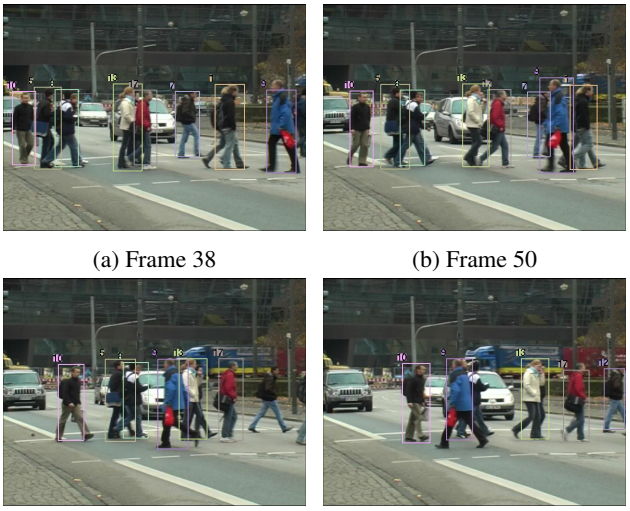
Multi-target Tracking by Lagrangian Relaxation to Min-Cost Network Flow

Asad A. Butt and Robert T. Collins
The Pennsylvania State University, University Park, PA. 16802, USA
{asad,rcollins}@cse.psu.edu

Abstract

We propose a method for global multi-target tracking that can incorporate higher-order track smoothness constraints such as constant velocity. Our problem formulation readily lends itself to path estimation in a trellis graph, but unlike previous methods, each node in our network represents a candidate pair of matching observations between consecutive frames. Extra constraints on binary flow variables in the graph result in a problem that can no longer be solved by min-cost network flow. We therefore propose an iterative solution method that relaxes these extra constraints using Lagrangian relaxation, resulting in a series of problems that ARE solvable by min-cost flow, and that progressively improve towards a high-quality solution to our original optimization problem. We present experimental results showing that our method outperforms the standard network-flow formulation as well as other recent algorithms that attempt to incorporate higher-order smoothness constraints.



(a) Frame 38 (b) Frame 50

(c) Frame 68 (d) Frame 78

Figure 1: Example result of our algorithm using high order motion model on the TUD sequence [1]. Track labels remain unchanged after targets occlude each other.

1. Introduction

Multi-frame, multi-target tracking is a significant and challenging problem. We work within the paradigm of detect-then-track, where an object detector is run on each frame to hypothesize objects of interest, followed by a data association stage to link detections into multi-frame trajectories. This second, multi-frame data association stage is of particular interest, as it is a combinatorial optimization problem of significant complexity. Indeed, except for limited special cost functions that factorize into purely pairwise terms, the multi-frame assignment problem is NP-hard. Developing multi-frame search algorithms that yield good quality approximate solutions in polynomial running time has therefore become a problem of considerable research interest in the field.

Early approximation methods proposed greedy bipartite data association on a frame-by-frame basis [16] to extend a gradually lengthening set of trajectories over time. The two frame bipartite assignment problem, also known as the linear assignment problem, can be solved exactly in polynomial time by methods such as the Kuhn-Munkres (Hungarian) algorithm. However, these one-pass greedy algorithms do not work well when there is target interaction or occlusion in a scene [17]. The same can be said for recursive filtering approaches such as the Kalman filter or particle filter, which have been well-studied in the tracking literature for single target tracking [3]. Such trackers do not perform well in multi-target settings, having a tendency to "jump" between similar targets that pass near each other, resulting in identity swap errors. Another drawback is that decisions, once made, cannot be undone when future information shows them to be suboptimal.

More recent methods have attempted to find globally optimal solutions across the entire sequence by creating network flow graphs [2, 13, 20] or by using iterative hierarchical methods to link tracklets [11, 18, 19]. Network flow formulations, in particular, can be solved optimally and efficiently by min-cost flow algorithms. However, a network

graph only contains pairwise edges between observations, thus cost information evaluating the quality of a trajectory must be able to be factored into the product/sum of pairwise costs on each frame-to-frame link. This limits evaluation of geometric track quality to terms based only on distance traveled between frames, e.g. shortest paths, but does not allow for specifying local path smoothness constraints that are functions of three or more nodes. An attempt to capture these higher-order smoothness constraints has motivated recent work [6, 7]. It is also the motivation for our work. In particular, we develop a graph formulation that allows for encoding constant velocity constraints to evaluate path smoothness over three adjacent frames. It is important to understand that our use of "constant velocity" constraints does not imply that we constrain objects to move in a straight line at constant speed through the image, which of course objects like pedestrians do not do. Our approach applies *piecewise* constant velocity, evaluated over adjacent subwindows of three frames, as a type of smoothness constraint that penalizes hypothesized paths that locally have high curvature or large changes in speed.

In the remaining sections we discuss related work (Section 2), present our graph formulation and Lagrangian relaxation solution method (Sections 3 and 4), and provide experimental results (Section 5). The paper concludes with a summary and discussion of future work.

2. Related Work

Recent approaches have formulated multi-frame, multi-target data association as a network flow problem [2, 13, 20]. Candidate object matches in consecutive frames form nodes and edges in a graph G, with edge costs based on pairwise appearance and distance relations. The number and best set of trajectories through the graph can be solved efficiently using min-cost flow to yield the globally optimal solution. Zhang et al. [20] create a network where two nodes are assigned to each detection and the link between these nodes is weighted by the probability that the detection is part of the solution. Links between nodes representing detections from consecutive frames are weighted by the cost of both detections being part of the same trajectory. Each link has a maximum capacity of 1, so the flow conservation constraint ensures no two trajectories share an observation. Min-cost flow is solved using a push-relabel method [10]. Pirsiavash et al. [13] and Berclaz et al. [2] propose using the more efficient successive shortest path algorithm to solve the min-cost flow problem, resulting in faster run times with the same globally optimal results, and also propose to use dynamic programming to yield approximate solutions very quickly. Although all these algorithms have polynomial time complexity, they do so by restricting the cost functions to products or sums of unary and pairwise edge weights. Hence, these methods can use information such as distance between corresponding observations in adjacent frames, but are unable to leverage higher order smoothness constraints such as constant velocity.

Brendel et al. [5] formulate data association as a maximum weight independent set (MWIS) problem. Their algorithm solves for two-frame tracklets independently, and then links these into complete tracks by using a learned distance measure. Long term occlusions are handled by using the MWIS solution method hierarchically to merge small tracklets into longer ones based on similarity of appearance and motion.

Li et al. [11] propose to progressively associate tracklets to obtain final trajectories. They use a ranking and classification algorithm (HybridBoost) to learn cost parameters for the tracklet affinity function. Yang et al. [18] create a CRF from the set of tracklets to remove the assumption of independence between them. The features for evaluating cost are selected using the RankBoost algorithm. Both of these algorithms require offline training, which may be infeasible for many one-off problems. In a more recent paper, Yang and Nevatia [19] propose an online CRF model to learn the parameters for tracklet association. To avoid ID switches, they learn appearance features that discriminate between targets that are close to each other.

In terms of representing higher-order constraints, Ochs and Brox [12] propose a method to project a hyper-graph onto its primal graph, allowing higher-order motion models to be represented by the edges of a normal graph. Their method is targeted towards motion segmentation of images but can readily be adapted to multi-target tracking as a way to project higher order motion constraints onto pairwise constraints in a regular flow network. Collins [7] presents an ICM-like approximate algorithm that iteratively improves an initial feasible multi-frame solution. The algorithm is capable of handling arbitrary higher-order cost functions defined over entire trajectories, however it is unclear whether the approach can be simplified to efficiently take advantage of cost functions with bounded order, such as constant velocity computed over temporal windows of three frames. Another recent paper by Butt and Collins [6] attempts to incorporate constant velocity constraints by solving a series of independent multi-dimensional assignment problems over frame triplets, which are then merged into longer trajectories and optionally sent to a network flow algorithm to span long temporal gaps due to occlusion. The approach does not have the ability to revisit and correct a trajectory.

An early pair of papers by Poore [14, 15] use Lagrangian relaxation to address the NP-hard multi-dimensional assignment problem at the heart of multi-frame tracking. Poore and Rijavec [14] use Lagrangian relaxation recursively to reduce a K-frame assignment problem to a $K-1$ dimensional one, and so on, until a two-frame assignment prob-

lem is reached and solved using the Hungarian algorithm. Instead of relaxing one frame of constraints at a time, Poore and Robertson III [15] relax constraints over $K-2$ frames simultaneously to get to the simplified bipartite assignment problem, again solved optimally. Feasible approximate solutions over longer and longer subsequences are then recovered recursively (in the first paper) or iteratively (in the second).

Our algorithm also employs Lagrangian relaxation to solve the multi-frame tracking problem, however our graph formulation and use of Lagrangian relaxation is totally different from Poore's early work. In particular, our relaxation reduces to a global network flow problem, solved optimally by min-cost flow, rather than reducing to a local two-frame assignment problem solved by the Hungarian algorithm. By making better use of global information over the entire sequence during each iteration, our approach is potentially capable of finding better solutions.

3. Our Approach

We propose an algorithm that can incorporate piecewise constant-velocity path smoothness constraints while maintaining a manageable computational complexity. Unlike [6], which builds up a solution from short tracks determined independently over small subsets of frames, our process optimizes globally over the entire sequence. We apply the principle of Lagrangian relaxation to develop an iterative solution method where, at each step, higher-order smoothness constraints are relaxed to form a modified-cost network flow problem that can be solved optimally and efficiently. This sequence of solutions gradually approaches a solution in which the higher-order constraints are satisfied, yielding a high-quality approximate solution to the original hard problem. An illustrative overview of our graph representation is presented in the next section, followed by a more rigorous problem formulation in subsequent sections.

3.1. Illustrative Overview

We motivate our problem representation with a simple example. The top graph in Figure 2 depicts a three frame sequence with three observations (1,2,3) in the first frame, two observations (4,5) in the second, and four observations (6 7 8 9) in the third. Directed edges in the graph connect pairs of observations from adjacent time frames that are candidate matches. We can associate a binary variable x_{ij} with each edge, allowing a match to be turned on or off, subject to constraints that the sum of variables on edges coming into a node must be equal to the sum of variables on edges leaving the node. With the addition of source and sink nodes connected to all the observations, unit capacities on all edges, and costs on each edge representing the cost of making a match, this would become a typical min-cost network flow formulation of multi-target, multi-frame matching. Note that costs, being associated with pairwise edges, are functions only of the candidate pair of observations connected by that edge, and thus limited to quantities that can be computed from two detections in adjacent frames.

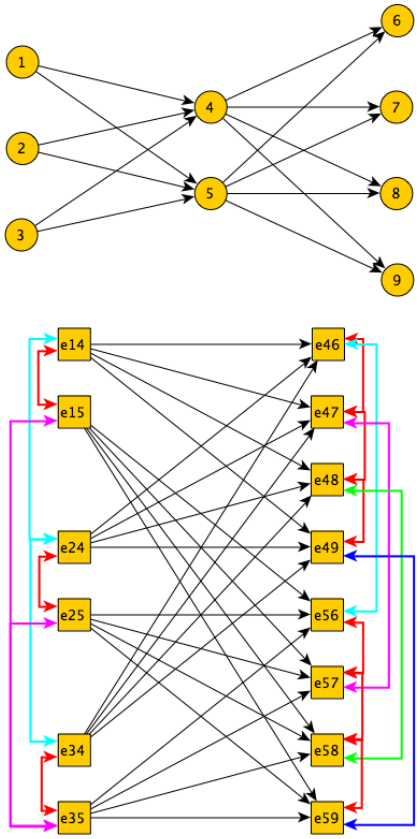


Figure 2: (Top) Graph depicting a three frame sequence with three observations in the first frame, two in the second, and four in the third. (Bottom) A new graph where candidate match pairs in the top graph have become nodes, thin black edges are added between match pairs that share an observation in frame 2, and thick colored hyperedges represent additional constraints that must be enforced so that each observation is used only once in the matching solution.

Now consider the bottom graph in Figure 2. Here, each square node represents an edge in the original graph, or equivalently, a candidate pair of matching observations, e.g. node e_{14} represents a candidate match between observation 1 in the first frame and 4 in the second. Thin black directed edges in this graph connect two nodes that share an observation in the second frame, that is, they connect a pair of candidate match pairs having the middle observation in common. As such, each of these edges represents a three-frame trajectory composed of one observation from each frame. For example, the edge between nodes e_{14} and e_{46} represents the path formed by observations 1-4-6. As

before, we can now add binary variables on these edges, source and sink nodes, unit capacities, and flow costs. In particular, note that costs on these edges now can encode higher-order motion information computed over three observations, such as constant velocity to evaluate whether an object's displacement between frame 2 and 3 is of roughly the same magnitude and direction as it's movement from frame 1 to 2.

We might be tempted at this point to use min-cost network flow on this new graph to find the optimal three-frame tracking solution. However, that would not be correct, because a feasible solution must satisfy additional constraints due to some nodes sharing observations within the same frame, which introduces a coupling between their edge variables. These extra constraints are shown as thicker, colored hyperedges in the graph, and they constrain either all the incoming edges or all outgoing edges of the set of nodes they connect to have at most one edge selected. For example, there is a hyperedge connecting nodes e_{14}, e_{24} and e_{34} in the graph, signifying that if we turn on one of the outgoing edges of node e_{14}, we can no longer select any outgoing edges from nodes e_{24} or e_{34}. This is so because all of these edges represent trajectories that pass through observation 4 in the second frame. Likewise, the hyperedge connecting nodes e_{46}, e_{47}, e_{48} and e_{49} means that only one of the incoming edges into that node set can be turned on.

Due to the introduction of these hyperedges, our problem is no longer equivalent to min-cost network flow. However, we will introduce an approximate solution method based on Lagrangian relaxation that uses min-cost network flow as a core subroutine.

3.2. Problem Formulation

Let l be the length of a video sequence, F_k be the set of observations in frame k, and r_k be the size of that set:

$$F_k = \{ob1^k_1, \ldots, ob1^k_{r_k}\} \quad k = 1, \ldots, l \ . \tag{1}$$

We form candidate matches between observations in consecutive frames. These matches are found based on appearance similarity as well as spatial proximity of the observations. Then, a match m^k_i is a 2-tuple $m^k_i = (ob1_{m^k_i}, ob2_{m^k_i})$ such that $ob1_{m^k_i} \in F_k$ and $ob2_{m^k_i} \in F_{k+1}$. The set of all candidate matches between frames k and $k+1$ is given as

$$P_k = \{m^k_1, \ldots, m^k_{n_k}\}, \tag{2}$$

where n_k is the total number of candidate matches in that frame pair. Since there are $l-1$ frame pairs, the entire sequence contains $n = n_1 + n_2 + \cdots + n_{l-1}$ match pairs, collected into a set $M = \{m_1, \ldots, m_n\}$.

We generate a graph $G = (V, E)$ as shown in Figure 3. For the remainder of this paper, we use the words nodes and vertices interchangeably, and links, edges and arcs interchangeably. Vertex set V contains a start node (s), a sink node (t), and two linked nodes $2i-1$ and $2i$ for each match $i = 1, \ldots, n$

$$V = \{s, t, 1, 2, \ldots, 2n-1, 2n\} \ . \tag{3}$$

For a match m_i, all of its incoming edges are connected to node $2i-1$ (referred to as an 'incoming node'), and the outgoing edges are connected to $2i$ (referred to as an 'outgoing node'). The link between the incoming node and the outgoing node ensures that by splitting each match into two nodes, at most a flow of 1 can pass through each match. Another advantage is that any unary and binary constraints can be placed on these edges, thus keeping them separate from the constant velocity constraints.

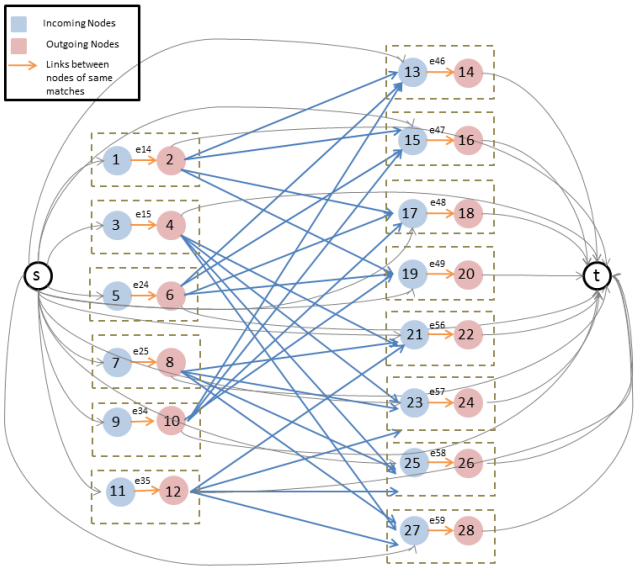


Figure 3: Network flow graph corresponding to the example in Figure 2. Each match (represented by a rectangle) has an incoming and an outgoing node, and the match number is labeled on the edge between these nodes. A link can have a flow of 0 or 1.

Links are created between nodes representing matches from different frame pairs. Hence, matches in P_k are connected to matches in P_{k-1} and P_{k+1}. We can represent our graph as a staged trellis where vertices representing matches from P_k appear in stage t_k and edges are directed only forward in time. We create an edge

$$(v_{2i}, v_{2j-1}) \in E$$

when candidate match pairs $m_i \in P_k$ and $m_j \in P_{k+1}$ share an observation in frame $k+1$. This edge represents the continuity of a trajectory through the three observations involved.

To make G a network flow graph, we also add edges from the source node to all the incoming nodes, as well as edges

from all outgoing nodes to the target node

$$(s, 2i-1) \in E; \quad i = 1, \ldots, n$$
$$(2i, t) \in E; \quad i = 1, \ldots, n \ . \tag{4}$$

These arcs ensure that object tracks may start or end at any point during the sequence. Each arc (i,j) has an associated binary flow variable $x_{i,j}$ that takes value 1 when the arc is part of a trajectory in the min-cost flow solution, and 0 otherwise. Occlusion Handling can also be incorporated into the formulation by connecting nodes between non-consecutive frame pairs. These links can be based on the proximity of the observations, or some other appropriate gap-spanning affinity measure.

3.3. Conflicts Between Matches

Similar to [2, 13, 20], graph G is a network flow graph and each node can be used in at most one track. However, since each node represents a pair of observations, we have additional constraints that must be imposed. Matches from the same pair of frames conflict when they have an observation in common, because their shared observation can be used in only one trajectory. We must therefore ensure that only one of those conflicting matches is used in the final flow solution.

For each observation $a \in F_k$, consider the set of match pairs $\{(a, *)\}$ in P_k having a as the incoming node of the pair. For each of these matches, edges entering node a from match pairs in P_{k-1} are in conflict: at most one of them can be selected and the rest must be 0. We therefore form an edge conflict constraint set EC to limit the sum of flow variables on edges entering incoming node a to be at most 1. Similarly, for the set of match pairs $\{(*, a)\}$ in P_{k-1} having a as the outgoing node of the pair, we form a conflict set to limit the sum of flow variables on edges exiting outgoing node a. Let the total number of conflict sets created in this way be q, and the set of all conflict sets be $\{\text{EC}_1, \ldots, \text{EC}_q\}$.

Our multi-frame, multi-target tracking problem can now be written as the following binary linear program

$$\min \ f(x) = \sum_{(i,j)\in E} c_{ij} x_{ij} \tag{5}$$

$$\text{s.t.} \quad x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in E \tag{6}$$

$$\sum_{(i,j)\in E} x_{ij} = \sum_{(j,k)\in E} x_{jk} \qquad \forall j \in V - \{s, t\} \tag{7}$$

$$\sum_{(i,j)\in EC_s} x_{ij} \leq 1 \qquad s = 1, \ldots, q \tag{8}$$

Similar to previous works, each edge (i, j) is assigned a cost c_{ij} based on the cost of linking match pairs, taking into account appearance similarity and path smoothness constraints. Unlike previous network flow approaches, our costs can take into account three-node smoothness measures such as constant velocity, because they are defined between pairs of match pairs, not pairs of observations. The linear objective function (5) is minimized with respect to three sets of constraints. The first constraint set (6) tells us that this is a binary optimization problem where edges are either selected or not, 1 or 0. The second constraint set (7) contains the standard flow conservation equations, saying that the flow entering a node is equal to the flow exiting it, ensuring continuous paths from s to t. Taken together, equations (5–7) specify a min-cost network flow problem that can be solved using standard algorithms. However, the third set of constraints (8) are needed to ensure that for each edge conflict set, at most one edge is selected, or in other words, that an observation shared by multiple match pairs can be used only once. This set of constraints cannot be directly handled in a network flow framework, and motivates our use of Lagrangian relaxation in the next section.

4. Lagrangian Relaxation

The key idea of Lagrangian relaxation is to take a constrained problem that is difficult and to generate a simpler approximation by converting some of the hard constraints into soft constraints. This is done by incorporating them into the cost function using Lagrange multipliers [4].

For example, consider the problem of minimizing a linear objective function $f(x) = w'x$ with respect to two sets of linear constraints, $Ax = b$ and $Cx = d$. Furthermore, suppose that it is easy to minimize $f(x)$ with respect to the $Ax = b$ constraints, but difficult when the additional $Cx = d$ constraints are included. In this case, we define a vector of Lagrange multipliers λ, one multiplier for each constraint in the $Cx = d$ set, and form a new objective function $L(x, \lambda) = w'x + \lambda'(Cx - d)$, leading to a Lagrangian relaxed problem of minimizing $L(x, \lambda)$ with respect to $Ax = b$, which by assumption is easy to solve for any fixed values of the multipliers λ. Note that instead of enforcing the hard constraints $Cx = d$, we now allow those constraints to be violated, but penalize those violations by an amount controlled by λ. We can penalize any $c_i'x > d_i$ by setting $\lambda_i > 0$, thus increasing the value of the objective function. Likewise, $c_i'x < d_i$ is penalized by setting $\lambda_i < 0$. If all constraints $Cx = d$ are satisfied exactly, that is, if $(Cx - d) = 0$, the x that minimizes objective function $L(x, \lambda)$ is also the solution to the original objective function $f(x)$ with respect to the full set of constraints (in the case of equality constraints).

Looking back at our binary linear program, it is clear that without the set of conflict constraints (8) this would be a min-cost network flow problem, which we know how to solve efficiently. Hence, we define q Lagrange multipliers $\lambda = \{\lambda_1, \ldots, \lambda_q\}$ and relax these constraints by incorporating them into the objective function. The new optimization

problem becomes:

$$\begin{aligned} \min\ L(x,\lambda) = & \sum_{(i,j)\in E} c_{ij}x_{ij} + \sum_{s=1}^{q}\lambda_s\Big(\sum_{(i,j)\in EC_s} x_{ij} - 1\Big) \\ \text{s.t.}\quad & x_{ij} \in \{0,1\} \qquad \forall (i,j)\in E \\ & \sum_{(i,j)\in E} x_{ij} = \sum_{(j,k)\in E} x_{jk} \quad \forall j \in V - \{s,t\} \end{aligned} \tag{9}$$

For a fixed value of the variables λ, the new cost function is just $\sum c'x - \alpha$ for some constant α and new costs c' that are functions of the old costs c and the current values of λ. The relaxed problem in (9) is a standard min-cost network flow problem, and can be solved efficiently by previous methods in the tracking literature [2, 13, 20].

Note that for each conflict constraint set EC_s, we now have a soft constraint term $\lambda_s(\sum x_{ij} - 1)$ in the objective function. The factor in parentheses measures how many more edges than 1 are turned on at the same time (recall that no more than one of them should be on). Setting a positive value of λ_s penalizes this excess by increasing the objective function value. Diving deeper into the details, each edge in such a violating conflict set will have its cost increased by λ_s, thus encouraging the min-cost flow solution to route flow elsewhere in the graph.

The network flow algorithm is now run iteratively, updating λ_s in each iteration. Because the relaxed objective function $L(x,\lambda)$ provides a lower bound to the original objective function $f(x)$, we seek a tighter lower bound at each iteration by choosing λ to increase $L(x,\lambda)$. Although $L(x,\lambda)$ is convex [4], it is not differentiable at all points, and thus we use a subgradient method for finding the next value of λ at each iteration. For each λ_s, the subgradient g_s is $(\sum_{(i,j)\in EC_s} x_{ij} - 1)$. If $g_s > 0$, λ_s increases, making it less likely for the edges in conflict to be selected.

4.1. Stopping Criteria

The iterative algorithm may not converge to satisfy all the Lagrangian constraints within a reasonable amount of time. Therefore, we need additional stopping criteria testing for a maximum number of iterations or an insignificant change in the value of λ. In cases where some soft constraints remain unsatisfied, we will have conflicting matches in the final trajectories, allowing an observation to be part of two or more different tracks. To remove these violations, we use the following greedy resolution phase:

1. Create tracks from the selected observation pairs.
2. For all observations in each frame, check if the observation is used in more than one track, and add those tracks to a 'competing tracks' list.
3. Compare the total cost of the competing tracks. Choose the track with the lowest total edge cost.
4. For all other tracks in the list, remove the conflicting observation. If the track has observations in the following frames, create a new track with these observations.

In our experiments, our algorithm either converged to the optimal solution, or else had very few conflicting matches as part of the final trajectories. For those examples where the algorithm halted with a non-feasible solution, the above greedy method of resolving the conflicts to reach feasibility provided us with good final trajectories.

5. Experiments

We divide our experimental results into two sections. Section 5.1 uses the publicly available data association dataset[1] of [7]. Section 5.2 compares with state of the art multi-target tracking methods on common video sequences from the literature[2] [3]

5.1. Benchmark Comparison

A block-ICM approach using a snake energy cost function to incorporate higher order motion information is proposed in [7]. They use two datasets, which have been made public, of trajectories from pedestrians walking through a building. One sequence has an average of 5 observations per frame (known as "sparse" sequence), and the other has an average of 20 observations per frame (known as "dense" sequence). Each sequence is 15 minutes long, and the ground truth is hand-labeled. Each sequence is subsampled to form datasets of 1 frame per second, 2 frames per second, and 3 frames per second. To compare against our results, we evaluate three other algorithms. The first algorithm (Projection) is based on [12], where the hypergraph with edges connecting observations in three frames (to obtain constant velocity measure) is projected onto a simple graph. The costs on the edges of the simple graph are chosen as the best value of constant velocity between the two connected nodes and a node in the future. Essentially, this is a way to be able to use higher order cost functions in a regular network flow problem. The other algorithms are a greedy sequential filtering method (Greedy) and the ICM like method (Block-ICM) of [7], for which we use the numbers reported by the authors.

We use the same error measure as in [7], which is the total mismatch error percentage. The mismatch error (mme) is the number of times an identity swap occurs within the estimated trajectories. The mismatch error percentage is then calculated as $100 \times (\sum_t mme(t) / \sum_t g(t))$, where $g(t)$ is the

[1] http://vision.cse.psu.edu/data/data.shtml
[2] https://www.d2.mpi-inf.mpg.de/node/382
[3] http://www.vision.ee.ethz.ch/~aess/dataset/.

Algorithm	Sparse Trajectories			Dense Trajectories		
	3fps	2fps	1fps	3fps	2fps	1fps
Projection	0.05	0.12	1.40	0.30	0.53	14.87
Greedy	**0.00**	0.06	1.36	0.12	0.34	6.83
Block-ICM	**0.00**	0.12	0.80	0.13	0.25	4.13
Ours	**0.00**	**0.00**	**0.41**	**0.10**	**0.17**	**1.46**

Table 1: The algorithms are compared on different sample rates for the sparse and dense sequences. The numbers reported are the mismatch error percentages, and lower numbers are better. The results clearly show the advantage of using our method.

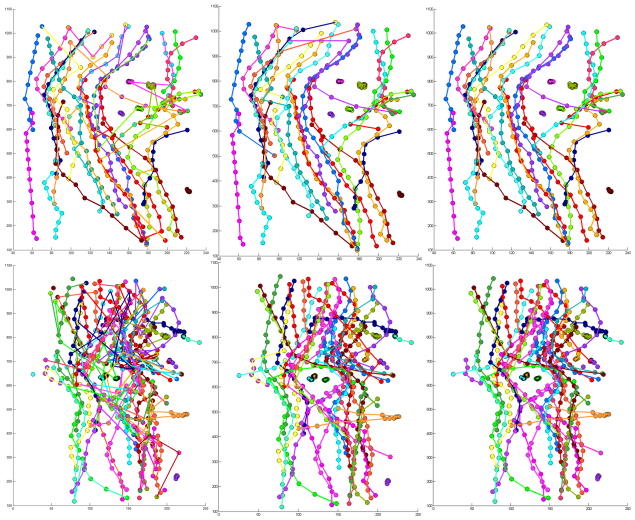


Figure 4: Comparison of the projection method adapted from [12] (left), block-ICM method from [7] (center), and our method (right). Each row shows the results of the three algorithms on a sequence from the dataset. Nodes are color coded according to the ground truth; correct trajectories should appear in the same color. For the top row, projection, block-ICM and our method had 33, 10 and 0 ID swaps respectively. For the bottom row, there were 112, 16 and 1 ID swaps respectively.

number of ground truth observations in frame t. A lower error percentage means that there are fewer ID switches between observations in different tracks.

Table 1 shows the tracking performance comparison. Our algorithm easily outperforms the competing algorithms regardless of the target density or frame subsampling. Figure 4 shows that our algorithm maintains ID labels in tough sequences where targets move close to each other.

5.2. Tracking in Video

We compare our algorithm with state of the art approaches on the popular TUD sequence [1], and ETHMS dataset [8]. We use the pre-trained pedestrian tracker of [9], which was also used in [13]. The quantitative metric that we use is the number of mismatches or ID switches. We also note the total number of (correct) observations used in the final trajectories by the algorithms. Table 2 shows the number of mismatches and the total number of detections for the TUD sequence, and the first 350 frames of the ETHMS sequence. We also show the tracking results on the ground truth detections in the ETHMS sequence. Our algorithm provides better results in all cases when compared with the dynamic programming (DP) algorithm of [13]. For the min-cost network flow algorithm of [20] the total number of observations that are part of the trajectories should be noted. Specifically, while their number of mismatches appears to be fewer, it is due to a much larger number of false negatives. Figure 5 illustrates the superiority of our algorithm.

Algorithm	TUD	ETHMS	ETHMS (GT)
DP	32/768	37/1387	25/1648
Ours	14/819	23/1514	14/1783
MCNF	9/433	11/1057	5/922

Table 2: We compare our algorithm with the dynamic programming (DP) algorithm of [13] and the min-cost network flow (MCNF) algorithm of [20] for the TUD and ETHMS (first 350 frames) sequences. The entries in the table are (number of mismatches)/(total number of observations used in the trajectories). Columns 1 and 2 use the pre-trained detector of [9]. Column 3 shows the results when ground truth detections are used. We allow occlusion handling of up to 8 frames for all algorithms.

5.3. Computational Time

Our code is implemented in MATLAB, and can be further optimized. For the TUD sequence with 200 frames, our algorithm obtained the solution in 1.43 seconds. For the ETHMS sequence with 1000 frames, our algorithm took 59.04 seconds. An advantage of our iterative solution approach is that each relaxed network flow subproblem has a special structure that can be leveraged to achieve substantial speed increase over general integer programming solvers, while still yielding high-quality results.

6. Conclusion

We have proposed a framework that uses higher-order constraints for multi-target tracking. Instead of observations, candidate match pairs of observations are used as nodes in the graph, allowing each graph edge to encode a cost based on observations in three frames. However, this higher order information comes with additional constraints, which must be relaxed to yield a min-cost flow network. We use Lagrangian relaxation to form a series of min-cost flow problems that yield solutions that gradually improve to approximate the solution to our original problem.

While the algorithm nearly always converged in our experiments, convergence is not guaranteed, and we have pro-

Figure 5: Rows 1 and 2 compare results of the dynamic programming (DP) algorithm of [13] and our algorithm respectively on the ETH sequence (frames 50, 90 and 106 shown). A pre-trained pedestrian tracker is used, and no occlusion handling is done for either algorithm. Rows 3 and 4 show tracking results using ground truth detections (frames 274, 304 and 319 shown). Row 3 shows the results from the DP algorithm, and row 4 shows our results. In this experiment, we allowed occlusion handling of up to 8 frames for both methods. The results show that our algorithm maintains ID labels more reliably.

posed stopping criteria and a greedy algorithm to enforce feasibility in such cases. We have shown our algorithm to be superior to competing methods, including other methods that attempt to use higher order information [7, 12].

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

References

[1] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2008. 1, 7

[2] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(9):1806 –1819, September 2011. 1, 2, 5, 6

[3] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Sys*. Artech House, Norwood, MA, 1999. 1

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. 5, 6

[5] W. Brendel, M. Amer, and S. Todorovic. Multiobject tracking as maximum weight independent set. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1273 –1280, June 2011. 2

[6] A. Butt and R. Collins. Multiple target tracking using frame triplets. In *Asian Conference on Computer Vision (ACCV)*, November 2012. 2, 3

[7] R. Collins. Multitarget data association with higher-order motion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2012. 2, 6, 7, 8

[8] A. Ess, B. Leibe, K. Schindler, and L. van Gool. A mobile vision system for robust multi-person tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2008. 7

[9] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627 –1645, September 2010. 7

[10] A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22:1–29, 1992. 2

[11] Y. Li, C. Huang, and R. Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2953 –2960, June 2009. 1, 2

[12] P. Ochs and T. Brox. Higher order motion models and spectral clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2012. 2, 6, 7, 8

[13] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2011. 1, 2, 5, 6, 7, 8

[14] A. Poore and N. Rijavec. A lagrangian relaxation algorithm for multidimensional assignment problems arising from multitarget tracking. *SIAM Journal on Optimization*, 3(3):544–563, 1993. 2

[15] A. B. Poore and A. J. Robertson III. A new lagrangian relaxation based algorithm for a class of multidimensional assignment problems. *Computational Optimization and Applications*, 8:129–150, 1997. 2, 3

[16] C. J. Veenman, M. J. T. Reinders, and E. Backer. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(1):54 –72, January 2001. 1

[17] B. Wu and R. Nevatia. Tracking of multiple, partially occluded humans based on static body part detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 951 – 958, June 2006. 1

[18] B. Yang, C. Huang, and R. Nevatia. Learning affinities and dependencies for multi-target tracking using a crf model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1233 –1240, June 2011. 1, 2

[19] B. Yang and R. Nevatia. An online learned crf model for multi-target tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2012. 1, 2

[20] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1 –8, June 2008. 1, 2, 5, 6, 7