Optimizing Multiple Object Tracking and Best View Video Synthesis

Hao Jiang, Sidney Fels, and James J. Little

***Abstract*—We study schemes to tackle problems of optimizing multiple object tracking and best-view video synthesis. A novel linear relaxation method is proposed 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. This scheme models object tracking as multi-path searching. 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 can find the global optimum with high probability. Given the tracking data from our method, optimizing best-view video synthesis using multiple-view videos is further studied, which is formulated as a recursive decision problem and optimized by a dynamic programming approach. The proposed object tracking and best-view synthesis methods have found successful applications in MyView—a system to enhance media content presentation of multiple-view video.**

***Index Terms*—Dynamic programming, linear programming, multiple object tracking, video synthesis.**

I. Introduction

OPTIMIZING multiple object tracking and best-view video synthesis is key for many multimedia applications, such as surveillance, smart rooms, sports analysis and video presentation enhancement. In this paper, we propose an efficient multi-object tracking method based on multiple shortest path searching and a recursive dynamic programming approach for best-view selection. The proposed methods have been successfully applied to MyView, a multimedia system to enhance interactive multiple-view video presentation.

A. Multiple Object Tracking

Tracking is a challenging task when there are complex interactions between targets. It is important to be able to track multiple objects simultaneously to obtain good results [1]. We categorize object interactions into two classes. The first class of interactions constrain an object's relative location, 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 tracking objects separately. Naive exhaustive search quickly becomes intractable as the problem scales up. More efficient schemes such as extended dynamic programming [1] are still too complex to be applied to problems with a medium number of observations and objects. Thus, for large scale problems, approximation schemes are preferred. We propose a linear relaxation scheme [8] 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. Fig. 1 illustrates an example of tracking squash players in a double match using the proposed method.

Object tracking has been studied extensively. For example, Kalman filtering [3] has been a classic scheme for single object tracking using a Gaussian noise model. Particle filtering [4] is a more general sequential inference method. It has been used for tracking multiple objects such as ants [2] with complex interactions. Particle filtering has also been studied for tracking hockey players [9] in which object interaction is not explicitly modeled. Finite set statistics [7] have recently been discovered to be able to unify the recursive filtering formulation for both multiple object and single object tracking. Unfortunately, optimal multiple object recursive filtering is computationally demanding. Approximation searching schemes need to be applied.

Different schemes have been studied to improve the efficiency of optimization. Multiple hypothesis tracking (MHT) [5], [6] updates a subset of best hypotheses and delays the objects states decision. Greedy schemes [12], [13] have also been proposed for finding trajectories of feature points in image sequences.

Dynamic programming (DP) is widely applied in 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 in each frame, 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 [10] has been studied to find objects paths with heuristics used to determine the sequence of paths assignments in a multiple-camera setting. While simple heuristics such as best-track-first assignment have low computational complexity, it does not always give correct solutions when objects have complex mutual occlusions.

Belief propagation (BP) [17] is another approach which has been used for optimizing hand tracking. Occlusion is explicitly

Manuscript received December 18, 2007; revised April 20, 2008. Current version published October 24, 2008. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Alex C. Kot.

H. Jiang is with the Computer Science Department, Boston College, Chestnut Hill, MA 02467 USA (e-mail: hjiang@cs.bc.edu).

S. Fels and J. J. Little are with the University of British Columbia Vancouver, BC V6T 1Z4 Canada (e-mail: ssfels@ece.ubc.ca; little@cs.ubc.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2008.2001379

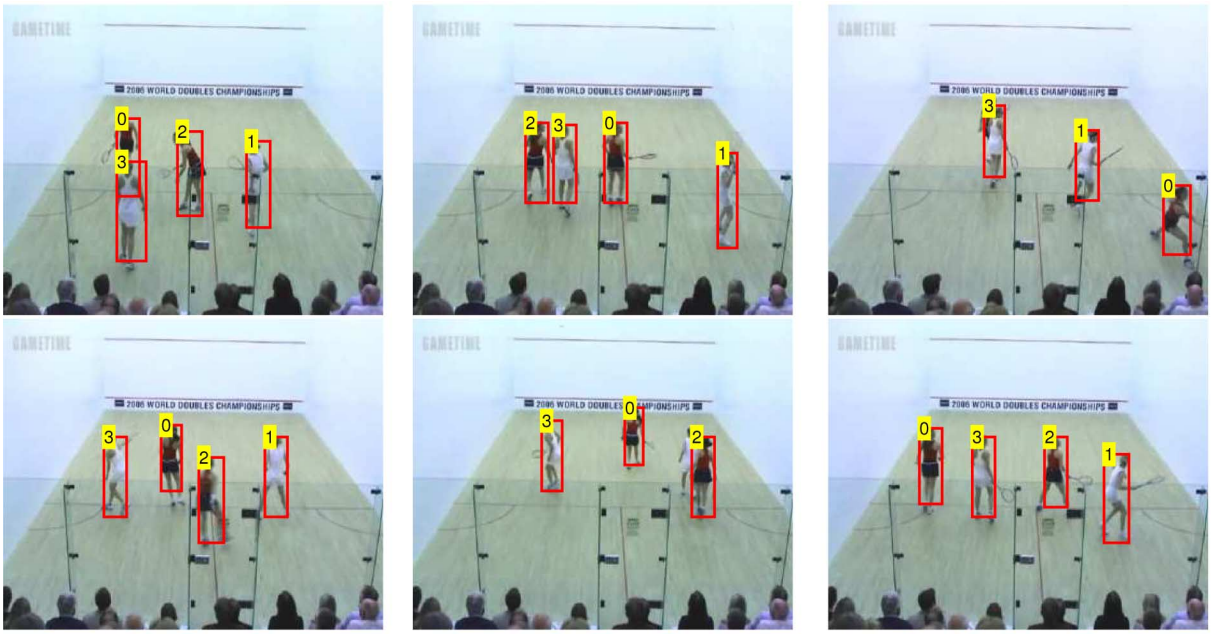


Fig. 1. Example frames from tracking Squash players in a doubles match. The algorithm assigns labels to the different players as they move around. Notice in the third and fifth frames that one player occludes another player. Our algorithms handles this occlusion by representing this situation as a distinct state.

modeled in this method. However, multiple object tracking results in a loopy graph structure making it difficult to guarantee convergence to a global optimum. Recently, message propagation in Bayesian networks has been applied to optimizing trajectories of football players in videos [11]. This approach does not consider track interaction among objects.

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 [14] has been studied for radar data association. This formulation is different from our 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 are studied in [15], [16], which turn out to be quite similar to the sequential DP method [10]. Unlike previous LP methods, our proposed scheme is based on a multiple-shortest-path model that tries to connect edges into paths and therefore has much fewer variables.

Even though intensively studied, robust and efficient tracking of multiple objects with complex interactions remains unsolved. We propose a novel global 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 penalty terms are included to ensure that the layout of the objects is consistent over time, 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 program relaxation effectively solves the path searching problem when paths overlap and objects occlude each other. The relaxation solution is then rounded to an integer solution to obtain the object locations and occlusion states. As our results illustrate, the linear program almost always yields integer solutions that globally optimize object tracks and has low order polynomial average complexity.

B. Best View Selection

Apart from tracking multiple objects simultaneously, another important task is automatically selecting the best views for a specific object and generating a smooth and natural video sequence. Different approaches have been proposed for best-view selection based on view quality measurements. In computer graphics, measurements such as the projection of polygons or the view entropy [19] can be used for best-view selection. Selecting the best view in computer animation [20] also uses measurements such as visibility, relevance, redundancy and eccentricity and an optimization step can be further applied to the whole sequence. Face appearance [22] is a natural feature for determining the view quality in video surveillance applications. 3-D locations of objects and their relative poses to cameras are other features [21] often used in best-view selection. Even though view quality measurements have been extensively studied, there is still little work about optimizing best-view video sequence synthesis using noisy view quality measurements. A naive method of video synthesis by selecting the best view at each time instant simply based on view quality usually works poorly, since it contains annoying abrupt view changes due to unavoidable view quality measurement errors.

We propose a new method to optimize the synthesis of a smooth view transition sequence based on noisy view quality measurements. The best view video synthesis is optimized using a dynamic programming approach. The optimization works in a sliding window fashion that involves one previous best-view decision and a sequence of buffered best-view candidates. Dynamic programming globally optimizes each video segment based on the quality of each view and a smoothness

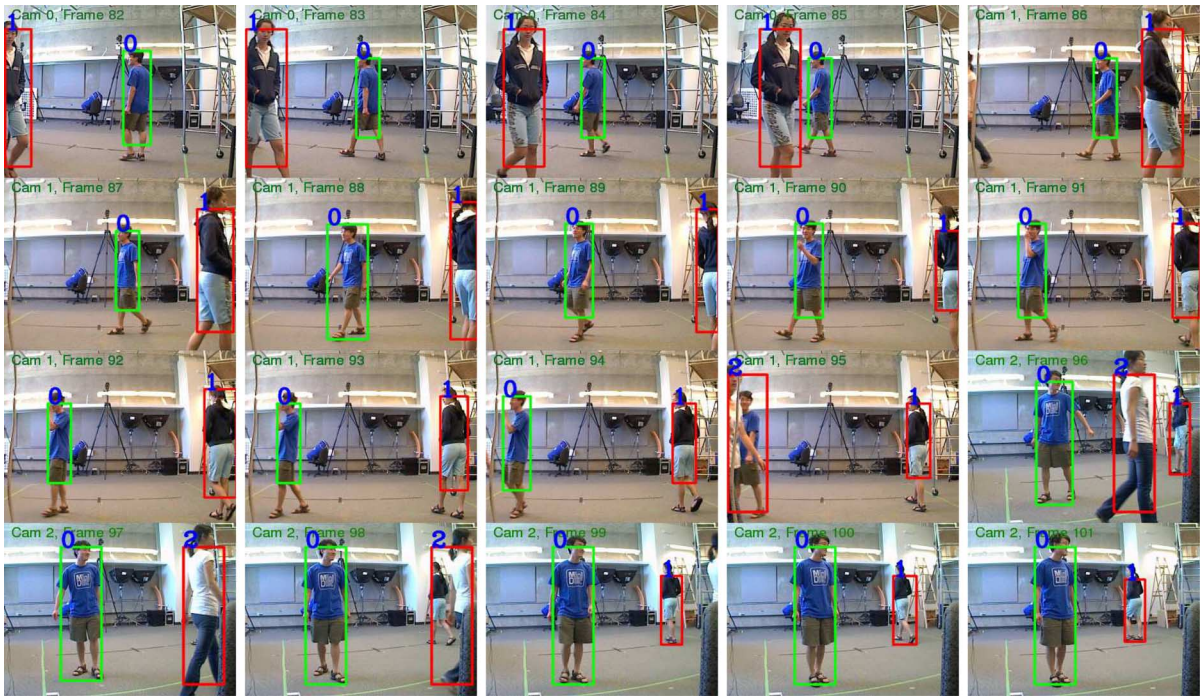


Fig. 2. Segment of best-view video for object 0 based on the dynamic programming method from three camera views. The bounding boxes show the tracked objects using the linear relaxation method. The green bounding boxes indicate the target object of the best-view video. The best-view measurement is based on the size of the bounding boxes.

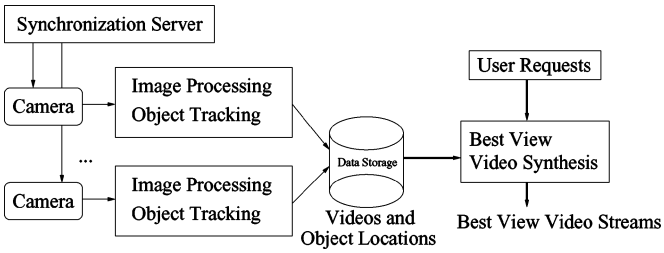


Fig. 3. Block diagram of the MyView system. Camera data is streamed to the server with tracking meta-data added based on our LP tracking algorithm. Clients request "best" view video data of objects which are streamed to the client's view application.

constraint. Experiments show that the proposed scheme generates pleasing and smooth synthesis best-view video focusing on specific objects. Fig. 2 shows a best-view sequence for object 0 generated in real time from three synchronized camera views using the proposed scheme.

C. Application in MyView

The object tracking and best-view selection algorithms form the basis of a system denoted as MyView whose block diagram is shown in Fig. 3. MyView is a media system for enhancing multiple-view video presentation. In MyView, multiple cameras capture video simultaneously of scenes such as sports events. Objects in the videos are automatically tracked using our proposed scheme in real time. Based on users' requests, synthesized video sequences focusing on the "best" view of a specific object is generated on the fly and streamed to the user client application running on a computer such as a laptop PC or handheld device with multimedia support.

In the following sections, we elaborate the proposed method in tracking, best-view video synthesis and their application in MyView. The arrangement of the paper is as follows. In Section II, we describe the multiple path searching model for object tracking, the optimization formulation and the linear relaxation method. In Section III, we describe the dynamic programming approach for optimizing best-view video synthesis. Results of object tracking and best video synthesis in MyView are discussed in Section IV. Section V concludes the paper.

II. Optimizing 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 the locations of objects 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. 4. 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

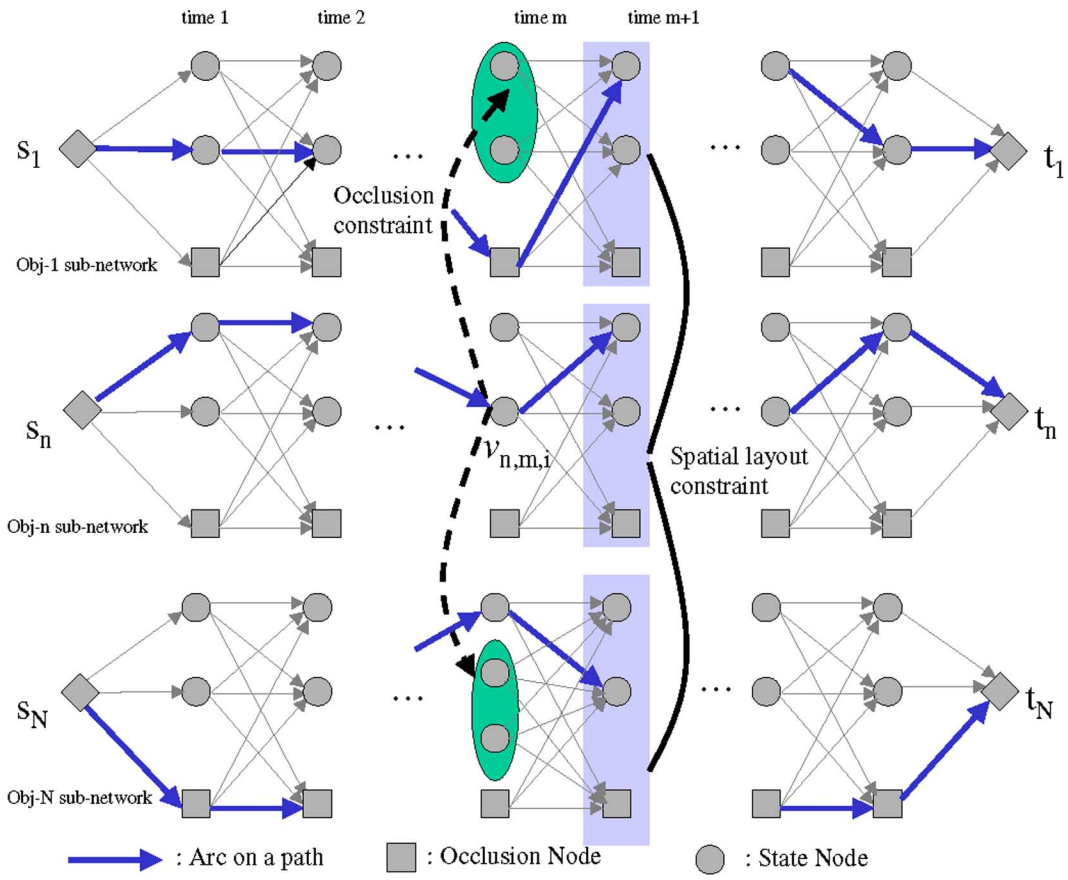


Fig. 4. Network model for multiple object tracking. Round nodes represent possible locations and appearance states of objects. Nodes shown as rectangles represent the state when an object is not in the frame due to occlusion. Nodes conflicting with a round node at a given time are shown in ovals. The spatial locations for the objects at each time instant also subject to spatial layout constraints. A spatial trajectory of a given object is a path through the graph from the source node to the sink node as shown by an arc. The LP algorithm finds that path.

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.

A. Problem Statement

In multiple object tracking, we need to locate objects 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 the scene.

B. Network Model of Multiple Object Trajectories

In the following, we study multiple object tracking based on a network model in which submodels in our formulation interact with each other. This approach contrasts with the previous trellis model used in single-chain dynamic programming. Fig. 4 illustrates the network model of the multiple object tracking that we use.

In Fig. 4, 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 subnetwork for each object may contain a different number of nodes. The nodes represented as rectangles in Fig. 4 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. 4 are also included for each object subnetwork 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. 4 the spatial conflict set for node $v_{n,m,i}$ includes the node itself and nodes in the ovals in the other object subnetworks that would overlap the region of $v_{n,m,i}$. As an example, in Fig. 5, objects one and two conflict in space if the two bounding boxes are sufficiently close. Note that the occlusion node for each object never

has a spatial conflict, so it will never be in a spatial conflict set. This setting enforces an object to either occupy a spatial location that does not conflict with other objects or disappear in the video frame. 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 object 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. 4 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 shown at the bottom of the page, where function $g(\cdot)$ compares the similarity of an object appearance corresponding to nodes in the network, e.g., by comparing color histograms in bounding boxes; $d(\cdot)$ 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. c^a_{const} compensates for the mean position and color difference for an object at successive frames. c^b_{const} is a constant greater than the color histogram difference for the same object and less than the one for different objects. These constants are therefore the functions of sensor detection, state transition, death and spawning probabilities and can be set by training. But, they are not sensitive and therefore manually setting is found sufficient. 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.

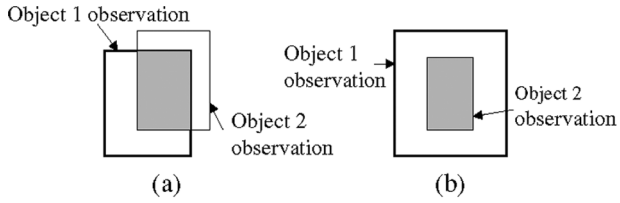


Fig. 5. Overlapped regions. (a) Partially overlapped regions; (b) Fully overlapped regions.

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 subnetworks 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. 5(a). The second one includes completely overlapped regions as shown in Fig. 5(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.

Apart from the occlusion constraint, we also would like to keep the spatial layout of objects stable over a short period of

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

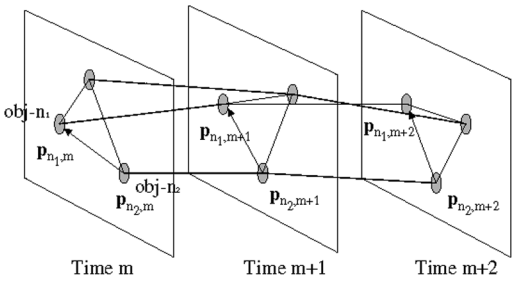


Fig. 6. Spatial layout consistency: the relative locations between each pair of objects tend to keep unchanged through time.

time. To model this constraint, we keep the spatial displacement vectors between objects as similar as possible across time. As shown in Fig. 6, 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 2-D. 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.

C. Discrete Optimization

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

$$\begin{aligned}&\min_{\text{paths}}\sum_{n,m}\sum_{(v_{n,m,i},v_{n,m+1,j})\text{ on a path}} c(v_{n,m,i},v_{n,m+1,j})\\&\quad+\mu\sum_{m}\sum_{\{n_1,n_2\}\in\mathcal{N}}\|(\mathbf{p}_{n_1,m+1}-\mathbf{p}_{n_2,m+1})\\&\quad-(\mathbf{p}_{n_1,m}-\mathbf{p}_{n_2,m})\|\\&\text{s.t. at most one path goes through}\\&\quad O(v_{n,m,i}),\forall v_{n,m,i}\end{aligned}$$

where $\mathbf{p}_{n,m}$ is the location of object n at time instant m. For instance, if we use bounding boxes to quantify 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 xy coordinate of the bounding box and $(p_{n,m,3}, p_{n,m,4})$ is the right-bottom corner xy 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 $\|\cdot\|$ 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 a 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.

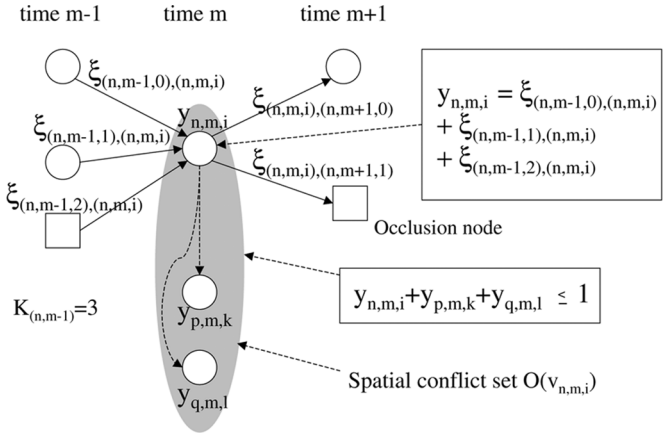


Fig. 7. Linear programming formulation. Each edge in the network is associated with a variable ξ and each node with a variable y. These variables are 1 if the corresponding edges or nodes are on the shortest paths and otherwise 0. The nodes in the gray regions correspond to the conflict nodes in which only one of them can be on the shortest paths.

D. Linear 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. 7 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:

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

subject to

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

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

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

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

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

$$m = 1 \cdots M, \forall n, i$$

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

$\phi_l(\cdot)$ 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 common linear programming trick [18] 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. 7.

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 program 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 0 s 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.

If the solution is not integral, we have to round the result into an integer solution. The rounding process is as follows. At each time instant, we find the maximum y in the corresponding column of each object subnetwork. Recall that y indicates the probability of an object occupying a spatial location or in the occlusion state. We then round the largest y of all nodes at the time instant to 1 and y of its conflict nodes and nodes on the same object subgraph at this time instant to be 0. These nodes are then labeled as visited. We repeat the above process to set the next largest y to be 1 and zeros the other conflict nodes and nodes in the same object subgraph. We repeat this process until we traverse all the nodes. Now, we have a rounding result, where the non-zero $y_{n,m,i}$ indicates object n at time m has state i, which either corresponds to a spatial location or an occlusion state. The simplex method for linear programming has exponential complexity in the worst case. Linear programming is fast for real applications [18]; for our LP formulation, its average complexity is approximately $O(n^2km)(2\log(k) + 2\log(n) + \log(m))$, 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 two 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 two 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 subnetworks 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. The tracking result is shown in Figs. 8 and 9. The top-left corner x and y-coordinate of the bounding boxes for both objects are shown in Fig. 8(a) and (b). For this example, LP relaxation has integer solutions for ξ and therefore achieves its global optimum. As shown in Figs. 8 and 9 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

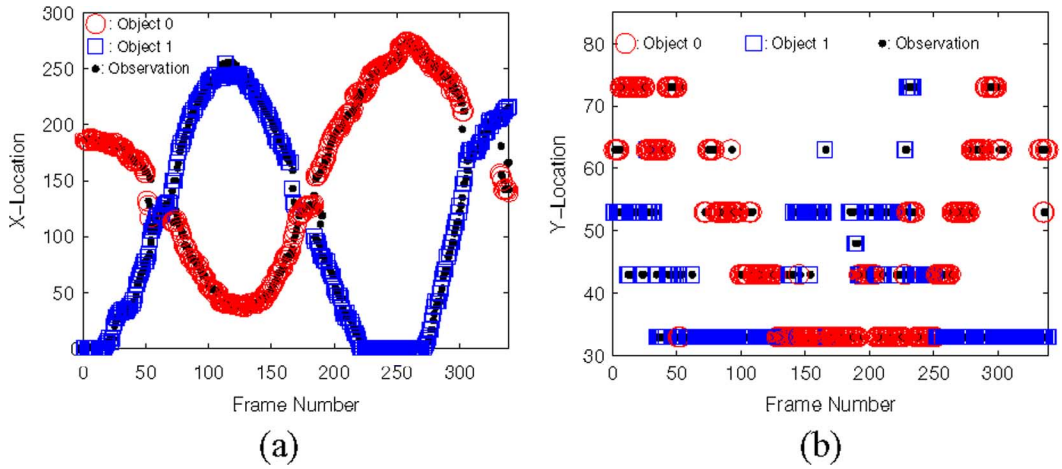


Fig. 8. Tracking result of the proposed method for example 1.

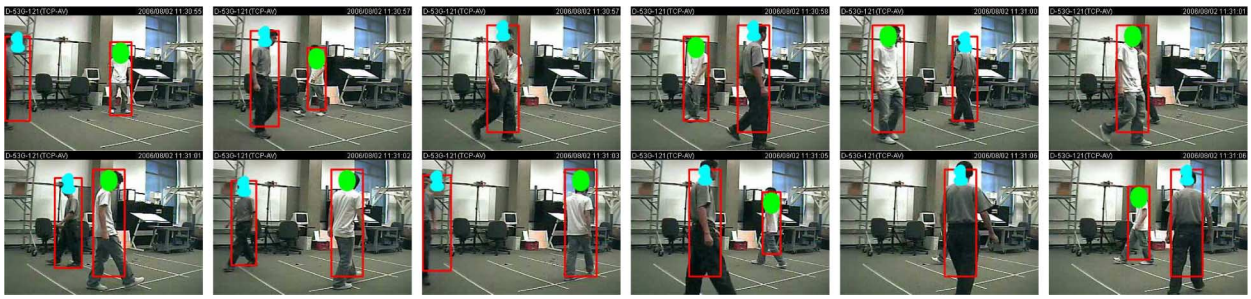


Fig. 9. Selected images of tracking two objects in 340 successive video frames using the proposed scheme. Green and blue labels indicate object 0 and 1 respectively.

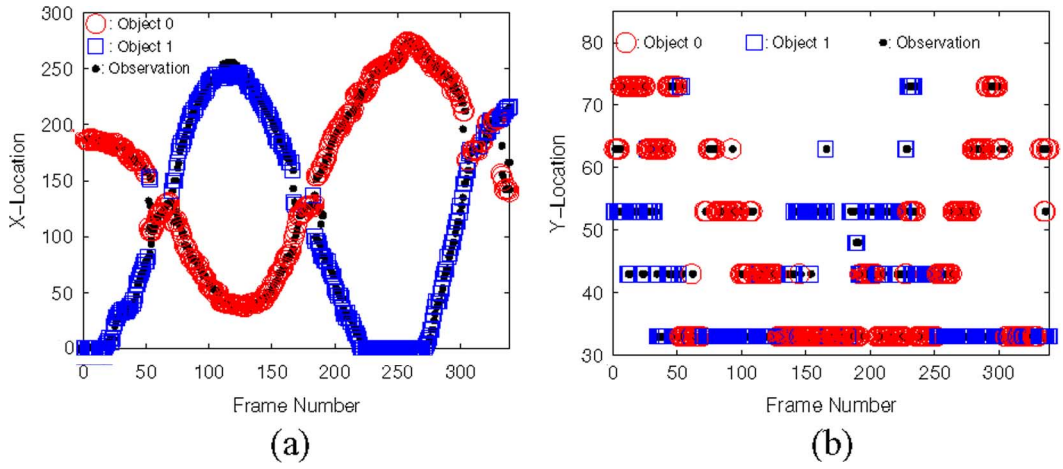


Fig. 10. Tracking result of approximate DP for example 1.

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. Figs. 10 and 11 show the tracking result of approximate 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.

E. 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 one 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.

III. Optimizing Best View Sequence Synthesis

In this section, we further study an algorithm to optimize the best-view selection so as to generate a smooth video sequence. Finding the best view in a scene relates to the objects that are in the view, thus, is a natural extension to the tracking objective in the previous section. The locations of objects in video are represented as bounding boxes in each video frame. For each object, there is a "best" view in all the camera views. The best view is extracted from among all the camera's views by using some view quality measure. For example, the size of the bounding box for an object at one time instant could be used for such measure. Other measurements such as face orientation, body pose and object activity in the scene could also be taken into consideration. The focus of our work is not to improve or define the best-view measurement but rather to design a scheme to optimize the best-view video synthesis using a given noisy best-view measurement that has been defined for the particular content in the scene.

The naive method of selecting the best view at each time instant based on view quality measurement does not typically yield smooth video transitions. Errors and noisy measurement cause error decisions and result in poor synthesized video with fast cuts from one view to another. To solve this problem, we propose to jointly optimize a cost function that depends on both the view quality and the view transition smoothness so that a pleasing best-view video sequence can be composed. As described below, our approach is like a constrained filter that attempts to optimize the best-view selection over time constrained by the temporal smoothness. The best-view decision filter works in a sliding window fashion, in which the sliding window spans from a previous frame (indexed as -1), the current frame (0) to the next $N-1$ frames. The optimization problem for each sliding window is written as

$$\min_{b_n, n=0\cdots N-1} \left\{ \sum_{n=0}^{N-1} c(n, b_n) + \lambda \sum_{n=0}^{N-1} d(b_n, b_{n-1}) \right\}$$

where n is the frame number; b_n is the best view for time n and b_{-1} is the previous labeled best view; $c(n, b_n)$ is the cost of selecting b_n as the best view at time instant n. $d()$ is a distance function that penalizes jumps from views in adjacent time instants. Here we define $d(x, y) = \delta(x - y)$, which thus has a 0 cost for choosing the same camera view for two successive time instants or a constant cost otherwise. The output of the optimization is a sequence of view selections $\{b_n^*, n = 0 \ldots N-1\}$ for the current sliding window. We pick the best view for the current frame as b_0^*. The sliding window is then moved to the next frame and we repeat the optimization procedure iteratively.

The above optimization problem can be solved using dynamic programming. The procedure of best-view estimation based on DP is as follows. We assume that the sliding window length is N and there are a total of M views in the optimization (i.e., M cameras). Remember, each frame has an associated view quality measurement which changes depending upon the context requiring this algorithm to possibly run many times depending upon the user's needs.

Algorithm: Best View Selection

1. Choose b_{-1} as the view with the best-view measurement;
2. $t(-1, b_{-1}) = 0,\ t(-1, m) = +\infty,\ m = 0..M-1,\ m \neq b_{-1}$;
3. For $n = 0$ to $N-1$
   For $m = 0$ to $M-1$
   $v_{n-1} = \arg\min_{i \in [0..M-1]} \{\lambda d(m, i) + t(n-1, i)\}$;
   $t(n, m) = c(n, m) + \lambda d(m, v_{n-1}) + t(n-1, v_{n-1})$;
   $\text{prev}(n, m) = v_{n-1}$;
4. $f_{N-1} = \arg\min_{m \in [0..M-1]} t(N-1, m)$;
5. For $n = N-2$ to 0
   $f_n = \text{prev}(n+1, f_{n+1})$;
   Output current best view $b_0^* = f_0$;
6. $b_{-1} = b_0^*$; Move the sliding window forward; Goto 2;

IV. Experiment Results for Tracking and Best-View

This section describes results obtained for both our LP tracking approach (Section IV-A) and Best-View method (Section IV-B). The tracking results are considered for various settings including moving toy objects, people moving in a lab setting, a single squash games and a multi-person squash games which is quite challenging. We also compare the tracking using LP with a DP approach for some of the experiments. The best-view study looks at people in a lab setting to present how it works in MyView.

A. Linear Relaxation Tracking

We report our results using our method for tracking multiple objects on four different video sequences. These video sequences are in CIF format with frame rates ranging from 15 to 30 frames/s.

1) Tracking Two Stuffed Animals: Fig. 12 shows the tracking result of the proposed method for a 307-frame video. Two toy

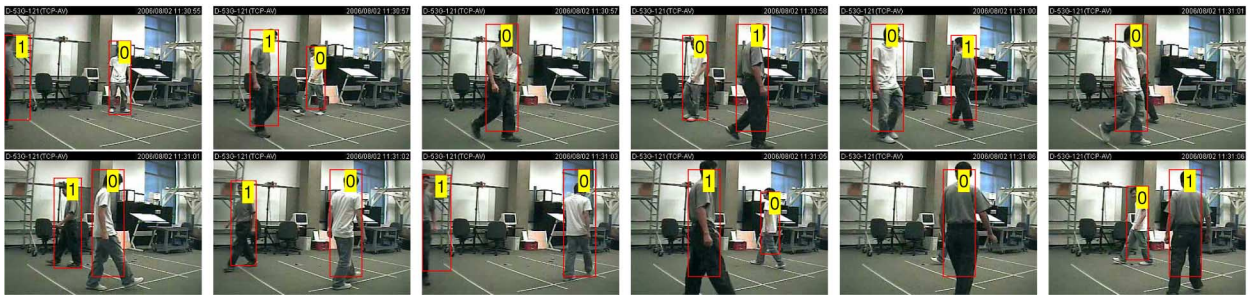


Fig. 11. Selected images for approximate DP result in Example 1.

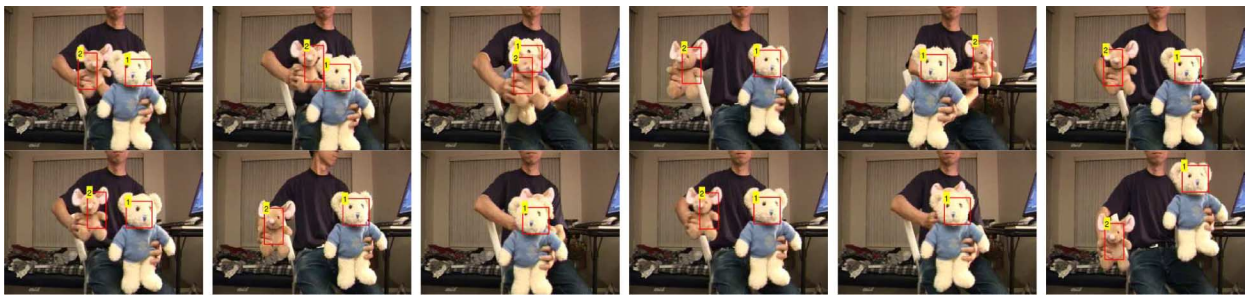


Fig. 12. Tracking two toy objects with the proposed scheme. Selected frames from 307 frames.

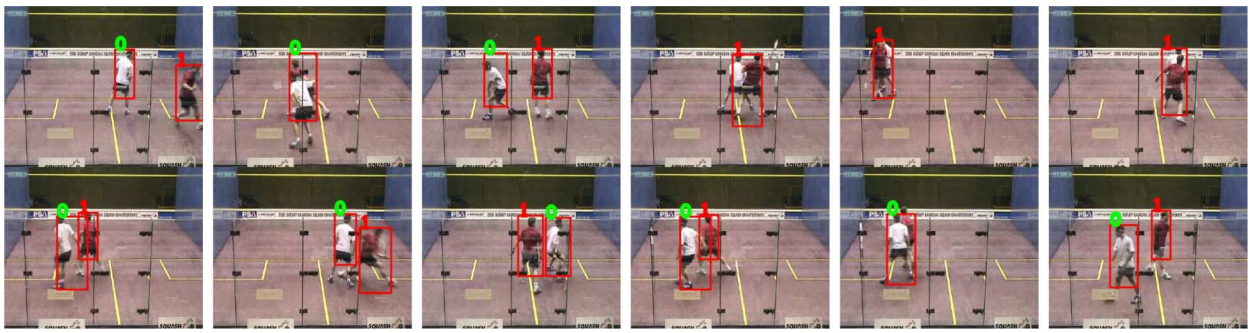


Fig. 13. Squash. Selected frames from 1351 frames.

objects are tracked through the video sequence. There are complex occlusions between the two objects. The templates for the two objects are set using the first video frame. A subimage 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.

2) Tracking Fast Moving Squash Players: In another experiment, we apply the proposed scheme to a 1351-frame squash video sequence with two players as shown in Figs. 13 and 14. The candidate objects are detected by background subtraction similar to the method used in [10]. The thresholded binary background subtraction image is convolved with rectangular boxes with width-height ratio of a standing pose human in different scales. The result is a 3-D image with coordinates x–y and scale. Non-maximum suppression is then applied to the 3-D image to find all the possible object locations and scales. The bounding boxes further shrink to fit objects boundaries tightly. The video includes complex object interaction and mutual occlusion. Noisy background subtraction also makes object tracking a hard task.

In this experiment, we convert the color sequences into grayscale ones and use a rough 64-bin histogram for the features. The proposed linear program relaxation is then applied to the video sequence in a sliding window fashion as done in the first experiment. The proposed scheme accurately follows the object locations through the video sequence. Fig. 13 illustrates sample frames of the tracking result and Fig. 14 shows the object locations at each time instant through time (occluded objects are not shown). In the 1351-frame video sequence, object 0 has seven wrong label assignments and object 1 has five wrong detections. The average object tracking error rate is about 0.01 per frame for this example. LP also has a high probability of directly obtaining the global optimal solution.

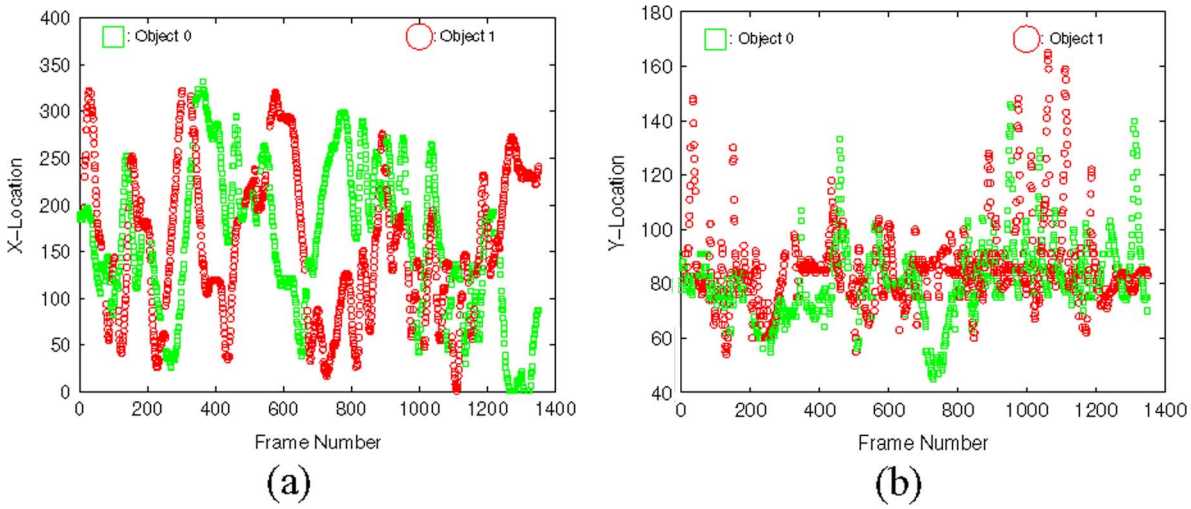


Fig. 14. Object locations for two squash players. (a) X-Locations of objects; (b) Y-Locations of objects.

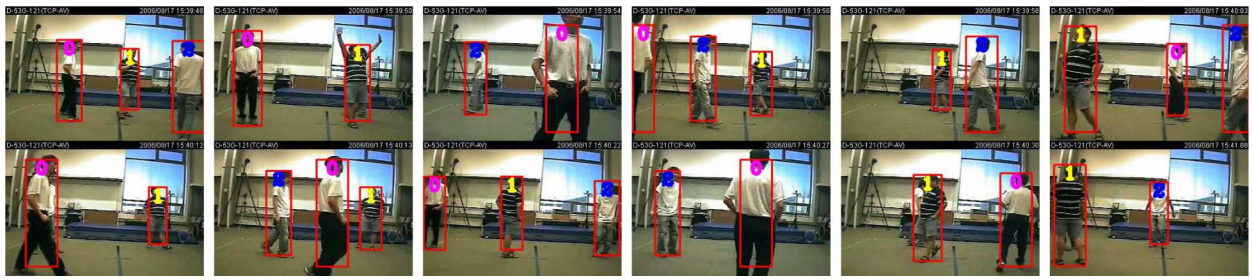


Fig. 15. Tracking three people with the proposed scheme. Selected frames from 2431 frames.

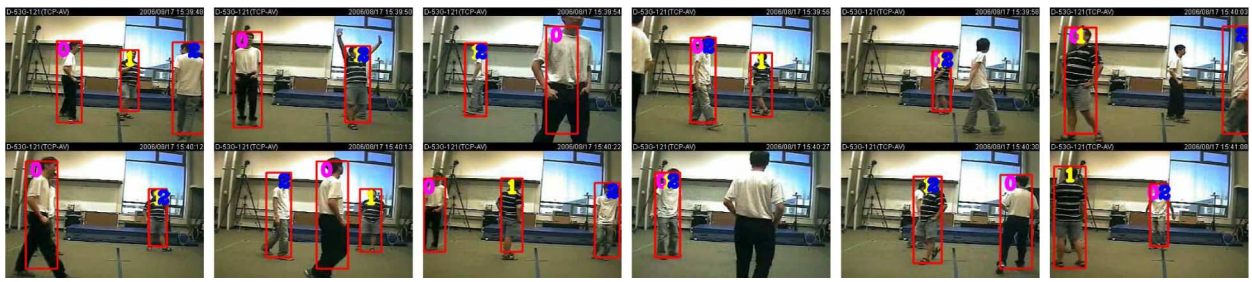


Fig. 16. Tracking three people with separate DP for each object. Selected frames from 2431 frames.

Only three segments do not have fully integer solutions for ξ in 75 video segments.

3) Comparison With DP on Tracking Three People Walking in an Office: Fig. 15 and and Fig. 18 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 Section II-E.

The proposed scheme can deal with complex occlusions and objects moving out of the scene and coming back. Object 0 has five wrong detections, object 1 has 22 wrong detections and object 2 has 125 wrong detections. Overall the error rate is 0.06 per frame. In this experiment, four 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. 16. 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 error rate is 0.17 per frame. Fig. 17 shows sample video frames where the LP approach improves the tracking result.

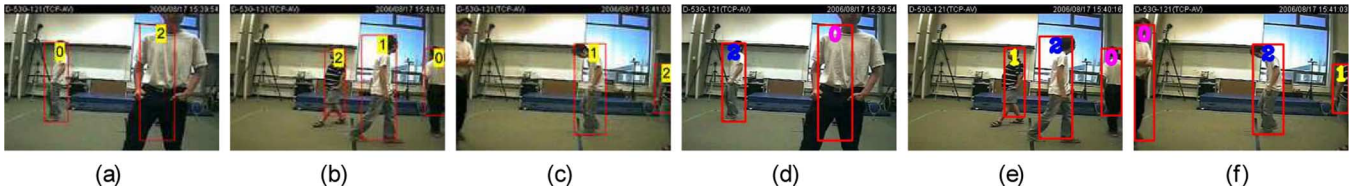


Fig. 17. Sample frames where DP with simple heuristics does not yield correct solution while the proposed scheme does. (a)–(c) show sequence DP frames. (d)–(f) show results with the proposed method.

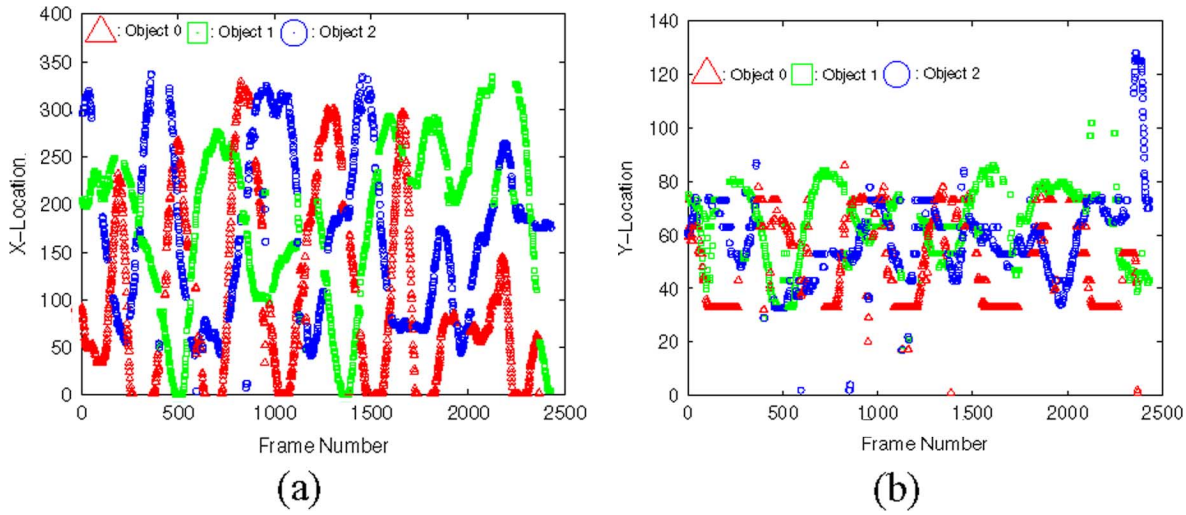


Fig. 18. Objects locations for 3-people tracking. (a) X-Locations of objects; (b) Y-Locations of objects.

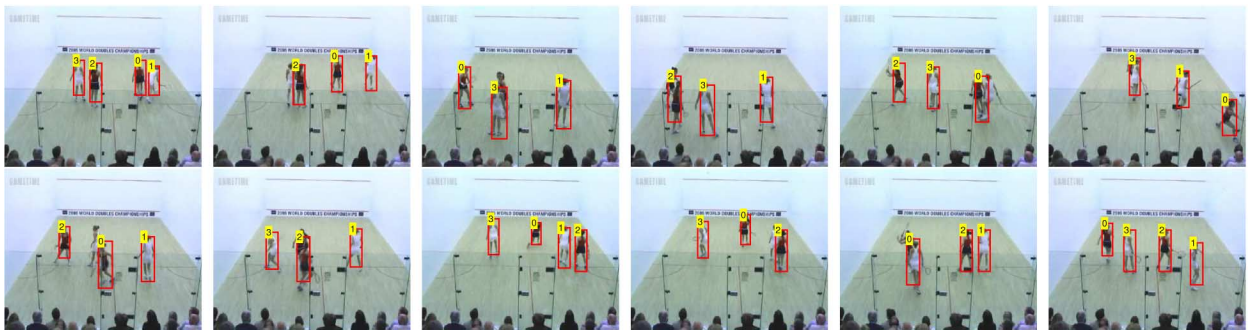


Fig. 19. Double Squash. Selected frames from 500 frames.

4) Tracking 4 Players in a Double-Squash Game: In Figs. 19 and 20 we applied our method to a 500-frame double-squash video sequence. There are four objects in the video and there are about ten 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 using a sliding window. 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 three minutes using a 2.6 GHz PC which is much faster than extended DP which needs about an hour to compute the result. Since the LP solution is very near the global optimum, branch and bound converges very quickly. We use the branch and bound method here to obtain a global optimum so that we can have a fairer comparison with extended DP.

As shown in Figs. 19 and 20, the tracker works well in following multiple objects during a long sequence. In Fig. 20, 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.

5) Processing Time Considerations: Fig. 21 shows typical average running times of the linear program using a 2.6 GHz PC. Random observations and color histograms are generated in

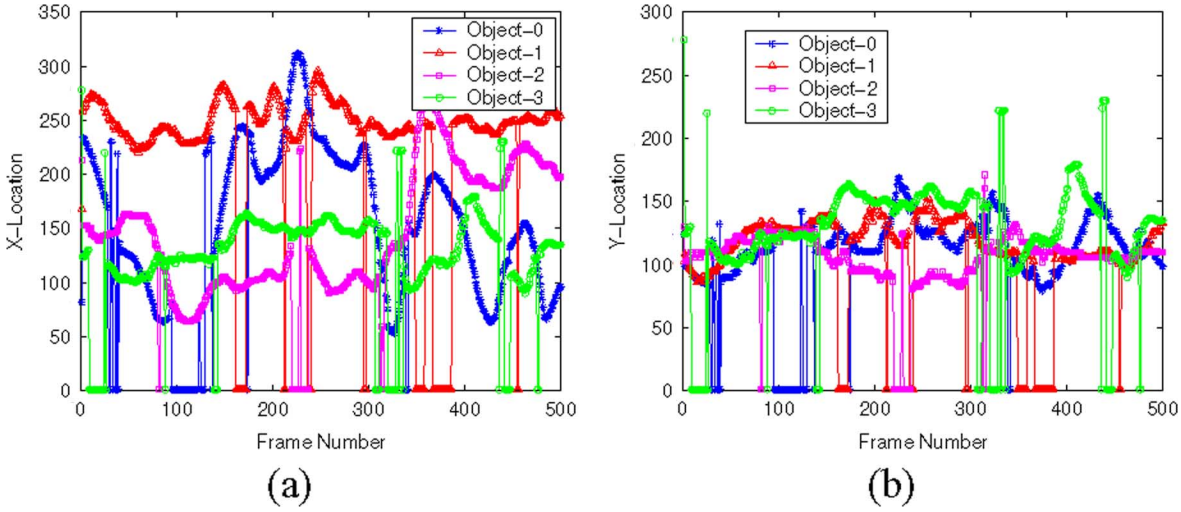


Fig. 20. Objects locations for four squash players. (a) X-Locations of objects; (b) Y-Locations of objects.

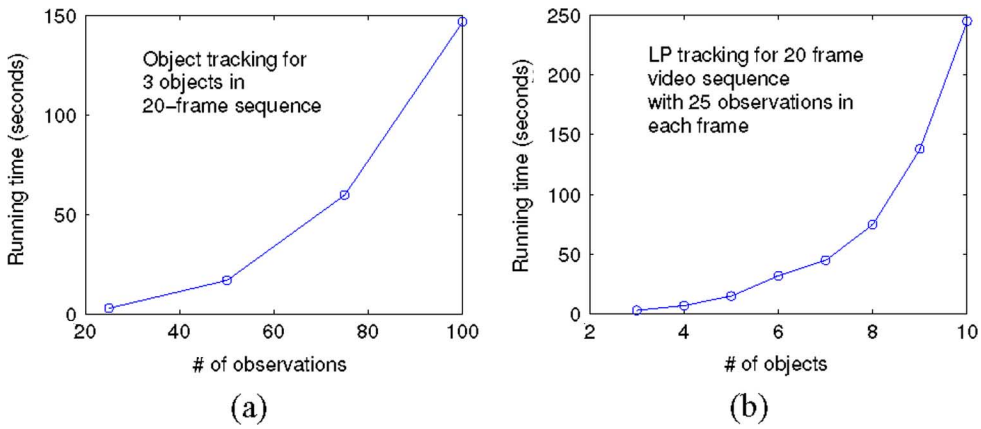


Fig. 21. Complexity of the proposed scheme.

TABLE I
TYPICAL RUNNING TIME FOR THE PROPOSED METHOD AND EXTENDED DP FOR OPTIMIZING 20 VIDEO FRAMES WITH THREE OBJECTS AND 50 OBSERVATIONS IN EACH FRAME

	This Paper	Extended DP
Time	20 secs	18800 secs

each frame. Each experiment is repeated ten times and running times are averaged. Fig. 21(a) shows the typical running time of our method for different numbers of observations. Fig. 21(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 as shown in Table I, extended DP takes more than five hours to optimize three objects in 20 video frames with 50 observations in each frame, while the proposed scheme converges in tens of seconds. Thus, our method requires considerably less computation time than extended DP and still achieves good accuracy.

B. DP Best View Selection

We test the best-view selection method for multiple-view video sequences captured in our MyView system. In this experiment, three cameras captured videos synchronously. Network cameras are used in the experiment. The low quality of the video makes reliable object tracking a challenge. Camera synchronization is done by pulses propagated over TCP/IP along an Ethernet generated by a synchronization server. There are three human subjects moving around in the scene. Because our tracking method works robustly even when there are occlusions and complex object interactions, we can track multiple objects using each camera independently providing good estimates of the three subjects in each view. Video from each camera is processed by a 2.6 GHz Linux machine. Object tracking is done online with the frame rate of about 20 frames/s. The quality of view measure for a given object is estimated by the size of the bounding box for that object, thus, we assume that we want to see views of an object that show the largest view. The best-view video for each object is also generated on the fly by the video server, which optimizes the synthesized video by the proposed

Fig. 22. Selected best-view video frames. Row 1: Best-view video frames for object 0. Row 2: Best-view video frames for object 1. Row 3: Best-view video frames for object 2. Each column corresponds to the same time instant. Notice that in each view the object of interest is generally large and close to the camera selected.

Fig. 23. Video generated focusing on object 0 using videos of three video cameras. The bounding box of the focus object is shown as green.

dynamic programming method and then streams the video to the client.

Fig. 22 shows the best-view selection result for each of the three objects. Each column corresponds to the same time instant for the multiple-view video; Each row corresponds to the selected best-view frames for a given object. Bounding boxes are tracking results using the linear relaxation method. The green bounding box indicates the focus object in each row. Fig. 22 clearly shows the effectiveness of the proposed method in focusing the best view on the specific objects.

Figs. 23–25 are expanded video sequences of Fig. 22 for each object, which illustrates the smooth video transitions. The camera selection curves are shown in Fig. 26(a)–(c). Experiments showed that the dynamic programming approach not only yields the optimum decision in a recursive fashion but also generates visually pleasing best-view video sequences.

V. Conclusion

In this paper, we address the problems of optimizing object tracking and best-view video synthesis. 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 global 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 and has a high chance of obtaining the global optimum. It also gives better results in finding multiple trajectories simultaneously than greedy search schemes such as sequence DP. We further propose a dynamic programming approach for optimizing best-view video sequences. Realtime video tracking and best video generation have been implemented in our MyView application. The proposed tracking and best-view selection methods are also useful for applications such as surveillance, smart rooms, sports game analysis, and broadcasting systems. Object tracking in very large systems such as tracking players on a sports field using hundreds of cameras is still a challenging problem in both computation complexity and robustness. We are now actively studying how to apply the proposed method to real-time large scale

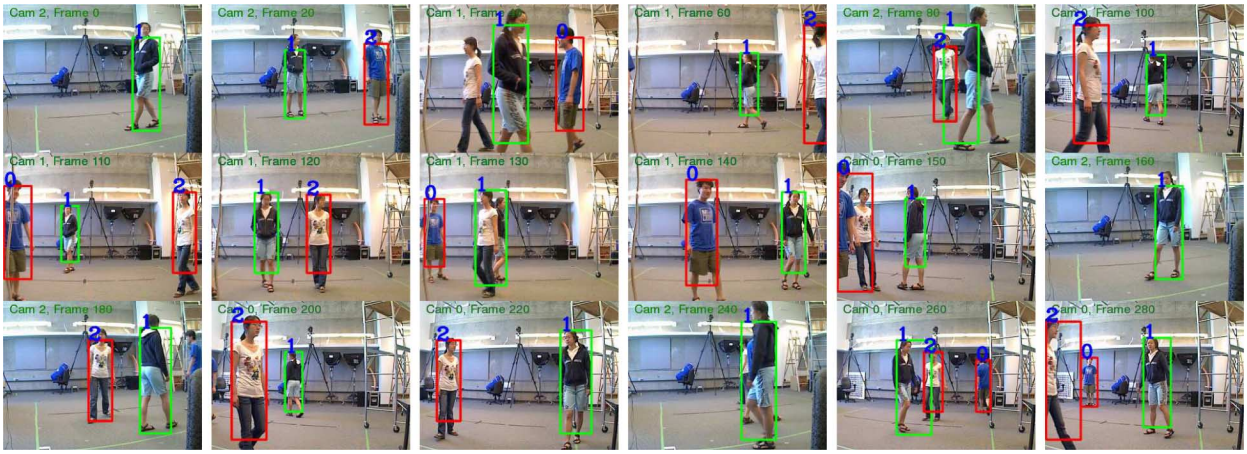


Fig. 24. Video generated focusing on object 1 using videos of three video cameras. The bounding box of the focus object is shown as green.

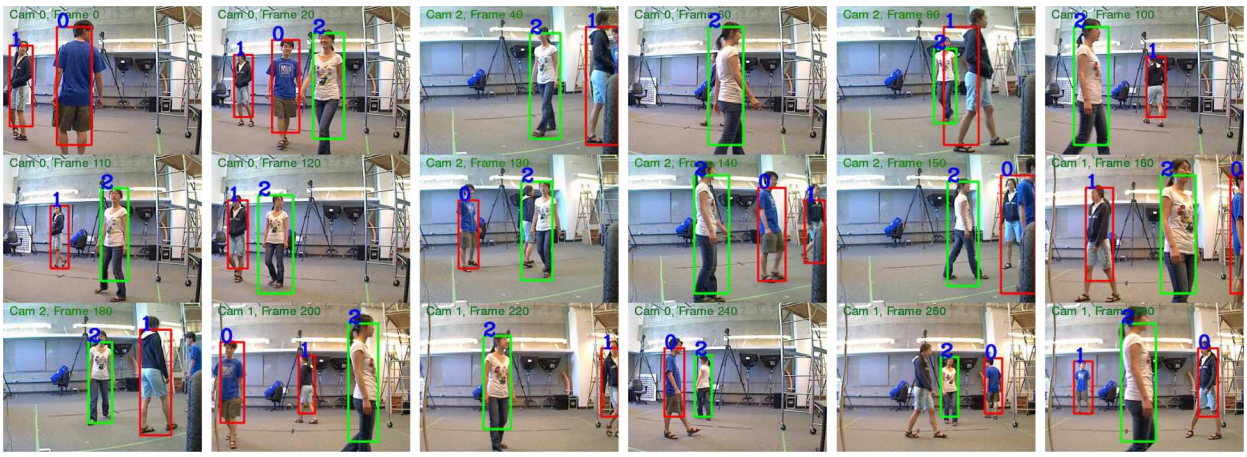


Fig. 25. Video generated focusing on object 2 using videos of 3 video cameras. The bounding box of the focus object is shown as green.

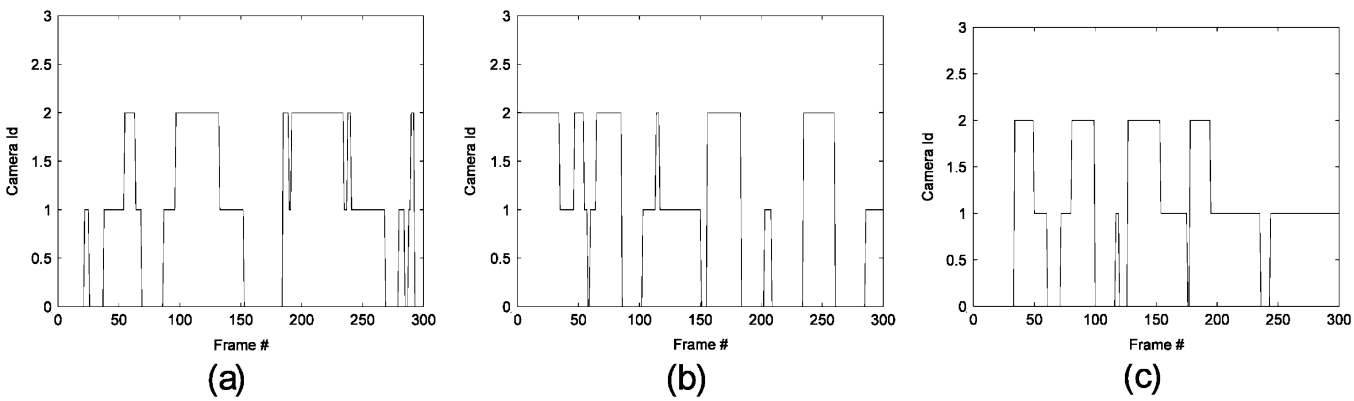


Fig. 26. Camera selection through time. (a). View selections for object 0. (b). View selections for object 1. (c). View selections for object 2.

systems such as a camera network for surveillance and sports broadcasting.

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. Aerosp. Electron. Syst.*, vol. AES-25, no. 2, pp. 287–295, 1989.

[2] Z. Khan, T. Balch, and F. Dellaert, "An MCMC-based particle filter for tracking multiple interacting targets," in *Eur. Conf. Computer Vision*, 2004.

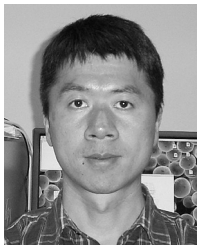
[3] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association.*. New York: Academic, 1988.

[4] G. Kitagawa, "Non-Gaussian state-space modeling of nonstationary time series," *J. Amer. Statist. Assoc.*, vol. 82, pp. 1032–1063, 1987.

[5] D. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. Automat. Contr.*, vol. AC-24, pp. 843–853, 1979.

[6] I. Cox and S. Hingorani, "An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, no. 2, pp. 138–150, 1996.

[7] R. P. S. Mahler, "Multitarget Bayes filtering via first-order multitarget moments," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 39, no. 4, pp. 1152–1178, 2003.
[8] H. Jiang, S. Fels, and J. J. Little, "A linear programming approach for multiple object tracking," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2007.
[9] K. Okuma, A. Taleghani, N. D. Freitas, J. J. Little, and D. G. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *Eur. Conf. Computer Vision*, 2004.
[10] J. Berclaz, F. Fleuret, and P. Fua, "Robust people tracking with global trajectory optimization," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2006.
[11] P. Nillius, J. Sullivan, and S. Carlsson, "Multi-target tracking—linking identities using bayesian network inference," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2006.
[12] I. K. Sethi and R. Jain, "Finding trajectories of feature points in a monocular image sequence," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, pp. 56–73, 1987.
[13] V. Salari and I. K. Sethi, "Feature point correspondence in the presence of occlusion," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 87–91, 1990.
[14] C. L. Morefield, "Application of 0-1 integer programming to multitarget tracking problems," *IEEE Trans. Automat. Contr.*, vol. AC-22, no. 3, pp. 302–312, 1977.
[15] A. B. Poore, "Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking," *Comput. Optim. Applicat.*, vol. 3, no. 1, pp. 27–57, Mar. 1994.
[16] P. P. A. Storms and F. C. R. Spieksma, "An LP-based algorithm for the data association problem in multitarget tracking," *Comput. Oper. Res.*, vol. 30, no. 7, pp. 1067–1085, 2003.
[17] E. B. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky, "Distributed occlusion reasoning for tracking with nonparametric belief propagation," *Neural Inform. Process. Syst.*, 2004.
[18] *Linear Programming*, V. Chvátal, Ed. New York: W. H. Freeman, 1983.
[19] P. Vazquez and M. Sbert, "Fast adaptive selection of best views," in *Int. Conf. Computational Science and its Applications, ICCSA'2003*, 2003, (LNCS 2669).
[20] P. M. Moreira, L. P. Reis, and A. A. de Sousa, "Best multiple-view selection for the visualization of urban rescue simulations," *Int J. Simul. Model*, vol. 5, no. 4, pp. 167–173, 2006.
[21] K. C. Ng, H. Ishiguro, M. Trivedi, and T. Sogo, "An integrated surveillance system—human tracking and view synthesis using multiple omni-directional vision sensors," *Image Vis. Comput.*, June 2002.
[22] K. Nummiaro, E. Koller-Meier, T. Svoboda, D. Roth, and L. V. Gool, "Color-based object tracking in multi-camera environments," *Lecture Notes in Computer Science, Pattern Recognition*, vol. 2781, 2003.



Hao Jiang received the Ph.D. degree from Simon Fraser University, Burnaby, BC, Canada, in 2006.

He was a Postdoctoral Researcher at the University of British Columbia, Vancouver, in 2006 and an Associate Researcher with Microsoft Research Asia in 1999. He is currently an Assistant Professor in the Computer Science Department, Boston College, Chestnut Hill, MA. His research interests include computer vision, image processing, multimedia, and graphics.

Sidney Fels received the B.A.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1988, and the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Toronto, Toronto, ON, in 1990 and 1994, respectively.

He has been with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada, since 1998. He is currently the Director of the Media and Graphics Interdisciplinary Centre (MAGIC) and heads the Human Communication Technologies (HCT) Laboratory. He was a Visiting Researcher at ATR Media Integrations and Communication Research Laboratories, Kyoto, Japan, from 1996 to 1997. His research interests are in human-computer interaction, biomechanical anatomical modeling, intelligent agents, neural networks, new interfaces for musical expression and interactive arts.

James J. Little received the A.B. degree from Harvard College, Cambridge, MA, in 1972 and the M.Sc. and Ph.D. degrees in computer science from the University of British Columbia (UBC), Vancouver, BC, Canada, in 1980 and 1985, respectively.

From 1985 to 1988, he was a Research Scientist at the MIT Artificial Intelligence Laboratory. Currently, he is a Professor of Computer Science at the UBC and Director of the Laboratory for Computational Intelligence. His research interests include computational vision, robotics, and spatio–temporal information systems with a particular interest in stereo, motion, tracking, mapping, and motion interpretation.