Globally Optimal Solution to Multi-Object Tracking with Merged Measurements

João F. Henriques, Rui Caseiro and Jorge Batista
Institute of Systems and Robotics, University of Coimbra
{henriques,ruicaseiro,batista}@isr.uc.pt

Abstract

Multiple object tracking has been formulated recently as a global optimization problem, and solved efficiently with optimal methods such as the Hungarian Algorithm. A severe limitation is the inability to model multiple objects that are merged into a single measurement, and track them as a group, while retaining optimality. This work presents a new graph structure that encodes these multiple-match events as standard one-to-one matches, allowing computation of the solution in polynomial time. Since identities are lost when objects merge, an efficient method to identify groups is also presented, as a flow circulation problem. The problem of tracking individual objects across groups is then posed as a standard optimal assignment. Experiments show increased performance on the PETS 2006 and 2009 datasets compared to state-of-the-art algorithms.

1. Introduction

Recent years have seen encouraging developments in the automatic detection of objects and isolated events. It remains a challenge, however, to maintain the identity of targets over long spans of time, which requires that not a single mismatch occurs during the whole tracking period. Stable identities are a crucial requirement for the inference of higher-level meaning and events, enabling important applications such as automatic surveillance and security, action recognition and video querying.

Our work improves on several ideas proposed recently in the literature, which formulate tracking as a high-level optimization problem. By making use of global information, as opposed to low-level, local features, this approach promises to solve the drifting and track loss issues of previous methods. In this work, we solve the long-standing issue of properly modeling merge and split events in a global optimization framework, solvable in polynomial time. We also devise a method to identify and track across groups.

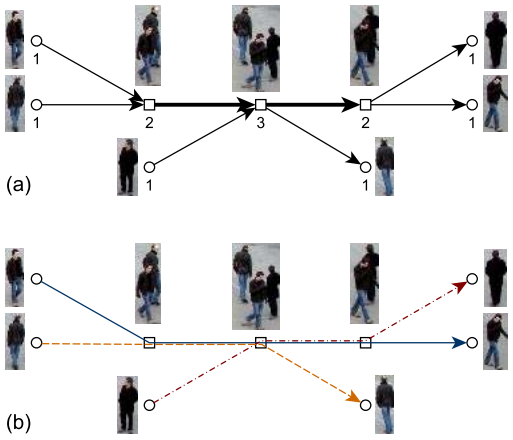

Figure 1. **(a)** A portion of the track graph from PETS 2009 sequence S2.L1, with total flows below each node. The groups subgraph (where flows are greater than 1) is represented with bold arcs and square nodes. **(b)** The individual objects' tracks, after matching them across the groups subgraph.

1.1. Related work

A great research effort has been dedicated to following a single object by keeping a model of low-level features and searching for its new location in each frame [16, 10, 19]. It is hoped that, given a good enough model of appearance and motion, a tracker will always be able to recognize its target. The goals are often contradictory: a model must be highly specific and stable to ignore occlusions and similar objects, but also highly general and adaptable to cope with legitimate changes in appearance. Even though there are probabilistic models for the interactions between objects and the association of objects to detections [7, 17, 9], binding decisions are always made locally in time, in a greedy fashion. A thorough survey on the subject can be found in [22].

The realization that local information sometimes is not enough to make a correct decision led to the development

The authors would like to acknowledge support by BRISA - Autoestradas de Portugal, S.A.. Rui Caseiro acknowledges support by Fundação para a Ciência e Tecnologia through grant SFRH/BD/74152/2010, and João F. Henriques through the grant SFRH/BD/75459/2010.

of *global optimization trackers*, which can operate on larger time scales and make use of high-level reasoning to resolve ambiguities. By posing the problem as a minimum-cost graph matching (optimal assignment or *optimal matching*), efficient methods that run in polynomial time such as the Hungarian algorithm have been used successfully [15, 8, 20]. The key idea is that each detection must be matched to another detection that occurs later in time, and chaining these matches yields tracks. All tracks are optimized simultaneously, and the number of tracks, initializations and terminations are decided as to maximize a well-defined posterior probability. In contrast, greedy strategies can get trapped in local maxima of the objective function.

Since the optimal matching problem is a special case of the minimum-cost flow problem, which in turn can also be solved as a more generic linear program (LP), several authors have used them to define additional restrictions. The LP solution presented in [11] introduces a term that encourages relative positions of objects to remain constant, however it assumes the number of objects to be fixed. A min-cost flow formulation is presented in [24], but since the flows are restricted to values of 0 or 1 it is effectively equivalent to an optimal matching. Their main contribution is an iterative procedure to solve occlusions.

The authors of [13] go further and use Quadratic Boolean Programming, but this class of problems cannot be solved without hypothesis pruning, which may yield sub-optimal solutions. The works [2] and [1] are notable for using min-cost flow and LP to track simultaneously with multiple views. In contrast, we use a single, uncalibrated camera.

Occlusion models to allow matches across occlusions caused by other objects are prevalent among all of these methods, since they optimize one-to-one matches. A different approach is to track objects that may occlude each other as a group, which allows them to be tracked for far longer than existing occlusion models can cope with. One-to-one matches cannot represent objects and groups merging or splitting, so many-to-one and one-to-many matches (*merges* and *splits*) are needed. The authors of [15] proposed an iterative method but it may require solving a number of one-to-one optimal matchings that is exponential in the number of detections. Markov Chain Monte Carlo (MCMC) methods [23] can also incorporate merges and splits.

We should point out that Multiple Hypothesis Tracking [18] and MCMC also attempt to model the joint trajectories of all objects, but cannot guarantee a globally optimal solution in sub-exponential time. It is also fair to note that the greedy trackers referenced earlier are still useful as a computationally cheap way to reduce the search space by pre-computing short track segments or *tracklets*. Optimal matchings can also be used on a frame-by-frame basis as in [21], but from the point of view of a whole video these methods are still considered greedy.

1.2. Outline and main contributions

Given the output of a low-level detection process, we propose a system to obtain the individual tracks of an unknown number of objects.

The main contributions in this paper are as follows:

1. A novel graph structure that allows polynomial algorithms to obtain many-to-one and one-to-many optimal matchings. Previous algorithms only allow one-to-one, or scale exponentially with detections [15].

2. A flow circulation formulation for the problem of counting objects in a track graph, which can be solved in polynomial time. Previous methods have no known bounds and cannot handle all cases [14].

3. A method to resolve individual identities exactly, by matching the individual objects across portions of the track graph where their identities are ambiguous.

The matching algorithm taking into account merges and splits is presented in Section 2. Section 3 describes the method to retrieve object counts, and Section 4 the process of matching across groups. Section 5 discusses the finer details of our implementation. Finally, Section 6 presents experimental results.

2. Optimal matching

In this section we will define the basic maximum a-posteriori (MAP) problem and then extend it to encompass merge and split events. We will then show how to modify the cost matrix of an optimal matching algorithm to solve the extended MAP problem.

2.1. MAP framework

Consider the output of a generic object detector, either a classifier [13] or foreground segmentation [15], for a given video sequence. Tracking by global optimization amounts to linking all the detections together over the whole video, choosing links so that the total probability is maximized; each disjoint string of detections represents the track of a different object.

As a matter of computational efficiency, it's standard practice to pre-link detections with a very high probability of association. The authors of [8] use a set of conservative association rules, and [15] use simple Kalman filters. This yields a set of track segments, or *tracklets*, without any dubious links. The tracklets are then linked by a global optimization process.

A more precise definition of these terms follows. A tracklet T_j is a set of detections, and the set of all tracklets is $\mathcal{T}$. A candidate solution is a set of tracks $\mathcal{S} = \{S_i\}$, and

each track S_i is a set of n_i tracklets[1] $S_i = \{T_j^i\}$. No two tracks may share a tracklet ($S_i \cap S_j = \emptyset$, $\forall S_i, S_j \in \mathcal{S}$), we will refer to this as the *no-overlap restriction*. The optimal solution $\mathcal{S}^*$ is obtained by solving a MAP problem [8],

$$\begin{aligned} \mathcal{S}^* &= \arg\max_{\mathcal{S}} P(\mathcal{S}|\mathcal{T}) = \arg\max_{\mathcal{S}} P(\mathcal{T}|\mathcal{S})P(\mathcal{S}) \\ &= \arg\max_{\mathcal{S}} P(\mathcal{S}) \prod_{T_j \in \mathcal{T}} P(T_j|\mathcal{S}) \end{aligned} \quad (1)$$

subject to the no-overlap restriction. The last step assumes the likelihoods of the tracklets $P(T_j|\mathcal{S})$ are conditionally independent given $\mathcal{S}$. A tracklet T_j that is not in any track is rejected as a false alarm with likelihood P_j^-, and the likelihood of being accepted is P_j^+.

$$P(T_j|S) = \begin{cases} P_j^+ & \text{if } \exists S_i \in \mathcal{S} : T_j \in S_i \\ P_j^- & \text{otherwise} \end{cases} \quad (2)$$

The association prior of a single track $P(S_i)$ is defined as a Markov chain with an initialization term $P_{init}(T_0^i)$, link terms $P_{link}(T_k^i, T_{k+1}^i)$, and a termination term $P_{term}(T_{n_i})$. Assuming tracks to be independent of each other,

$$P(\mathcal{S}) = \prod_{S_i \in \mathcal{S}} P(S_i) \quad (3)$$

$$P(S_i) = P_{init}(T_0^i) \left(\prod_{k=0}^{n_i - 1} P_{link}(T_k^i, T_{k+1}^i) \right) P_{term}(T_{n_i}) \quad (4)$$

These terms are obtained from simple parametric models of the tracklet data, and will be explained in Section 5.

2.2. MAP estimation with merges and splits

In many realistic scenarios, nearby objects cannot be distinguished and will be merged into a single track. This situation is unavoidable with all detection methods but is even more severe with foreground segmentation[2]. This section will address the issue of extending the MAP framework to allow merge and split events.

With merges and splits, representing the solution as a set of disjoint tracks $\mathcal{S}$ is no longer appropriate. A more general representation is the *track graph* used in [14]. The track graph $\mathcal{G}_{link}$ has the tracklets $\mathcal{T}$ as the nodes set, and a set of directed arcs $\mathcal{A}_{link}$. Each track S_i is equivalent to a series of arcs connecting each tracklet to the next in the same track. Formally, $\mathcal{A}_{link} = \bigcup_{S_i \in \mathcal{S}} A_i$, with $A_i = \{(T_0^i, T_1^i), (T_1^i, T_2^i), \ldots, (T_{n_i-1}^i, T_{n_i}^i)\}$. Note that direction is important; an arc points *from* T_i *to* T_j.

[1] We use T_j^i to index the jth tracklet of track S_i, and T_j when the associated track is not relevant.

[2] The standard formulation assumes tracklets represent single objects. Occlusions are handled as missing tracklets, making association difficult.

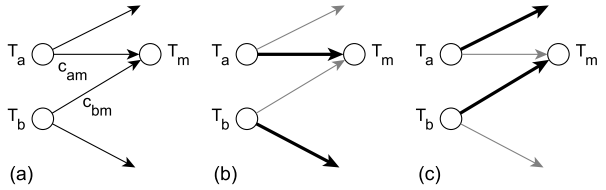

Figure 2. **(a)** Two tracklets, T_a and T_b, can link to tracklet T_m, but not at the same time (merge), due to the no-overlap restriction. **(b, c)** The two solutions (bold links).

The MAP problem from Eq. 1 is maintained, since $\mathcal{S}$ is equivalent to $\mathcal{A}_{link}$. Let $\mathcal{T}_{init}$ and $\mathcal{T}_{term}$ be the set of tracklets that initiate and terminate tracks, respectively. Combining Eq. 3 and 4, and rearranging the factors, yields

$$P(\mathcal{A}_{link}) = P_{inits}\, P_{terms}\, P_{links} \quad (5)$$

where

$$P_{inits} = \prod_{T_i \in \mathcal{T}_{init}} P_{init}(T_i) \quad (6)$$

$$P_{terms} = \prod_{T_i \in \mathcal{T}_{term}} P_{term}(T_i) \quad (7)$$

$$P_{links} = \prod_{(T_i, T_j) \in \mathcal{A}_{link}} P_{link}(T_i, T_j) \quad (8)$$

We can now define an augmented graph $\mathcal{G}$ that includes merge and split arcs, $\mathcal{A} = \mathcal{A}_{link} \cup \mathcal{A}_{merge} \cup \mathcal{A}_{split}$. A pair of merge arcs links two tracklets to a single tracklet, while a pair of split arcs links one tracklet to two. This is unlike the arcs in $\mathcal{A}_{link}$, since they represent a disjoint set of tracks (by the no-overlap restriction on $\mathcal{S}$). Denote the set of tracklets that arcs in $\mathcal{A}_{merge}$ link to as $\mathcal{T}_{merge}$, and tracklets that arcs in $\mathcal{A}_{split}$ link from as $\mathcal{T}_{split}$. Then we can define a more broad prior with additional terms as

$$P(\mathcal{A}) = P_{inits}\, P_{terms}\, P_{links}\, P_{merges}\, P_{splits} \quad (9)$$

$$P_{merges} = \prod_{T_i \in \mathcal{T}_{merge}} P_{merge}(T_i) \quad (10)$$

$$P_{splits} = \prod_{T_i \in \mathcal{T}_{split}} P_{split}(T_i) \quad (11)$$

assuming merges and splits are conditionally independent given the tracklets that interact in each merge or split. Section 5 explains how we compute $P_{merge}(T_i)$ and $P_{split}(T_i)$, the likelihoods of these events. Without any merge or split arcs, this reduces to the previous formulation.

2.3. Global optimization

We now turn to optimal matching to solve this problem. A set of candidate links and their respective costs can be encoded as an $n \times n$ matrix $\overline{C} = [\bar{c}_{ij}]$, with n the number

of tracklets and $\bar{c}_{ij}$ the cost of linking the tracklet T_i to T_j. Costs may be infinite, expressing impossible links, otherwise they are usually obtained as $\bar{c}_{ij} = -\log P_{link}(T_i, T_j)$.

Optimal matching methods such as the Hungarian algorithm [12] find the assignment matrix $X = [x_{ij}]$ that minimizes the total cost

$$\begin{array}{rl} X^* = & \arg\min_X \sum_{i,j} \bar{c}_{ij} x_{ij} \\ \text{subject to:} & \sum_i x_{ij} = 1, \forall j \\ & \sum_j x_{ij} = 1, \forall i \end{array} \tag{12}$$

with $i, j \in \{1, \ldots, n\}$, and $x_{ij} \in \{0, 1\}$. The output X^* is the adjacency matrix of the track graph $\mathcal{G}_{link}$. These equations model the no-overlap restriction.

As shown in [8], $\overline{C}$ can be replaced by an augmented $2n \times 2n$ matrix C with initialization and termination blocks

$$C = \begin{bmatrix} C_{link} & C_{term} \\ C_{init} & \mathbf{0}_{n\times n} \end{bmatrix} \tag{13}$$

The $n \times n$ block $C_{link} = [c_{ij}]$ incorporates the link probabilities and tracklet likelihoods

$$c_{ij} = \begin{cases} -\log P_i^-, & \text{if } i = j \\ -\log \left[\sqrt{P_i^+} P_{link}(T_i, T_j) \sqrt{P_j^+} \right], & \text{otherwise} \end{cases} \tag{14}$$

Let $\text{diag}_\infty(e_k)$ be an $n \times n$ matrix where diagonal elements are e_k and off-diagonal elements are infinite, then the initialization and termination blocks are defined as

$$\begin{aligned} C_{init} &= \text{diag}_\infty(c_{init,k}),\ c_{init,k} = -\log \left[P_{init,k} \sqrt{P_k^+} \right] \\ C_{term} &= \text{diag}_\infty(c_{term,k}), c_{term,k} = -\log \left[P_{term,k} \sqrt{P_k^+} \right] \end{aligned} \tag{15}$$

Any solution X corresponds univocally to a set of tracks[3] $\mathcal{S}$, and it can be verified that the corresponding total cost is $-\log P(\mathcal{S}|\mathcal{T})$. A proof of this equivalence is given in [8].

This formulation solves the MAP problem from Section 2.1 without merges and splits. Fig. 2-a illustrates two tracklets, T_a and T_b, that can be linked to another T_m. Alternative links are also represented. Due to the no-overlap restriction, either T_a is linked to T_m or T_b to T_m, but not both. The two solutions involving T_m are shown in Fig. 2b-c.

2.4. Solution with merges and splits

Considering the previous situation, when there is a chance that T_a and T_b merge into T_m (Section 5 discusses how these hypotheses are generated), we would like to allow a third possibility, a merge, where all three are linked

[3] Taking the upper-left $n \times n$ block of the solution as the adjacency matrix of $\mathcal{G}_{link}$.

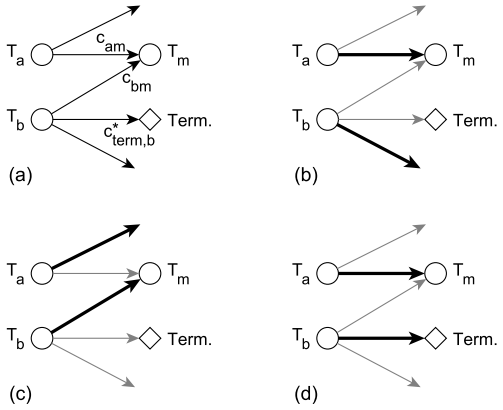

Figure 3. **(a)** The same subgraph of Fig. 2, with T_b linked to a virtual termination node (its termination cost is modified to $c^*_{term,b}$). **(b-d)** The three solutions (bold links); **(d)** models a merge event.

with cost $c_{merge,m} = -\log P_{merge}(T_m)$. A heuristic procedure was presented in [15], but we arrived at an exact solution, with some restrictions.

A merge event will be represented as T_a linking to T_m, and T_b terminating (as shown in Fig. 3-d). The graph needs to be modified so that:

1. The total cost of linking T_a to T_m and terminating a track at T_b is the cost of the merge:

$$c_{am} + c^*_{term,b} = c_{merge,m} \tag{16}$$

2. If a track is terminated at T_b, then T_a must link to T_m.

The first condition ensures a correct total cost, and implies that the termination cost of T_b must be $c^*_{term,b} = c_{merge,m} - c_{am}$. The second condition makes sure that T_b can only be terminated in a merge event, and not independently. It requires that we disable all links to T_m from tracklets other than T_a and T_b. This way, if T_b does not link to T_m (because it terminates or links to another tracklet), then T_a must link to T_m, by the restrictions in Eq. 12. The modified graph can be seen in Fig. 3-a, and the three possible solutions in Fig. 3b-d.

Splits are modeled in exactly the same way, by reversing the direction of the links (track termination is replaced with track initialization).

We will now show how this relates to the optimal matching, by defining a modified $2n \times 2n$ cost matrix $C^* = [c^*_{ij}]$. For each merge hypothesis, let $T^1_{merge,k}$ and $T^2_{merge,k}$ be the two tracklets that may merge into T_k. Conversely, for each split hypothesis, the two tracklets that split from T_k are $T^1_{split,k}$ and $T^2_{split,k}$. Then fulfilling the first condition amounts to modifying the initialization and termination costs from Eq. 15 to

$$c^*_{term,k} = \begin{cases} c_{merge,k} - c_{ak} & \text{if } T_k \in \mathcal{T}_{merge},\ T_a = T^1_{merge,k} \\ c_{term,k} & \text{otherwise} \end{cases}$$

$$c^*_{init,k} = \begin{cases} c_{split,k} - c_{ak} & \text{if } T_k \in \mathcal{T}_{split},\ T_a = T^1_{split,k} \\ c_{init,k} & \text{otherwise} \end{cases} \tag{17}$$

The elements of the updated matrix can then be obtained by making the costs of the disabled arcs infinite

$$c^*_{ij} = \begin{cases} \infty & \text{if } T_j \in \mathcal{T}_{merge} \wedge T_i \notin \left\{T^1_{merge,j}, T^2_{merge,j}\right\} \\ \infty & \text{if } T_i \in \mathcal{T}_{split} \wedge T_j \notin \left\{T^1_{split,i}, T^2_{split,i}\right\} \\ c_{ij} & \text{otherwise} \end{cases} \tag{18}$$

2.4.1 Practical considerations

This strategy gives the optimization algorithm the choice of enabling or not the proposed merge and split hypotheses, turning a one-to-one optimal matching into a many-to-many optimal matching. However, modeling these events requires some compromises, so we explain our choices here.

A track can no longer terminate at $T^2_{merge,k}$ independently of the merge; the same applies to $T^2_{split,k}$ initiating a track independently of the respective split. In practice such a situation would be very difficult to distinguish from the merge or split. We found that modeling such a rare event is not worth the additional complexity.

Also, when there is a merge hypothesis at T_k, no tracklets can link to it except $T^1_{merge,k}$ and $T^2_{merge,k}$. But the fact that there is a merge hypothesis at all means that $T^1_{merge,k}$ and $T^2_{merge,k}$ are spatially very close to T_k (with the criteria of Section 5). This makes it highly unlikely that these two tracklets must be ignored and some other, farther away tracklet should be linked to T_k, so disabling other links is justified. The same reasoning applies to split hypotheses.

3. Object counts as a flow circulation problem

Having made the associations through merges and splits, it is obvious that some links may represent the transit of more than one object: for example, after a merge, two or more objects are being tracked in a single tracklet. Lone objects can be tracked unambiguously but as soon as a group is formed, there is uncertainty in matching each single object that joins the group to each single object that leaves it. Nevertheless, modeling this uncertainty is a tremendous advantage when ambiguities persist for long periods of time.

In order to isolate groups, we must count the number of objects passing through each link. In [14], an iterative scheme is proposed to count objects in a track graph, but the authors admit that the algorithm can get trapped in an inconsistent state and so cannot process some portions of their test video sequence.

We present a simple solution, posing the problem as a minimum-cost flow circulation, for which algorithms of polynomial complexity do exist (e.g., [5]). Modeling the number of objects as a flow is a natural fit since standard network restrictions ensure the number of objects remains constant; source arcs at track initializations produce flow and sink arcs at track terminations consume it.

Let $\mathcal{J}$ be the set of all arc indexes in a graph, and $\mathcal{I}$ the set of indexes of all nodes. For a node $i \in \mathcal{I}$, $in(i)$ and $out(i)$ denote the indexes of incoming and outgoing arcs in the graph. For each arc $j \in \mathcal{J}$, lower and upper flow bounds are given by l_j and u_j, respectively, and the cost per unit flow is $\hat{c}_j$. Solving the min-cost flow circulation of Eq. 19 yields the optimal arc flows $f^* = \left(f^*_j | j \in \mathcal{J}\right)$.

$$\begin{aligned} f^* \quad = \quad & \arg\min_f \sum_{j \in \mathcal{J}} \hat{c}_j f_j \\ \text{subject to:} \quad & \sum_{j \in out(i)} f_j - \sum_{j \in in(i)} f_j = 0,\ \forall i \in \mathcal{I} \\ & l_j \le f_j \le u_j,\ \forall j \in \mathcal{J} \end{aligned} \tag{19}$$

To solve the object counts problem, we form a graph where each unit of flow represents the transit of one object. We use the track graph from Section 2, adding a source or sink arc to each node labeled as a track initialization or termination, respectively. Upper bounds are unused ($u_j = \infty, \forall j \in \mathcal{J}$). Lower bounds are 1, requiring each arc, including source and sink arcs, to carry at least one object ($l_j = 1$). Costs encourage more flow through larger detections: $\hat{c}_j = \max(L_i/S_i, L_k/S_k)$, where arc j links tracklet T_i to T_k, L_i is the length of T_i (number of detections), and S_i the average size of the detections in T_i.

After computing the optimal solution with the Edmonds-Karp algorithm [5], the optimal flows through the arcs (f^*_j) reveal the number of objects that traverse each link. Fig. 1-a illustrates the flows on a portion of a track graph obtained from a real test sequence. The total flow (sum of incoming flows[4]) is shown below each node.

4. Recovering object identities

As explained in Section 3, counting objects is necessary in order to isolate groups of objects. With this knowledge it is now possible to match single objects across groups.

We will refer to the subgraph of $\mathcal{G}$ that represents groups as the *groups subgraph*, $\mathcal{G}_{group}$. It can be constructed by taking the subset of arcs and nodes with flow greater than 1:

[4] By Eq. 19, the sum of incoming and outgoing flows through a node must be equal so either definition of total flow is correct.

$$\mathcal{T}_{group} = \left\{ T_i \mid \sum_{j \in in(i)} f_j^* > 1,\, T_i \in \mathcal{T} \right\} \quad (20)$$
$$\mathcal{A}_{group} = \{(T_i, T_j) \mid f_k^* > 1,\, k \in out(i) \wedge k \in in(j)\}$$

We can now define the isolated objects to be matched. Single-object tracklets that enter a group have an outgoing arc to $\mathcal{G}_{group}$, while those that leave a group have an incoming arc from $\mathcal{G}_{group}$:

$$\mathcal{T}_{in} = \{T_i \mid \exists (T_i, T_k) \in \mathcal{A},\, T_i \notin \mathcal{T}_{group},\, T_k \in \mathcal{T}_{group}\}$$
$$\mathcal{T}_{out} = \{T_i \mid \exists (T_k, T_i) \in \mathcal{A},\, T_i \notin \mathcal{T}_{group},\, T_k \in \mathcal{T}_{group}\} \quad (21)$$

Fig. 1-a illustrates these notions: $\mathcal{T}_{group}$ is depicted as square nodes, $\mathcal{A}_{group}$ as bold links, the three round nodes to the left represent $\mathcal{T}_{in}$ and the remaining round nodes $\mathcal{T}_{out}$.

Considering whether two tracklets $T_i \in \mathcal{T}_{in}$ and $T_j \in \mathcal{T}_{out}$ can be matched across groups, they may represent the same object only if T_j is reachable from T_i using a directed path in $\mathcal{G}_{group}$ (and the two arcs that link T_i and T_j to $\mathcal{G}_{group}$). We use this information to build a final cost matrix to find the optimal matching between them. Let the number of tracklets in $\mathcal{T}_{in}$ and $\mathcal{T}_{out}$ be r and s, respectively, then the elements of the $r \times s$ matrix $C' = \left[c'_{ij}\right]$ are given by:

$$c'_{ij} = \begin{cases} -\log P_{link}(T_i, T_j) & \text{if } T_j \text{ is reachable from } T_i \\ \infty & \text{otherwise} \end{cases} \quad (22)$$

This stage allows matches across potentially very long ranges. The link likelihoods $P_{link}(T_i, T_j)$ are calculated in the same way as for the short-range links in Section 2.

Since nodes in different connected components of $\mathcal{G}_{group}$ are, by definition, not reachable from each other, a convenient optimization is to find connected components first and treat them separately. This expresses the intuitive notion that matching objects across non-interacting groups can be done independently.

After obtaining the optimal matching with the Hungarian Algorithm, we finally have the complete tracking history of each object, as they enter and exit several groups. For the practical purpose of tracking an object within a group (i.e., knowing exactly which tracklets in a group are associated with the object) we simply find the shortest path in $\mathcal{G}_{group}$ from $T_i \in \mathcal{T}_{in}$ to $T_j \in \mathcal{T}_{out}$. If $r \neq s$ then one or more objects must have entered or exited the scene while inside the group, which can happen when a group of objects enter or exit the scene together. In this case, some (or all) rows or columns of C' have no match. These correspond to the entering or exiting objects, respectively, and instead we find the shortest path to the closest tracklet where a track initialization or termination occurs.

5. Implementation details

Although the focus of this work is on the optimization framework, we present here the detection and tracklet generation processes, as well as the probabilistic models used to describe low-level features. Though there are numerous alternatives in the literature reviewed in Section 1.1, our main goal was to balance simplicity with good enough performance. It should be noted that the optimization framework is fairly independent of these particular choices.

Object detection: For each sequence a simple background image is learned as the per-pixel medoid of 100 sample images [4]. Segmentation is performed by a two-label graph-cut algorithm [3] with likelihoods proportional to the color distances between image and background. Connected components of the segmented image represent detections.

Tracklet generation: We use the simple data association proposed in [15], summarized here. Any unassociated detections initialize Kalman filters with position, size and velocity states. Every frame, each Kalman filter is updated and associated with a single detection inside a validation gate. If there are no detections, or more than one in the validation gate, that tracker is terminated. Also, if more than one tracker is associated with the same detection, they are terminated. The detections associated to each tracker constitute a tracklet. This conservative set of rules ensures that tracklets reject any dubious associations.

Data likelihoods: The link likelihoods we use were proposed in [8], except we compare appearances using the Region Covariance Matrix dissimilarities of [16] instead of histogram distances. Link likelihoods take into account motion and appearance. Let $\mathcal{N}(x, \Sigma)$ be a zero-mean Gaussian with the (possibly 1×1) covariance matrix Σ evaluated at x, then the likelihood of matching tracklets T_i and T_j is:

$$P_{link}(T_i, T_j) = \mathcal{N}(d_{ij}^a,\, \Sigma_a)\, \mathcal{N}(d_{ij}^m,\, \delta_t \Sigma_m)\, \mathcal{N}(d_{ji}^m,\, \delta_t \Sigma_m) \quad (23)$$

where d_{ij}^a is the appearance dissimilarity between T_i and T_j, d_{ij}^m is the difference between the predicted Kalman state of T_i and the state of the closest detection (in time) in T_j, and δ_t is their time difference. The covariances Σ_a and Σ_m are estimated from training data. Since the motion covariance is proportional to time, long-range links rely less on this component. Links can only be made forward in time ($\delta_t \leq 0 \Rightarrow P_{link}(T_i, T_j) = 0$). Additionally, in the first stage (Section 2) only short-range links are considered ($\delta_t \geq 3 \Rightarrow P_{link}(T_i, T_j) = 0$).

Making use of the observation that objects usually appear or disappear near the edges of the image, we calculate the track initialization likelihoods as $T_{init}(T_i) = \mathcal{N}(d_i^e, \Sigma_e)$, where d_i^e is the distance of the first detection of T_i to the closest edge of the image, and Σ_e is a fixed parameter. Termination likelihoods $T_{init}(T_i)$ are estimated

Method	Metric	PETS'09	PETS'06		
		S2-L1	S1-T1	S3-T7	S4-T5
Multi-match	MOTA	0.966	0.785	0.816	0.883
	Mismatches	10	16	4	7
	Precision	0.985	0.882	0.847	0.932
	Recall	0.986	0.908	0.997	0.954
1-to-1 match	MOTA	0.806	0.643	0.775	0.650
	Mismatches	18	28	19	13
	Precision	0.915	0.845	0.869	0.896
	Recall	0.900	0.785	0.920	0.738

Table 1. Evaluation results for both methods. The PETS'09 MOTA score of the baseline algorithm (1-to-1 match) is in line with most contestants of the PETS'09 and '10 workshops [6]. The proposed algorithm (multi-match) scored higher than every contestant.

the same way, but using the last detection of T_i.

Finally, the likelihood of T_i being accepted or rejected are, respectively, $P_i^+ = (1-\beta)^{|T_i|}$ and $P_i^- = \beta^{|T_i|}$, with $|T_i|$ the number of detections in T_i and β the expected failure rate of the detector, which we set to 10^{-5}.

Merges and splits: The hypothesis of tracklets T_a and T_b merging into T_m is considered if they occur within a few frames of each other (in our implementation, 2 frames) and obey the area overlap criteria:

$$A(T_a \cap T_m) \geq \alpha A(T_m) \ \wedge \ A(T_b \cap T_m) \geq \alpha A(T_m) \quad (24)$$

where $A(T_m)$ is the area of the bounding box of the first detection of T_m, and $A(T_i \cap T_m)$ is the area of the intersection of the bounding boxes of the last detection of T_i and the first detection of T_m. We set $\alpha = 0.5$. Split hypothesis are obtained in the same way but with the time axis reversed.

For each merge hypothesis, consider the pixels of both T_a and T_b jointly as if they were a single tracklet $\hat{T}_{ab}$. Then the likelihood of the merge is the chance of this virtual tracklet linking to T_m; that is, $P_{merge}(T_m) = P_{link}(T_m, \hat{T}_{ab})$. Split likelihoods $P_{split}(T_m)$ are obtained in exactly the same way.

6. Experiments

We evaluated the proposed system on challenging videos from two publicly available datasets, PETS 2006 and PETS 2009. Both datasets provide multiple views, but we used a single uncalibrated camera: view 4 and view 1, respectively. The sequence S2-L1 from PETS 2009 is intended as a benchmark for multiple object tracking methods [6] and shows a scripted outdoors scene with several pedestrians crossing each other, totaling 795 frames. The PETS 2006 videos were recorded at a train station and represent a typical surveillance scenario with complex interactions, with a total of 8443 frames over 3 sequences.

Our system is better equipped to handle undersegmentation (groups) than oversegmentation. Since PETS 2009 S2-L1 has an occluding lightpost with a large sign that results in oversegmentation of groups in the middle of the screen, the likelihood of a pixel in this region being considered foreground is taken as the weighted average of its neighborhood with a gaussian kernel of $\sigma = 6$. This results in detections being connected across these pixels. Finding these regions automatically will be the subject of future work. Still images of the results are shown in Fig. 4.

For a performance comparison, we use the *same* system as a baseline, but with merges and splits disabled. It is a state-of-the-art global optimization tracker, which relies on one-to-one matches. Instead of forming groups, it handles inter-object occlusions as long range matches, so we allow links with large time gaps ($\delta_t \geq 60 \Rightarrow P_{link}(T_i, T_j) = 0$).

The comparative results with ground truth annotations every 3 frames are summarized in Table 1. We considered every moving person in the ground truth, as well as bags that separate from their owners, but not people who are sitting down through the whole video.

For a quantitative analysis we calculated the Multiple Object Tracking Accuracy [6], a metric widely used in the literature in recent years. A score of exactly 1 represents zero errors. We also report the raw number of identity mismatches. Finally, precision and recall were calculated. They do not take mismatches into account so are more indicative of the performance of the first two stages.

Most errors are caused by mistakes of the foreground segmentation: oversegmentation, small movements of sitting people, and pigeons in PETS 2006. Situations where high-level inference is not informative and people are very similar in appearance can cause mismatches in PETS 2009.

As expected, the use of high-level reasoning about identity ambiguities is a tremendous advantage. One-to-one matching handles occlusions by matching across them, so it performs well in simple cases such as objects crossing. However, large time gaps result in hundreds of possible matches, which increase the likelihood of errors in case of ambiguity. Reasoning about group formation severely restricts the matches to those that are physically possible, which results in improved performance in the most difficult cases, such as people traveling together for long periods.

Our framework outperforms all the state-of-the-art systems evaluated in the PETS'09 and PETS'10 workshops, as reported in [6] (the top contestant's MOTA was 0.960).

7. Conclusion

In this work we propose a new method that allows the multiple-matching tracking problem to be solved as an ordinary one-to-one matching. We have also presented a rigorous treatment of the formation of groups, and tracking objects across them. Experiments on challenging videos illustrate the advantage of considering merged measurements. We expect this framework to enable future systems to handle scenes of increasingly complex situations.

Figure 4. Tracking results for PETS'09 S2-L1 (first row) and PETS'06 S1-T1-C (second row). Identities are color-coded so this image is best viewed in color. Objects in a group are identified by nested outlines with their respective colors. Trails show the short-term trajectory of each object. Note that sudden changes in position are not tracking errors, but due to merged objects sharing the same position when in a group. Even though most tracks go through complicated interactions, reasoning about the formation of groups allows the tracker to recover their identities.

References

[1] A. Andriyenko and K. Schindler. Globally optimal multi-target tracking on a hexagonal lattice. In *ECCV*, 2010. 2

[2] J. Berclaz, F. Fleuret, and P. Fua. Multiple object tracking using flow linear programming. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (Winter-PETS)*, 2009. 2

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE-TPAMI*, 26(9):1124–1137, 2004. 6

[4] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE-TPAMI*, pages 1337–1342, 2003. 6

[5] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19:248–264, 1972. 5

[6] A. Ellis and J. Ferryman. PETS2010 and PETS2009 evaluation of results using individual ground truthed single views. 2010. 7

[7] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8(3):173–84, 1983. 1

[8] C. Huang, B. Wu, and R. Nevatia. Robust object tracking by hierarchical association of detection responses. In *ECCV*, 2008. 2, 3, 4, 6

[9] J. Huang, C. Guestrin, and L. Guibas. Fourier theoretic probabilistic inference over permutations. *The Journal of Machine Learning Research*, 10:997–070, 2009. 1

[10] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *IJCV*, 29(1):5–8, 1998. 1

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

[12] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955. 4

[13] B. Leibe, K. Schindler, and L. V. Gool. Coupled detection and trajectory estimation for multi-object tracking. In *IEEE-ICCV*, 2007. 2

[14] P. Nillius, J. Sullivan, and S. Carlsson. Multi-target tracking-linking identities using bayesian network inference. In *IEEE-CVPR*, 2006. 2, 3, 5

[15] A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu. Multi-Object tracking through simultaneous long occlusions and Split-Merge conditions. In *IEEE-CVPR*, 2006. 2, 4, 6

[16] F. Porikli, O. Tuzel, and P. Meer. Covariance tracking using model update based on means on riemannian manifolds. In *IEEE-CVPR*, 2006. 1, 6

[17] W. Qu, D. Schonfeld, and M. Mohamed. Real-time interactively distributed multi-object tracking using a magnetic-inertia potential model. In *IEEE-ICCV*, 2005. 1

[18] D. B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979. 2

[19] S. Stalder, H. Grabner, and L. V. Gool. Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In *IEEE-ICCV*, 2009. 1

[20] J. Xing, H. Ai, and S. Lao. Multi-Object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses. In *IEEE-CVPR*, 2009. 2

[21] M. Yang, F. Lv, W. Xu, and Y. Gong. Detection driven adaptive multi-cue integration for multiple human tracking. In *IEEE-ICCV*, 2009. 2

[22] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13–es, 2006. 1

[23] Q. Yu and G. Medioni. Multiple-Target tracking by spatiotemporal monte carlo markov chain data association. *IEEE-TPAMI*, 31(12):2196–2210, 2009. 2

[24] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *IEEE-CVPR*, 2008. 2