A Linear Programming Approach for Multiple Object Tracking

Hao Jiang, Sidney Fels and James J. Little
University of British Columbia, Vancouver, BC, V6T 1Z4, Canada
haoj,ssfels@ece.ubc.ca, little@cs.ubc.ca

Abstract

We propose a linear programming relaxation scheme for the class of multiple object tracking problems where the inter-object interaction metric is convex and the intra-object term quantifying object state continuity may use any metric. The proposed scheme models object tracking as a multi-path searching problem. It explicitly models track interaction, such as object spatial layout consistency or mutual occlusion, and optimizes multiple object tracks simultaneously. The proposed scheme does not rely on track initialization and complex heuristics. It has much less average complexity than previous efficient exhaustive search methods such as extended dynamic programming and is found to be able to find the global optimum with high probability. We have successfully applied the proposed method to multiple object tracking in video streams.

1. Introduction

Tracking multiple objects simultaneously is key for many vision applications, such as visual navigation and object activity recognition. Even though each object can be tracked separately, tracking objects together is important for obtaining good results if objects have complex interactions [1]. We categorize object interactions into two classes. The first type of interaction constrains the object relative locations, i.e., objects tend to keep relative positions or spatial layout during a short period of time. The second type of interaction is object mutual occlusion, i.e., an object in front occludes other objects in the same region. Explicitly modeling interaction of objects enables tracking multiple objects more robustly, especially in cluttered environments. But, the search space also increases drastically compared to that of tracking objects separately. Naive exhaustive search becomes intractable. Efficient exhaustive searching schemes such as extended dynamic programming [1] are still too complex to be applied to problems with a medium number of observations and objects. We propose a linear programming relaxation scheme for a specific class of multiple object tracking problems, in which the metric for inter-object position interaction term is convex while the intra-object terms quantifying object state continuity along time may use any metric. The proposed scheme explores a large search space efficiently and almost always gives a global optimum because of the special structure of the formulation.

Multiple object tracking has been studied intensively. For example, Kalman filtering has been a classic scheme for object tracking. Recently, particle filtering has been popular for tracking multiple objects such as ants [2] with complex interactions. Particle filtering has also been studied for tracking hockey players [3] in which object interaction is not explicitly modeled. Bayesian networks have been applied to optimizing trajectories of football players in video [5]. This approach does not consider track interaction among objects.

Dynamic programming (DP) is also widely applied to multiple object tracking. The single chain Viterbi algorithm can be extended [1] to optimize multiple tracks simultaneously. The computational complexity of extended DP is $O(mk^{2n})$, where k is the number of observations at each stage, n is the number of objects and m is the length of the sequence. Extended DP is thus hard to apply to large scale problems. An efficient approximate dynamic programming scheme [4] has been studied to find a single object's path with heuristics used to determine the sequence of path assignments in a multiple-camera setting. While simple heuristics such as best-track-first assignment works well for multiple camera tracking, it does not always give correct solutions when objects have complex mutual occlusion patterns, especially for single camera applications.

Linear programming (LP) is another approach that can be used for more efficient search in object tracking. Optimizing object tracks using 0-1 Integer Programming [6] has been studied for radar data association. This formulation is different from the proposed scheme in that a variable is defined for each feasible trajectory and object tracking is solved as a set packing problem. Other approximation methods for solving similar integer LP formulations as [6] are studied in [7, 8], which turn out to be quite similar to the sequential DP method [4]. Unlike previous LP methods, our proposed scheme is based on a multiple-shortest-path model that tries to connect edges into paths and has much fewer variables. Belief Propagation (BP) [9] has also been used for optimizing hand tracking. Occlusion is explicitly modeled in this method. However, multiple object tracking results in a loopy graph structure making it difficult to guarantee convergence to a global optimum.

Even though intensively studied, robust and efficient tracking of multiple objects with complex interactions remains unsolved. In this paper, we propose a novel linear programming relaxation scheme to optimize multiple object tracks simultaneously by explicitly modeling spatial layout constraints and mutual occlusion constraints. We formulate object tracking as a multipath searching problem. Each path is composed of a sequence of states, e.g., locations and appearances, of an object through time represented by nodes in a graph. Different tracks are constrained so that objects cannot occupy the same spatial region. Convex

1-4244-1180-7/07/$25.00 ©2007 IEEE

penalty terms are included to constrain the consistent objects' layout in space, i.e., the objects' relative positions do not change abruptly from frame to frame. The state continuity metric term along time may use any metric. Based on the special structure of our formulation, a linear programming relaxation approach effectively solves the path searching problem when paths overlap and objects occlude each other. As our results illustrate, the linear program almost always yields integer solutions that globally optimize object tracks and has low order polynomial average complexity.

2. Multiple Object Tracking

In this section, we describe our linear programming based method for optimizing multiple object tracks in continuous video frames. Intuitively, at each frame we represent all the possible spatial locations of each object from the observations as nodes based on attributes of the objects. (In our examples, we determine possible bounding boxes for objects' locations based on background subtraction or appearance characteristics of objects. These bounding boxes are also used to determine what it means for one object to occlude another.) Over a window of frames, these nodes form a graph where a path connecting nodes represents a possible spatial trajectory of an object over time in the video. This is represented in Fig. 1. However, if one object occludes another, there is a break in the track of one object. We have a special occlusion node that allows the path for an occluded object to be accounted for in that particular frame if there is no other non-overlapping location for the potentially occluded object. This graph forms the basis for formulating a cost function based on all the possible paths and constraints, leading to a linear program that may be efficiently solved. The algorithm optimizes the states for all the objects together. Thus, it finds consistent paths for all the objects over a window of video frames and assigns a meaningful interpretation of location or status of occlusion to each object as described more formally below.

2.1. Problem Statement

In multiple object tracking, we need to locate objects (positions, poses etc.) through a sequence of video frames. For each video frame, we assume that there is a set of observations for each object, which are obtained by using methods such as background subtraction or template matching. These observations are not reliable and may contain many false positives. Misdetection of an object may also occur. We wish to obtain object locations in a sequence of video frames based on the assumption that an object usually does not change appearance and location abruptly. Apart from finding the correct trajectories for all the objects, we also need to determine whether an object is visible in a video frame: objects may disappear due to occlusion or moving out of scene.

2.2. Network Model

In the following, we study multiple object tracking based on a network model in which sub-models in our formulation interact with each other. This approach contrasts with the previous trellis model used in single-chain dynamic programming. Fig. 1 illustrates the network model of the multiple object tracking.

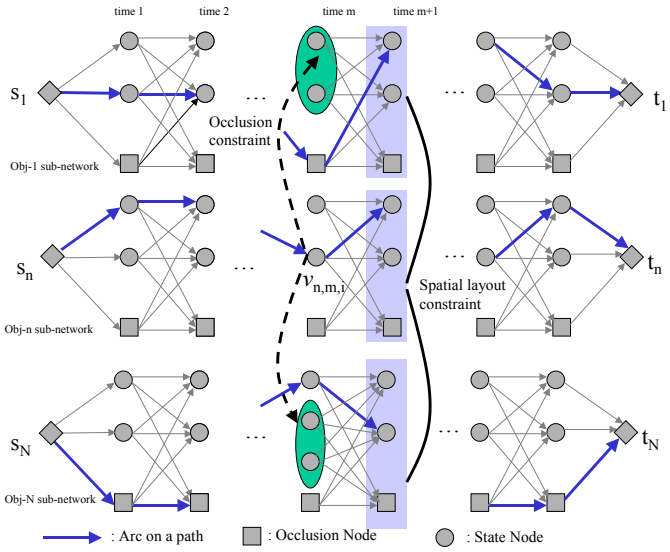


Figure 1. The network model for multiple object tracking.

In Fig. 1, an object's possible location and appearance states are represented as round nodes. For a given frame, hypothesized locations (i.e., observations) for each object may be different, and therefore the sub-network for each object may contain a different number of nodes. The rectangular nodes in Fig. 1 are the occlusion nodes that provide a node to represent that an object is occluded and does not have a spatial location. A source node and a sink node, shown as diamond nodes in Fig. 1 are also included for each object sub-network to represent the start and end of the object tracking sequence. Sink nodes are included just for convenience; they do not correspond to states of objects. The solid arcs between nodes indicate possible state transitions. A connected set of nodes between a source and sink node represents the spatial trajectory of an object.

We also model mutual occlusion among objects in the network. A spatial conflict set is defined for each node in the network. Nodes in a spatial conflict set correspond to object states occupying the same spatial location. As shown in Fig. 1 the spatial conflict set for node $v_{n,m,i}$ includes the node itself and nodes in the ovals in the other objects' sub-networks that would overlap the region of $v_{n,m,i}$. Note that the occlusion node for each object never has a spatial conflict, so it will never be in a spatial conflict set. Only one node in a spatial conflict set may be selected for connecting an object path as this represents the visible object at that location in space. Once a node is selected for one of the objects, all the other objects must either select a node that includes a different spatial location for that frame or the occlusion node. The above condition is defined as the object mutual occlusion constraint. We also include a spatial layout constraint for all the objects. This is defined in the network model to constrain objects' relative locations at each time instant. Multiple object tracking can thus be modeled as finding optimal paths from the source nodes to the sink nodes for all objects, which satisfies the object interaction constraints.

We use the following notation to precisely define the

problem in an LP framework. For object n, its source node is denoted as s_n and its sink node as t_n. s_n corresponds to the location and appearance of object n in frame 0. The source node also provides an initial *template* node for computing trajectory costs as described below. For each video frame, we insert nodes corresponding to all the observations of object n at each time instant together with an occlusion node. $v_{n,m,i}$ denotes the node indicating that object n is assigned state i in frame m. The occlusion node is always the node with the largest state number i. The source node s_n is also denoted as $v_{n,0,0}$, and the sink node t_n as $v_{n,M+1,0}$, where M is the length of video sequence. We connect nodes in successive frames with arcs as shown in Fig. 1 using a fully connected pattern. For most applications, partially connected patterns can also be used to simplify the problem based on heuristics, for example, that objects do not move far between successive frames.

A cost $c(v_{n,m,i}, v_{n,m+1,j})$ is assigned to each arc, which indicates the cost of state i at time m and state j at time $m+1$ being on the trajectory of object n. The cost function can be convex or non-convex. An arc's cost usually contains two parts: the cost of choosing a state at a time instant and the cost of state transition from i to j. In this paper, the cost of arc connecting node $v_{n,m,i}$ and $v_{n,m+1,j}$ is defined as

$$c(v_{n,m,i}, v_{n,m+1,j}) = \begin{cases} g(s_n, v_{n,m+1,j}) + \\ \lambda_1 \cdot g(v_{n,m,i}, v_{n,m+1,j}) + \\ \lambda_2 \cdot d(v_{n,m,i}, v_{n,m+1,j}) & \begin{array}{c} v_{n,m,i}, v_{n,m+1,j} \\ \text{are not occlusion} \\ \text{nodes and not} \\ \text{sink nodes} \end{array} \\ c^a_{const} + g(s_n, v_{n,m+1,j}) & \begin{array}{c} v_{n,m,i} \text{ is occluded} \\ \text{and } v_{n,m+1,j} \text{ is not} \end{array} \\ c^b_{const} & otherwise \end{cases}$$

where function $g(.)$ compares the similarity of an object appearance corresponding to nodes in the network, e.g., by comparing color histograms in bounding boxes; $d(.)$ computes the spatial distances of two states, e.g., the distance of two bounding boxes. λ_1 and λ_2 are constant coefficients to control the weight of temporal smoothness. c^b_{const} and c^a_{const} are constant costs penalizing when an object disappears or reappears. Thus, if an arc leads into an occlusion node or a sink node, it bears a constant cost. The cost of an arc from an occlusion node to a nonocclusion node includes the similarity measurement of the destination node to the template object (the source node) plus a constant. When both of the nodes are non-occlusion nodes, the edge connecting the nodes has weight equaling the summation of three terms: the similarity of the target node to the template object, the appearance similarity of detections in two successive frames and a term that penalizes large spatial displacement between video frames.

In modeling the object occlusion constraint, we need to specify the spatial conflict set for each non-occlusion node $v_{n,m,i}$. The spatial conflict set for node $v_{n,m,i}$ is denoted as $O(v_{n,m,i})$ which includes $v_{n,m,i}$ and nodes from other sub-networks whose regions are highly overlapping with the region of node $v_{n,m,i}$. To determine whether nodes are included in a spatial conflict set, we consider two types of overlapping regions. The first one includes partially overlapped regions as shown in Fig. 2 (a). The second one includes completely overlapped regions as shown in Fig. 2 (b). There are multiple approaches to determine whether to include a node in the spatial conflict set. For example, one approach uses the probability of two bounding boxes overlapping. This probability is calculated using the ratio of the overlapping area to the average area of the rectangular regions. If the ratio is sufficiently large, the two regions cannot be visible at the same time and nodes corresponding to these regions are in the same spatial conflict set. Another approach uses a simpler measurement based on the total city-block distance of the 4 corners of the two bounding boxes. In this case, if the difference is below some threshold, then the two bounding boxes are overlapping and the nodes should be included. If the difference is large then either the objects are not overlapping or the size of two objects is very different and the corresponding nodes do not belong to a spatial conflict set. We use this latter approach in our examples.

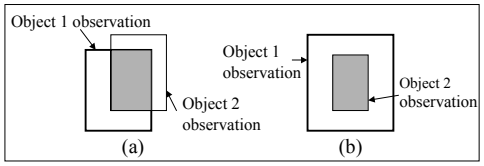


Figure 2. Overlapped regions. (a): Partially overlapped regions; (b): Fully overlapped regions.

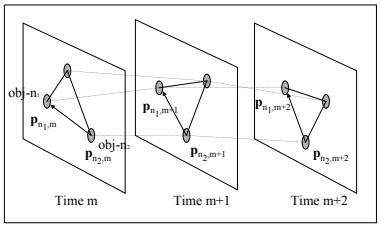


Figure 3. Spatial layout consistency.

Apart from the occlusion constraint, we also would like to keep the spatial layout of objects stable over a short period of time. To model this constraint, we keep the spatial displacement vectors between objects as similar as possible across time. As shown in Fig. 3, the vectors from object n_2 to object n_1 tend to remain unchanged at time instant m and instant $m+1$, i.e., $||(\mathbf{p}_{n_1,m+1} - \mathbf{p}_{n_2,m+1}) - (\mathbf{p}_{n_1,m} - \mathbf{p}_{n_2,m})||$ tends to be a small number. In fact, vector $\mathbf{p}$ can be more than 2D. For example, $\mathbf{p}$ can be a 4D vector representing the 2 corners of bounding boxes. This second constraint is a soft one and implemented as a regularization term in the objective function.

2.3. Discrete Optimization

An energy function for optimizing object tracks can thus be written as follows.

$$\min_{paths} \sum_{n,m} \sum_{\substack{(v_{n,m,i}, v_{n,m+1,j}) \\ \text{on a path}}} c(v_{n,m,i}, v_{n,m+1,j}) + \mu \sum_{m} \sum_{\substack{\{n_1,n_2\} \\ \in \mathcal{N}}} ||(\mathbf{p}_{n_1,m+1} - \mathbf{p}_{n_2,m+1}) - (\mathbf{p}_{n_1,m} - \mathbf{p}_{n_2,m})||$$

$$s.t. \text{ at most one path goes through } O(v_{n,m,i}), \forall v_{n,m,i}$$

where $\mathbf{p}_{n,m}$ is the location of object n at time instant m. For instance, if we use bounding boxes to quan-

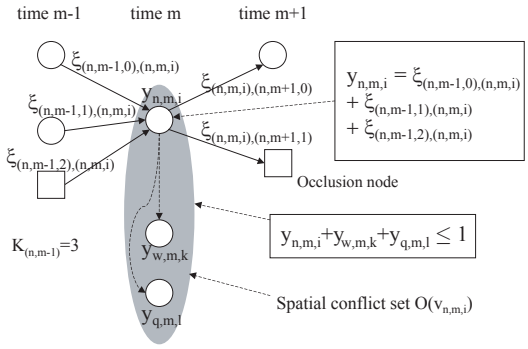


Figure 4. Linear programming formulation.

tify the location of an object, $\mathbf{p}_{n,m}$ is a 4-element vector $(p_{n,m,1}, p_{n,m,2}, p_{n,m,3}, p_{n,m,4})$ in which $(p_{n,m,1}, p_{n,m,2})$ is the top-left corner x-y coordinate of the bounding box and $(p_{n,m,3}, p_{n,m,4})$ is the right-bottom corner x-y coordinate. $\mathcal{N}$ is the set of neighboring objects. μ is a coefficient to control the weight of the spatial layout regularization term. In this paper, we assume all the object pairs are neighbors, i.e., $\mathcal{N}$ contains all the object pairs. We assume that the norm $||.||$ is the L_1 norm. Using the L_1 norm enables us to relax the optimization into a simpler linear program. In fact, the L_2 norm can also be used and the relaxation is a quadratic program which can also be efficiently solved. In the following, we use the L_1 norm and LP relaxation to illustrate the concept.

Because of path interaction, searching algorithms need to consider all the paths simultaneously and thus have to search a large space. Naive exhaustive search is not an tractable option. This optimization problem has convex (L_1) inter-object regularization terms, while the intra-object regularization term embedded in the arc cost may use any metric. As shown in the following section, this type of problem can be relaxed into a convex program that can be efficiently solved.

2.4. Linear Programming Relaxation

To convert the above discrete optimization problem into a linear programming relaxation we embed the discrete search space into a continuous one as follows.

We convert the objective function into a linear one by introducing variable $\xi_{(n,m,i),(n,m+1,j)}$ to indicate whether arc $(v_{n,m,i}, v_{n,m+1,j})$ is on the path of object n. If the arc is indeed on a path, the variable should be 1 and otherwise is 0. We also define variable $y_{n,m,i}$ to be the summation of ξ corresponding to all the incoming arcs of node $v_{n,m,i}$. Let $K_{(n,m-1)}$ be the number of nodes for object n at time $m-1$, $y_{n,m,i} = \sum_{j=0}^{K_{(n,m-1)}-1} \xi_{(n,m-1,j),(n,m,i)}$. Thus, $y_{n,m,i}$ indicates whether node $v_{n,m,i}$ is on the path of object n. In the ideal case, $y_{n,m,i}$ will be 1 if the node is on the path and 0 otherwise. Object location is represented with variables p. $p_{n,m,l}$ is the lth element of the location of object n at time m. $p_{n,m,l}$ equals the linear combinations of observations with coefficients $y_{n,m,i}$. Fig. 4 illustrates these notations with a simple case. Based on the energy function defined, the cost of a path is thus the linear combination of edge costs plus an L_1 norm regularization term. By introducing non-negative auxiliary variables, we can further turn the L_1 norm terms into linear functions. The path finding can therefore be relaxed into the following linear program:

$$\min \sum_{\text{For all edges } (v_{n,m,i}, v_{n,m+1,j})} \xi_{(n,m,i),(n,m+1,j)} \cdot c(v_{n,m,i}, v_{n,m+1,j}) + \mu \sum_{m,l} \sum_{\{n_1,n_2\}\in\mathcal{N}} (p^+_{n_1,n_2,m,l} + p^-_{n_1,n_2,m,l})$$

subject to:

$$\sum_{j=0}^{K_{(n,1)}-1} \xi_{s_n,(n,1,j)} = 1, \forall n$$

$$\sum_{i=0}^{K_{(n,M)}-1} \xi_{(n,M,i),t_n} = 1, \forall n$$

$$\sum_{i=0}^{K_{(n,m-1)}-1} \xi_{(n,m-1,i),(n,m,j)} = \sum_{l=0}^{K_{(n,m+1)}-1} \xi_{(n,m,j),(n,m+1,l)},\ m = 1..M, \forall n, j$$

$$y_{n,m,i} = \sum_{j=0}^{K_{(n,m-1)}-1} \xi_{(n,m-1,j),(n,m,i)},$$

$$y_{n,m,i} + \sum_{v_{q,m,j}\in O(v_{n,m,i}), q\neq n} y_{q,m,j} \leq 1,\ m = 1..M, \forall n, i$$

$$p_{n,m,l} = \sum_{i=0}^{K_{(n,m)}-1} \phi_l(\mathbf{r}_{n,m,i}) y_{n,m,i},\ m = 1..M, \forall n, l$$

$\phi_l(\cdot)$ extracts the lth element of location vector

$$p^+_{n_1,n_2,m,l} - p^-_{n_1,n_2,m,l} = p_{n_1,m,l} - p_{n_1,m+1,l} - p_{n_2,m,l} + p_{n_2,m+1,l}, \{n_1, n_2\} \in \mathcal{N}, m = 1..M-1,$$

$$\xi, p^+, p^- \geq 0$$

In the above equation, $\mathbf{r}_{n,m,i}$ is the location vector, e.g., bounding box coordinates, corresponding to node $v_{n,m,i}$. Occlusion nodes correspond to a special location, e.g., zero-size bounding box at the center of an image. $p^+_{n_1,n_2,m,l}$ and $p^-_{n_1,n_2,m,l}$ are non-negative auxiliary variable pairs, which are used to turn the L_1 norm smoothness term into a linear function.

We use a standard linear programming trick [10] to convert an absolute value term into a linear function. In the constraint, the difference of the auxiliary variable pair $p^+_{n_1,n_2,m,l}$ and $p^-_{n_1,n_2,m,l}$ equals the location vector difference of two neighboring objects, for which we would like to compute the absolute value. When the linear program is finally optimized, at least one of the auxiliary variables in each pair will be zero. Otherwise, we can always subtract the smaller one of the pair from each variable and get a feasible solution with smaller objective function and one variable in the pair becomes zero, which contradicts the optimum solution assumption. Therefore the sum of the auxiliary variables in the objective function equals the absolute

value of the spatial consistency term in our formulation, when the LP is optimized. The linear program is equivalent to the original discrete optimization if the linear cost term equals the original cost term, which will be the case if ξ are further constrained to be 0 or 1. The linear program is thus a linear approximation or relaxation of the discrete optimization problem.

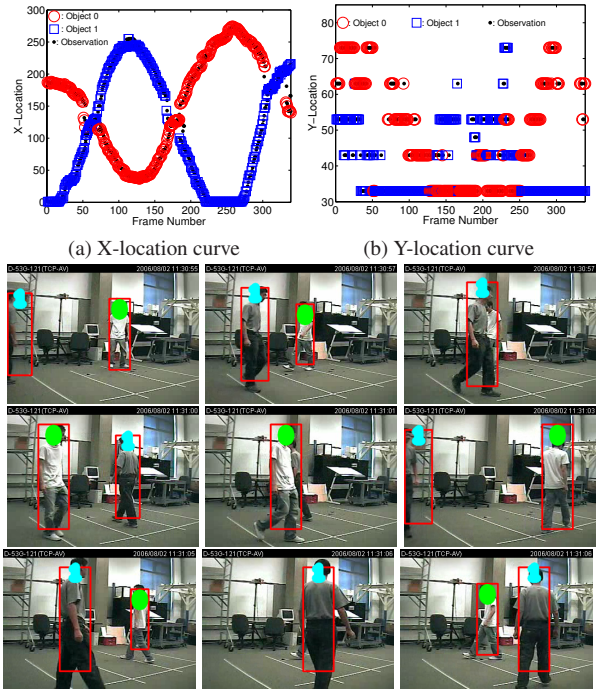
The first three constraints set out the unity flow continuity constraints that are necessary conditions for the solution to be a path for each object. The constraint on y guarantees that no two paths go through the same spatial conflict set, i.e., if one path goes through a position other tracks tend to pass these positions will be occluded. The spatial conflict set is also illustrated in Fig. 4.

If we constrain the variables of ξ to be 0 or 1, the integer program exactly solves the multiple object tracking problem. We drop the integer constraint and obtain a linear programming relaxation which can be solved efficiently. There is no guarantee that the linear program always gives integer solutions for ξ. For real problems, most of ξ are indeed 0s or 1s and therefore gives the globally optimized solution. As shown in the experiments, the linear program has a high probability of directly giving the global optimal solution.

The simplex method for linear programming has exponential complexity in the worst case. Linear programming is fast for real applications [10]; for our LP formulation, its average complexity is approximately $O(n^2km)(2log(k) + 2log(n) + log(m))$, in which k is the number of observations for each object, n is the number of objects and m is the number of frames in optimization. In comparison to extended DP, the linear program has much lower average complexity.

Example 1: To illustrate how our approach works we track 2 objects in 340 consecutive video frames. We assume that object histograms are known. At each time instant, potential object locations are detected as bounding boxes. Each bounding box is represented using a 4-element vector representing 2 opposite corners. Spatial conflict sets are then determined for each bounding box. In this example, all the bounding boxes detected are candidates for object 0 or 1, hence, the sub-networks for each object are the same. Grayscale color histograms with 64 bins are used as the features for object appearance identification. In this example, a neighboring set only contains one pair {0,1}. We build a linear program for this problem based on our proposed LP relaxation scheme. The LP takes 4628 simplex iterations. Values of p give locations of objects. If the value of y at an occlusion node is greater than 0.5, the object is set occluded at the time instant. The tracking result is shown in Fig. 5. The top-left corner x and y-coordinate of the bounding boxes for both objects are shown in Figs. 5 (a) and (b). For this example, LP relaxation has integer solutions for ξ and therefore achieves its global optimum. As shown in Fig 5 the object paths are quite good for both x and y coordinates even when the objects overlap each other.

As a comparison, we apply DP with best-track-first assigned heuristics to the same data. The energy function of DP is the same as the proposed scheme except for the spatial layout consistency term. Approximate DP is not easily extended to include such regularization terms since it optimizes each track separately and then assigns tracks sequentially. Fig. 6 shows the tracking result of approximate



(a) X-location curve (b) Y-location curve

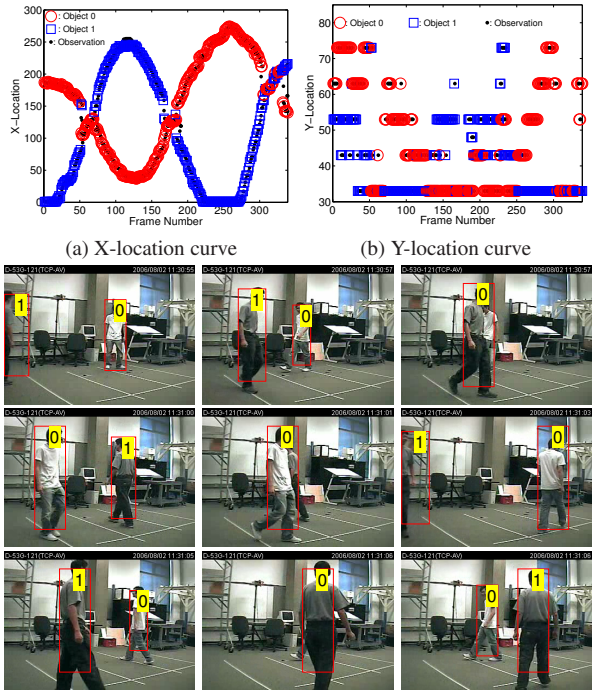
Figure 5. Tracking 2 objects in 340 successive video frames using the proposed scheme. Green and blue labels indicate object 0 and 1 respectively.

DP. In this example, DP first picks the object 0 track as a better fit and determines the track for object 1 after removing assigned boxes for object 0. As shown in this example, greedy track assignment selected wrong labels at the first and third occlusion instances. Simply reducing the occlusion label cost will not solve the problem and it also causes many missed detections.

2.5. Online Multiple Object Tracking

We have studied an LP based method to track multiple objects by optimizing tracks in a sequence of video frames. This scheme can be extended to online video tracking by applying the tracking scheme as a moving window filter. For our long video sequences we use a video segment window size of between 15 to 300 frames with 1 frame overlapping between segments. An object list keeps the histogram of object templates. The locations of object templates are also updated at the end of each video segment. The tracking network is constructed by using the templates as "observations" in the zero stage and another M successive video frames are used in constructing the rest of the network.

Objects can also be detected automatically for background subtraction based object tracking. If we find a consistent object which is not on the track of previous video segment, we insert it into the object list. The consistency is measured by a backward and forward testing approach based on the proposed tracking scheme. We check the duration of visibility and the cost of track in backward and forward tracking. If a new object has track cost lower than a threshold and appears in more than 75% of the testing period, it is inserted into the template list.



(a) X-location curve (b) Y-location curve

Figure 6. Tracking result using approximate DP for Example 1.

3. Experiment Results

We report our results using our method for tracking multiple objects on 4 different video sequences. These video sequences are in CIF format with frame rate 15–30 frames/second.

3.1. Tracking Two Stuffed Animals

Fig. 7 shows the tracking result of the proposed method for a 307-frame video. 2 toy objects are tracked through the video frames. There are complex occlusions between the two objects. The templates for the two objects are set using the first video frame. A sub-image is used as the feature in tracking. Object observations are obtained at local peaks of the template matching map. Approximately 80 detections are found for each object in each video frame which appear as nodes in the graph providing many path possibilities. In this experiment, LP optimizes each 20-frame segment including the template frame in a sliding window fashion. Despite complex occlusions, the proposed method tracks the objects correctly along the video sequence.

3.2. Tracking Fast Moving Squash Players

In another experiment, we apply the proposed scheme to a 1351-frame squash video sequence with 2 players as shown in Fig 8. The candidate objects are detected by background subtraction similar to method used in [4]. The video includes complex object interaction and mutual occlusion. Noisy background subtraction also makes object tracking a hard task.

In this experiment, we convert color image into grayscale and use a rough 64-bin histogram as features. The proposed linear programming relaxation is then applied to the video sequence in sliding window fashion the same as the first experiment. The proposed scheme accurately follows the object locations through the video sequence. Fig. 8 illustrates sample frames of the tracking result and Fig. 9 shows the object locations at each time instant through time (occluded objects are not shown). In the 1351-frame video sequence, object 0 has 7 wrong label assignments and object 1 has 5 wrong detections. The average object tracking precision is about 99% for this example. LP also has a high probability of directly obtaining the global optimal solution. Only 3 segments do not have fully integer solutions for ξ in 75 video segments.

3.3. Comparison with DP on Tracking Three People Walking in an Office

Fig. 10 and and Fig. 13 show the result of tracking three objects with the proposed method for a 2431-frame video. In this experiment, we use background subtraction to detect bounding boxes for potential object locations. The features of objects are grayscale image histograms with 64 bins inside a bounding box. Bounding boxes detections are noisy because of the large compression ratio of the video and complex object interaction. The scales of bounding boxes are also not accurate, which results in large portions of the background inside some bounding boxes. The sliding window setting is the same as previous experiments. Objects are automatically detected in this example using the method in Sec. 2.5.

The proposed scheme can deal with complex occlusions and objects moving out of the scene and coming back. Object 0 has 5 wrong detections, object 1 has 22 wrong detections and object 2 has 125 wrong detections. Overall the accuracy rate is 94% per frame. In this experiment, 4 segments do not have fully integer solutions for ξ in a total of 135 video segments.

To compare methods, we apply DP to each single person with exactly the same network weight settings. The result is shown in Fig.11. Because no object interaction constraint is enforced, DP often assigns different labels to the same object and sometimes fails to locate an object in the scene. Simple heuristics do not always give the correct solution. DP with best-track-first assigned heuristics has 67, 37 and 319 wrong tracking errors for object 0, 1 and 2 respectively. The accuracy is 83% per frame. Fig. 12 shows sample video frames where the LP approach improves the tracking result.

3.4. Tracking 4 Players in a Double-Squash Game

In Fig. 14, we applied our method to a 500-frame double-squash video sequence. There are four objects in the video and there are about 10 detections in each frame. The players in the same team wear the same clothing. In this experiment, we use the proposed scheme to optimize tracking in the whole video sequence rather than shorter segments. We would like to obtain a global optimal solution considering only the occlusion constraint.

We use a basic branch and bound method to obtain the global solution. Our method finds the global optimal in 3 minutes using a 2.6GHz PC which is much faster than extended DP which needs about an hour to compute the result. Since LP solution is very near the global optimum, branch and bound converges very soon. We use branch and bound method here to obtain a global optimum so that we can have a fair comparison with extended DP.

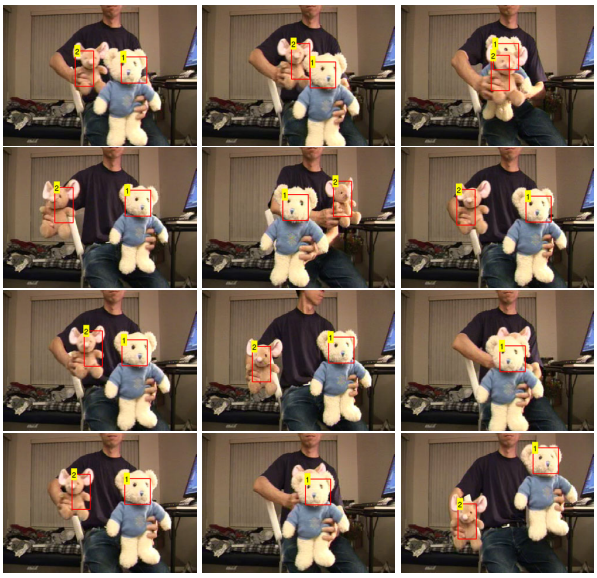
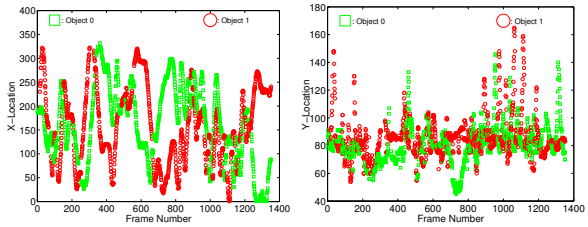


Figure 7. Tracking 2 toy objects with the proposed scheme. Selected frames from 307 frames.

Figure 8. Squash. Selected frames from 1351 frames.

As shown in Fig. 14 and Fig. 15 , the tracker works well in following multiple objects during a long sequence. In Fig. 15, when objects are occluded, their spatial locations are set to (1,1), which are shown as abrupt drops in the curves. Even though we obtain a global optimal solution, the result is not perfect. Sometimes errors occur for dark team players (player 0 and 2) when the two players occlude each other and cause their identities to be exchanged. Such errors happen due to both unreliable bounding box detection using background subtraction and occlusion between objects with very similar appearance.

Fig. 16 shows typical average running times of the linear program using a 2.6GHz PC. Random observations and color histograms are generated in each frame. Each exper-



(a) (b)

Figure 9. Object locations for 2 squash players. (a): X-Locations of objects; (b): Y-Locations of objects.

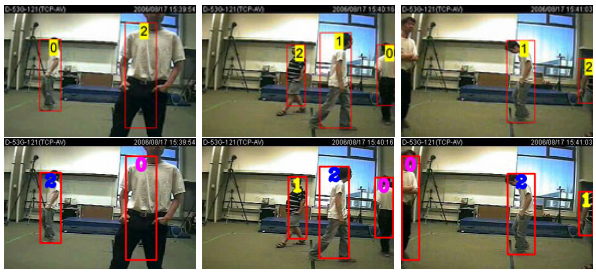
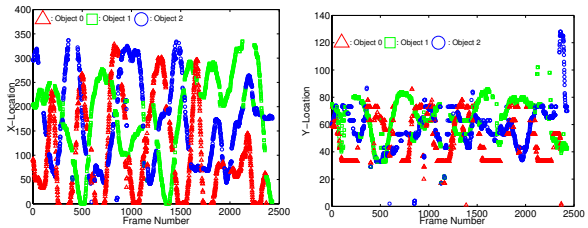


Figure 12. Sample frames where DP with simple heuristics does not yield correct solution while the proposed scheme does. The first row shows sequence DP frames. Second row shows results with the proposed method.



(a) (b)

Figure 13. Objects locations for 3-people tracking. (a): X-Locations of objects; (b): Y-Locations of objects.

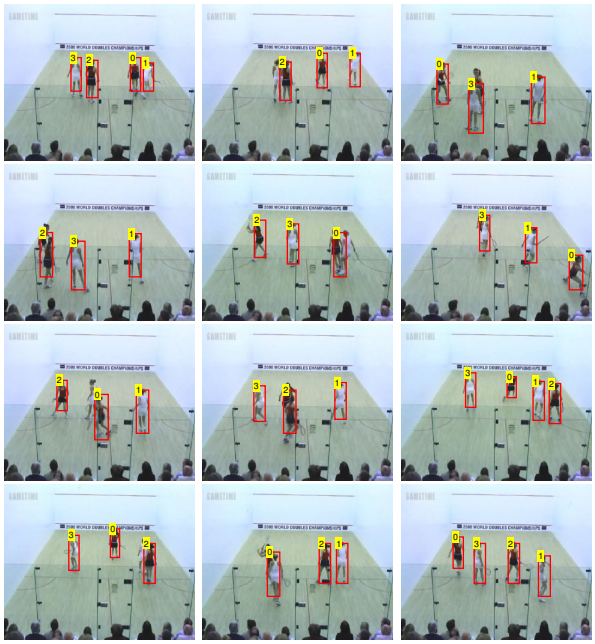


Figure 14. Double Squash. Selected frames from 500 frames.

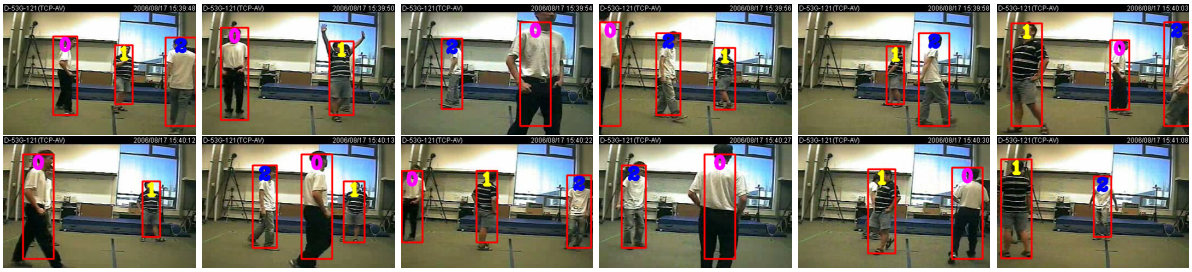


Figure 10. Tracking 3 people with the proposed scheme. Selected frames from 2431 frames.

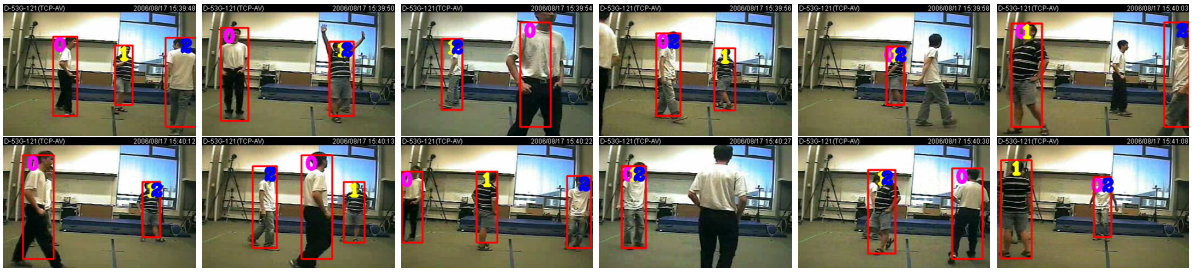
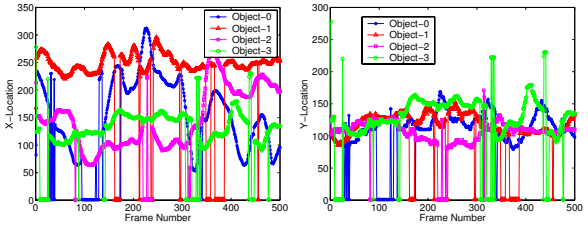
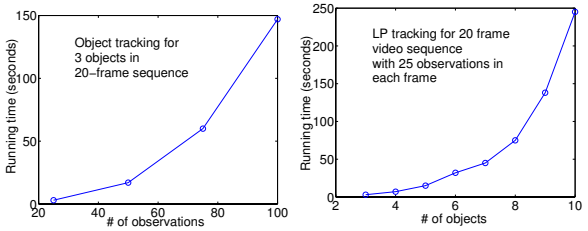


Figure 11. Tracking 3 people with separate DP for each object. Selected frames from 2431 frames.



(a) (b)

Figure 15. Objects locations for 4 squash players. (a): X-Locations of objects; (b): Y-Locations of objects.



(a) (b)

Figure 16. The complexity of the proposed scheme.

iment is repeated 10 times and running times are averaged. Fig. 16 (a) shows the typical running time of our method for different numbers of observations. Fig. 16 (b) shows the typical running time of our method for different numbers of objects. Simultaneously optimizing all the tracks using the Viterbi algorithm has considerably higher spatial and temporal complexity. In one case, extended DP takes about 6 hours to optimize 3 objects in 20 video frames with 50 observations in each frame, while the proposed scheme converges in tens of seconds as shown in Fig. 16 (a). Thus, our method requires considerably less computation time than other approaches and still achieves good accuracy.

4. Conclusion

In this paper, we propose a novel framework for optimizing multiple object tracking that can be solved efficiently based on a linear programming relaxation. The proposed scheme explicitly models track interaction such as the spatial layout constraint and object mutual occlusion. Experiments show that the proposed scheme works robustly in tracking objects with complex interactions in long video sequences. The linear program relaxation can also be solved more efficiently than previous methods such as extended dynamic programming. Thus, we believe our approach provides a useful method for multiple object tracking in video sequences.

References

[1] J.K. Wolf, A.M. Viterbi and G.S. Dixson, “Finding the best set of K paths through a trellis with application to multitarget tracking”, IEEE Trans. on Aerospace and Electronic Systems, pp.287-295, vol.AES-25, no.2, 1989.

[2] Z. Khan, T. Balch, and F. Dellaert, “An MCMC-based particle filter for tracking multiple interacting targets”, ECCV 2004.

[3] K. Okuma, A. Taleghani, N.D. Freitas, J.J. Little, D.G. Lowe, “A Boosted Particle Filter: Multitarget Detection and Tracking”, ECCV 2004.

[4] J. Berclaz, F. Fleuret, and P. Fua, “Robust people tracking with global trajectory optimization”, CVPR 2006.

[5] P. Nillius, J. Sullivan, and S. Carlsson, “Multi-target tracking – linking identities using Bayesian network inference”, CVPR 2006.

[6] C.L. Morefield, “Application of 0-1 integer programming to multitarget tracking problems”, IEEE Trans. on Automatic Control, pp.302-312, vol. AC-22, no.3, 1977.

[7] Aubrey B. Poore, “Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking”, Computational Optimization and Applications, v.3 n.1, pp.27-57, March 1994.

[8] P. P. A. Storms, F. C. R. Spieksma, “An LP-based algorithm for the data association problem in multitarget tracking”, Computers and Operations Research, vol.30 , no.7, pp.1067-1085, 2003.

[9] E.B. Sudderth, M.I. Mandel, W.T. Freeman, and A.S. Willsky, “Visual hand tracking using nonparametric belief propagation”, NIPS 2004.

[10] V. Chvátal, Linear Programming, W.H. Freeman and Co. New York 1983.