IDIAP RESEARCH REPORT

MULTIPLE OBJECT TRACKING USING FLOW LINEAR PROGRAMMING

Jerome Berclaz Francois Fleuret Pascal Fua

Idiap-RR-10-2009

JUNE 2009

Centre du Parc, Rue Marconi 19, P.O. Box 592, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

Multiple Object Tracking using Flow Linear Programming

Jérôme Berclaz[1*] François Fleuret[2†]
Pascal Fua[1]
[1] CVLab, EPFL, Lausanne, Switzerland
[2] Idiap Research Institute, Martigny, and EPFL, Lausanne, Switzerland

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 problem results in a convex problem that can be solved using standard Linear Programming techniques. In addition, this new approach is far simpler formally and algorithmically than existing techniques and lets us demonstrate excellent performance in two very different contexts.

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.

*supported by the Indo Swiss Joint Research Programme (ISJRP)
†supported by the Swiss National Science Foundation under the National Centre of Competence in Research (NCCR) on Interactive Multimodal Information Management (IM2).

However, while it is easy to design a statistical trajectory model with all 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 [14] or by greedy Dynamic Programming in which trajectories are estimated one after another [5]. None of these approaches guarantees a global optimum. A notable exception is a recent approach [8] that relies on Linear Programming [3] to find a global optimum 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 were 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 problem results in a convex problem that can also be solved using Linear Programming techniques, but without any of the above limitations or even having to require an appearance model. The latter does of course not mean that one should not be used if available but making it optional increases the range of applicability of our method.

Furthermore, our approach is far simpler formally and algorithmically than existing techniques and 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*, 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 do not appear or disappear except at specified entrances and exits, do not move too quickly, and cannot share a location with another object. These assumptions are 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. It solution is a set of flows that are both consistent and binary so that linking detections becomes trivial.

Our main contribution is therefore 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.

2 Related Work

Kalman filtering is an efficient way to address multi-target tracking [2, 10, 7, 18] when the number of objects remains small. However, when it increases, mistakes become more frequent and are difficult to correct due to the recursive nature of the method. Particle filtering can avoid this by exploring multiple hypotheses [16, 6, 14]. It has been used to great effect to follow multiple hockey

players [12] and to track multiple people in the ground and image planes simultaneously [4]. However, in our experience, it typically requires careful tuning of several metaparameters, which reduces the generality of methods that rely on it.

Less conventional is the approach of [11], which formulates the multi-object tracking as a Bayesian network inference problem and applies this method to tracking multiple soccer players. They assume that a track graph has already been produced and concentrate on linking identities in the provided track graph.

Dynamic Programming can be used to link multiple detections over time, and therefore solve the multi-target tracking problem. Moreover, it can be extended to enable the optimization of several trajectories simultaneously [17]. Unfortunately, the computational complexity of such an approach can be prohibitive. To overcome this limitation, [5] sequentially applies Dynamic Programming over individual trajectories, which are assumed to be 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 [13], where a directed graph, with nodes standing for actual detections, represents the multi-frame 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 on air radar detections [15] or tackle multiple people tracking [8]. 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 [13]. Objects hiding each other are modeled by specifying spatial conflicts within nodes. Occlusions are handled by introducing a special node type and arc costs are chosen according to object appearances and motion model. Additionally, another soft constraint helps ensuring spatial layout consistency.

Due to its reduced state-space, this method is computationally efficient. However, it requires *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 only consisting of observations, as opposed to all possible locations as in our approach, it can not smoothly interpolate trajectories in case of false negatives. Moreover, the choice of arc costs is rather ad-hoc and involves many parameters, which have to be tuned for each possible application, reducing the generality of the method. In comparison, our model is far simpler, with the neighbourhood size being the only value that needs to be adapted.

3 Algorithm

In this section, we first formulate multi-target tracking as a discrete Linear Programming problem. Since such a problem is NP-Complete in its discrete version, we solve a continuous version of it, which is far easier to do. This results in a set of flow variables that can easily be linked to provide complete trajectories. We discuss these steps in more detail below.

3.1 Formalization

Table 1: Notation

K	number of spatial locations;
T	number of time steps;
$\mathbf{I} = (\mathbf{I}^1, \ldots, \mathbf{I}^T)$	captured images;
$\mathcal{N}(k) \subset \{1, \ldots, K\}$	neighborhood of location k;
$f_{i,j}^t$	estimated number of objects moving from location i at time t to location j at time $t+1$;
m_i^t	estimated number of objects at location i at time t;
M_i^t	random variable standing for the true number of objects at location i at time t;
$\mathfrak{F}$	set of occupancy maps physically possible.

We discretize the physical area of interest into K locations, and the time interval into T instants. For any location k, let $\mathcal{N}(k) \subset \{1, \ldots, K\}$ denote the neighbourhood of k, that is, 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 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_i^t standing for the number of objects located at i at time t. Each edge is labeled with a discrete variable $f_{i,j}^t$ standing for the number of objects moving from location i at time t to location j at time $t+1$, as shown in Fig. 1. For instance, the fact that an object remains at location i between times t and $t+1$ is represented by $f_{i,i}^t = 1$

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

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

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

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

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

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

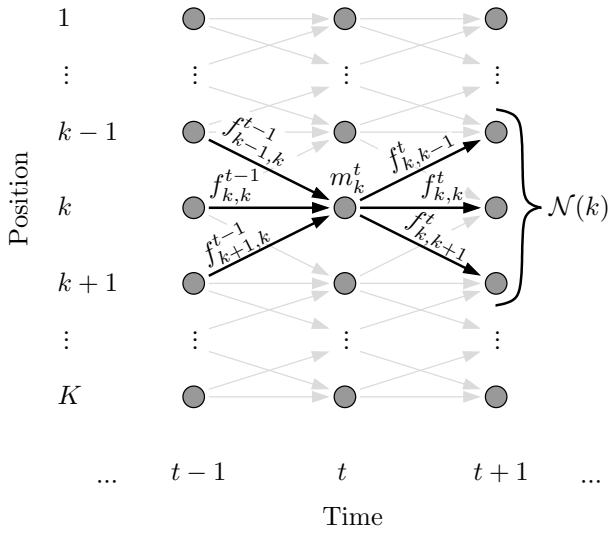


Figure 1: Simplified flow model, which does not use a virtual position. Positions are arranged on one dimension and neighborhood is reduced to 3 positions.

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

$$\rho_i^t = \hat{P}(M_i^t = 1 \,|\, \mathbf{I}^t)\,, \tag{4}$$

where $\mathbf{I}^t$ is the signal available at time t. For the multi-camera pedestrian-tracking application, $\mathbf{I}^t$ 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 Eqs. 1, 2, and 3, and we define $\mathfrak{F}$ the set of feasible maps. Our goal then becomes solving

$$\mathbf{m}^* = \arg\max_{\mathbf{m}\in\mathfrak{F}} \hat{P}(\mathbf{M} = \mathbf{m} \,|\, \mathbf{I}) \quad . \tag{5}$$

Assuming conditional independence of the M_i^t, given the $\mathbf{I}^t$, the optimization

problem of Eq. 5 can be re-written as

$$\begin{aligned}
\mathbf{m}^* &= \arg\max_{\mathbf{m}\in\mathfrak{F}} \log \prod_{t,i} \hat{P}(M_i^t = m_i^t \,|\, \mathbf{I}^t) && (6)\\
&= \arg\max_{\mathbf{m}\in\mathfrak{F}} \sum_{t,i} \log \hat{P}(M_i^t = m_i^t \,|\, \mathbf{I}^t) \\
&= \arg\max_{\mathbf{m}\in\mathfrak{F}} \sum_{t,i} (1 - m_i^t) \log \hat{P}(M_i^t = 0 \,|\, \mathbf{I}^t) \\
&\qquad\qquad + m_i^t \log \hat{P}(M_i^t = 1 \,|\, \mathbf{I}^t) && (7)\\
&= \arg\max_{\mathbf{m}\in\mathfrak{F}} \sum_{t,i} m_i^t \log \frac{\hat{P}(M_i^t = 1 \,|\, \mathbf{I}^t)}{\hat{P}(M_i^t = 0 \,|\, \mathbf{I}^t)} && (8)\\
&= \arg\max_{\mathbf{m}\in\mathfrak{F}} \sum_{t,i} \left(\log \frac{\rho_i^t}{1 - \rho_i^t} \right) m_i^t \,, && (9)
\end{aligned}$$

where (6) is true under the assumption of conditional independence of the M_i^t given $\mathbf{I}^t$, (7) is true because m_i^t is 0 or 1 according to (2), and (8) is true by ignoring a term which does not depend on $\mathbf{m}$. Hence, our objective function (9) is a linear expression of the m_i^t.

In general, the number of tracked objects may vary with 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 an additional virtual location υ into our state space, which is linked to all the positions through which objects can enter or exit the area, such as doors or borders of the camera field of view.

As opposed to other flows, those originating from υ are not subject to the constraint of Eq. 1 and 2, because the virtual location υ theoretically contains all targets not present in the monitored area. Also, several objects can simultaneously enter or exit the area, as long as there are enough entrance points.

3.2 Optimization

For a sequence of T frames with K positions, each having N neighbors, our optimization problem has TK occupancy variables and TKN flow variables. And since, for every location at every time frame there are two constraints in addition to positivity, the variables are linked by $2TK$ constraints. In other words, for a 1000-frame people tracking sequence on a grid of 1000 locations, each with 9 neighbors, our minimization problem involves 9,000,000 variables and 2,000,000 constraints, which is far beyond what discrete optimization algorithms can handle.

This is however not true of continuous optimization schemes and treating the m_i^t and $f_{i,j}^t$ as real-valued variables and optimizing our criterion under the constraints of Eqs. 1, 2, and 3 yields a convex Linear Programming problem whose global optimum can actually be computed on a standard PC. The complexity can be further decreased by pruning the graph and performing batch processing, as described in § 3.2.2.

Furthermore, we observe experimentally that solving the real-valued system always produces Boolean values, which is not surprising: If the objective func-

tion can be increased by putting mass along certain edges, there is no reason not to reach the constraint of Eq. (2). For instance, if false alarms generated by the detector make two different trajectories possible, the slightest numerical difference between the sum of the $\log \frac{\rho_i^t}{1-\rho_i^t}$ over either one will break the symmetry and the global optimum will correspond to 1s over the path with highest value and 0s over the other.

While we can imagine pathological situations, such as targets moving at extremely close range for a long period of time, which may lead to a non-Boolean optimum, such cases are extremely rare in practice. After processing all sequences shown in § 4, our system predicted a total of $26,400,000$ values, $26,375,847$ of which are in the range $[0, 0.01]$, and $24,153$ in the range $[0.99, 1]$. We observed no value in the interval $[0.01, 0.99]$.

3.2.1 Flow Formulation

We optimize with respect to the $f_{i,j}^t$ rather than the m_i^t because there is no natural way to express the flow continuity in terms of the latter. We therefore solve the following Linear Programming problem:

$$\begin{aligned} \text{Maximize} \quad & \sum_{t,i} \log\left(\frac{\rho_i^t}{1-\rho_i^t}\right) \sum_{j \in \mathcal{N}(i)} f_{i,j}^t \\ \text{under} \quad & \forall t, i, j, \ f_{i,j}^t \geq 0 \\ & \forall t, i, \sum_{j \in \mathcal{N}(i)} f_{i,j}^t \leq 1 \\ & \forall t, j, \sum_{i : j \in \mathcal{N}(i)} f_{i,j}^{t-1} = \sum_{k \in \mathcal{N}(j)} f_{j,k}^t \ . \end{aligned} \tag{10}$$

In practice, we use the Simplex algorithm [3] for which standard implementations exist [9].

3.2.2 Complexity Reduction

As discussed above, the number of variables of our optimization problem is very high and solving it directly is only practical for moderately sized grids. This limitation can be overcome using two simple techniques.

Pruning the Graph 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. In other words, given loose upper bounds on the speed of the objects to track and on the maximum number of false negatives the detector can produce successively, we can build a criterion to remove nodes of the graph which are very unlikely to ever be occupied.

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

$$\max_{\substack{\|j-k\|<\tau_1 \\ t-\tau_2<u<t+\tau_2}} \rho_j^u \ . \tag{11}$$

If it is found to be below a threshold, the location is considered as unused because no object could reach it with any reasonable level of probability. All flows to and from it are then removed from the model. Applying this method allows us to reduce the number of variables and constraints up to a factor of 10.

Batch Processing Instead of directly optimizing a whole video sequence, one can separate it into several batches of frames and optimize over them individually. 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

$$\forall k \in \{1, \ldots, K\}, \sum_{j \in \mathcal{N}(k)} f_{k,j}^{-1} = \mu_k, \tag{12}$$

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 Linear Programming framework.

3.3 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, we can follow the motion back in time by moving along the edges whose $f_{i,j}^t$ are greater than 0, and build the corresponding long trajectories.

4 Results

In this section, we present results in two very different contexts. First, we use a multi-camera setup in which the cameras are located at shoulder level to track pedestrians who may walk in front of each other. This produces numerous occlusions and allows us to demonstrate the ability of our approach to deal with them. 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 a state-of-the-art linking method based on sequential Dynamic Programming [5].

4.1 Test Data

The first set of data we acquired consists of a multi-camera video sequence of pedestrians. The chosen location for data acquisition is an underground passageway leading to a train platform and corresponds to a realistic video-surveillance environment, with all the associated shortcomings. Over the whole recording, the sequence shows various levels of activity. We extracted the most active segments and used them for testing.

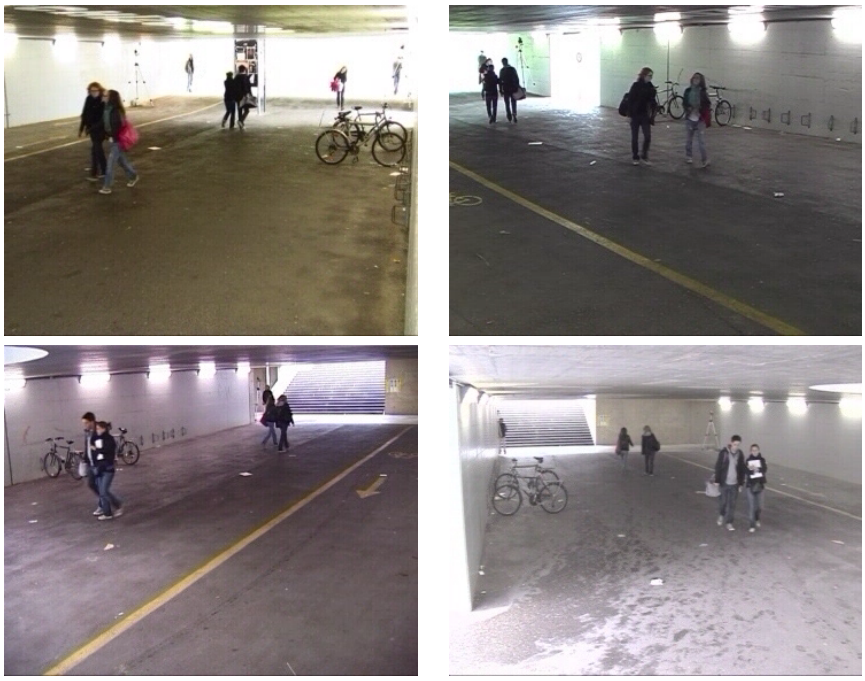


Figure 2: The multi-camera setup we used for testing. Each image corresponds to a different camera. The images are small and the lighting conditions far from ideal.

The area was filmed by 4 Digital Video cameras with overlapping field of view. The video format is DV PAL, down-sampled to 360×288 pixels at 25 fps and the 4 video streams have been synchronized manually after data acquisition. The area covered by the system is about 12m × 30m wide and has been discretized into a ground plane grid of 40 × 100 locations, displayed on Fig. 3.

As illustrated in the first three rows in Fig. 6, this is a very challenging environment for several reasons. First, lighting conditions are very poor, representative of what can be expected in a real-world surveillance application. Most images are under-exposed, except near the exits where they often are saturated. Second, the area covered by the system is large, which means that people can get very small when reaching the far end, making their precise localization challenging. Third, because of the low ceiling, the cameras had to be placed relatively low, which generates occlusions and makes localization more error-prone. Finally, large parts of the area of interest, especially near its edges, are seen by only two or even one cameras.

All these difficulties put together greatly affect the performance of the POM detector described below. As illustrated by Fig. 4, the detection maps are generally very noisy, with some people wrongly located or simply ignored for significant numbers of consecutive frames. On those sequences, the performance of the detector is varying between 70% and 80% of correct detections. Extracting correct paths out of this data is therefore a challenging task.

To demonstrate the ability of our algorithm to track a larger number of

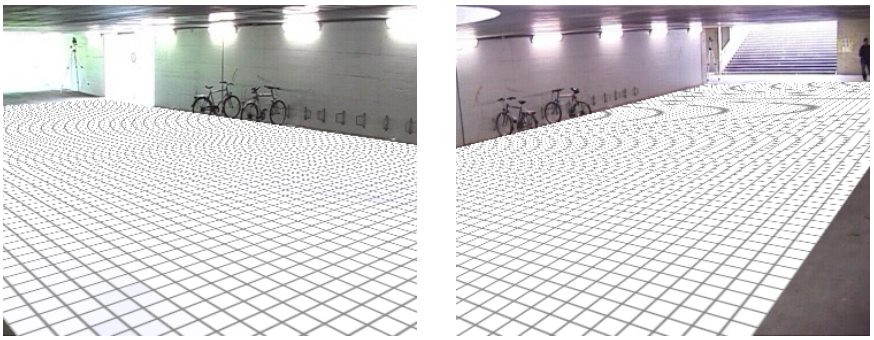


Figure 3: Ground plane grid used for people tracking.

people with frequent occlusions, we also applied it to a multi-camera video sequence featuring several people in a small 10m $\times$ 15m area, shown in the fourth row of Fig. 6. We used the same video equipment as before.

The second set of data is a video sequence in which 24 table tennis balls were launched across the field of view, with up to 10 appearing simultaneously on screen. Those were filmed by a single DV camera, placed facing down about 1.5m above the ground. The videos were cropped to a resolution of 500$\times$300 pixels, and the corresponding area was discretized into a grid of 50 $\times$ 30 locations.

4.2 Probabilistic Occupancy Map

For all our sequences, the detection data used as input by our tracker was generated with the publicly available implementation [1] of the Probabilistic Occupancy Map (POM) algorithm [5].

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 Eq. 6.

In the multi-camera 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 cm $\times$ 30 cm, as illustrated by Fig. 3. Correspondences 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.

To process the single-camera 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.

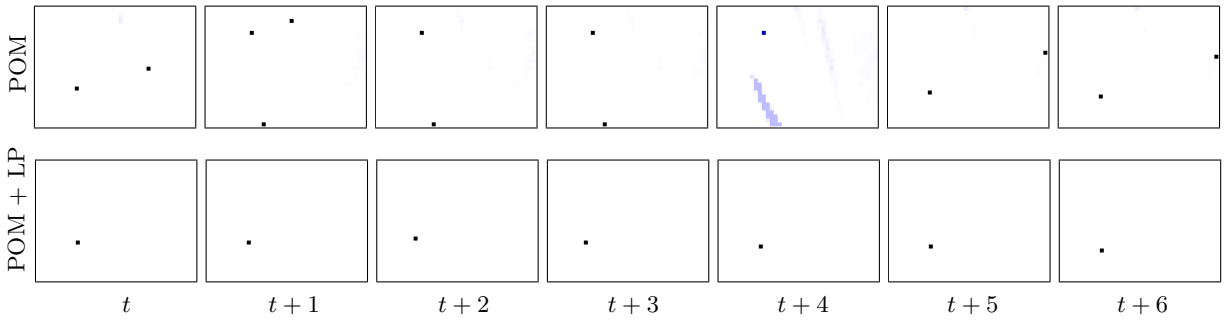


Figure 4: Original probabilistic occupancy maps for 7 consecutive frames of a pedestrian sequence (upper row) compared to the output of our Linear Programming algorithm (lower row). The darker the color, the higher the probability of presence. Note that the POM maps are extremely noisy, as evidenced by the fact that the number of probability peaks and their locations vary wildly. By contrast, only one peak remains in the LP output, and it moves slowly, which is consistent with the motion of a person over 1/4th of a second.

4.3 Baseline

To provide a baseline for comparison, we implemented the sequential Dynamic Programming approach of [5]. 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 [5] 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. In the rest of this section we refer to our Linear Programming method as 'LP' and to the sequential Dynamic Programming as 'DP'.

4.4 Pedestrian Tracking Results

For pedestrians tracking we define the graph of Fig. 1 as follows: Every interior location of the ground plane at time t is linked to its 9 direct neighbors at time $t+1$, which means that a pedestrian can move at most from one location to its immediate neighbors between frames. Border locations through which access to the area is possible are connected to the virtual location v. This arrangement is consistent with our chosen grid quantization at 25 fps.

To quantify our results, we manually labeled some of the test sequences. We proceeded by localizing people on the ground plane once every 25 frames. We not only marked their position, but also their identity, in order to assess how well the tracker follows people without switching their identities. We split this ground truth data into three sequences from the passageway data set:

- Seq. 1: A 2,500-frame sequence with 6 people, including one riding a bicycle;

- Seq. 2: A 1,000-frame sequence with 5 people, 4 of whom are walking side-by-side;
- Seq. 3: A 1,000-frame sequence with 4 people, one of whom is running.

In all three sequences, some of the people enter and exit the field of view at different times. We perform the LP optimization on a standard PC with 4 Gigabits of RAM, which is not particularly large anymore. The average run time is around one hour per 500-frame batch on the large grid of Fig. 3 containing about 4000 locations.

We measure the performance of both DP and LP methods using two metrics. The first one, plotted on Fig. 5, is expressed in terms of true positive and false positive rates. A true positive is counted when the tracker retrieves an object with its correct identity. This means that if two identities are switched, this generates two false positives - 2 objects are found at the wrong place - and two false negatives - the same 2 objects are not found at their correct location. The second metric focuses on trajectories and is displayed in Table 2. A ground truth trajectory is judged good, if the tracker associates the same person to it for at least 80% of the time. It is called mixed, when explained at least 80% of the time, but with identity switches. Finally, a trajectory is considered as lost if 20% or more of its locations are left unexplained.

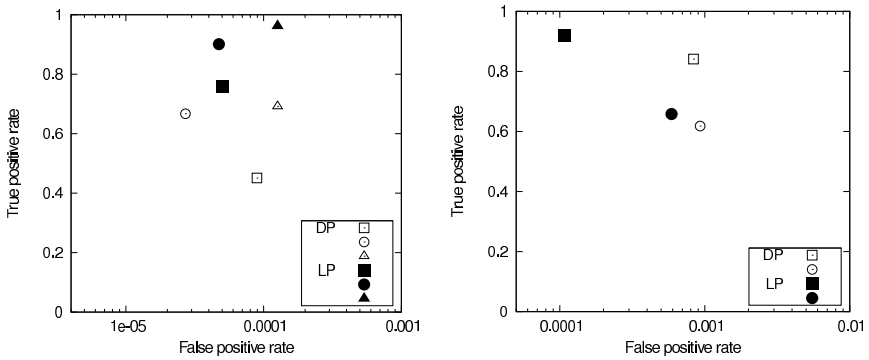
As evidenced on Fig. 5.a, LP produces increased true positive rates for every test sequence, with almost no impact on the false positive rate. Table 2.a also shows the superior performance of LP in the quality of the trajectories extracted. These results are all the more significant that the LP method makes far fewer assumptions than the DP one, which relies on an appearance model that must remain constant over time and is therefore vulnerable to appearance changes, such as those caused by lighting changes. It is also much simpler from an algorithmic point of view. Examples of LP tracking results are depicted in Fig. 6 by overlaying rectangles representing detections over the original images.

4.5 Ball Tracking Results

Since the balls move much faster than pedestrians, we handle them by simply extending the size of the location's neighborhood in the graph of Fig. 1. For this experiment, we allow a ball to travel as far as 4 locations between two consecutive frames, thus defining a location's neighborhood to be its 81 closest neighbors. As for the pedestrians, border locations are connected to the virtual location v that allows entrances and exits.

Ground truth was generated for two balls sequences of about 1000 frames each in the same manner as for the pedestrians data set. Because of the balls' higher speed, one in every 3 frames was labeled.

The results of both DP and LP methods according to the two metrics defined above are shown in Fig. 5.b and Table 2.b. Since the balls all have the same appearance, we turned off the appearance model in the DP implementation. In both test sequences, LP produces more true positives and fewer false positives, while yielding an even more striking improvement in terms of trajectories. Some LP balls tracking results are depicted in the last row of Fig. 6



(a) pedestrians (b) balls

Figure 5: True positive vs. false positive rates. Each graph represents error rates obtained using Dynamic Programming (DP) in white, and Linear Programming (LP) in black. Different shapes correspond to different sequences. (a) shows results for pedestrian tracking and (b) for ball tracking. Note that LP results are always further up than the corresponding DP ones and, except in one case, to the left.

	Seq. 1		Seq. 2		Seq. 3	
	DP	LP	DP	LP	DP	LP
good	4 / 13	8 / 13	5 / 6	6 / 6	4 / 5	5 / 5
mixed	1 / 13	0 / 13	0 / 6	0 / 6	0 / 5	0 / 5
lost	8 / 13	5 / 13	1 / 6	0 / 6	1 / 5	0 / 5

(a) Sequences with pedestrians

	Seq. 1		Seq. 2	
	DP	LP	DP	LP
good	33 / 40	39 / 40	50 / 72	64 / 72
mixed	3 / 40	1 / 40	8 / 72	3 / 72
lost	4 / 40	0 / 40	14 / 72	5 / 72

(b) Sequences with balls

Table 2: Comparison of the trajectories extracted by the Dynamic Programming (DP) and Linear Programming (LP) methods on both pedestrians (a) and balls (b) video sequences.

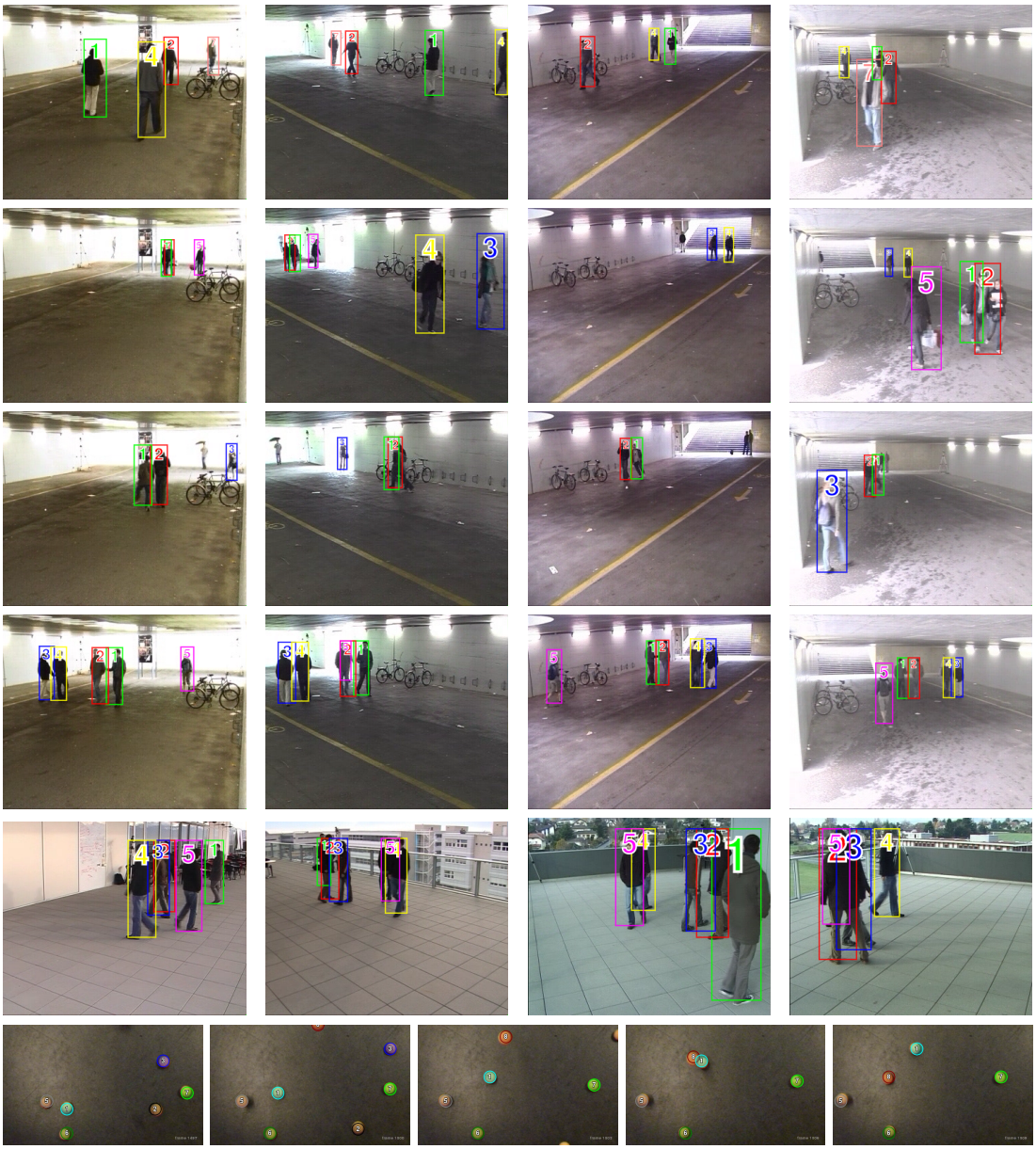


Figure 6: Examples of tracking results by the Linear Programming approach. Each one of the first 4 rows shows the images taken by all 4 cameras at the same time. The first 3 rows are from the underground passageway data set and the 4th from our second pedestrian data set. The last row shows tracking results on a sequence with multiple balls, with only one in every three frames displayed.

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 component of a multi-object 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. 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 results in 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.

Our approach moves the burden to the optimization scheme and the number of variables we have to handle in our approach prevents us from exploiting its full potential. The ability to handle very fine spatial resolutions and long time sequences jointly would further increase the appeal of the method.

We are therefore now investigating the optimization scheme itself. We have so far used a standard optimizer that does not exploit the specificities of our problem, which is why the processing is slow. However, the optimization could be made much less costly by performing it directly in the subspace spanned by maps that conserve mass instead of introducing mass conservation as a constraint. Other techniques based on hierarchical partitioning of the graph may also provide alternative ways to very significantly reduce the computational cost.

References

[1] J. Berclaz, F. Fleuret, and P. Fua. Pom: Probabilistic occupancy map, 2007. http://cvlab.epfl.ch/software/pom/index.php.

[2] J. Black, T.J. Ellis, and P. Rosin. Multi-view image surveillance and tracking. In *IEEE Workshop on Motion and Video Computing*, 2002.

[3] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.

[4] Wei Du and Justus Piater. Multi-camera people tracking by collaborative particle filters and principal axis-based integration. In *Asian Conference on Computer Vision*, pages 365–374, 2007.

[5] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multi-Camera People Tracking with a Probabilistic Occupancy Map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):267–282, February 2008.

[6] J. Giebel, D.M. Gavrila, and C. Schnorr. A bayesian framework for multi-cue 3d object tracking. In *Proceedings of European Conference on Computer Vision*, 2004.

[7] S. Iwase and H. Saito. Parallel tracking of all soccer players by integrating detected positions in multiple view images. In *International Conference on Pattern Recognition*, volume 4, pages 751–754, August 2004.

[8] Hao Jiang, Sidney Fels, and James J. Little. A linear programming approach for multiple object tracking. In *Conference on Computer Vision and Pattern Recognition*, pages 744–750, 2007.

[9] A. Makhorin. Glpk- gnu linear programming kit, 2008. http://www.gnu.org/software/glpk/.

[10] A. Mittal and L. Davis. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *International Journal of Computer Vision*, 51(3):189–203, 2003.

[11] Peter Nillius, Josephine Sullivan, and Stefan Carlsson. Multi-target tracking - linking identities using bayesian network inference. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2187–2194, 2006.

[12] K. Okuma, A. Taleghani, N. de Freitas, J.J. Little, and D.G. Lowe. A boosted particle filter: multitarget detection and tracking. In *ECCV*, Prague, Czech Republic, May 2004.

[13] Khurram Shafique and Mubarak Shah. A noniterative greedy algorithm for multiframe point correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):51–65, January 2005.

[14] K. Smith, D. Gatica-Perez, and J.-M. Odobez. Using particles to track varying numbers of interacting people. In *Conference on Computer Vision and Pattern Recognition*, 2005.

[15] P. P. A. Storms and F. C. R. Spieksma. An lp-based algorithm for the data association problem in multitarget tracking. *Computers & Operations Research*, 30(7):1067–1085, June 2003.

[16] J. Vermaak, A. Doucet, and P. Perez. Maintaining multimodality through mixture tracking. In *International Conference on Computer Vision*, volume 2, pages 1110–1116, October 2003.

[17] J.K. Wolf, A.M. Viterbi, and G.S. Dixon. Finding the best set of k paths through a trellis with application to multitarget tracking. *Aerospace and Electronic Systems, IEEE Transactions on*, 25(2):287–296, Mars 1989.

[18] Ming Xu, J. Orwell, and G. Jones. Tracking football players with multiple cameras. In *International Conference on Image Processing*, volume 5, pages 2909–2912, October 2004.