Tracking Interacting Objects Optimally Using Integer Programming

Xinchao Wang[1]*, Engin Türetken[1]*, François Fleuret[2,1], and Pascal Fua[1]**

[1] Computer Vision Laboratory, EPFL, Lausanne, Switzerland
[2] Computer Vision and Learning group, Idiap research institute, Martigny, Switzerland

Abstract. In this paper, we show that tracking different kinds of interacting objects can be formulated as a network-flow Mixed Integer Program. This is made possible by tracking all objects simultaneously and expressing the fact that one object can appear or disappear at locations where another is in terms of linear flow constraints. We demonstrate the power of our approach on scenes involving cars and pedestrians, bags being carried and dropped by people, and balls being passed from one player to the next in a basketball game. In particular, we show that by estimating jointly and globally the trajectories of different types of objects, the presence of the ones which were not initially detected based solely on image evidence can be inferred from the detections of the others.

1 Introduction

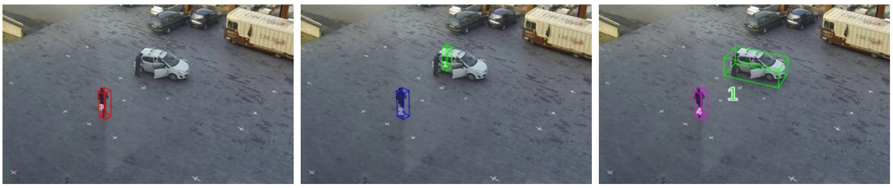
Tracking people or objects over time can be achieved by first running detectors that compute probabilities of presence in individual images and then linking high probabilities of detections into complete trajectories. This can be done recursively [6, 19], using dynamic programming [26, 11, 23], or using Linear Programming [25, 15, 5].

Most of these approaches focus on one kind of object, such as pedestrians or cars, and only model simple interactions, such as the fact that different instances may repel each other to avoid bumping into each other or synchronize their motions to move in groups [20, 28]. In this paper, we introduce a Mixed Integer Programming framework that lets us model the more complex relationship between the presence of objects of a certain kind and the appearance or disappearance of objects of another. For example, when tracking people and cars on a parking lot, this enables us to express that people may only appear or disappear either at the edge of the field of view or as they enter or exit cars that have stopped. Similarly, when attempting to check if a bag has been abandoned in a public place where we can track the people, we can express that this can only happen at locations through which somebody has been the instant before. The same goes for the ball during a basketball match; it is usually easiest to detect when it has left the hands of one player and before it has been caught by another.

We will show that enforcing the fact that one object can only appear or disappear at locations where another is or has been can be done by imposing linear flow constraints.

* Authors contributed equally.

** This work was funded in part by the SNSF DACH Project "Advanced Learning for Tracking and Detection in Medical Workflow Analysis".



(a) POM [11] (b) KSP [5] (c) Ours

Fig. 1. Motivation for our approach. (a) Thresholding the detector [11] scores for cars and people produces only one strong detection in this specific frame of a complete video sequence. (b) Linking people detections across frames [5] reveals the presence of an additional person. (c) This additional person constitutes evidence for the presence of a car he will get in. This allows our algorithm to find the car as well in spite of the car detection failure. Because we treat people and cars symetrically, the situation could have been reversed: The car could have been unambiguously detected and have served as evidence for the appearance of a person stepping out of it. This would not be the case if we tracked cars first and people potentially coming out of them next.

This results in a Mixed Integer Programming problem, for which the global optimum can be found using standard optimization packages [14]. Since different object types are handled in symmetric fashion, the presence of one can be evidence for the appearance of the other and vice-versa. For example, Fig. 1 depicts a case where simply thresholding the response of the car detector we use leads to a car being missed. However, because people are properly detected disappearing at a location in the middle of the parking lot, the algorithm eventually concludes correctly that there must have been a car there which they entered. So, in this scenario, not only does the presence of a vehicle "allow" the disappearance of pedestrians but the disappearance of pedestrians is treated as evidence for the presence of a vehicle.

This is much more general than what is done in approaches such as [28], in which the appearance of people is used to infer the possible presence of a static entrance. It also goes beyond recent work on interaction between people and objects [2]. Due to the global nature of the optimization and the generality of the constraints, we can deal with objects that may be completely hidden during large portions of the interaction and do not require any training data.

Our contribution is therefore a mathematically principled and computationally feasible approach to accounting for the relationship between flows representing the motions of different object types, especially with regard to their container/containee relationship and appearance/disappearance. We will demonstrate this in the case of people entering and leaving cars, bags being carried and dropped, and balls being passed from one player to the next in a ball-game.

2 Related Work

Multiple target tracking has a long tradition, going back many years for applications such as radar tracking [7]. These early approaches to data association usually relied on gating and Kalman filtering, which have later made their way into our community [6, 19]. Because of their recursive nature, they are prone to errors that are difficult to recover from by using a post processing step. Particle-based approaches such as [13, 24, 8], among many others, partially address this issue by simultaneously exploring multiple hypotheses. However, they can handle only relatively small batches of temporal frames without their state space becoming unmanageably large, and often require careful parameter setting to converge.

In recent years, techniques that optimize a global objective function over many frames have emerged as powerful alternatives. They rely on Conditional Random Fields [17, 27], belief Propagation [29, 9], Dynamic or Linear Programming [3, 10]. Among the latter, some operate on graphs whose nodes can either be all the spatial locations of potential people presence [26, 11, 5, 1], only those where a detector has fired [25, 15], or short temporal sequences of consecutive detections that are very likely to correspond to the same person [21, 30, 23, 4].

On average, these more global techniques are more robust than the earlier ones but, especially among those that focus on tracking people, do not handle complex interactions between them and other scene objects. In papers such as [18], which looks into the behavior of sports players, their trajectories are assumed to be given. In [20, 28], group behavior is considered during the tracking process by including priors that account for the fact that people tend to avoid hitting each other and sometimes walk in groups.

In [28], there is also a mechanism for guessing where entrances and exits may be by recording where tracklets start and end. However, this is very different from having objects that may move, thereby allowing objects of a different nature to appear or disappear at varying locations. In [2], person-to-person and person-to-object interactions are exploited to more reliably track all of them. This approach relies on a Bayesian Network model to enforce frame-to-frame temporal coherence, and on training data to learn object types and appearances. Furthermore, it requires the objects to be at least occasionally visible during the interaction. By contrast, we propose a global optimization framework that does not require training and can handle objects that remain invisible during extended periods of time, such as a person inside a car or a ball being carried and hidden by a player.

3 Method

In this section, we first formulate the problem of simultaneously tracking multiple instances of two kinds of objects, one of which can contain the other, as a constrained Bayesian inference problem. Here, we take "contain" to mean either fully enclosing the object, as the car does to its occupants, or simply being in possession of and partially hiding it, as a basketball player holding the ball. We then discuss these constraints in more details and show that they result in a Mixed Integer Program (MIP) on a large graph, which we solve by first pruning the graph and then using a standard optimizer.

3.1 Bayesian Inference

Given image sequences from one or more cameras with overlapping fields of view, we will refer to the set of images acquired simultaneously as a *temporal frame*. Let the number of time instants be T and the corresponding set of temporal frames $\mathbf{I} = (\mathbf{I}^1, \ldots, \mathbf{I}^T)$.

Assuming the position of target objects to be completely defined by their ground plane location, we discretize the area of interest into a grid of L square cells, which we will refer to as *spatial locations*. Within each one, we assume that a target object can be in any one of O *poses*. In this work, we define this pose space to be the set of regularly spaced object orientations on the ground plane.

For any pair k of location l and orientation o, let $\mathcal{N}(k) \subset \{1,\ldots,LO\}$ denote the neighborhood of k, that is, the locations and orientations an object located at l and oriented at o at time t can reach at time $t+1$. Let also $l(k)$ and $o(k)$ respectively denote the location and orientation of k.

Similar to [5], which treats spatial locations as graph vertices, we build a directed acyclic graph $G = (V, E)$ on the locations and orientations, where the vertices $V = \{v_k^t\}$ represent pairs of orientation angles and locations at each time instant, and the edges $E = \{e_{kj}^t\}$ represent allowable transitions between them. More specifically, an edge $e_{kj}^t \in E$ connects vertices v_k^t and v_j^{t+1} if and only if $j \in \mathcal{N}(k)$. The number of vertices and edges are therefore roughly equal to OLT and $|\mathcal{N}(.)|\,OLT$, respectively.

Recall that we are dealing with two kinds of objects, one of which can contain the other. Let $\mathbf{X} = \{X_k^t\}$ be the vector of binary random variables denoting whether location $l(k)$ is occupied at time t by a *containee* type object with orientation $o(k)$, and $\mathbf{x} = \{x_k^t\}$ a realization of it, indicating presence or absence of a *containee* object. Similarly, let $\mathbf{Y} = \{Y_k^t\}$ and $\mathbf{y} = \{y_k^t\}$ respectively be the random occupancy vector and its realization for the *container* object class.

As will be discussed in Section 4, we can estimate image-based probabilities $\rho_k^t = P(X_k^t = 1 \mid \mathbf{I}^t)$ and $\beta_k^t = P(Y_k^t = 1 \mid \mathbf{I}^t)$ that a containee or container object is present at grid location $l(k)$, with orientation $o(k)$, and at time t in such a way that their product over all k and t is a good estimate of the joint probability $P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y} \mid \mathbf{I})$. Among other things, this is done by accounting for objects potentially occluding each other.

Given the graph G and the probabilities ρ_k^t and β_k^t, we look for the optimal set of paths as the solution of

$$
\begin{aligned}
(\mathbf{x},\mathbf{y})^* &= \operatorname*{argmax}_{(\mathbf{x},\mathbf{y})\in\mathcal{F}} P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y} \mid \mathbf{I}) && (1)\\
&\approx \operatorname*{argmax}_{(\mathbf{x},\mathbf{y})\in\mathcal{F}} \prod_{t,k} P(X_k^t = x_k^t \mid \mathbf{I}^t) P(Y_k^t = y_k^t \mid \mathbf{I}^t) && (2)\\
&= \operatorname*{argmax}_{(\mathbf{x},\mathbf{y})\in\mathcal{F}} \sum_{t,k} \log P(X_k^t = x_k^t \mid \mathbf{I}^t) + \log P(Y_k^t = y_k^t \mid \mathbf{I}^t) \\
&= \operatorname*{argmax}_{(\mathbf{x},\mathbf{y})\in\mathcal{F}} \sum_{t,k} x_k^t \log \rho_k^t + (1 - x_k^t) \log(1 - \rho_k^t) \\
&\qquad\qquad + y_k^t \log \beta_k^t + (1 - y_k^t) \log(1 - \beta_k^t) && (3)\\
&= \operatorname*{argmax}_{(\mathbf{x},\mathbf{y})\in\mathcal{F}} \sum_{t,k} \log\left(\frac{\rho_k^t}{1-\rho_k^t}\right) x_k^t + \log\left(\frac{\beta_k^t}{1-\beta_k^t}\right) y_k^t && (4)
\end{aligned}
$$

where $\mathcal{F}$ stands for the set of all feasible solutions as defined in the following section. Eq. 2 comes from the above-mentioned property that the product of image-based probabilities is close to true posterior of Eq. 1, which will be discussed in more details in

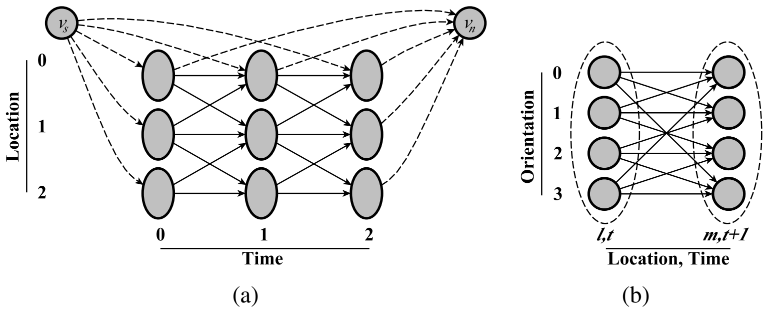


Fig. 2. A graph representing 3 spatial locations at 3 consecutive times. (a) Each ellipse corresponds to one spatial location at one time instant. Some are connected to a source and a sink node to allow entrances and exits. (b) Within each ellipse are four nodes, one for each possible orientation and the arrows represent possible transitions from one location and orientation to those in the neighboring ellipse.

§ 4, and from the assumption that all feasible transitions from time t to time $t+1$ are equally likely. Eq. 3 is true because both x_k^t and y_k^t are binary variables. Finally, Eq. 4 is obtained by dropping constant terms that do not depend on x_k^t or y_k^t. The resulting objective function is therefore a linear combination of these variables.

However, not all assignments of these variables give rise to a plausible tracking result. Therefore, the optimization of Eq. 4 must be performed subject to a set of constraints defined by $\mathcal{F}$, which we describe next.

3.2 Flow Constraints

To express all the constraints inherent to the tracking problem we introduce two additional sets of binary indicator variables that describe the flow of objects between pairs of discrete spatial locations and orientations at consecutive time instants. More specifically, we introduce the flow variables f_{kj}^t and g_{kj}^t, which stand respectively for the number of containee and container type objects moving from orientation $o(k)$ and location $l(k)$ at time t to orientation $o(j)$ and location $l(j)$ at time $t+1$.

In the following, in addition to the integrality constraints on the flow variables, we define six sets of constraints to obtain structurally plausible solutions.

Upper Bound on Flows: We set an upper-bound of one to the sum of all incoming flows to a given spatial location because it cannot be simultaneously occupied by multiple objects of the same kind.

$$\sum_{\substack{k:l=l(k),\\ i:k\in\mathcal{N}(i)}} f_{ik}^{t-1} \leq 1, \qquad \sum_{\substack{k:l=l(k),\\ i:k\in\mathcal{N}(i)}} g_{ik}^{t-1} \leq 1, \qquad \forall t,l\,. \tag{5}$$

Spatial Exclusion: As detailed in § 4.1, we model objects such as cars or people as rectangular cuboids, whose size is usually larger than that of a single grid cell. We impose spatial exclusion constraints to disallow solutions that contain overlapping cuboids in the 3D space. Let $\mathcal{N}_f(k)$ and $\mathcal{N}_g(k)$ denote the spatial exclusion neighborhoods for the containee and container objects respectively. We write

$$\sum_{i:k\in\mathcal{N}(i)} f_{ik}^{t-1} + \sum_{\substack{j\in\mathcal{N}_f(k),\\ i:j\in\mathcal{N}(i)}} f_{ij}^{t-1} \leq 1, \qquad \sum_{i:k\in\mathcal{N}(i)} g_{ik}^{t-1} + \sum_{\substack{j\in\mathcal{N}_g(k),\\ i:j\in\mathcal{N}(i)}} g_{ij}^{t-1} \leq 1, \qquad \forall t,k\,. \tag{6}$$

Flow Conservation: We require the sum of the flows incoming to a graph vertex v_k^t to be equal to the sum of the outgoing flows for each container object type.

$$y_k^t = \sum_{i:k\in\mathcal{N}(i)} g_{ik}^{t-1} = \sum_{j\in\mathcal{N}(k)} g_{kj}^t, \;\; \forall t,k\,. \tag{7}$$

This ensures that the container objects cannot appear or disappear at locations other than the ones that are explicitly designated as entrances or exits. Graph vertices associated to these entrance and exit points serve respectively as a source and a sink for the flows. To allow this, we introduce two additional vertices v_s and v_n into our graph G, which are linked to all the vertices representing positions through which objects can respectively enter or leave the observed area. Furthermore, we add directed edges from v_s to all the vertices of the first time instant and from all the vertices of the last time instant to v_n, as illustrated by Fig. 2.

To ensure that the total container flow is conserved in the system, we enforce the amount of flow generated at the source v_s to be equal to the amount consumed at the sink v_n.

$$\sum_{j\in\mathcal{N}(s)} g_{sj} = \sum_{i:n\in\mathcal{N}(i)} g_{in} \quad . \tag{8}$$

Consistency of Interacting Flows: We allow a containee type object to appear or disappear at a location not designated as entrance or exit only when it comes into contact with or is separated from a container object. We write

$$- \sum_{\substack{m:l(k)=l(m),\\ i:m\in\mathcal{N}(i)}} g_{im}^{t-1} \leq a(t,k) \leq \sum_{\substack{m:l(k)=l(m),\\ j\in\mathcal{N}(m)}} g_{mj}^t, \;\; \forall t,k \tag{9}$$

$$a(t,k) = \sum_{i:k\in\mathcal{N}(i)} f_{ik}^{t-1} - \sum_{j\in\mathcal{N}(k)} f_{kj}^t \tag{10}$$

In Eq. 9, the total amount of container flow passing through the location k is denoted by the two sums on both sides of the inequality. When they are zero, these constraints impose the conservation of flow for the containee objects at location k. When they are equal to one, a containee object can appear or disappear at k. Note that, here we assume the containee objects never come to interact with the container one at exactly the same moment. For example, at one time instance only one person is allowed to enter the car.

Note that all four sums in Eqs. 9 and 10 can be equal to one. As a result, these constraints allow for a container and a containee object to coexist at the same location and at the same time instant, which can give rise to several undesirable results as shown

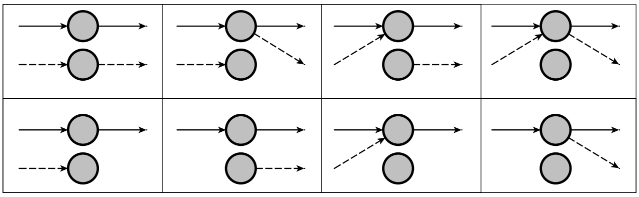


Fig. 3. Flow constraints in a two-orientation case. In each of the eight examples shown here, the two circles represent two orientation nodes at the same spatial location. The solid and the dotted arrows represent respectively non-zero flows g^t_{kj} and f^t_{kj} of the container and of the visible containee objects. **Top Row.** Forbidden configurations, which are all cases where a containee and a container coexist at the same location and at the same time instant without interacting with each other. For example, the configuration on the left could be interpreted as someone jumping in and out of the car at the same time. **Bottom Row**: Feasible configurations.

in the top row of Fig. 3. To avoid this, we bound the total amount of containee flow incoming to and outgoing from a location by one when there is a container object at that location.

$$\sum_{\substack{k:l=l(k),\\ i:k\in\mathcal{N}(i)}} f^{t-1}_{ik} + \sum_{\substack{k:l=l(k)\\ j\in\mathcal{N}(k)}} f^t_{kj} \leq 2 - \sum_{\substack{k:l=l(k)\\ j\in\mathcal{N}(k)}} g^t_{kj}, \ \forall t, l \tag{11}$$

Tracking the Invisible: We say a containee object is *invisible* when it is carried by a container. The four sets of constraints described above do not allow us to keep track of the number of invisible instances carried by a container object at a time. To facilitate their tracking even when they are invisible, we introduce additional flow variables h^t_{kj}, which stand for the number of invisible containees moving from orientation $o(k)$ and location $l(k)$ at time t to orientation $o(j)$ and location $l(j)$ at time $t+1$. These variables act as counters that are incremented or decremented when a containee object respectively disappears or appears in the vicinity of a container.

$$\sum_{\substack{k:l=l(k)\\ j\in\mathcal{N}(k)}} h^t_{kj} = \sum_{\substack{k:l=l(k),\\ i:k\in\mathcal{N}(i)}} h^{t-1}_{ik} + \sum_{\substack{k:l=l(k),\\ i:k\in\mathcal{N}(i)}} f^{t-1}_{ik} - \sum_{\substack{k:l=l(k)\\ j\in\mathcal{N}(k)}} f^t_{kj}, \ \forall t, l \tag{12}$$

$$h^t_{kj} \leq c * g^t_{kj}, \ \forall t, k, j : j \in \mathcal{N}(k) \tag{13}$$

where c is a fixed integer constant standing for the maximum number of containee instances a container can hold. For example, in the case of cars and people, this constant is set to 5. As a result, unlike the flow variables f^t_{kj} and g^t_{kj} that are binary, h^t_{kj} usually have a higher but finite upper bound. Note that, in Eq. 12, the h^t_{kj} variables are incremented or decremented always by an integer value. Therefore, during the optimization, we allow these variables to be continuous, except only those that are connected to the source, i.e., h_{sj}, which we restrict to be integers. Our experimental results show that allowing $h^t_{kj:k\neq s}$ to be continuous slightly speeds up the optimization, compared to imposing the integrality constraints on these variables.

Additional Bound Constraints: Finally, we impose additional upper or lower bound constraints on the flow variables when the maximum or minimum number of object instances of a certain type in the scene is known *a priori*. For instance, during a basketball game, the number of balls in the court is bounded by one. We write this as

$$\sum_{\substack{v_k^t \in V(t),\\ j \in \mathcal{N}(k)}} h_{kj}^t + \sum_{\substack{v_k^t \in V(t),\\ j \in \mathcal{N}(k)}} f_{kj}^t \leq 1\,, \qquad \forall t \tag{14}$$

where $V(t)$ denotes the set of graph vertices of time instant t. Together with the invisible flow constraints expressed in Eqs. 12 and 13, these constraints allow us to keep track of where the ball is and who has possession of it even when it is invisible. Another interesting case arises from the fact that a moving vehicle must have a driver inside. We express this as

$$h_{kj}^t \geq g_{kj}^t, \qquad \forall t, k, j : j \in \mathcal{N}(k), l(k) \neq l(j) \tag{15}$$

3.3 Mixed Integer Programming

The formulation defined above translates naturally into a Mixed Integer Program (MIP) with variables f_{kj}^t, g_{kj}^t, h_{kj}^t and a linear objective

$$\sum_{\substack{t \in \{1,\cdots,T\},\\ v_k^t \in V(t)}} \sum_{j \in \mathcal{N}(k)} \left(\alpha_k^t \, f_{kj}^t + \gamma_k^t \, g_{kj}^t\right), \tag{16}$$

with

$$\alpha_k^t = -\log\left(\frac{\rho_k^t}{1-\rho_k^t}\right), \text{ and } \gamma_k^t = -\log\left(\frac{\beta_k^t}{1-\beta_k^t}\right). \tag{17}$$

This objective is to be minimized subject to the constraints introduced in the previous section. Since there is a deterministic relationship between the occupancy variables (x_k^t, y_k^t) and the flow variables (f_{kj}^t, g_{kj}^t), this is equivalent to maximizing the expression of Eq. 4.

Solving the Linear Program (LP) obtained by relaxing the integrality constraints may, in some cases, result in fractional flow values as will be shown in the results section. That is why, we explicitly enforce the integrality constraints in our final results.

3.4 Graph Size Reduction

In most practical situations, the MIP of Eq. 16 has too many variables to be handled by ordinary solvers. To reduce the computational time, we eliminate spatial locations whose probability of occupancy is low.

A naive way to do this would be to simply eliminate grid locations $l(k)$ whose purely image-based probabilities ρ_k^t and β_k^t are below a threshold. However, this would be self-defeating because it would preclude the algorithm from doing what it is designed to do, such as inferring that a car that was missed by the car detector must nevertheless be present because people are seen to be coming out of it.

Instead, we implemented the following two-step algorithm. First, we designate all grid locations as potential entries and exits, and run the K-Shortest Paths Algorithm (KSP) [5] for containers and containees independently. In our experiments, we used the publicly available KSP code, which is shown to be very efficient. This produces a set of container and containee tracklets that can start and end anywhere and anytime on the grid. Second, we connect all these tracklets both to each other and to the original entrance and exit locations using the Viterbi algorithm [12]. Finally, we obtain a subgraph of G, whose nodes belong either to the tracklets or the paths connecting them.

In this way, the resulting subgraph still contains the low ρ_k^t and β_k^t locations that may correspond to missed detections while being considerably smaller than the original grid graph. For example, on a 20-frame *PETS2006* [22] image sequence that will be introduced in the results section, this procedure reduces the number of edges from around 22M to 17K. The resulting graphs are small enough to solve the MIP of Eq. 16 on batches of 500 to 1000 frames using the branch-and-cut procedure implemented in the Gurobi optimization library [14]. It minimizes the gap between a lower bound obtained from LP relaxations and an upper bound obtained from feasible integer solutions. The algorithm stops when the gap drops below the specified tolerance value. In practice, we set it to $1e^{-4}$ indicating the solution it finds is very close to the global optimum.

4 Estimating Probabilities of Occupancy

Our approach to computing the image-based probabilities of presence ρ_k^t and β_k^t that appear in Eq. 3 and Eq. 4 is an extension of the one proposed in [11].

This earlier algorithm was designed to estimate such probabilities for pedestrians given the output of background subtraction on a set of images taken at the same time. Its basic ingredient is a generative model that represents humans as cylinders that it projects into the images to create synthetic ideal images we would observe if people were at given locations. Under this model of the image given the true occupancy, the probabilities of occupancy at every location are taken to be the marginals of a product law minimizing the Kullback-Leibler divergence from the "true" conditional posterior distribution. This makes it possible to evaluate the probabilities of occupancy at every location as the fixed point of a large system of equations.

Importantly, probabilities computed in this way exhibit the property that allows us to go from Eq. 1 to Eq. 2 in our derivation of the objective function. We have therefore extended the approach to handling multiple classes of objects simultaneously as follows.

4.1 Oriented Objects

To handle objects such as cars or bags, we extend [11] by introducing simple wireframe models to represent them, as shown in Fig. 4. The only difficulty is that in the case of cylinders, orientation is irrelevant whereas the projections of our wireframe models depend on it. We solve this by allowing the generative model to model objects of any type at any one of the O regularly spaced orientations. This means that the projections of our 3D models can have arbitrary shapes and that we cannot use the integral image

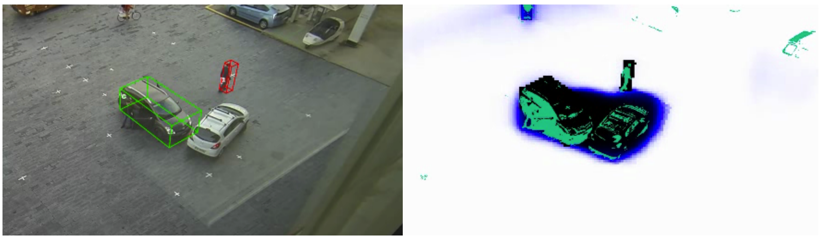


Fig. 4. Simultaneously detecting people and cars. (a) A person and a car is detected, as indicated by the red and green wireframes. (b) The same boxes are projected and filled as black boxes to create a synthetic image that approximates as closely as possible the background subtraction results, shown in green. Note that the white car is the same as the one that appears in Fig. 1. It remains undetected because the background subtraction algorithm fails to extract it.

trick of the publicly available software anymore [11]. We therefore use an "integral line" variant, which is comparably efficient. More specifically, we compute an integral image by taking integral of the image values only along the horizontal axis. At detection time, we then take the difference between the left-most and right-most integral pixels of a projected region and sum the resulting differences obtained from each row. This lets us detect objects of different types simultaneously and compute the probabilities of occupancy ρ_k^t and β_k^t introduced in § 3.1.

Note however, that the white car in Fig. 4 is missed because its color is similar to that of the background used for training, which is taken under direct sunlight. Arguably, we could have used a more powerful car detector but all detectors sometime fail and the point of this paper is that our technique can recover from such failures by leveraging information provided by other objects, in this case the people getting out of the car.

4.2 Objects off the Ground Plane

In [11], objects of interest are assumed to be on the ground and the fact that they can move in the vertical direction, such as when people jump, is ignored. For people, this is usually not an issue because the distance of their feet to the ground tends to be small compared to their total height and the generative model remains roughly correct. However, in the case of an object such as a ball, which is small and can be thrown high into the air, this is not true anymore.

In theory, this could be handled by treating height over ground as a state variable, much as we do for orientation. However, in the specific case of the basketball competition we show in the result section that when the ball is in the air it also often is in front of the spectators, making the background non-constant and the results of [11] unsatisfactory. Therefore, in this specific case, we use a discriminative approach and run a ball detector based on color and roundness in each one of the frames taken at the same time, triangulate the 2D detections to obtain candidate 3D detections, and project the resulting probability estimate on the ground plane. Due to the small size of the ball compared to that of people, its presence or absence in a frame has little effect on the estimated probabilities of presence of people and we can assume conditional independence of

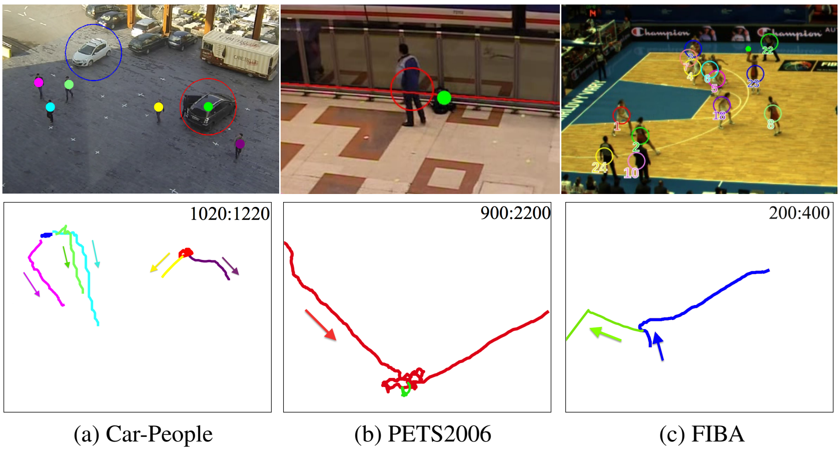


Fig. 5. Tracking results on three representative subsequences taken from our datasets. **Top row.** Sample frames with the detected container objects highlighted with circles and containee ones with dots. **Bottom Row.** Corresponding color-coded top-view trajectories for interacting objects in the scene. The arrows indicate the traversal direction. Note that, in the FIBA case, even though there are many players in the field, we plot only two trajectories: one for the ball the other for the player first holding it and then throwing it.

presence of people and ball given the images, which means we can still multiply the probabilities as required for the derivation of Eq. 2.

5 Experiments

In this section, we briefly describe the sequences we used for validation and give implementation details of our approach. We then introduce several baseline methods and finally present our comparative results. We show that our approach outperforms state-of-the-art methods on complex scenes with multiple interacting objects.

5.1 Test Sequences

We tested our approach on three datasets featuring three very different scenarios: people and vehicles on a parking lot (Car-People dataset), people and luggage in a railway station (PETS2001 dataset), and basketball players and the ball during a high-level competition (FIBA dataset). These datasets are multi-view and we processed a total of about 15K temporal frames. They all involve multiple people and objects interacting with each other. In Fig. 5, we show one image from each dataset with recovered trajectories. We summary the datasets as follows.

- **Car-People Dataset:** We captured several 300- to 5000-frame sequences from 2 cameras that feature many instances of people getting in and out of the cars. We show experimental evaluation on two sequences in the manuscript and provide further results in the supplementary material.

- **PETS2006 Dataset:** We use a 3020-frame sequence acquired by 2 cameras that shows people entering and leaving a railway station while carrying bags. Notably, one person brings a backpack into the scene, puts it on the ground, and leaves.
- **FIBA Dataset:** We use a 2600-frame sequence captured by 6 cameras at the 2010 FIBA Women World Championship. It features two 5-player-teams, 3 referees and 2 coaches. This sequence is challenging due to the complex and frequent interactions between the players and the ball, which makes it hard to detect the ball.

5.2 Parameters and Baselines

To compute the probabilities of occupancy ρ_k^t and β_k^t of Section 4, we used 12 regularly distributed orientations for cars and 2 for luggages, which we found to be sufficient given the quality of the videos. For the outdoor scenes and the basketball court, we discretized the ground plane into 25cm$\times$25cm cells. For the railway station, the area of interest is relatively small, which allowed us to perform a finer sampling with a cell size of 10cm$\times$10cm to improve the localization accuracy.

We compared our approach, denoted as **OURS-MIP**, against six baseline methods, which we summarize below.

- **POM:** We keep those orientation nodes, for which one of the occupancy probabilities ρ_k^t or β_k^t is greater than 0.5, and suppress the others. The resulting detections lack temporal consistency and may not satisfy the constraints introduced in § 3.2.
- **SSP:** The Successive Shortest Path (SSP) [23] is a algorithm for tracking multiple objects. It first builds a graph by linking pairs of object detections in consecutive temporal frames and then applies Dynamic Programing to find solutions. We run the publicly available SSP code and compared the results with ours.
- **KSP-free:** As discussed in Section 3.4, the KSP approach of [5] can be used to compute object trajectories for the container and containee objects independently using their occupancy probabilities. We designate all the grid locations as potential entries and exits prior to running the KSP algorithm. As a result, this approach allows objects to appear or disappear at any location at a certain cost value, which we take to be 40.
- **KSP-fixed:** This algorithm is similar to KSP-free, except that we use the original entrances and exits of the scene, such as the edge of the field of view. Therefore, objects can only appear or disappear at these predetermined locations.
- **KSP-sequential:** We first use the KSP-fixed algorithm to track the container objects and designate all the nodes that belong to the resulting trajectories as potential entrances and exits for the containees. We then use the same algorithm to find the containee trajectories, which may emerge from or enter the container ones. In other words, unlike in our approach, the two object classes are *not* treated symmetrically.
- **OURS-LP:** The linear programming approach (LP) solves the problem introduced in § 3.3 with the integrality constraints relaxed. The resulting flow variables are then rounded to the nearest integer to obtain the final solution.

5.3 Results

We ran all the baseline algorithms and ours on all the test sequences introduced in § 5.1. We show some qualitative results in Fig. 5. In the following, we present quantitative results on a representative subset of the sequences. We provide additional ones as well as videos overlaid with detection results in the supplementary material.

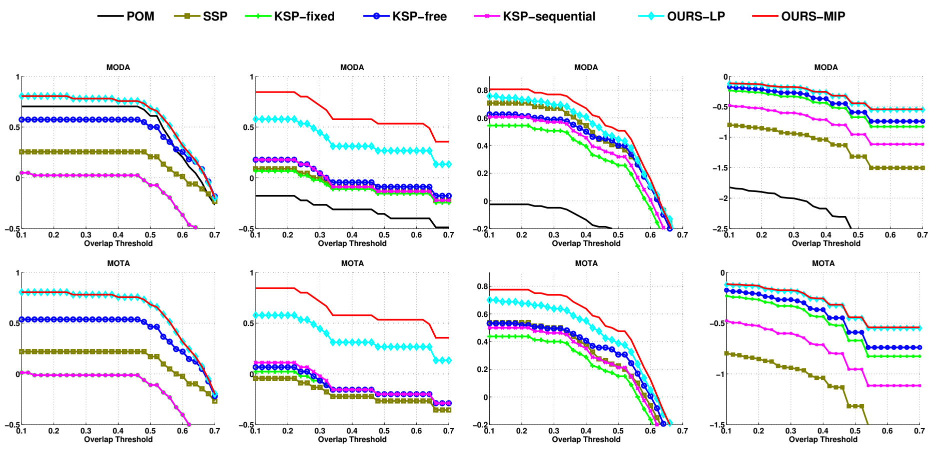
To quantify these results, we use the standard CLEAR [16] metrics, Multiple Object Detection Accuracy (MODA) and Multiple Object Tracking Accuracy (MOTA). MODA focuses on missed and false detections, while MOTA also accounts for identity switches. They are defined as a function of the amount of overlap between the detections and the ground-truth.

In Fig. 6, we plot MOTA and MODA for our approach (OURS-MIP) against those of our baselines on two sequences in the Car-People dataset, the PETS06 dataset, and the FIBA dataset. For the results of the remaining sequences in the Car-People dataset, we refer the reader to the supplementary material.

The sequence Car-People Seq.0 is the one from which we extracted the image shown in Fig. 1 and the corresponding results are shown in the first column of Fig. 6. It involves three people getting into a car stopped at the center of a parking lot. As discussed in § 4.1, the POM detector often fails to detect the car due to poor background subtraction. As a result, both KSP-fixed and KSP-sequential yield poor results because they do not create a car track, and hence are forced to explain the people in the scene by hallucinating them entering from the edges of the field of view. SSP and KSP-free do better by allowing the car to appear and disappear as needed but this does not correspond to physically plausible behavior. POM does even better because the people are in fact detected most of the time. OURS-MIP approach performs best because the evidence provided by the presence of the people along with the constraint that they can only appear or disappear in the middle of the scene, where there is a stopped car, forces the algorithm to infer that there is one at the right place.

The Car-People Seq.1 features two people getting into the first car, staying for a while, and getting out and entering the second one. Here, KSP-sequential and KSP-free do slightly better than KSP-fixed, which needs to hallucinate two false positive tracks to allow for the people emerging from the first car. The same happens in the PETS2006 sequence when the bag suddenly becomes visible in the middle of the image. Again, our approach performs best on both sequences mainly because we do not allow solutions that contain overlapping detections in the 3D space, which is enforced by the spatial exclusion constraints of § 3.2. In contrast, all the baseline methods produce overlapping spurious detections that are not physically plausible.

For the FIBA sequence, we show in Fig. 6(d) the MODA and MOTA scores for the ball only because the people detection scores for both the baselines and our approach are all very similar and the differences would not be visible in print. KSP-sequential yields a poor performance because of the weak image evidence that gives rise to several spurious ball detections. KSP-fixed eliminates some of these detections by forcing the ball to enter the scene only from the designated locations, and KSP-free does so by requiring that a cost to be paid for every appearance or disappearance of the ball. Our approach achieves the best performance by reasoning simultaneously for both players and ball, and enforcing that there can be at most one ball in the field during the game.



(a) Car-People Seq.0 (b) Car-People Seq.1 (c) PETS2006 (d) FIBA

Fig. 6. Comparing our proposed approach (OURS-MIP) against the baselines in terms of the MOTA and MODA scores. Our tracker yields a significant improvement on all datasets, thanks to the joint-global optimization on both container and containee objects.

Note that solving the LP problem of § 3.3 and subsequently rounding the resulting fractional flow variables as in OURS-LP systematically performs either very similarly or worse than explicitly imposing the integrality constraints as we do in the OURS-MIP approach. In Car-People Seq.1 and PETS2006, where OURS-MIP significantly outperforms OURS-LP, the ratio of fractional flows to non-zero flows using OURS-LP are 39% and 12% respectively. In the other two sequences the ratio are lower than 1%, therefore the performance of OURS-LP and OURS-MIP are very similar.

Finally, in the Car-People dataset, we observe a few failure cases where a person gets into the car but the associated counter variable is not incremented. This is because the car is parked on the boundary of the monitored area and the person is detected closer to the boundary than to the car, therefore the optimizer prefers the explanation that the person leaves the monitored area than he enters the car.

6 Conclusion

We have introduced a new approach to tracking multiple objects of different types and accounting for their complex and dynamic interactions. It relies on Mixed Integer Programming and ensures convergence to a global optimum using a standard optimizer. Furthermore, not only does it explicitly handle interactions, it also provides an estimate for the *implicit* transport of objects for which the only evidence is the presence of other objects that can contain or carry them.

We demonstrated our method on real-world sequences that feature people boarding and getting out of cars, carrying and dropping luggages, and passing the ball during a basketball match. The same approach could be applied to more complex situations and future work will aim at extending it to scenarios with more than two types of objects.

References

1. Andriyenko, A., Schindler, K., Roth, S.: Discrete-Continuous Optimization for Multi-Target Tracking. In: CVPR (2012)
2. Baumgartner, T., Mitzel, D., Leibe, B.: Tracking People and Their Objects. In: CVPR. pp. 3658–3665 (2013)
3. Bellman, R.: Dynamic programming and lagrange multipliers. Proceedings of the National Academy of Sciences 42, 767–769 (1956)
4. BenShitrit, H., Berclaz, J., Fleuret, F., Fua, P.: Multi-Commodity Network Flow for Tracking Multiple People. PAMI (2013)
5. Berclaz, J., Fleuret, F., Türetken, E., Fua, P.: Multiple Object Tracking Using K-Shortest Paths Optimization. PAMI 33, 1806–1819 (September 2011)
6. Black, J., Ellis, T., Rosin, P.: Multi-View Image Surveillance and Tracking. In: IEEE Workshop on Motion and Video Computing (2002)
7. Blackman, S.: Multiple-Target Tracking with Radar Applications. Artech House (1986)
8. Breitenstein, M., Reichlin, F., Leibe, B., Koller-Meier, E., Van Gool, L.: Online Multi-Person Tracking-By-Detection from a Single Uncalibrated Camera. PAMI (2010)
9. Choi, W., Savarese, S.: A Unified Framework for Multi-Target Tracking and Collective Activity Recognition. In: ECCV (2012)
10. Dantzig, G.B.: Linear Programming and Extensions. Princeton University Press (1963)
11. Fleuret, F., Berclaz, J., Lengagne, R., Fua, P.: Multi-Camera People Tracking with a Probabilistic Occupancy Map. PAMI 30(2), 267–282 (February 2008)
12. Forney, G.: The Viterbi Algorithm. In: Proceedings of IEEE. pp. 268–278 (March 1973)
13. Giebel, J., Gavrila, D., Schnorr, C.: A Bayesian Framework for Multi-Cue 3D Object Tracking. In: ECCV (2004)
14. Gurobi: Gurobi Optimizer (2012), http://www.gurobi.com/
15. Jiang, H., Fels, S., Little, J.: A Linear Programming Approach for Multiple Object Tracking. In: CVPR (2007)
16. Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Boonstra, M., Korzhova, V., Zhang, J.: Framework for Performance Evaluation of Face, Text, and Vehicle Detection and Tracking in Video: Data, Metrics, and Protocol. PAMI 31(2), 319–336 (February 2009)
17. Lafferty, J., Mccallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: ICML (2001)
18. Lucey, P., Bialkowski, A., Carr, P., Morgan, S., Matthews, I., Sheikh, Y.: Representing and Discovering Adversarial Team Behaviors Using Player Roles. In: CVPR (2013)
19. Mittal, A., Davis, L.: M2Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene. IJCV 51(3), 189–203 (2003)
20. Pellegrini, S., Ess, A., Schindler, K., Van Gool, L.: You'll Never Walk Alone: Modeling Social Behavior for Multi-Target Tracking. In: ICCV (2009)
21. Perera, A., Srinivas, C., Hoogs, A., Brooksby, G., Wensheng, H.: Multi-Object Tracking through Simultaneous Long Occlusions and Split-Merge Conditions. In: CVPR (2006)
22. PETS: Performance Evaluation of Tracking and Surveillance (2009), http://www.cvg.rdg.ac.uk/slides/pets.html
23. Pirsiavash, H., Ramanan, D., Fowlkes, C.: Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects. In: CVPR (June 2011)
24. Smith, K., Gatica-Perez, D., Odobez, J.M.: Using Particles to Track Varying Numbers of Interacting People. In: CVPR (2005)
25. Storms, P., Spieksma, F.: An Lp-Based Algorithm for the Data Association Problem in Multitarget Tracking. Computers and Operations Research 30(7), 1067–1085 (June 2003)

26. Wolf, J., Viterbi, A., Dixon, G.: Finding the Best Set of K Paths through a Trellis with Application to Multitarget Tracking. IEEE Transactions on Aerospace and Electronic Systems 25(2), 287–296 (March 1989)
27. Yang, B., Nevatia, R.: An Online Learned CRF Model for Multi-Target Tracking. In: CVPR (2012)
28. Yang, B., Nevatia, R.: Multi-Target Tracking by Online Learning of Non-Linear Motion Patterns and Robust Appearance Models. In: CVPR (2012)
29. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Generalized Belief Propagation. In: NIPS. pp. 689–695 (2000)
30. Zhang, L., Li, Y., Nevatia, R.: Global Data Association for Multi-Object Tracking Using Network Flows. In: CVPR (2008)