (11 errors for 0.01 MOTA)				
JPDA	0.926	0.045	0.027	0.002
MHT	0.896	0.081	0.015	0.008
Nillius et al. [11]	**0.985**	0.0074	0.0072	0.0004
Local Linking	0.970	0.015	0.015	0.0
Global Linking	0.945	0.032	0.022	0.001
Multiview Linking	0.944	0.031	0.024	0.001
B2 (58 errors for 0.01 MOTA)				
JPDA	0.886	0.087	0.022	0.005
MHT	0.913	0.072	0.004	0.011
Nillius et al. [11]	0.871	0.093	0.034	0.002
Local Linking	0.933	0.020	0.041	0.006
Global Linking	**0.940**	0.037	0.018	0.004
Multiview Linking	0.913	0.047	0.036	0.004
B3 (70 errors for 0.01 MOTA)				
JPDA	0.871	0.088	0.030	0.011
MHT	0.881	0.103	0.004	0.012
Nillius et al. [11]	0.854	0.103	0.039	0.004
Local Linking	0.910	0.030	0.050	0.010
Global Linking	**0.915**	0.053	0.026	0.006
Multiview Linking	0.875	0.078	0.040	0.007

Our global linking approach resulted in the best overall performance in terms of the MOTA metric for sequences B2 and B3; the global linking approach by Nillius et al. [11] won for the least dense sequence B1. The latter approach provided a globally optimal solution but was time-consuming. In contrast, our greedy algorithm was fast but computed an approximate solution (of an excellent logarithmic approximation ratio).

The performance difference between the global and local linking strategies was not as conclusive as it was in the simulation. A reason may be that some bats had nonlinear flying patterns so that the global cost function, which was based on linear dynamics, was not sufficiently descriptive

for true matches. The multi-view linking approach did not perform as well on the real as on the synthetic data. This may be a result of inaccuracies of camera calibration and errors in the detection step. Nonetheless, it is important to note that the multi-view approach is particularly relevant for imaging situations in which local or global information is sparse, e.g., objects look alike and move in highly nonlinear patterns. In these situations, stereoscopic geometry might be the only useful information to help tracking through occlusion. Finally, we report that our tracklet-level methods had a better performance than the measurement-level methods JPDA and MHT. The traditional formulations of JPDA and MHT do not handle merged measurements, so they tend to have a higher target miss rate (MR) and more track-switch errors (MMR). In contrast, our tracklet-level methods have a higher false positive rate (FPR), as sometimes they tend to "over-explain" the merged measurements.

5. Conclusion

In this paper, we proposed three tracklet-linking strategies that are suitable for occlusion reasoning in different imaging scenarios. The methods process a track-graph representation that models track merging and splitting events. We showed that the objective function in the *local linking approach* is straightforward to optimize, but the formulation is unable to handle frequent or long-term occlusions. The objective function in the *global linking approach* is hard to optimize, since its solution corresponds to the solution of the set-cover problem. We showed that a greedy approximation solution worked well for both synthetic and real data. We also proposed a novel linking formulation for multiple views that not only outputs the trajectories in each view but also provides track-to-track correspondences across views.

In future work, we plan to apply our methods on popular datasets for pedestrian and cell tracking, for which tracklet linking is important. The objective function does not need to be modeled as a linear function; so we will also explore more complicated forms of objective functions that encode group behavior of large numbers of objects.

References

[1] A. Andriyenko and K. Schindler. Globally optimal multi-target tracking on a hexagonal lattice. In *ECCV*, pages 1 – 8, 2010.

[2] Y. Bar-Shalom and T. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[3] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing*, 2008.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.

[5] S. Deb, M. Yeddanapudi, K. Pattipati, and Y. Bar-Shalom. A generalized s-d assignment algorithm for multisensor-multitarget state estimation. *IEEE Trans. AES*, 33:523–538, 1997.

[6] C. Huang, B. Wu, and R. Nevatia. Robust object tracking by hierarchical association of detection responses. In *ECCV*, pages 788–801, 2008.

[7] H. Jiang, S. Fels, and J. Little. A linear programming approach for multiple object tracking. In *CVPR*, pages 1–8, 2007.

[8] Z. Khan, T. Balch, and F. Dellaert. MCMC data association and sparse factorization updating for real time multitarget tracking with merged and multiple measurements. *IEEE Trans. PAMI*, 28:1960 – 1972, December 2006.

[9] K. Li, M. Chen, T. Kanade, E. Miller, L. Weiss, and P. Campbell. Cell population tracking and lineage construction with spatiotemporal context. *Medical Image Analysis*, 12(1):546 – 566, October 2008.

[10] L. Lovasz. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, pages 383 – 390, 1975.

[11] P. Nillius, J. Sullivan, and S. Carlsson. Multi-target tracking: Linking identities using Bayesian network inference. In *CVPR*, 2006.

[12] S. Oh, S. Russell, and S. Sastry. Markov chain monte carlo data association for general multiple target tracking problems. In *the 43rd IEEE Conference on Decision and Control*, pages 1–8, 2004.

[13] A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu. Multi-object tracking through simultaneous long occlusions and split-merge conditions. In *CVPR*, pages 666–673, 2006.

[14] A. B. Poore. Multidimensional assignment formulation of data association problems rising from multitarget and multisensor tracking. *Computational Optimization and Applications*, 3(1):27–57, 1994.

[15] K. Shafique, M. Lee, and N. Haering. A rank constrained continuous formulation of multi-frame multi-target tracking. In *CVPR*, pages 1–8, 2008.

[16] V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[17] Z. Wu, N. I. Hristov, T. L. Hedrick, T. H. Kunz, and M. Betke. Tracking a large number of objects from multiple views. In *ICCV*, Kyoto, Japan, September 2009.

[18] Z. Wu, N. I. Hristov, T. H. Kunz, and M. Betke. Tracking-reconstruction or reconstruction-tracking? comparison of two multiple hypothesis tracking approaches to interpret 3d object motion from several camera views. In *Proceedings of the IEEE Workshop on Motion and Video Computing (WMVC)*, Utah, December 2009.

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

[20] J. Yang and J. Y.-T. Leung. A generalization of the weighted set covering problem. *Naval Research Logistics*, 52:142–149, 2005.

[21] Q. Yu, G. Medioni, and I. Cohen. Multiple target tracking using spatio-temporal Markov Chain Monte Carlo data association. In *CVPR*, pages 1–8, 2007.

[22] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *CVPR*, 2008.