Multiple Object Tracking Using K-Shortest Paths Optimization

Jérôme Berclaz, François Fleuret, Engin Türetken, and Pascal Fua, *Senior Member*, *IEEE*

Abstract—Multi-object tracking can be achieved by detecting objects in individual frames and then linking detections across frames. Such an approach can be made very robust to the occasional detection failure: If an object is not detected in a frame but is in previous and following ones, a correct trajectory will nevertheless be produced. By contrast, a false-positive detection in a few frames will be ignored. However, when dealing with a multiple target problem, the linking step results in a difficult optimization problem in the space of all possible families of trajectories. This is usually dealt with by sampling or greedy search based on variants of Dynamic Programming which can easily miss the global optimum. In this paper, we show that reformulating that step as a constrained flow optimization results in a convex problem. We take advantage of its particular structure to solve it using the k-shortest paths algorithm, which is very fast. This new approach is far simpler formally and algorithmically than existing techniques and lets us demonstrate excellent performance in two very different contexts.

Index Terms—Data association, multiobject tracking, K-shortest paths, linear programming.

1 Introduction

Multi-object tracking can be decomposed into two separate steps that address independent issues. The first is time-independent detection, in which a prediction scheme infers the number and locations of targets from the available signal at every time step independently. It usually involves either a generative model of the signal given the target presence or a discriminative machine learning-based algorithm. The second step relies on modeling detection errors and target motions to link detections into the most likely trajectories.

In theory at least, such an approach is very robust to the occasional detection failure. For example, false positives are often isolated in time and can readily be discarded. Similarly, if an object fails to be detected in a frame but is detected in previous and following ones, a correct trajectory should nevertheless be produced.

However, while it is easy to design a statistical trajectory model with all of the necessary properties for good filtering, estimating the family of trajectories exhibiting maximum posterior probability is NP-complete. This has been dealt with in the literature either by sampling and particle filtering [1], linking short tracks generated using Kalman filtering [2], or by greedy Dynamic Programming in which trajectories are estimated one after another [3]. While effective, none of these approaches guarantees a global optimum. A notable exception is a recent approach [4] that relies on Linear Programming [5] to find a global optimum with high probability, but at the cost of a priori specifying the number of objects being tracked and restricting the potential set of locations where objects can be found to those where the detector has fired. The former is restrictive, while the latter is fine as long as the detector never produces false negatives, but may lead to erroneous trajectories in the more realistic case where it does.

By contrast, we show that reformulating the linking step as a constrained flow optimization results in a convex problem that fits into a standard Linear Programming framework. This formulation, however, yields a very large system that is hardly tractable using generic Linear Programming solvers. Therefore, we then demonstrate that, due to its particular structure, our problem can be solved very efficiently using the k-shortest paths algorithm [6], which yields real-time performance on realistically sized problems. Our method does not present any of the limitations mentioned above, nor does it require an appearance model. The latter does, of course, not mean that one should not be used if available, but making its use optional increases the range of applicability of our approach. Moreover, it is far simpler both formally and algorithmically than existing techniques and we will show that it performs well in two difficult real-world scenarios:

- tracking multiple balls of similar color, which is a case where an appearance model would not help;
- tracking multiple people with multiple cameras set at shoulder level so that there are significant occlusions.

In both cases, we use an object detector that produces a *probabilistic occupancy map* (POM), that is, a set of probabilities of presence of objects at a discrete set of locations at each time step independently. These probabilities may of course be noisy and inaccurate. Our only assumptions are that objects neither appear nor disappear anywhere but at specified entrances and exits, do not move too quickly, and cannot share a location with another object. These assumptions are

- *J. Berclaz, E. Türetken, and P. Fua are with the École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland. E-mail: {jerome.berclaz, engin.turetken, pascal.fua}@epfl.ch.*
- *F. Fleuret is with the Idiap Research Institute, CH-1920 Martigny, Switzerland. E-mail: francois.fleuret@idiap.ch.*

Manuscript received 27 Nov. 2009; revised 30 Aug. 2010; accepted 7 Jan. 2011; published online 28 Jan. 2011.
Recommended for acceptance by P. Felzenszwalb.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2009-11-0788.
Digital Object Identifier no. 10.1109/TPAMI.2011.21.

minimal and generally applicable. We formulate the search for a map that obeys them, while being as close as possible to the original one as a convex Linear Programming problem. Its solution is a set of flows that are both consistent and binary so that linking detections becomes trivial.

Our main contribution is two-fold: First, we introduce a generic and mathematically sound multiple object tracking framework, which only requires an occupancy map from a detector as input. Very few parameters need to be set and the algorithm handles unknown, and potentially changing, numbers of objects while naturally filtering out false positives and bridging gaps due to false negatives. Second, we demonstrate that this Linear Programming problem can be solved very effectively using the k-shortest paths algorithm [6].

2 Related Work

Multiple object tracking is an intensively studied area of research. A wide range of approaches relies on the recursive update of tracks with the most recent detections. For instance, Kalman filtering is an efficient way to address multitarget tracking [7], [8], [9], [10], [11] when the number of objects remains small. It is also well suited for real-time applications. However, when the number of objects increases, identity switches become more frequent and are difficult to correct due to the recursive nature of the method. The work of Wu and Nevatia [12], which tracks multiple humans using the mean-shift algorithm, also suffers from the same weakness.

Particle filtering can address some of the limitations of Kalman filtering by exploring multiple hypotheses [13], [1], [14], [15], [16], [17]. This technique has been used to great effect to follow multiple hockey players [18] or to track multiple people in the ground and image planes simultaneously [19]. In the same spirit, Yu et al. [20] rely on data-driven MCMC to recover trajectories of targets using a batch of observations. Maggio et al. [21] apply a Probability Hypothesis Density filter to tracking multiple objects from noisy observations which therefore falls into this family of algorithms. Despite their success, in our experience, those sampling-based methods typically require careful tuning of several meta-parameters, which reduces the generality of systems that rely on it. Besides, they usually look at small time windows because their state space grows exponentially with the number of frames.

In an attempt to increase tracking reliability, some methods rely on a hybrid approach. Detections are first connected into short tracks, which are then linked together using a higher level method. For example, Perera et al. [2] rely on Kalman filtering to obtain basic tracks and then try to merge and split the tracks using the Hungarian algorithm. Huang et al. [22] explore the hierarchical version of the same concept, while Li et al. [23] use a variant of AdaBoost to automatically learn the best criterion for linking low-level tracks together. Similarly, Beleznai et al. [24] turn observations into trajectory segments using local PCA and then link those segments based on their spatial proximity and smoothness constraints. Ge and Collins [25] rely on mean shift or particle filtering to generate tracklets from detection results. In a second stage, they use MCMC data association to combine the tracklets into full tracks and to automatically estimate the best parameters for the model. Eshel and Moses [26] use a motion model and nearest neighbor to build tracks out of heads detected from a top-mounted calibrated camera. The tracks thus generated are then merged and split into the final trajectories using heuristics based on overlap, directions, and speed. Brostow and Cipolla [27] propose another method to tracklet generation in a crowded environment, without, however, going all the way to combining them into complete tracks. It detects multiple people and creates tracklets by applying Bayesian clustering on simple tracked image features. By contrast, Nillius et al. [28] concentrate on the high-level task. The authors assume that a track graph has already been produced and focus on linking identities in the provided track graph. They formulate the multi-object tracking as a Bayesian network inference problem and apply this method to tracking multiple soccer players.

This class of methods is a good compromise: The two-stage architecture allows them to scale efficiently, while at the same time taking into account a wider observation window. However, while exhibiting good results in some situations, those methods rely on an ad hoc mathematical formulation, which does not guarantee convergence to a global optimum. They are therefore prone to mistakes such as identity switches. To improve robustness to wrong identity assignment, research has recently focused on linking detections over a larger time window using various optimization schemes. For example, Khan and Shah [29] apply graph cuts to extract trajectories from a batch of people detections obtained using homographic constraints on images from multiple cameras. Leibe et al. [30] simultaneously optimize detections and tracks, coupled into a Quadratic Boolean Problem and solved by an EM algorithm.

Dynamic Programming [31] can be used to link multiple detections over time and therefore solve the multitarget tracking problem. Moreover, it can be extended to enable the optimization of several trajectories simultaneously [32]. Unfortunately, the computational complexity of such an approach can be prohibitive. While efficient for very small state-space, it does not scale to the size of problems we generally deal with. To overcome this limitation, in earlier work [3] we sequentially applied Dynamic Programming over individual trajectories, which were assumed independent. While this approach greatly reduces the optimization cost, it tends to mix trajectories when the targets are densely located. It is also quite sensitive to false negatives and exhibits a tendency to ignore trajectories when the detection information is not good enough. A different formulation is chosen by Shafique and Shah [33], where a directed graph, with nodes standing for actual detections, represents the multiframe point correspondence problem. A greedy optimization algorithm is introduced to efficiently solve the problem, but without a guarantee to find a global optimum.

By contrast, Linear Programming is an optimization method that has been applied to find global optima and solve the data association problem for airplane radar tracking [34] or multiple people tracking [4]. Starting from the output of simple object detectors, this last approach builds a network graph in which every node is an observation fully connected to future and past observations, in much the same way as in Shafique and Shah [33]. Occlusions among objects are modeled by specifying spatial conflicts between nodes. Additional nodes are created to specifically handle occluded objects. Finally, arc costs are chosen according to object appearances and a motion model, and soft constraints are introduced to ensure spatial layout consistency. A relatively

TABLE 1
Notation

K	number of spatial locations;
T	number of time steps;
$\mathbf{I}$	$= (\mathbf{I}^1, \dots, \mathbf{I}^T)$ captured images;
$\mathcal{N}(k)$	$\subset \{1, \dots, K\}$ neighborhood of location k;
$e^t_{i,j}$	directed edge from location i at time t to location j at time $t+1$;
$f^t_{i,j}$	estimated number of objects moving from location i at time t to location j at time $t+1$;
m^t_i	estimated number of objects at location i at time t;
M^t_i	random variable standing for the true number of objects at location i at time t;
$\mathcal{F}$	set of occupancy maps physically possible;
$\mathcal{H}$	set of flows physically possible, i.e. satisfying the constraints of Eqs. 1, 2, 3, and 4.

similar graphical model, with nodes representing detections, is built by Zhang et al. [35] and Ma et al. [47] for multi-people tracking. The global optimum is searched using a min-cost flow algorithm in the former article and a variant of the k-shortest path algorithm [6] in the latter. Both optimization methods exploit the specific structure of the graph to reach the optimum faster than Linear Programming.

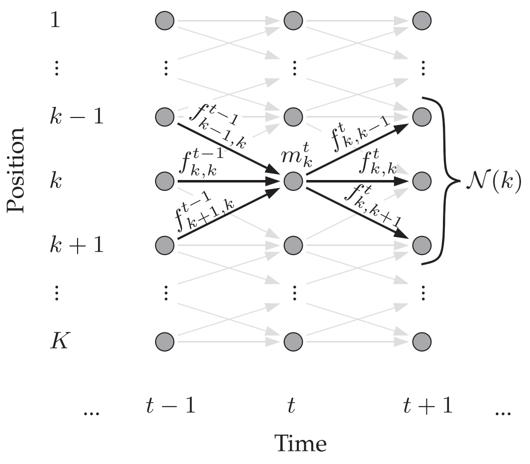
Due to their reduced state-space, these methods are computationally efficient. However, Jiang et al. [4] require a priori knowledge of the number of objects to be tracked, which seriously limits its applicability in real life situations. Also, with a state-space consisting of only observations, as opposed to all possible locations as in our approach, they cannot smoothly interpolate trajectories when there are false negatives. Finally, the choice of arc costs is ad hoc and involves many parameters which have to be tuned for each possible application, reducing the generality of the methods. By contrast, our model is far simpler, with the neighborhood size being the only value that needs to be adapted. We optimize over the entire space of locations for fine trajectory interpolation, and deal with the large resulting size of our problem by trading standard Linear Programming optimization for a very efficient formulation based on the k-shortest paths algorithm.

3 Algorithm

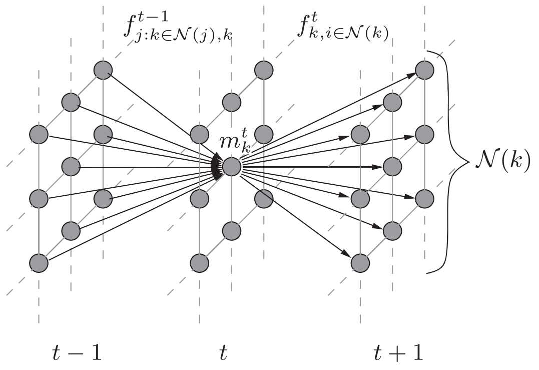
In this section, we first formulate multitarget tracking as an Integer Programming (IP) problem, using the notation summarized in Table 1. Although such a problem is NP-hard in many cases, we show that a relaxation of it as a Linear Program yields the optimal solution, and hence the problem is solvable in polynomial time. Despite our simple and clean formulation, the large number of variables and constraints makes it tractable only for small areas and short sequences. Thus, in a second step, we demonstrate how the k-shortest paths algorithm can be used to solve this problem much more efficiently than generic Linear Programming solvers can.

3.1 Formalization

We discretize the physical area of interest into K locations, and the time interval into T instants. For any location k, let $\mathcal{N}(k) \subset \{1, \dots, K\}$ denote the neighborhood of k, that is,



(a) Simplified flow model



$f^{t-1}_{j:k\in\mathcal{N}(j),k}$
$f^t_{k,i\in\mathcal{N}(k)}$
m^t_k
$\mathcal{N}(k)$
t − 1
t
t + 1

(b) Grid flow model

Fig. 1. (a) A simplified flow model which does not use virtual positions. Positions are arranged on one dimension and neighborhood is reduced to three positions. (b) A flow model used for tracking objects moving on a 2D grid, such as in pedestrian tracking. For the sake of readability, only the flows to and from location k at time t are printed.

the locations an object located at k at time t can reach at time $t+1$.

To model occupancy over time, let us consider a labeled directed graph with $K\,T$ vertices, which represents every location at every instant. Its edges correspond to admissible object motions, which means that there is one edge $e^t_{i,j}$ from (t,i) to $(t+1,j)$ if, and only if, $j \in \mathcal{N}(i)$. To allow objects to remain static, there is always an edge from a location at time t to itself at time $t+1$.

Each vertex is labeled with a discrete variable m^t_i standing for the number of objects located at i at time t. Each edge is labeled with a discrete variable $f^t_{i,j}$ standing for the number of objects moving from location i at time t to location j at time $t+1$, as shown in Fig. 1a. For instance, the fact that an object remains at location i between times t and $t+1$ is represented by $f^t_{i,i} = 1$.

Given these definitions, for all t, the sum of flows arriving at any location j is equal to m^t_j, which also is the sum of outgoing flows from location j at time t. We must therefore have

$$\forall t, j, \quad \underbrace{\sum_{i:j\in\mathcal{N}(i)} f^{t-1}_{i,j}}_{\text{Arriving at } j \text{ at } t} = m^t_j = \underbrace{\sum_{k\in\mathcal{N}(j)} f^t_{j,k}}_{\text{Leaving from } j \text{ at } t} . \tag{1}$$

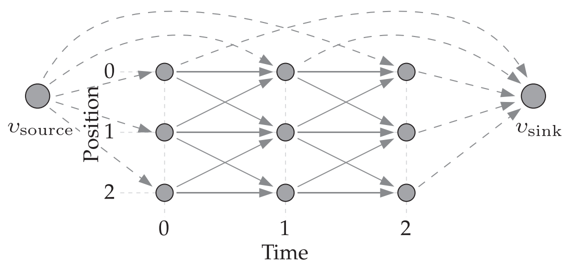


Fig. 2. A complete flow system for a simple graph consisting only of three positions and three time frames. Here, we assume that position 0 is connected to the virtual positions, and therefore is a possible entrance and exit point. Flows to and from the virtual positions are shown as dashed lines.

Furthermore, since a location cannot be occupied by more than one object at a time, we can set an upper bound of 1 to the sum of all outgoing flows from a given location and impose

$$\forall k, t, \sum_{j \in \mathcal{N}(k)} f_{k,j}^t \leq 1. \tag{2}$$

A similar constraint applies to the incoming flows, but we do not need to explicitly state it since it is implicitly enforced by (1). Finally, the flows have to be nonnegative and we have

$$\forall k, j, t,\ f_{k,j}^t \geq 0. \tag{3}$$

In general, the number of tracked objects may vary over time, meaning that objects may appear inside the tracking area and others may leave. Thus, the total mass of the system changes and we must allow flows to enter and exit the area.

We do this by introducing two additional nodes, v_{source} and v_{sink}, into our graph, which are linked to all of the nodes representing positions through which objects can respectively enter or exit the area, such as doors or borders of the camera field of view. In addition, a flow goes from v_{source} to all the nodes of the first frame, and reciprocally, a flow goes from all of the nodes of the last frame to v_{sink}. We call v_{source} and v_{sink} *virtual locations* because, as opposed to the other nodes of the graph, they do not represent any physical location. The resulting complete graph is shown in Fig. 2.

Finally, we introduce an additional constraint that ensures that all flows departing from v_{source} eventually end up in v_{sink}:

$$\underbrace{\sum_{j \in \mathcal{N}(v_{\text{source}})} f_{v_{\text{source}},j}^t}_{\text{Leaving } v_{\text{source}}} = \underbrace{\sum_{k: v_{\text{sink}} \in \mathcal{N}(k)} f_{k,v_{\text{sink}}}^t}_{\text{Arriving at } v_{\text{sink}}}. \tag{4}$$

Let M_i^t denote a random variable standing for the true presence of an object at location i at time t. The object detector used to process the sequence provides, for every location i and every instant t, an estimate of the marginal posterior probability of the presence of an object

$$\rho_i^t = \hat{P}(M_i^t = 1 \mid \mathbf{I}^t), \tag{5}$$

where $\mathbf{I}^t$ is the signal available at time t. For the multicamera pedestrian-tracking application described in Section 4, $\mathbf{I}^t$ denotes the series of pictures taken by all the cameras at time t.

Let $\mathbf{m}$ be an occupancy map that is a set of occupancy variables m_i^t, one for each location and for each instant. We say that $\mathbf{m}$ is *feasible* if there exists a set of flows $f_{k,j}^t$ that satisfies (1), (2), (3), and (4), and we define $\mathcal{F}$ as the set of feasible maps. Our goal then becomes solving

$$\mathbf{m}^* = \arg\max_{\mathbf{m} \in \mathcal{F}} \hat{P}(\mathbf{M} = \mathbf{m} \mid \mathbf{I}). \tag{6}$$

Assuming conditional independence of the M_i^t, given the $\mathbf{I}^t$, the optimization problem of (6) can be rewritten as

$$\begin{aligned}\mathbf{m}^* &= \arg\max_{\mathbf{m} \in \mathcal{F}} \log \prod_{t,i} \hat{P}\big(M_i^t = m_i^t \mid \mathbf{I}^t\big) \\ &= \arg\max_{\mathbf{m} \in \mathcal{F}} \sum_{t,i} \log \hat{P}\big(M_i^t = m_i^t \mid \mathbf{I}^t\big),\end{aligned} \tag{7}$$

$$\begin{aligned}&= \arg\max_{\mathbf{m} \in \mathcal{F}} \sum_{t,i} \big(1 - m_i^t\big) \log \hat{P}\big(M_i^t = 0 \mid \mathbf{I}^t\big) \\ &\quad +\ m_i^t \ \log \hat{P}\big(M_i^t = 1 \mid \mathbf{I}^t\big)\end{aligned} \tag{8}$$

$$= \arg\max_{\mathbf{m} \in \mathcal{F}} \sum_{t,i} m_i^t \ \log \frac{\hat{P}(M_i^t = 1 \mid \mathbf{I}^t)}{\hat{P}(M_i^t = 0 \mid \mathbf{I}^t)} \tag{9}$$

$$= \arg\max_{\mathbf{m} \in \mathcal{F}} \sum_{t,i} \left(\log \frac{\rho_i^t}{1 - \rho_i^t}\right) m_i^t, \tag{10}$$

where (7) is true under the assumption of conditional independence of the M_i^t given $\mathbf{I}^t$, (8) is true because m_i^t is 0 or 1 according to (2), and (9) is obtained by ignoring a term which does not depend on $\mathbf{m}$. Hence, the objective function of (10) is a linear expression of the m_i^t.

3.2 Linear Programming Formulation

The formulation defined above translates naturally into the Integer Program

$$\begin{aligned}\text{Maximize} \quad & \sum_{t,i} \log\left(\frac{\rho_i^t}{1 - \rho_i^t}\right) \sum_{j \in \mathcal{N}(i)} f_{i,j}^t \\ \text{subject to} \quad & \forall t, i, j, \quad f_{i,j}^t \geq 0 \\ & \forall t, i, \quad \sum_{j \in \mathcal{N}(i)} f_{i,j}^t \leq 1 \\ & \forall t, i, \quad \sum_{j \in \mathcal{N}(i)} f_{i,j}^t - \sum_{k: i \in \mathcal{N}(k)} f_{k,i}^{t-1} \leq 0 \\ & \sum_{j \in \mathcal{N}(v_{\text{source}})} f_{v_{\text{source}},j} - \sum_{k: v_{\text{sink}} \in \mathcal{N}(k)} f_{k,v_{\text{sink}}} \leq 0.\end{aligned} \tag{11}$$

In this system, the optimization is carried out with respect to the flows $f_{i,j}^t$ rather than the occupancies m_i^t because there is no natural way to express the flow continuity constraints in terms of the latter. This is equivalent to maximizing the objective function of (10) because $\forall t, j, m_j^t = \sum_{k \in \mathcal{N}(j)} f_{j,k}^t$.

Note that the constraints of (1), (2), (3), and (4) are expressed as inequalities, to have the linear program in *canonical form*. This new formulation is strictly equivalent to the original one and no additional constraint is needed. The inequalities are indeed sufficient to ensure that no flow can ever appear or disappear within the graph.

Under this formulation, our Integer Program can be solved by any generic LP solver. However, due to the very large size of our problem, this solution would hardly be practical as IP solving is NP-complete. The usual workaround is to relax the integer assumption and solve a continuous Linear Program instead, which has polynomial-time average-case complexity. The drawback of this method is that the Linear Program is unlikely to converge to the optimal solution of the original IP.

In our case, however, the relaxed Linear Program always converges toward an integer solution because its constraint matrix exhibits a property known as *total unimodularity*, as will be shown in Appendix A, which can be found in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2011.38 . As a consequence, we could use a generic LP solver to optimize our multitarget tracking framework. However, this approach would only be tractable for moderately sized problems and does not scale to most practical applications. Therefore, in the next section, we introduce a more efficient optimization scheme, which takes into account the specificity of our problem to tremendously reduce the complexity.

3.3 K-Shortest Paths Formulation

The relaxation of the original integer problem yields a large scale LP problem, which can be solved by generic LP solvers that, in general, rely on variants of the Simplex algorithm [5] or interior-point-based methods [36]. However, these algorithms do not make use of the specific structure of our problem and have very high worst-case time complexities. In the following, we show that this complexity can be drastically reduced by reformulating the problem as a *k-shortest node-disjoint paths* problem on a directed acyclic graph (DAG).

Given a pair of nodes, namely, the source υ_{source} and the sink υ_{sink}, in a graph G, the k-shortest paths problem is to find the k paths $\{p_1, \dots, p_k\}$ between these nodes such that the total cost of the paths is minimum. The problem is well studied in the network optimization literature and the results have been widely applied in the field of network connection routing and restoration. There exist many variants of the algorithm, each targeted at a specific problem instance.[1]

In our specific case, we are interested in the particular instance where the graph is directed and paths are both node-disjoint—i.e., two separate paths cannot share the same node—and node-simple—i.e., a path visits every node in the graph at most once. We use the graph structure with a single source and a single sink illustrated in Fig. 2. Any path between υ_{source} and υ_{sink} in this graph represents the flow of a single object in the original problem along the edges of the path. The node-disjointness constraint means that no location can be shared between two paths, hence two objects. This is thus equivalent to the constraint of (2). Moreover, by only looking for paths between the source and sink nodes, we ensure that no flow can ever be created or suppressed anywhere in the graph but at the virtual locations. This enforces the constraints of (1) and (4). Finally, the node-simple characteristic of the paths simply stems from the fact that our graph is a DAG, hence acyclic.

1. For a complete list of references, see the online bibliography at http://liinwww.ira.uka.de/bibliography/Theory/k-path.html.

A directed edge $e^t_{i,j}$ from location i at time t to location j at time $t+1$ is assigned the cost value

$$c(e^t_{i,j}) = -\log\left(\frac{\rho^t_i}{1-\rho^t_i}\right). \tag{12}$$

The cost value of the edges emanating from the source node is set to zero to allow objects to appear at any entrance position and at any time instant at no cost. We formulate our problem as a minimization problem by negating the objective function of (11).

Let $\mathcal{H}$ denote the set of feasible solutions of the original LP formulation of (11), satisfying the constraints given in (1), (2), (3), and (4). Then, the optimal solution $\mathbf{f}^*$ of the k-shortest paths problem can be written as

$$\mathbf{f}^* = \arg\min_{\mathbf{f}\in\mathcal{H}} \sum_{t,i} c\big(e^t_{i,j}\big) \sum_{j\in\mathcal{N}(i)} f^t_{i,j}, \tag{13}$$

where $c(e^t_{i,j})$ represents the cost of the edge $e^t_{i,j}$ defined in (12). Note that any node-disjoint k paths between υ_{source} and υ_{sink} with arbitrary k is in the feasible set of solutions $\mathcal{H}$. In addition, any solution in $\mathcal{H}$ can be expressed as a set of k node-disjoint paths.

Let p^*_i be the shortest path computed at the ith iteration of the algorithm and $P_l = \{p^*_1, \dots, p^*_l\}$ be the set of all l shortest paths computed up to iteration l. We start by finding the single shortest path in the graph p^*_1 and compute its total cost

$$\text{cost}(p^*_l) = \sum_{e^t_{i,j}\in p^*_l} c\big(e^t_{i,j}\big). \tag{14}$$

We then compute iteratively the l-shortest paths for $l = 2, 3, 4, \dots$, and for each l, we calculate the total cost of the shortest paths

$$\text{cost}(P_l) = \sum_{i=1}^{l} \text{cost}(p^*_i). \tag{15}$$

At each new iteration $l+1$, the total cost $\text{cost}(P_{l+1})$ is compared to the cost at the previous iteration $\text{cost}(P_l)$. The optimal number of paths k^* is obtained when the cost of iteration k^*+1 is higher than the one of iteration k^*. The procedure is summarized by the pseudocode of Algorithm 1, in Appendix B, which can be found in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2011.38.

To compute such k-shortest paths, we use the *disjoint paths* algorithm [6], which is an efficient iterative method based on signed paths. For the sake of completeness, we give a brief description of this algorithm in Appendix B, which can be found in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2011.38.

The equivalence of the LP and the k-shortest paths formulations follows from the exact procedure we use to select an optimal k such that the objective function is minimized. Since path costs are monotonically increasing

$$\text{cost}(p^*_{i+1}) \geq \text{cost}(p^*_i) \qquad \forall i, \tag{16}$$

the total cost function $\mathrm{cost}(P_l)$ is convex with respect to l. Therefore, the global minimum is reached when $\mathrm{cost}(p_i^*)$ changes sign and becomes nonnegative:

$$\mathrm{cost}(P_{k^*-1}) \geq \mathrm{cost}(P_{k^*}) \leq \mathrm{cost}(P_{k^*+1}). \tag{17}$$

This is set as the stopping criterion of the algorithm, as presented in Algorithm 1. Finally, among the set of all consecutive values that may satisfy the above condition, we select the smallest one to avoid erroneous splitting of paths.

As discussed in Appendix B, which can be found in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2011.38, the worst-case complexity of the algorithm is $O(k(m + n \cdot \log n))$, where k is the number of objects appearing in a given time interval, m is the number of edges and n is the number of nodes in the final transformed graph. This is more efficient than the min-cost flow method of Zhang et al. [35], which exhibits a worst-case complexity of $O(kn^2 m \log n)$. Furthermore, due to the mostly acyclic nature of our graph, the average complexity is almost linear with the number of nodes, which is reflected by our experimental results in Fig. 7b. This is much faster than general LP solvers, and a speed gain of up to a factor 1,000 can be expected, as illustrated by the runtime comparison in Section 4.10.

3.4 Batch Processing and Complexity Reduction

Processing a whole video sequence is possible but impractical for applications such as broadcasting, in which the result must be supplied quickly. When dealing with such cases, we split the sequence into batches of 100 frames, which yields good results and can be done in real time. This results in a constant 4 second delay between input and output, which is nevertheless compatible with many applications.

To enforce temporal consistency across batches, we add the last frame of the previously optimized batch to the current one. We then force the flows out of every location of this frame to sum up to the location's value in the previous batch:

$$\forall k \in \{1, \dots, K\}, \sum_{j \in \mathcal{N}(k)} f_{k,j}^{-1} = \mu_k, \tag{18}$$

where μ_k is the score at location k of the last frame of the previous batch and $f_{k,j}^{-1}$ is a flow from location k of the last frame of the previous batch to location j in the first frame of the current batch. This is implemented as an additional constraint in our framework.

Further reducing the system's size might be needed for extremely large problems, to limit their time and memory consumption. It could be achieved by pruning the detection graph. Since most of the probabilities of presence estimated by the detector are virtually equal to zero, we can use this sparsity to reduce the number of nodes to consider in the optimization, thus reducing the computational cost.

Formally, for every position k and every frame t, we check the maximum detection probability within a given spatio-temporal neighborhood:

$$\max_{\substack{\|j-k\|<\tau_1 \\ t-\tau_2<u<t+\tau_2}} \rho_j^u. \tag{19}$$

If this value is below a threshold, the location is considered not reachable by an object with any reasonable level of probability. All flows to and from it are then removed from the model. Applying this method would reduce the number of variables and constraints up to an order of magnitude. In the examples presented in this paper, we have not found it necessary to do so.

3.5 Algorithm Output

Estimating the $f_{i,j}^t$ indirectly provides the m_i^t values and the feasible occupancy map $\mathbf{m}^*$ of maximum posterior probability. This data can be used as a cleaned up version of the original occupancy map in which most false positives and negatives have been filtered out. However, the $f_{i,j}^t$ themselves provide, in addition to the instantaneous occupancy, estimates of the actual motions of objects. From these estimated flows of objects, one can follow the motion back in time by moving along the edges whose $f_{i,j}^t$ are not 0, and build the corresponding long trajectories.

4 Results

In this section, we present results in two very different contexts. First, we use a multicamera setup in which the cameras are located at shoulder level to track pedestrians who may walk in front of each other. The frequent occlusions between people produce noisy detections, which our algorithm nevertheless links very reliably. As a result, our approach was shown to compare favorably against other state-of-the-art algorithms in the PETS 2009 evaluation [37]. Second, to highlight the fact that we do not depend on an appearance model, we track sets of similar-looking bouncing balls seen from above. In both cases, we compare our results to those of our earlier tracking method based on sequential Dynamic Programming [3] and show that we can obtain good results even when using a single camera.

4.1 PETS 2009 Evaluation

The results of our approach on the PETS 2009 S2/L1 multicamera sequence have been submitted to the Winter-PETS 2009 workshop. The results of this comparative evaluation are presented in Ellis et al. [37] and illustrated in Fig. 3. They show that, for the tracking task, our current approach outperforms the other submitted methods. Furthermore, our earlier Dynamic Programming-based approach [3] is also shown to perform well. In the remainder of this section, we thus use this previous approach as the baseline, and extensively compare our new algorithm against it.

4.2 Test Data

Our main data set consists of a series of multicamera video sequences of pedestrians. The chosen locations for data acquisition include various environments: a crowded outdoor terrace, an indoor basketball court, as well as a very difficult dark underground passageway. Additionally, we also perform our own detailed evaluation on a sequence from the PETS 2009 [39] data set.

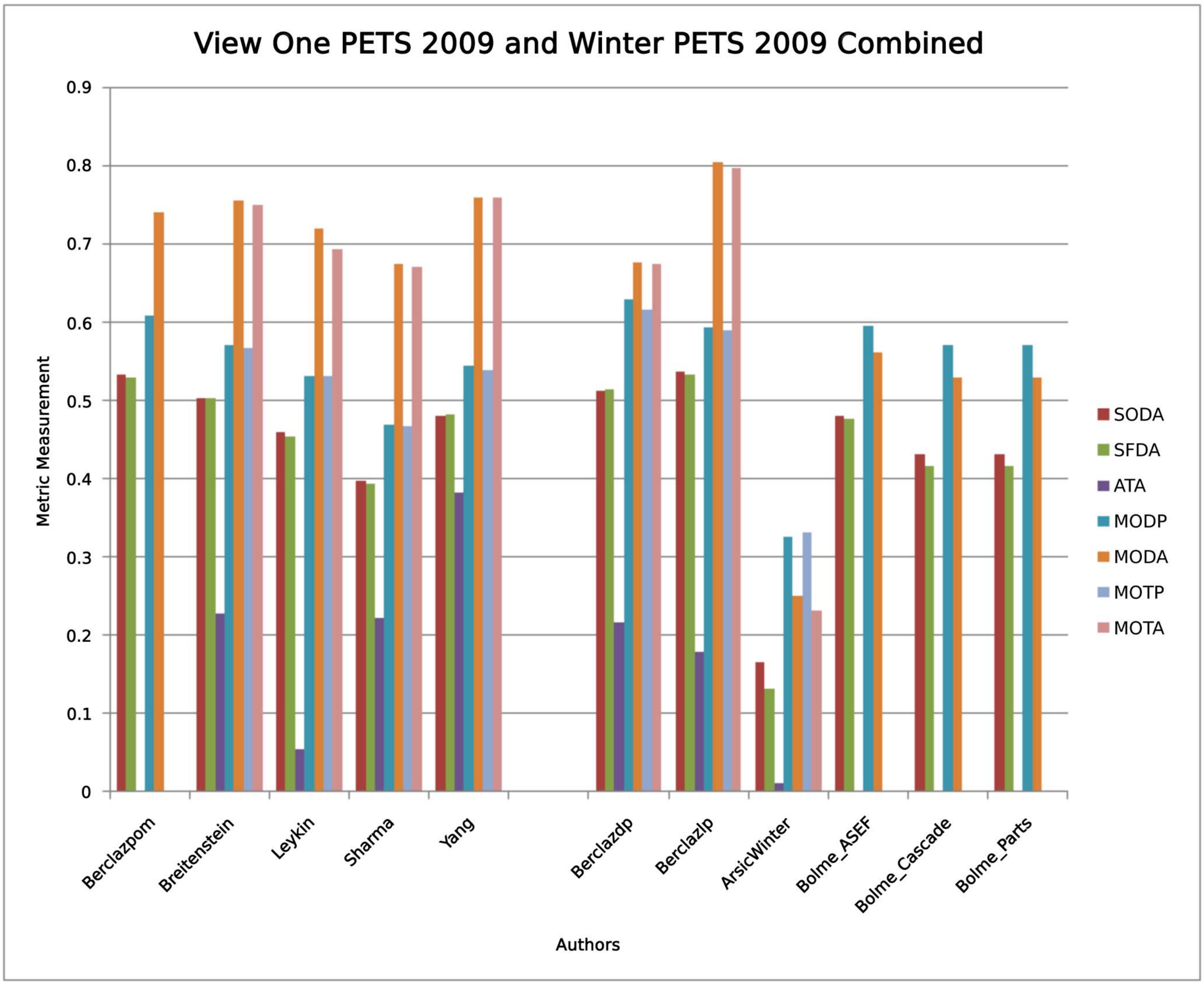


Fig. 3. Winter-PETS 2009 [37] results comparison chart, showing the performance of various tracking methods on the S2/L1 sequence. The results are evaluated with the CLEAR (MODA, MODP, MOTA, and MOTP, described below) and VACE metrics [38] (SODA, SFDA, and ATA). In the chart, our algorithm is referred to as "Berclazlp" and our earlier approach based on dynamic programming as "Berclazdp." Figure courtesy of James Ferryman and Ali Shahrokni from the University of Reading.

4.2.1 *Laboratory Sequence*

This 3 1/2 minute outdoor sequence consists of up to nine people appearing, one after the other, and walking in front of the cameras. It tests the ability of our algorithm to cope with a moderately crowded environment.

4.2.2 *Basketball Sequence*

This sequence involves 10 basketball players in a game on half a basketball court. It is a difficult sequence with fast moving people and many occlusions.

4.2.3 *Passageway Sequences*

These sequences involve several people passing through a public underground passageway. They are very challenging for several reasons. First, lighting conditions are very poor, which is typical in real-world surveillance situations. A large portion of the images is either underexposed or saturated. Second, the area covered by the system is large, and people get very small on the far end of the scene, making their precise localization challenging. Finally, large parts of the scene are filmed by only two or even a single camera.

All of these difficulties greatly affect the quality of the probabilistic occupancy maps we use as input and the detection maps can get very noisy, with some people poorly localized or ignored over significant numbers of frames. On these noisy sequences, if we were to detect people by simply thresholding the maps in individual frames, the true positive rate would drop to 70-80 percent, thus making the linking task challenging.

4.2.4 *PETS 2009 Sequence*

This sequence was filmed at a road corner of a university campus and involves about 10 people. Important light changes between the background model and the sequence, as well as precision issues in camera calibration make the people detection results noisy. Moreover, the sequence was acquired at a low frame rate of 7 fps, which creates an additional difficulty since people can move significantly from one frame to the next.

TABLE 2
Dimensions of the Areas Used for Pedestrian Tracking

Scene	dimensions	grid size	locations
Laboratory	7m×10m	30×45	1,350
Basketball	15m×14m	47×50	2,350
Passageway	12m×30m	40×100	4,000
PETS 09	18.5m×20m	56×61	3,416

4.2.5 Camera Setup

In the first three pedestrian environments, the scene was filmed by four DV cameras with overlapping fields of view, each of which was placed in a corner of the monitored area. The video format is DV PAL, downsampled to 360×288 pixels at 25 fps, and the four video streams were synchronized manually after data acquisition. The dimensions of the four areas are summarized by Table 2.

The PETS 2009 sequence was filmed by seven cameras: three dedicated video surveillance cameras and four DV cameras. The DV cameras were placed at about 2 m above the ground, whereas the video surveillance cameras were located between 3-5 m above it, and significantly farther from the scene. The frame rate for all cameras was set to 7 fps. Due to calibration imprecision, only five out of the seven available camera views were used for people detection.

Our second data set consists of two video sequences in which 24 table tennis balls were thrown on the ground. Those were filmed by a single DV camera, placed about 1.5 m above the ground. The videos were cropped to a resolution of 600×400 pixels, and the corresponding area was discretized into a grid of 60×40 locations.

4.3 Probabilistic Occupancy Map

We used the publicly available implementation [40] of our earlier POM algorithm [3] to create the detection data needed as input by our tracker.

This method first performs binary background/foreground segmentation in all images taken at the same time and then uses a generative model to estimate the most likely locations of targets given the observed foreground regions. More precisely, it relies on a decomposition of the space of possible object positions into a discrete grid. Then, at every time frame t and for every location i of the grid, it produces an estimate ρ_i^t of the marginal posterior probability of presence of a target at that location, given all input images captured at that instant. POM specifically estimates the ρ_i^t such that the resulting product law closely approximates the joint posterior distribution, which justifies the assumption of conditional independence in (7).

In the multicamera setup for which POM was designed, the grid of positions models the ground plane on which people walk and is made of square elements of typically $30 \text{ cm} \times 30 \text{ cm}$. Correspondence between camera and top views is ensured through camera calibration. The generative model at the heart of POM represents people as cylinders that project to rectangles in the images.

Note that, in our model, the resolution of the ground grid is independent of the target's size. If grid cells are smaller than a target, the spatial precision of detections is improved at the cost of increased computation time, but the detections do not spread over several cells. In effect, POM implicitly performs a nonmaximum suppression: The best fitted position receives a high probability, while surrounding locations are considered as empty. As a result, POM occupancy maps are normally peaky, which is what our linking algorithm expects.

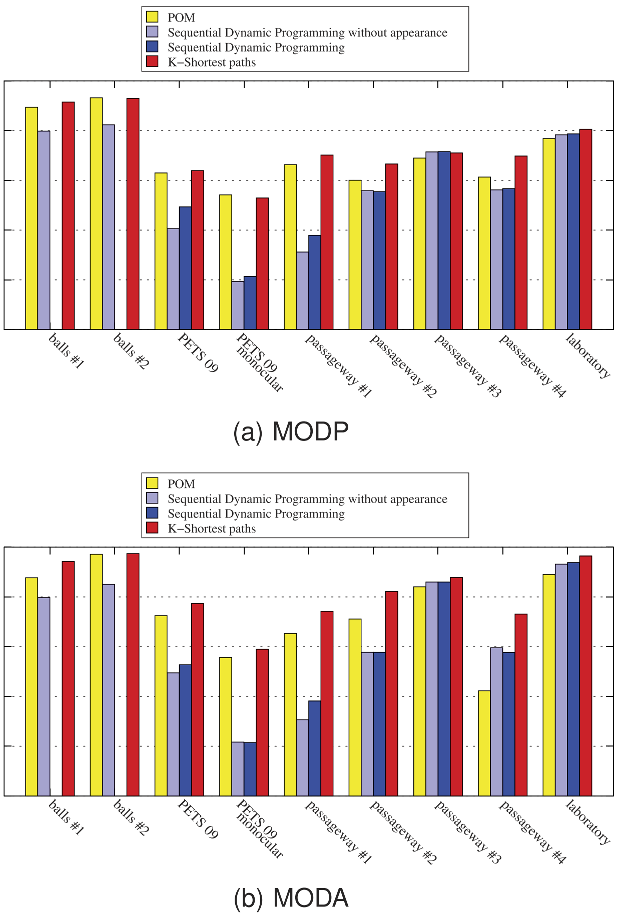
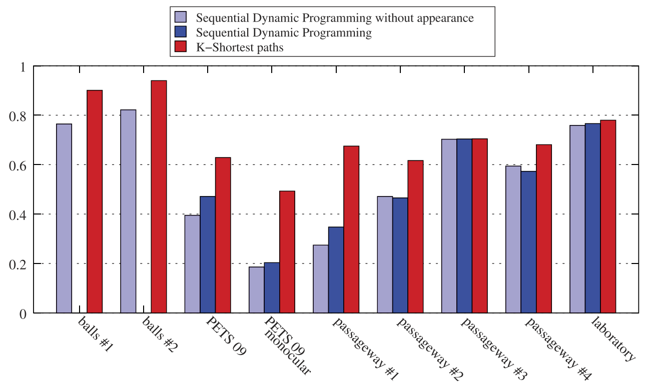


Fig. 4. Detection precision (MODP) and accuracy (MODA) measures applied to the results of the original detections (POM), the sequential Dynamic Programming (DP)—with and without appearance model—and the proposed (KSP) trackers on various sequences.

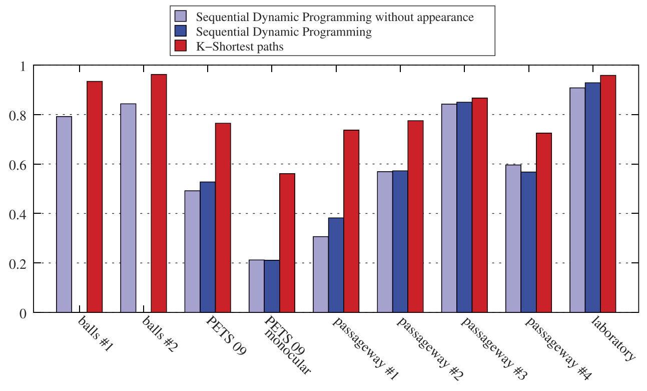
To process the monocular sequence of bouncing balls, we slightly modified the original POM code to represent the balls as squares and work directly in the top view, without having to project from oblique images into it.

4.4 Baseline

We compare our new algorithm to our previous sequential Dynamic Programming approach [3]. It involves estimating likely trajectories one after another in a greedy way using a standard Dynamic Programming procedure. The most likely trajectories are selected first and, once a trajectory has been found, the corresponding locations are removed from consideration. Note that the results reported in Fleuret et al. [3] were obtained with both a motion and an appearance model, while our results rely only on the very weak motion model implied by the graph's connectivity. For a fair comparison against our new approach, the method was applied both with and without its appearance model in the evaluation of Figs. 4 and 5. As expected, the appearance model slightly improves Dynamic Programming's results.



(a) MOTP



(b) MOTA

Fig. 5. Tracking precision (MOTP) and accuracy (MOTA) measures applied to the results of the sequential Dynamic Programming (DP)—with and without appearance model—and the proposed (KSP) trackers on various sequences.

In the rest of this section, we refer to our Linear Programming framework solved using the k-shortest paths algorithm as "KSP" and to the sequential Dynamic Programming as "DP."

4.5 Evaluation Metrics

To quantify our results, we manually labeled some of the test sequences. We marked both the position of the people and their identity to be able to detect identity switches. The same process was used to label the ball sequences. Our ground truth data therefore consists of

- two ball sequences of approximately 1,000 frames each, labeled once every three frames;
- the 800 frame long PETS 2009 sequence S2/L1, labeled once every five frames;
- four video sequences from the passageway data set, measuring, respectively, 2,500, 800, 900, and 800 frames, and labeled once every 25 frames;
- the laboratory sequence of 5,000 frames, labeled once every 25 frames.

Our results are evaluated using the standard CLEAR metrics: *Multiple Object Detection Accuracy* and *Precision* (MODA and MODP), as well as *Multiple Object Tracking Accuracy* and *Precision* (MOTA and MOTP). The detection precision metric (MODP) roughly gauges the quality of the bounding box alignment in case of correct detection, while its accuracy counterpart (MODA) evaluates the relative number of false positives and missed detections. The tracking metrics also take the identity of detections into account: MOTP evaluates the alignment of tracks with the ground truth, and MOTA produces a score based on the amount of false positives, missed detections, and identity switches. These metrics have become standard for evaluation of detection and tracking algorithms, and we refer the interested reader to [41], [38] for more detailed descriptions and motivations.

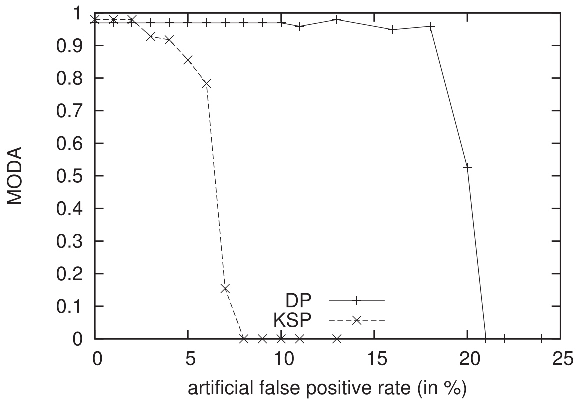
4.6 Pedestrian Tracking Results

For pedestrians tracking, we define the graph of Fig. 1a as follows: Every interior location of the ground plane at time t is linked to its nine direct neighbors at time $t+1$, as illustrated in Fig. 1b, which means that a pedestrian can only move from one location to its immediate neighbors in a consecutive frame. Border locations through which access to the area is possible are connected to the virtual locations υ_{source} and υ_{sink}. This arrangement is consistent with our chosen grid quantization at 25 fps. It even suits the 7 fps PETS 2009 sequence since the pedestrians do not move fast. To deal with an even lower frame rate, or faster moving objects, we could extend the neighborhood size, as we do in Section 4.8 to track table tennis balls. Detection results for all evaluated sequences are shown in Fig. 4 and tracking results shown in Fig. 5.

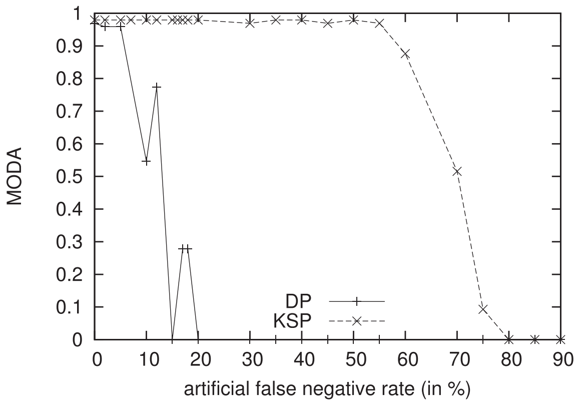
Since both DP and KSP link POM detections together, their precision score rarely exceeds the one of POM itself although it may happen that the interpolation performed by the trackers corrects some misalignment of POM, such as in the *laboratory* sequence of Fig. 4a. However, in both detection and tracking *precision*, KSP almost always scores significantly higher than DP.

Detection accuracy varies across sequences, depending on their difficulty. The *passageway* sequences, for instance, have lower accuracy than other sequences due to their poor image quality. So does the monocular PETS sequence because of the lack of precision resulting from the use of a single camera. Despite these variations, KSP's detection accuracy is always higher than POM. By accurately linking detections together while discarding isolated alarms, KSP efficiently filters the detections results, thus decreasing both false positive and missed detection counts. On the other hand, DP's accuracy is often lower than POM because of its tendency to ignore trajectories with missing detections. Note how the gap between KSP and DP generally widens as POM score gets lower, shown in Figs. 4 and 5. Both tracking methods deal efficiently with good detection results, but KSP proves much more robust than DP when the detection quality gets worse.

Recall that the occupancy maps are KSP's only input. The algorithm does not use any other source of information, such as the original images. Conversely, DP maintains a color model per tracked individual learned from images, in addition to the occupancy maps. In other words, KSP produces better results, even though it requires less information. This is valuable because, in some situations such as the ball tracking presented below, appearance models cannot be depended upon. Typical tracking results are illustrated in Fig. 8.

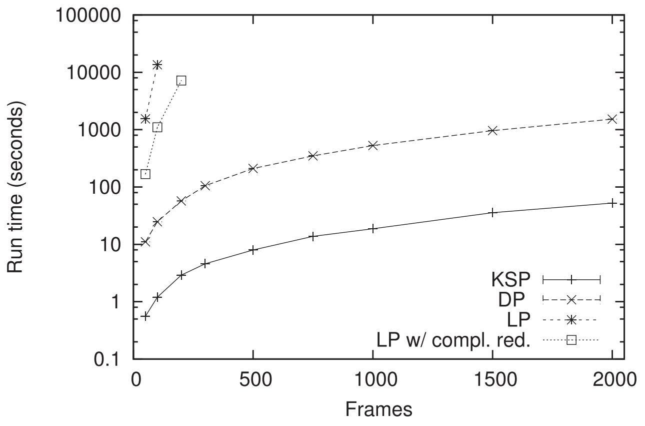


(a)

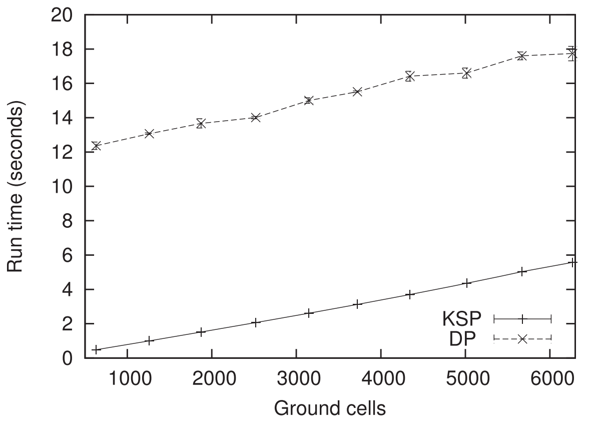


(b)

Fig. 6. Artificially increasing the number of detection (a) false positives and (b) false negatives, expressed as a percentage on the x-axis of the graphs.



(a)



(b)

Fig. 7. (a) Runtime comparison between our framework solved with a generic LP package (LP), our framework with a pruned graph solved with a generic LP package (LP w/comp. red.), our framework solved with the k-shortest paths algorithm (KSP), and our earlier Dynamic Programming method (DP). Note that the y-axis represents runtime and is plotted in log scale. (b) Both the DP and KSP algorithms scale almost linearly with the grid size.

4.7 Monocular Pedestrian Results

To further emphasize the strength of our approach, we generated the detection maps using only one of the seven available views of the PETS data set. Although POM still works on monocular sequences, ground plane localization is less precise: Without views from different angles, there is a depth ambiguity when estimating a pedestrian's position. Also, in the monocular case, occlusions often result in missed detection.

Under these challenging conditions, our algorithm shows its superiority over the sequential Dynamic Programming, even more clearly than in the multicamera case. This is illustrated in Figs. 4 and 5. In this context, DP's greedy strategy often leaves people outside the grid instead of trying to explain their very noisy detections. By contrast, KSP's joint optimization pays off and interpolates trajectories nicely. Monocular tracking results are depicted in Fig. 9.

4.8 Ball Tracking Results

Given the difference in grid scale, the balls move much faster than pedestrians and can cross more than one grid location between consecutive frames. To deal with this environment, we thus had to extend the location neighborhood to include the next closest 49 locations, which increases the maximum distance traveled between consecutive frames to three grid locations.

Detection and tracking results for the two ball sequences are also illustrated in Figs. 4 and 5, while screen shots are shown in Fig. 10. Detecting ping-pong balls is not a particularly difficult task, and thus POM's results are generally excellent, with very few false positives and false negatives. Because all balls have exactly the same appearance, DP's color model is useless and the comparison between the two algorithms is fair. As for the pedestrian environment, KSP outperforms DP on all four metrics. Here again, DP's greedy strategy is a disadvantage. Because it might be less costly to ignore some isolated detections, DP tends to leave out too many of them.

4.9 Failure Modes

Our tracking algorithm can be mainly affected by two elements: false detections and missing ones. To quantify the effect of both types of detection error, we carried out the following experiment. We selected a 1,000 frame excerpt of the *laboratory* sequence showing high detection accuracy and added various levels of random detection noise uniformly. We also randomly deleted detections from the same original sequence. That way, we artificially generated

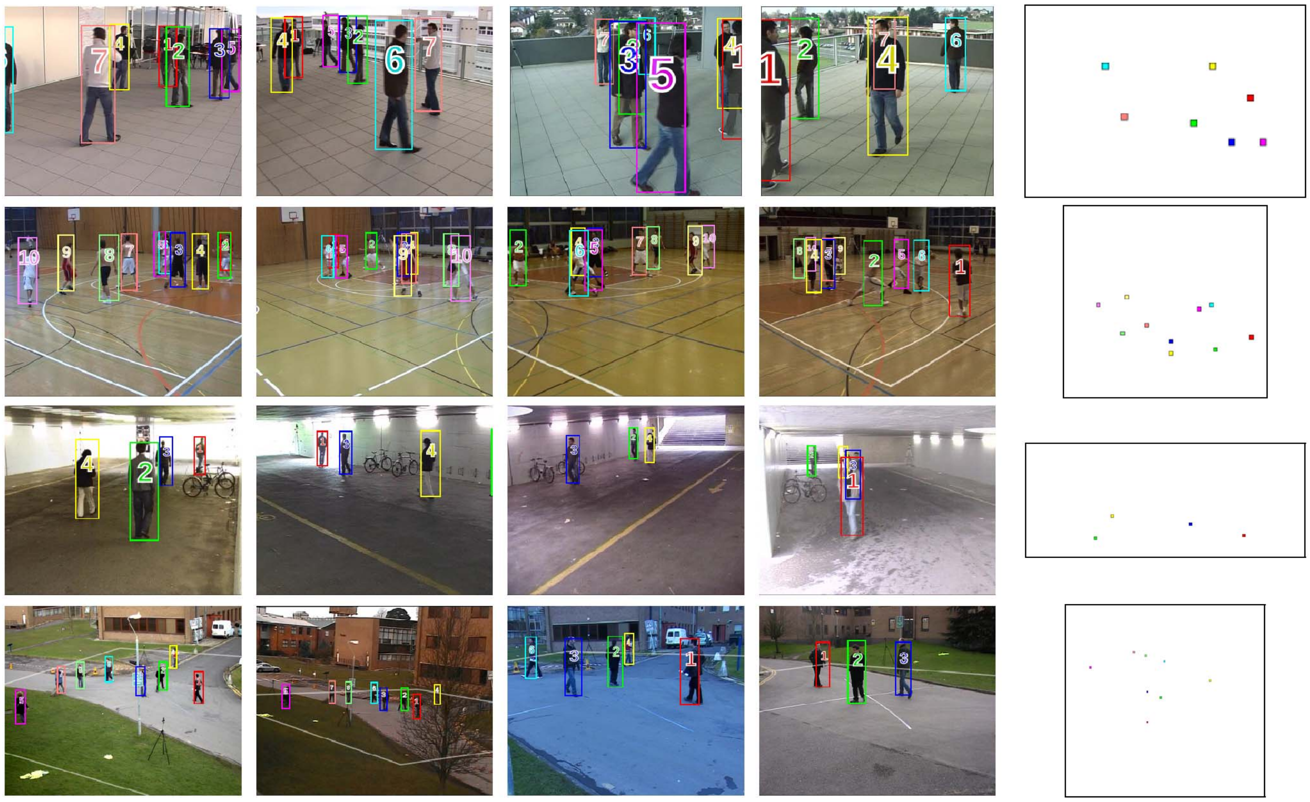


Fig. 8. Multicamera pedestrian tracking results in various environments. Each of the first four columns shows a different camera view. The fifth column displays the top view. The first row comes from the laboratory sequence, the second from the basketball environment, the third from the passageway, and the last one from PETS 2009.

controlled amounts of false positives and false negatives. The sequences thus generated were then processed by both DP and KSP tracking methods and evaluated using a known ground truth.

The results of this evaluation are presented in Fig. 6. The graph of Fig. 6a shows that KSP is more sensitive to false positives than DP. Beyond a density value, KSP is able to readily link false detections into—seemingly—coherent trajectories. Here, KSP's lack of motion model is a disadvantage over DP. Conversely, DP's tendency to leave out incomplete trajectories makes it more robust to this kind of noise. The graph of Fig. 6b shows the effect of missed detections. Both trackers react the same way: Beyond a false negative rate, the remaining detections are no longer linked together and remain unexplained. KSP nevertheless shows a much higher robustness to missed detections than DP does. This is consistent with our observations on real data: In Figs. 4 and 5, the difference between DP and KSP performance is usually larger when POM's occupancy maps have low accuracy.

Another problem to which our method is potentially vulnerable is identity switch. Since we rely entirely on detection data and do not use any appearance information or complex motion model, there is no way to distinguish two trajectories intersecting. In practice, we do not suffer much from this because, most of the time, the objects evolve outside of each other's neighborhood. Moreover, the joint optimization of all trajectories pays off in this regard, as opposed to DP's greedy strategy.

4.10 Runtime

Finally, we evaluate the speed of our new tracking algorithm. Solving the Linear Program with standard LP libraries [42] is slow, as evidenced by the curve labeled *LP* on the graph of Fig. 7a. Using the complexity reduction method of Section 3.4 decreases the computation time by a factor of 10, as shown by the curve labeled *LP w/compl. red*. Here, we pruned the graph by a radius of $\tau_1 = \tau_2 = 3$ (see (19)).

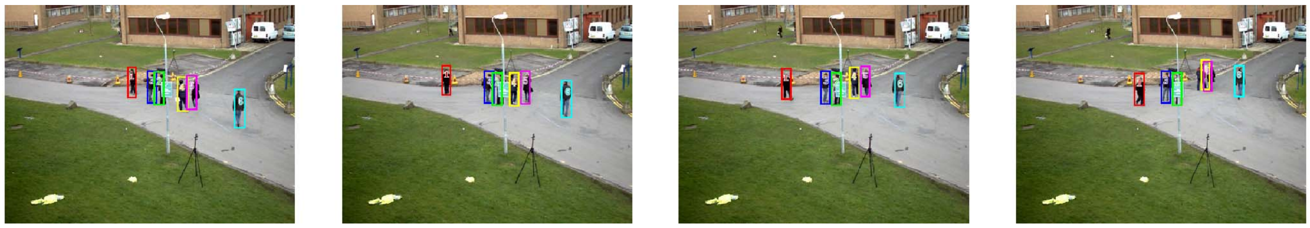


Fig. 9. Monocular pedestrian tracking results, from the PETS 2009 sequence.

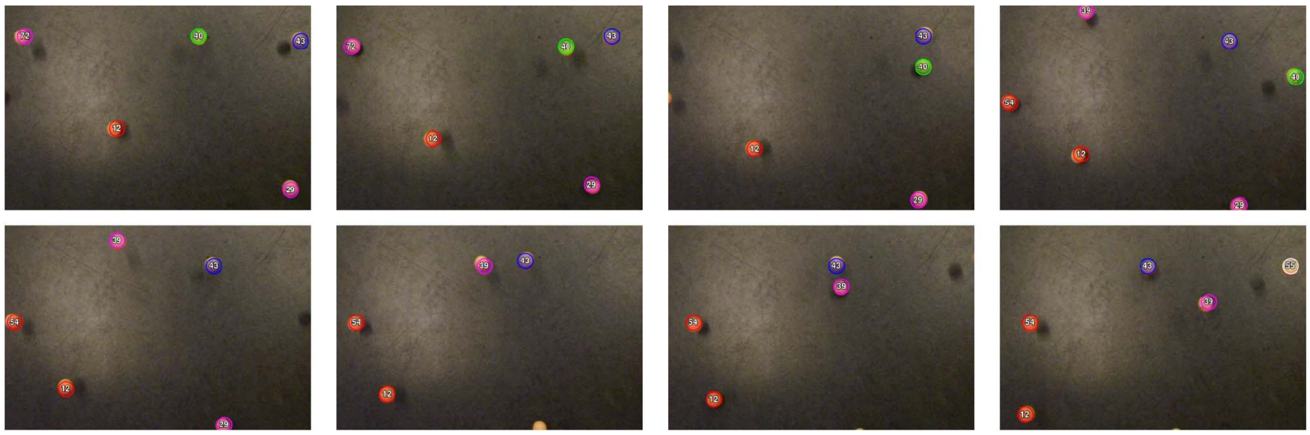


Fig. 10. Multiple ball tracking results. Successive screenshots are separated by three time frames.

By contrast, KSP is much faster: As illustrated in Fig. 7a, there is a considerable speed gain of a factor 100-1,000, compared to the generic LP solver [42]. And the gain is still very significant when compared to the LP solver using complexity reduction methods.

Compared to the DP algorithm, KSP is about 10 times faster. DP suffers from having to load videos in order to maintain its appearance models and from its redundant batch processing [3]. Interestingly, when dealing with 25 fps videos, KSP can, on average, process a batch of frames in less than half the time it takes to play it. This means that for a frame rate of 25 fps or less, our tracker can readily operate in real time.

Fig. 7b illustrates the grid size influence on runtime of both DP and KSP. Applying POM on a 200-frame excerpt of the *laboratory* sequence, we generated occupancy grids of different resolutions, which were then processed by the two tracking algorithms. Results show a linear dependency on the grid size, which is consistent with the average complexity of k-shortest paths.

All of the above experiments have been performed on a recent Linux PC, equipped with a 2.5 GHz Intel processor and 8 GB of memory. Tracking was applied to a part of the *laboratory* sequence in which five to seven people are present. For the k-shortest path, no particular optimization was performed nor did we use any of the complexity reduction methods of Section 3.4. The results of DP and KSP in Fig. 7 are the average of 20 runs, plotted with 95 percent confidence interval. This is barely noticeable because the values are very peaked around the average.

Finally note that whether solved with a generic LP package or the k-shortest paths algorithm, our framework always produces the exact same results, but KSP allows us to obtain them much faster.

5 Conclusion

Combining frame-by-frame detections to estimate the most likely trajectories of an unknown number of targets, including their entrances and departures to and from the scene, is one of the most difficult components of a multiobject tracking algorithm. We have shown that by formalizing the motions of targets as flows along the edges of a graph of spatio-temporal locations, we can reduce this difficult estimation problem to a standard Linear Programming one. By relying on the k-shortest paths algorithm for the optimization of our problem, we could reduce the complexity to a tiny fraction of the one from the original LP problem, yielding an efficient algorithm performing robust multi-object tracking in real time on a standard computer.

The resulting algorithm is far simpler than current state-of-the-art alternatives and its convexity ensures that a global optimum can be found. It obtains a better performance than a state-of-the-art method on difficult real-word applications, in spite of having access to a more limited signal and requiring fewer meta-parameters. Future work will focus on integrating additional cues to our framework, such as an appearance or a motion models, to robustly handle identities of intersecting trajectories.

Acknowledgments

The authors would like to thank Ali Shahrokni, Anna Ellis, and James Ferryman from the University of Reading for evaluating their results against the PETS 2009 ground truth. This work is supported by the Swiss National Science Foundation (SNSF) Sinergia project Aerial Crowds and by the SNSF under the National Centre of Competence in Research (NCCR) on Interactive Multimodal Information Management (IM2).

References

[1] J. Giebel, D. Gavrila, and C. Schnorr, “A Bayesian Framework for Multi-Cue 3D Object Tracking,” *Proc. European Conf. Computer Vision,* 2004.

[2] A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and H. Wensheng, “Multi-Object Tracking through Simultaneous Long Occlusions and Split-Merge Conditions,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 666-673, June 2006.

[3] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, “Multi-Camera People Tracking with a Probabilistic Occupancy Map,” *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 30, no. 2, pp. 267-282, Feb. 2008.

[4] H. Jiang, S. Fels, and J. Little, “A Linear Programming Approach for Multiple Object Tracking,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 744-750, 2007.

[5] G.B. Dantzig, *Linear Programming and Extensions.* Princeton Univ. Press, 1963.

[6] J.W. Suurballe, “Disjoint Paths in a Network,” *Networks,* vol. 4, pp. 125-145, 1974.

[7] J. Black, T. Ellis, and P. Rosin, "Multi-View Image Surveillance and Tracking," *Proc. IEEE Workshop Motion and Video Computing,* 2002.

[8] A. Mittal and L. Davis, "M2tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene," *Computer Vision and Image Understanding,* vol. 51, no. 3, pp. 189-203, 2003.

[9] S. Iwase and H. Saito, "Parallel Tracking of All Soccer Players by Integrating Detected Positions in Multiple View Images," *Proc. Int'l Conf. Pattern Recognition,* pp. 751-754, Aug. 2004.

[10] M. Xu, J. Orwell, and G. Jones, "Tracking Football Players with Multiple Cameras," *Proc. Int'l Conf. Image Processing,* pp. 2909-2912, Oct. 2004.

[11] D.R. Magee, "Tracking Multiple Vehicles Using Foreground, Background and Motion Models," *Image and Vision Computing,* vol. 22, no. 2, pp. 143-155, Feb. 2004.

[12] B. Wu and R. Nevatia, "Tracking of Multiple, Partially Occluded Humans Based on Static Body Part Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 951-958, June 2006.

[13] J. Vermaak, A. Doucet, and P. Perez, "Maintaining Multimodality through Mixture Tracking," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 1110-1116, Oct. 2003.

[14] K. Smith, D. Gatica-Perez, and J.M. Odobez, "Using Particles to Track Varying Numbers of Interacting People," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2005.

[15] Z. Khan, T. Balch, and F. Dellaert, "MCMC-Based Particle Filtering for Tracking a Variable Number of Interacting Targets," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 11, pp. 1805-1918, Nov. 2005.

[16] C. Yang, R. Duraiswami, and L. Davis, "Fast Multiple Object Tracking via a Hierarchical Particle Filter," *Proc. IEEE Int'l Conf. Computer Vision,* 2005.

[17] T. Mauthner, M. Donoser, and H. Bischof, "Robust Tracking of Spatial Related Components," *Proc. Int'l Conf. Pattern Recognition,* 2008.

[18] K. Okuma, A. Taleghani, N. de Freitas, J. Little, and D. Lowe, "A Boosted Particle Filter: Multitarget Detection and Tracking," *Proc. European Conf. Computer Vision,* May 2004.

[19] W. Du and J. Piater, "Multi-Camera People Tracking by Collaborative Particle Filters and Principal Axis-Based Integration," *Proc. Asian Conf. Computer Vision,* pp. 365-374, 2007.

[20] Q. Yu, G. Medioni, and I. Cohen, "Multiple Target Tracking Using Spatio-Temporal Markov Chain Monte Carlo Data Association," *Proc. IEEE Int'l Conf. Computer Vision,* 2007.

[21] E. Maggio, M. Taj, and A. Cavallaro, "Efficient Multi-Target Visual Tracking Using Random Finite Sets," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 18, no. 8, pp. 1016-1027, Aug. 2008.

[22] C. Huang, B. Wu, and R. Nevatia, "Robust Object Tracking by Hierarchical Association of Detection Responses," *Proc. European Conf. Computer Vision,* pp. 788-801, 2008.

[23] Y. Li, C. Huang, and R. Nevatia, "Learning to Associate: Hybridboosted Multi-Target Tracker for Crowded Scene," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* June 2009.

[24] C. Beleznai, B. Frühstück, and H. Bischof, "Multiple Object Tracking Using Local PCA," *Proc. Int'l Conf. Image Processing,* 2006.

[25] W. Ge and R.T. Collins, "Multi-Target Data Association by Tracklets with Unsupervised Parameter Estimation," *Proc. British Machine Vision Conf.,* Sept. 2008.

[26] R. Eshel and Y. Moses, "Homography Based Multiple Camera Detection and Tracking of People in a Dense Crowd," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2008.

[27] G.J. Brostow and R. Cipolla, "Unsupervised Bayesian Detection of Independent Motion in Crowds," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 594-601, 2006.

[28] P. Nillius, J. Sullivan, and S. Carlsson, "Multi-Target Tracking—Linking Identities Using Bayesian Network Inference," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 2187-2194, 2006.

[29] S. Khan and M. Shah, "Tracking Multiple Occluding People by Localizing on Multiple Scene Planes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 31, no. 3, pp. 505-519, Mar. 2009.

[30] B. Leibe, K. Schindler, and L.V. Gool, "Coupled Detection and Trajectory Estimation for Multi-Object Tracking," *Proc. Int'l Conf. Computer Vision,* Oct. 2007.

[31] R.E. Bellman, *Dynamic Programming.* Princeton Univ. Press, 1957.

[32] J. Wolf, A. Viterbi, and G. Dixon, "Finding the Best Set of K Paths through a Trellis with Application to Multitarget Tracking," *IEEE Trans. Aerospace and Electronic Systems,* vol. 25, no. 2, pp. 287-296, Mar. 1989.

[33] K. Shafique and M. Shah, "A Noniterative Greedy Algorithm for Multiframe Point Correspondence," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 1, pp. 51-65, Jan. 2005.

[34] P.P.A. Storms and 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, June 2003.

[35] L. Zhang, Y. Li, and R. Nevatia, "Global Data Association for Multi-Object Tracking Using Network Flows," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2008.

[36] N. Karmarkar, "A New Polynomial Time Algorithm for Linear Programming," *Combinatorica,* vol. 4, pp. 373-395, 1984.

[37] A. Ellis, A. Shahrokni, and J. Ferryman, "PETS 2009 and Winter-PETS 2009 Results: A Combined Evaluation," *Proc. 12th IEEE Int'l Workshop Performance Evaluation of Tracking and Surveillance,* Dec. 2009.

[38] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, M. Boonstra, V. Korzhova, and J. Zhang, "Framework for Performance Evaluation of Face, Text, and Vehicle Detection and Tracking in Video: Data, Metrics, and Protocol," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 31, no. 2, pp. 319-336, Feb. 2009.

[39] *Proc. 11th IEEE Int'l Workshop Performance Evaluation of Tracking and Surveillance,* http://pets2009.net, 2011.

[40] J. Berclaz, F. Fleuret, and P. Fua, "POM: Probabilistic Occupancy Map," http://cvlab.epfl.ch/software/pom/index.php, 2007.

[41] K. Bernardin and R. Stiefelhagen, "Evaluating Multiple Object Tracking Performance: The Clear MOT Metrics," *EURASIP J. Image and Video Processing,* vol. 2008, pp. 1-10, 2008.

[42] A. Makhorin, "GLPK—GNU Linear Programming Kit," http://www.gnu.org/software/glpk/, 2008.

[43] A. Ghouila-Houri, "Caractérisation Des Matrices Totalement Unimodulaires," *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences,* vol. 254, pp. 1192-1194, 1962.

[44] A.J. Hoffman and J.B. Kruskal, "Integral Boundary Points of Convex Polyhedra," *Linear Inequalities and Related Systems,* pp. 223-246, Princeton Univ. Press, 1956.

[45] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik,* vol. 1, pp. 269-271, 1959.

[46] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms,* second ed. MIT Press, 2001.

[47] Y. Ma, Q. Yu, and I. Cohen, "Target Tracking with Incomplete Detection," *Computer Vision and Image Understanding,* vol. 113, no. 4, pp. 580-587, 2009.

Jérôme Berclaz received the MS degree in communication systems in 2004 and the PhD degree in computer vision in 2010 from the EPFL (Swiss Federal Institute of Technology). He is now a postdoctoral researcher in the Computer Vision Laboratory and the Signal Processing Laboratory at the EPFL. His main research interest is computer vision.

François Fleuret received the PhD degree in probability from the University of Paris VI in 2000 and the habilitation degree in applied mathematics from the University of Paris XIII in 2006. He holds a senior researcher position at the Idiap Research Institute in Switzerland. Prior to that, he held positions at the University of Chicago, the French Institut de Recherche en Informatique et en Automatique (INRIA), and the École Polytechnique Fédérale de Lausanne (EPFL). His main research interests are at the interface between statistical methods and algorithmic, centered on the development of algorithmically efficient machine learning techniques.

Engin Türetken received the BSc and MSc degrees in electrical and electronics engineering from the Middle East Technical University in 2005 and 2008, respectively. He is currently working toward the PhD degree in the school of Computer and Communication Sciences at the Swiss Federal Institute of Technology (EPFL). His research interests include computer vision, graph theory, and combinatorial optimization.

Pascal Fua received the engineering degree from the Ecole Polytechnique, Paris, in 1984 and the PhD degree in computer science from the University of Orsay in 1989. He joined EPFL (Swiss Federal Institute of Technology) in 1996, where he is now a professor in the School of Computer and Communication Science. Before that, he worked at SRI International and INRIA Sophia-Antipolis as a computer scientist. His research interests include shape modeling and motion recovery from images, analysis of microscopy images, and augmented reality. He has (co)authored more than 150 publications in refereed journals and conferences. He has been an associate editor of the *IEEE journal Transactions on Pattern Analysis and Machine Intelligence* and has often been a program committee member, area chair, and program chair of major vision conferences. He is a senior member of the IEEE.