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, Switzerland

jerome.berclaz@epfl.ch, francois.fleuret@idiap.ch, pascal.fua@epfl.ch

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 yields excellent results on the PETS 2009 data set.

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 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 [16] or by greedy Dynamic Programming in which trajectories are estimated one after another [6]. None of these approaches guarantees a global optimum. A notable exception is a recent approach [9] that relies on Linear Programming [4] 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 is shown to perform well on some of the difficult sequences of the PETS 2009 data set.

For processing these sequences, 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

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

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 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 [3, 12, 8, 20] 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 [18, 7, 16]. It has been used to great effect to follow multiple hockey players [14] and to track multiple people in the ground and image planes simultaneously [5]. 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 [13], 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 [19]. Unfortunately, the computational complexity of such an approach can be prohibitive. To overcome this limitation, [6] 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 [15], 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 [17] or tackle multiple people tracking [9]. 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 [15]. 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.

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.

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 $\mathcal{N}(k) \subset \{1, \ldots, K\}$ denote the neighborhood of k, that is,

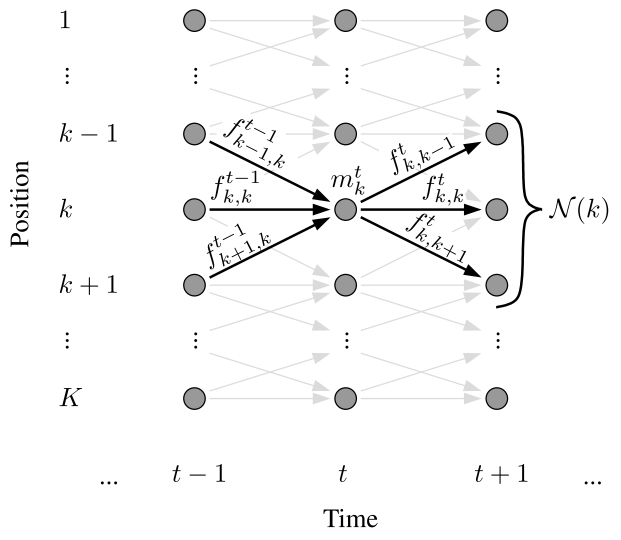


Figure 1: Simplified flow model, which does not use a virtual position. Positions are arranged on one dimension and neighborhood is reduced to 3 positions.

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

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

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

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

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

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

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 v are not subject to the constraint of Eq. 1 and 2, because the virtual location v 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 function 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 [4] for which standard implementations exist [11].

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

To demonstrate the effectiveness of our algorithm for people-tracking purposes, we use video sequences from the PETS 2009 data set. We rely on the Probabilistic Occupancy Map (POM) algorithm [6], which we briefly describe in § 4.2, to compute the required detections. We compare our results to those of our earlier approach using sequential Dynamic Programming [6], first in the multi-camera context, and then using only one camera.

4.1 Test Data

We test our algorithm on the multi-camera sequence S2-L1 from the PETS 2009 data set. In this video, 7 cameras observe several pedestrians under various angles. Among the cameras, 4 of them are located relatively close to the scene, and at roughly the height of people's heads. The 3 remaining cameras are located further from the monitored area and about 4-5m above the ground, giving a wide angle view of the situation. The frame rate for all cameras is about 7 fps.

The area monitored by our system is the $18\text{m}\times 20\text{m}$ rectangle shown in Fig. 2. This space is discretized into 55 $\times$ 61 = 3,355 locations. Correspondences between camera views and the grid is ensured through camera calibration. We adapt our graph flow to the pedestrian tracking framework as follows: Every interior location of the ground plane at time t is linked to its 9 direct neighbors at time $t+1$, as illustrated in Fig. 3, which means that a pedestrian can only move from one location to its immediate neighbors between consecutive frames. As explained in §3.1, border locations through which access to the area is possible are connected

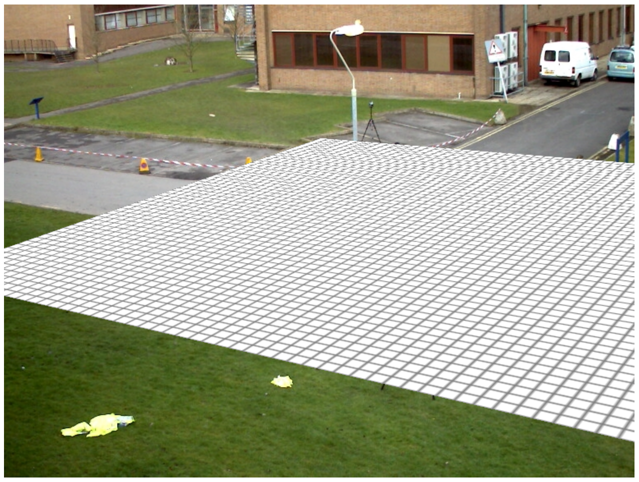


Figure 2: Ground plane grid used for pedestrian detection.

to the virtual location v. Despite the slow frame rate of the data set, this simple model is sufficient to obtain accurate tracking. Should we deal with even lower frame rates, we could easily modify this model to connect every location with its 25 closest neighbors, or more if needed.

4.2 Probabilistic Occupancy Map

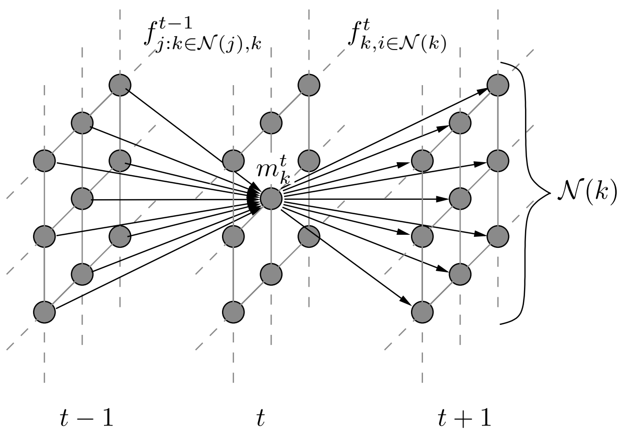
For this evaluation, the detection data used as input by our tracker is generated with the publicly available implementation [1] of our Probabilistic Occupancy Map (POM) algorithm [6].

It 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.

The generative model at the heart of POM represents people as cylinders that project to rectangles in the images. An evaluation of the POM detection algorithm against the PETS 2009 dataset can be found in [2].

4.3 Baseline

To provide a baseline for comparison, we use our earlier sequential Dynamic Programming approach [6]. It involves



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

Figure 3: Flow model for the pedestrian tracking application. Here, we plot flows arriving to and departing from location k at time t.

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 [6] 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 Multi Camera Results

To evaluate our algorithm and to compare it to the baseline, we first ran the POM detector on the sequence S2-L1. Due to some observed inaccuracies in the camera calibration data, we only used 5 of the 7 available views to perform detection. This first step generated a probabilistic occupancy map for every frame of the sequence.

The detection results were then fed to both the DP and LP algorithms, which produced individual trajectories. Examples of LP tracking results are displayed in Fig. 7. The tracking results of both methods have been evaluated with the CLEAR metrics [10] for detection and tracking, respectively shown on Fig. 4 and 5.

Note that we used our own implementation of the CLEAR metrics. We followed the description of [10] as closely as possible, but there might be some minor differences. We also generated our own ground truth for the test sequence. There might therefore be small discrepancies with the official PETS 09 results.

Both detection and tracking precision metrics (MODP and MOTP) roughly gauge the quality of the bounding box

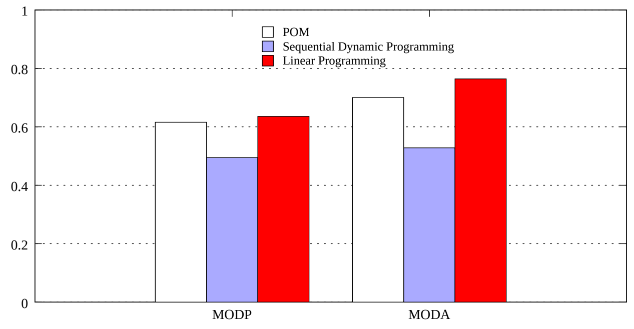


Figure 4: Detection precision and accuracy measures applied to the results of the original detection (POM) as well as the sequential Dynamic Programming and the proposed Linear Programming based trackers.

alignment, in the cases of correct detection. Since our method uses a ground plane grid with a finite precision of about 30cm, there is always some residual error in the alignment with the ground truth, which prevents us to score arbitrarily high. This shows up in the graph of Fig. 4, in which we see no significant difference between the original POM detections and LP. However, in both detection and tracking precision, LP achieves significantly higher scores than DP.

The detection accuracy metrics (MODA) evaluates the relative number of false positive and missed detections. Note that DP is lower than POM, because it tends to ignore trajectories for which some detections were missed. It thus produces more missed detections. By contrast, LP receives a higher score than POM and DP. By accurately linking detections together, while discarding isolated alarms, the LP efficiently filters the detections results, effectively decreasing both the false positives and missed detections counts.

Finally, the tracking accuracy measure (MOTA) is very similar to the detection one (MODA), with the exception that it also takes identity switches into account. Not surprisingly, LP scores again higher than DP.

Please recall that LP just used POM occupancy maps, whereas DP also looked at the original images and maintained a color model per tracked individual. In other words, LP produces better results, even though it requires less information. This is potentially valuable, because, in some situations, appearance models cannot be depended upon.

4.5 Monocular Results

To further emphasize the strength of the Linear Programming approach, we generated the detection maps using only one of the 7 available views. Although POM still works on monocular sequences, it is intrinsically less precise in the ground plane localization. Without several views from different angles, there is an inherent depth ambiguity when es-

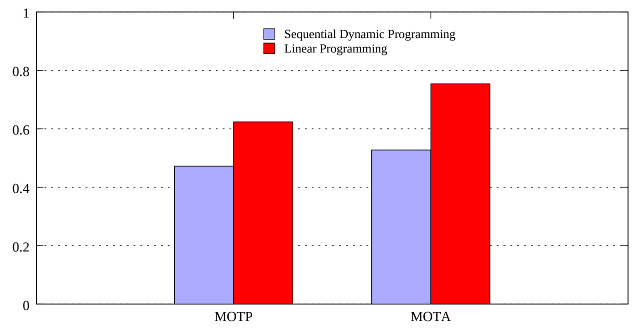


Figure 5: Tracking precision and accuracy measures applied to the results the sequential Dynamic Programming and the proposed Linear Programming based trackers.

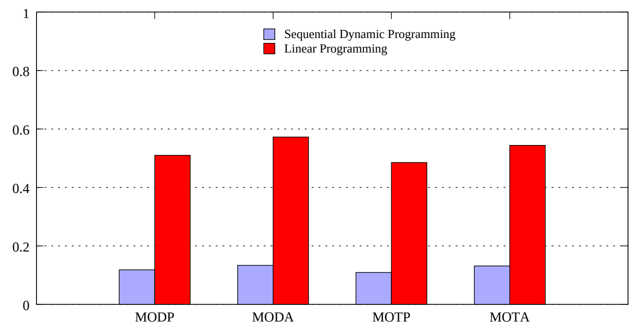


Figure 6: Performance of the DP and LP algorithms on detections generated from a monocular video.

timating a pedestrian's position, especially when the background subtraction blobs are noisy or incomplete. Also, in the monocular case, occlusions often result in missed detection.

In these challenging conditions, the Linear Programming method shows its superiority over the sequential Dynamic Programming one, even more clearly than in the multi-camera case. This is illustrated by Fig. 6. In this context, DP's greedy strategy often prefers leaving people outside the grid rather than trying to explain the very noisy detections. By contrast, LP's joint optimization pays off and interpolate trajectories nicely. Some monocular tracking results are shown in the last row of Fig. 7.

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 the difficult PETS 2009 data set, 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 specificity 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. Berclaz, A. Shahrokni, F. Fleuret, J. Ferryman, and P. Fua. Evaluation of probabilistic occupancy map people detection for surveillance systems. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 55–62, June 2009.

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

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

[5] W. Du and J. Piater. Multi-camera people tracking by collaborative particle filters and principal axis-based integration. In *ACCV*, pages 365–374, 2007.

[6] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multi-Camera People Tracking with a Probabilistic Occupancy Map. *TPAMI*, 30(2):267–282, February 2008.

[7] J. Giebel, D. Gavrila, and C. Schnorr. A bayesian framework for multi-cue 3d object tracking. In *ECCV*, 2004.

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

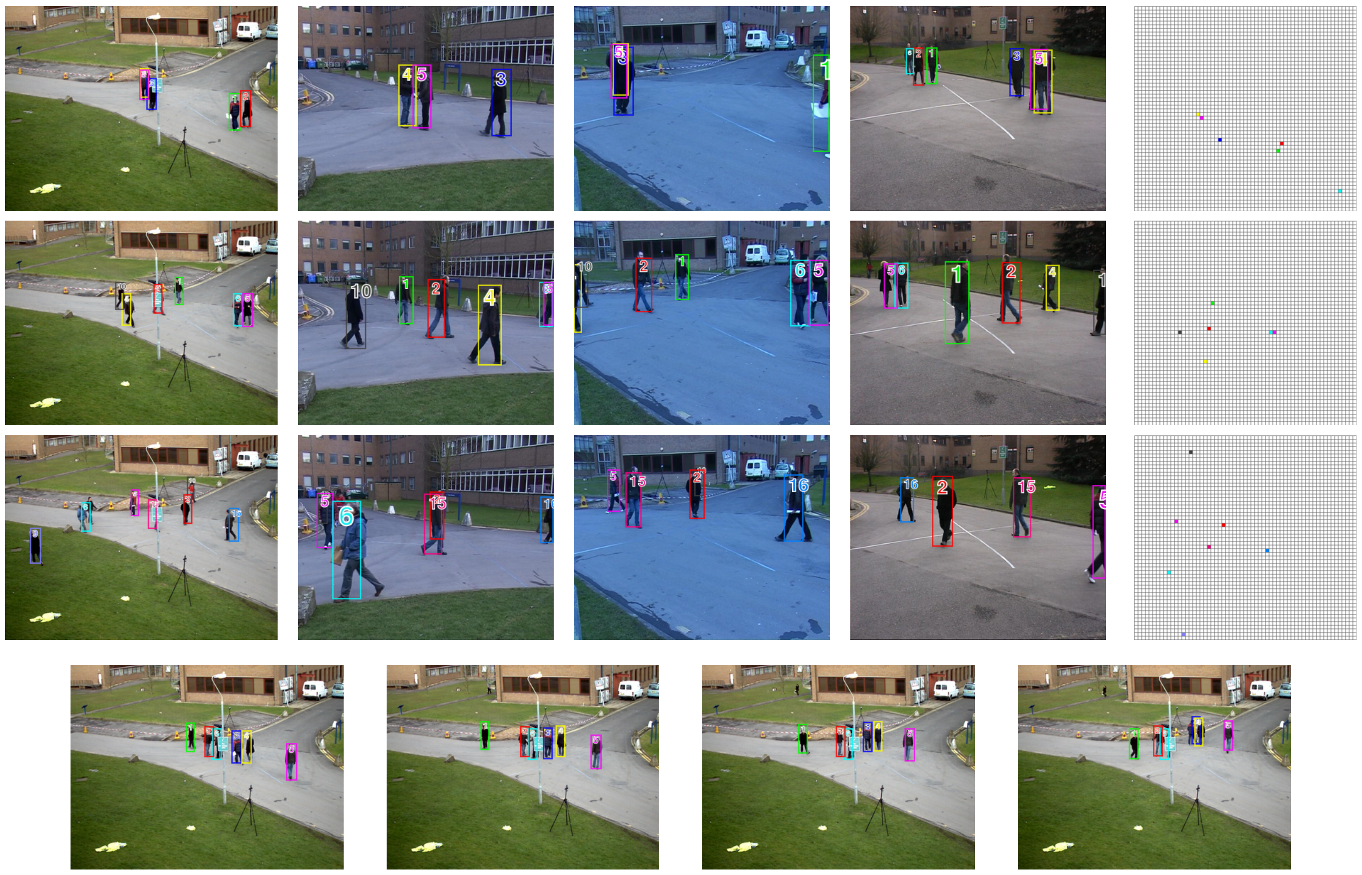


Figure 7: Some results of our LP-based tracker on a PETS 2009 sequence. Each of the three first rows shows a different time frame of the sequence. The first four columns display multi-camera tracking results in four of the five available camera views. The right-most column represents the corresponding detections on the ground plane. The last row shows four result frames in the monocular context. Since the people identities are color coded, this figure is best viewed in color.

[9] H. Jiang, S. Fels, and J. Little. A linear programming approach for multiple object tracking. In *CVPR*, pages 744–750, 2007.

[10] 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. *TPAMI*, 31(2), February 2009.

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

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

[13] P. Nillius, J. Sullivan, and S. Carlsson. Multi-target tracking - linking identities using bayesian network inference. In *CVPR*, volume 2, pages 2187–2194, 2006.

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

[15] K. Shafique and M. Shah. A noniterative greedy algorithm for multiframe point correspondence. *TPAMI*, 27(1):51–65, January 2005.

[16] K. Smith, D. Gatica-Perez, and J.-M. Odobez. Using particles to track varying numbers of interacting people. In *CVPR*, 2005.

[17] 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.

[18] J. Vermaak, A. Doucet, and P. Perez. Maintaining multimodality through mixture tracking. In *ICCV*, volume 2, pages 1110–1116, October 2003.

[19] J. Wolf, A. Viterbi, and G. 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.

[20] M. Xu, J. Orwell, and G. Jones. Tracking football players with multiple cameras. In *ICIP*, volume 5, pages 2909–2912, October 2004.