Tracking Interacting Objects Using Intertwined Flows

Xinchao Wang* , Engin Türetken*, François Fleuret, and Pascal Fua, *Fellow, IEEE*

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 using intertwined flow variables and expressing the fact that one object can appear or disappear at locations where another is in terms of linear flow constraints. Our proposed method is able to track invisible objects whose only evidence is the presence of other objects that contain them. Furthermore, our tracklet-based implementation yields real-time tracking performance. 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 team sports. 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.

Index Terms—Multi-object tracking, interactions, network flows, mixed integer programming

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 probability detections into complete trajectories. This can be done recursively [1], using dynamic programming [2], [3], or using Linear Programming [4], [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 [6], [7]. In this paper, we introduce a network-flow 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 kind. 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. Similarly, when attempting to check if a bag has been abandoned in a public place, 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 or soccer match; it is usually easiest to detect the ball when it has left one player and before it has been caught by another.

We will show that tracking interacting objects simultaneously can be achieved by modeling their motions with intertwined flow variables, and by imposing linear flow constraints to enforce the fact that one object can only appear or disappear at locations where another is or has been. This results in a Mixed Integer Programming problem. In theory, it is NP-hard. In practice however, a very close approximation to the global optimum can be found using a standard optimization package [8]. Since different object types are handled simultaneously, the presence of any one of them can be evidence for the appearance of any other. Fig. 1(a) 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, our algorithm eventually concludes correctly that there must have been a car there which they entered, as shown in Fig. 1(c). In this scenario, not only does the presence of a vehicle explain the apparent disappearance of pedestrians but also their disappearance is evidence of the presence of a vehicle.

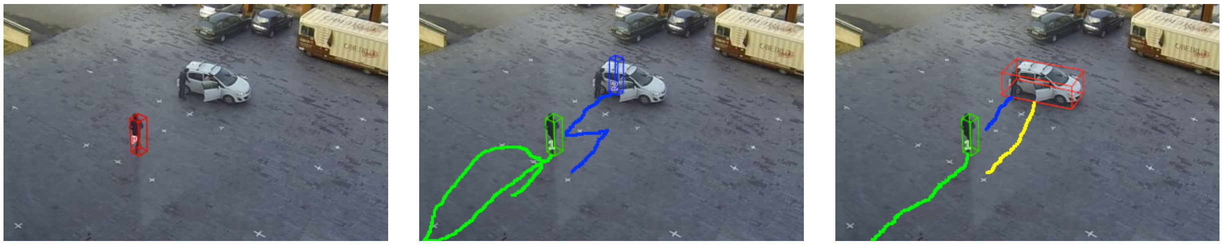
This is much more general than what is done in approaches such as [7], in which the appearance of people is used to infer the possible presence of a static entrance. It also goes beyond recent work on interaction between people and objects [9]. Due to the global nature of the optimization and the generality 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. We first introduced this approach in a conference paper [10] and described a relatively slow implementation. In this paper, we introduce a faster tracklet-based one that preserves optimality while yielding real-time performance.

Our contribution is therefore a mathematically principled 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, as well as an efficient tracklet-based

* *The authors contributed equally.*

- *X. Wang and P. Fua are with the Computer Vision Laboratory, IC Faculty, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne CH-1015, Switzerland. E-mail: {xinchao.wang, pascal.fua}@epfl.ch*
- *E. Türetken is with Swiss Center for Electronic and Microtechnology, Neuchâtel CH-2002, Switzerland. E-mail:engin.turetken@epfl.ch*
- *F. Fleuret is with the Computer Vision and Learning Group, Idiap research institute, Martigny CH-1920, Switzerland, and with École Polytechnique Fédérale de Lausanne (EPFL), Lausanne CH-1015, Switzerland. E-mail: francois.fleuret@idiap.ch*

- *This project was supported in part by the Swiss National Science Foundation.*



(a) POM [2] (b) KSP [5] (c) Ours

Fig. 1. Motivation for our approach. (a) Thresholding the Probability Occupancy Map (POM) detector [2] 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 using the K-Shortest Paths (KSP) algorithm [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 symmetrically, 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.

implementation that yields real-time performance. 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 during a game.

2 Related Work

In this section, we first discuss approaches to multiple object tracking and then review those that focus on tracking interacting objects.

2.1 Tracking Multiple Objects

Existing multiple object tracking approaches can be broadly divided into two categories: tracking by model evolution and tracking by detection. In this section, we briefly review the state-of-the-art representatives from both categories.

2.1.1 Tracking by Model Evolution

Early approaches of this category focused on tracking a single object and relied on gating and Kalman filtering [1]. Because of their recursive nature, they are prone to errors such as drift, which are difficult to recover from. Particle-based approaches such as [11], [12], [13], [14], among many others, partially address this issue by exploring multiple hypotheses. However, they can handle only relatively small batches of frames without their state space becoming unmanageably large and often require careful parameter setting.

Many approaches in this category aim at improving tracking accuracy by updating a classifier frame by frame. It can be a support vector machine [15], boosting [16], [17], naive Bayes [18] or an ensemble classifier [19]. The approach of [20] updates its classifier based on a combination of matting and tracking while that of [21] ignores the temporal order of the frames and selects easy-to-track ones first. These techniques have proved effective for single object tracking. However, when dealing with multiple objects that interact with each other, such as basketball players competing for possession of the ball, our experience is that they are prone to drift and identity switches, and thus not suitable for our purpose.

2.1.2 Tracking by Detection

In recent years, techniques that optimize a global objective function over many frames have emerged as powerful alternatives. They rely on Conditional Random Fields [22], [23], [24], Belief Propagation [25], [26], Dynamic or Linear Programming [27], [28], or Network Flow Programming [29], [30]. Among these algorithms, some operate on graphs whose nodes can either be all the spatial locations where an object can be present [2], [5], [31], only those where a detector has fired [4], [32], [33], [34], mid-level features of objects [35], [36], [37], [38], [39], [40], or short temporal sequences of consecutive detections that may correspond to the same target [41], [42], [43], [44], [45], [46], [47].

The K-Shortest Paths (KSP) algorithm [5] works on the graph of all potential locations over all time instants, and finds the ground-plane trajectories that yield the overall minimum cost. However, it assumes that the nodes in the graph are independent and it can not handle additional constraints such as spatial exclusion. Furthermore, the KSP approach is designed to handle only a single type of object and can not explain a container/containee relationship between different object classes. Running it twice, first on the containers and then on the containees, is suboptimal and can result in errors as shown in Fig. 1. As the KSP algorithm, the Successive Shortest Paths (SSP) approach [3] links detections using sequential dynamic programming. However, the SSP approach works on sparse graphs and links detections on the image plane, meaning that it is not able to track an object that is missed by the detector.

Other algorithms use sampling techniques to solve the data association problem. Unlike KSP and SSP that rely on minimum cost flow algorithms, they perform simple local operations, such as growing trajectories or merging detections, to minimize an energy function [48], [49]. Such algorithms may explore a large portion of the state space but often involve many parameters that needs to be learned or tuned.

Some recent tracking algorithms incorporate higher-order motion models. Many of them collapse detections from consecutive temporal frames into single vertices. They are then used to build spatio-temporal graphs in which the motion costs are encoded either in the vertices [50], [51] or in the edges connecting them [52], [53]. The final trajectories are found by minimizing a cost function defined on that potentially complicated graph and that usually involves many variables. In practice, relaxation techniques are often used to speed up the optimization at the cost of not guaranteeing that the solution is the true global optimum [52], [54]. Furthermore, all these approaches focus on a single class of objects, and when applied on multi-class interacting objects such as containers and containees, we have to run the algorithms twice and thus they suffer the same problem as KSP.

Many of the approaches discussed above rely on discriminant classifiers to detect target objects in individual images, which makes them sensitive to occlusions. To alleviate this problem, some approaches add occlusion hypotheses and enlarge the state space to allow for occluded tracks [55], while others attempt to boost the probabilities of detections corresponding to potentially occluded targets [56], [57]. The approach of [58] introduces a confidence map based on geometric information, while the approach of [59] explicitly trains people detector on the failure cases of a tracker to exploit the occlusion patterns. In our framework, the detector [2] is explicitly designed to account for occlusions. Furthermore, because we model the container-containee relationship, we can handle not only short-term occlusions but also extended ones where objects remain completed occluded.

2.2 Tracking Interacting Objects

On balance, global techniques such as [5], [3] tend to be more robust than others. But, among those designed for people tracking, few can handle complex interactions between them and other scene objects. Some existing approaches model the group behavior of pedestrians during the tracking process by including priors that only account for the fact that people tend to avoid hitting each other and sometimes walk in groups [6], [7], [60], [61], [62]. In [7], there is also a mechanism for guessing where entrances and exits may be by recording where tracklets start and end. However, there is no provision for mobile entry points to allow objects of different natures to appear or disappear at varying locations. The approach of [62] focuses on tracking individual targets across groups by modeling split and merge events. It handles only one type of targets and relies on a post-processing step to recover their identities within a group.

There have been several recent attempts at modeling people interactions in specific scenarios, such as playing team sports [63], [64] or time-varying regular patterns [65]. However, such approaches require a new model to be trained for each specific scenario. In the case of [63], which looks into the behavior of sports players, the player trajectories are also assumed to be given. As a result, even though these approaches yield state-of-the-art performance in specific cases, it is not clear how general they are.

The approach of [9] exploits person-to-person and person-to-object interactions to more reliably track both people and objects. It relies on a Bayesian Network model to enforce frame-to-frame temporal consistency, 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 completely invisible during extended periods of time, such as a person inside a car.

T	the number of frames
$\mathbf{I}$	the set of all temporal frames
L	the number of spatial locations on the ground
O	the number of poses within each location
i, j, k, r	indices of the state of an object, which is a triple of spatial location, pose and time
l, m, n	indices of the spatial locations of an object
$l(k)$	the spatial location of state k
$o(k)$	the pose of state k
$t(k)$	the time of state k
v_k	the vertex representing an object at state k
e_{kj}	the edge between v_k and v_j
V	the set of all vertices v_k
E	the set of all edges e_{kj}
G	the directed acyclic graph, $G = (V, E)$
$\mathcal{N}(k)$	the neighborhood of state k, i.e., the set of states that can be reached from state k
X_k	the binary variable denoting the occupancy event by a containee object at state k
Y_k	the binary variable denoting the occupancy event by a container object at state k
ρ_k	the detector-estimated posterior of the occupancy event by a containee object at state k, $\rho_k = P(X_k = 1 \mid \mathbf{I}^{t(k)})$
β_k	the detector-estimated posterior of the occupancy event by a container object at state k, $\beta_k = P(Y_k = 1 \mid \mathbf{I}^{t(k)})$
f_{kj}	the binary flow variable denoting a containee object moving from state k to state j
g_{kj}	the binary flow variable denoting a container object moving from state k to state j
h_{lm}	the counter variable denoting the number of containee objects contained by a container that moves from location l to m
$\mathcal{N}_f(k)$	the spatial exclusion neighborhood for a containee object at state k
$\mathcal{N}_g(k)$	the spatial exclusion neighborhood for a container object at state k
$T_s(l)$	the temporal span of spatial location l
$T_p(k)$	the temporal span of state k
$\mathcal{N}_m(l)$	the movement neighborhood of a car at location l
$\mathcal{S}$	the set of all joint spatial vertices
$\mathcal{K}$	the set of all non-joint spatial tracklets
$\mathcal{N}_s(l)$	the spatial neighborhood of spatial location l
τ_q	a set of spatial vertices that can be either a trajectory or a spatial tracklet
t_q	the first time instant of τ_q
T_q	the last time instant of τ_q
$\hat{v}^t(\tau_q)$	the spatial vertex at time $t_q - 1$ in the neighborhood of τ_q
$\hat{v}^T(\tau_q)$	the spatial vertex at time $T_q + 1$ in the neighborhood of τ_q
γ_q^i	a pose tracklet on τ_q
G'	the tracklet graph

TABLE 1
Notations

3 Formulation

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 an object, as a 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 the

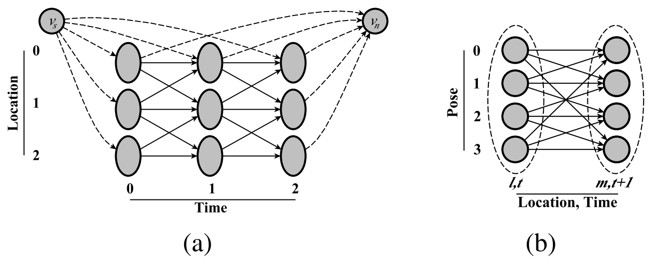


Fig. 2. A graph representing three spatial locations at three consecutive times. (a) Each ellipse denotes a *spatial vertex*, representing a spatial location at a time instant. Some are connected to a source and a sink to allow entrances and exits. (b) Each circle inside an ellipse denotes a *pose vertex*, representing a pose on a spatial location or a *state* of an object. In this case, there are four possible poses on each spatial location.

constraints and show that they result in a Mixed Integer Program (MIP) on a large graph. We will discuss in the following section our approach to nevertheless solving it fast.

3.1 Bayesian Inference

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

We discretize the ground plane of the monitored area 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*. For oriented objects such as cars, we define the pose space to be the set of regularly spaced object orientations on the ground plane; for non-oriented objects such as basketball and soccer ball, we define the pose space to be the regularly discretized height of the ball.

Let k denote the state of a target object, which we define to be the triple of location l, pose o and time t. In other words, we say an object occupies state k if it is located at l with pose o at time t. Let $\mathcal{N}(k)$ denote the neighborhood of k, that is, the states an object at state k at time t can reach at the next time instant $t+1$. Note that, the cardinality of $\mathcal{N}(k)$ depends on the target velocity and the frame rate. Let also $l(k)$, $o(k)$ and $t(k)$ respectively denote the location, pose and time of k.

Similar to [5], which treats spatial locations as graph vertices, we build a directed acyclic graph (DAG) $G = (V, E)$ on both the locations and poses, where the vertices $V = \{v_k\}$ represent the states of objects, and the edges $E = \{e_{kj}\}$ represent allowable transitions between them. More specifically, an edge $e_{kj} \in E$ connects vertices v_k and v_j 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. We show an example of such DAG in Fig. 2.

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

As will be discussed in Appendix C, we can estimate image-based probabilities $\rho_k = P(X_k = 1 \mid \mathbf{I}^{t(k)})$ and $\beta_k = P(Y_k = 1 \mid \mathbf{I}^{t(k)})$ that a containee or container object occupies state k at time $t(k)$ in such a way that their product over all k 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 and β_k, we look for the optimal set of paths as the solution of

$$(\mathbf{x},\mathbf{y})^* = \operatorname*{argmax}_{(\mathbf{x},\mathbf{y})\in\mathcal{F}} P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y} \mid \mathbf{I}) \tag{1}$$

$$\approx \operatorname*{argmax}_{(\mathbf{x},\mathbf{y})\in\mathcal{F}} \prod_k P(X_k = x_k \mid \mathbf{I}^{t(k)}) P(Y_k = y_k \mid \mathbf{I}^{t(k)}) \tag{2}$$

$$= \operatorname*{argmax}_{(\mathbf{x},\mathbf{y})\in\mathcal{F}} \sum_k \log P(X_k = x_k \mid \mathbf{I}^{t(k)}) + \log P(Y_k = y_k \mid \mathbf{I}^{t(k)}) \tag{3}$$

$$= \operatorname*{argmax}_{(\mathbf{x},\mathbf{y})\in\mathcal{F}} \sum_k x_k \log \rho_k + (1 - x_k) \log(1 - \rho_k) + y_k \log \beta_k + (1 - y_k) \log(1 - \beta_k) \tag{4}$$

$$= \operatorname*{argmax}_{(\mathbf{x},\mathbf{y})\in\mathcal{F}} \sum_k \log\left(\frac{\rho_k}{1-\rho_k}\right) x_k + \log\left(\frac{\beta_k}{1-\beta_k}\right) y_k \tag{5}$$

where $\mathcal{F}$ stands for the set of all feasible solutions as defined in the following section. Eq. 2 comes from the aforementioned property that the product of image-based probabilities is close to true posterior of Eq. 1, which will be discussed in more details in Appendix C, and from the assumption that all feasible transitions between two consecutive time instants are equally likely. Eq. 3 is obtained by taking the log of the product of probabilities. Eq. 4 is true because both x_k and y_k are binary variables. Finally, Eq. 5 is obtained by dropping constant terms that do not depend on x_k or y_k. 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. 5 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 two states at consecutive time instants. More specifically, we introduce flow variables f_{kj} and g_{kj}, which stand respectively for the number of containee and container type objects moving from state k to state $j \in \mathcal{N}(k)$. The flow variables f_{kj} and g_{kj} are defined on the edge e_{kj} and intertwined together.

In the following, in addition to the integrality constraints on the flow variables, we define five sets of constraints to obtain structurally plausible solutions. Our only assumption is that at each time instant, one container object can interact with at most one containee object.

Spatial Exclusion: As detailed in Appendix C.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

$$\sum_{i:k\in\mathcal{N}(i)} f_{ik} + \sum_{\substack{j\in\mathcal{N}_f(k),\\ i:j\in\mathcal{N}(i)}} f_{ij} \quad \leq \quad 1, \tag{6}$$

$$\sum_{i:k\in\mathcal{N}(i)} g_{ik} + \sum_{\substack{j\in\mathcal{N}_g(k),\\ i:j\in\mathcal{N}(i)}} g_{ij} \quad \leq \quad 1, \quad \forall k \ . \tag{7}$$

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

$$y_k = \sum_{i:k\in\mathcal{N}(i)} g_{ik} = \sum_{j\in\mathcal{N}(k)} g_{kj}, \quad \forall k \ . \tag{8}$$

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. We write

$$\sum_{j\in\mathcal{N}(s)} g_{sj} = \sum_{i:n\in\mathcal{N}(i)} g_{in} \ . \tag{9}$$

Consistency of Interacting Flows: We allow a containee type object to appear or disappear at the locations designated as the entrances or exits, and when it comes into contact with or leaves a container object. We write

$$-\sum_{\substack{r:l(k)=l(r),\\ i:r\in\mathcal{N}(i)}} g_{ir} \leq a(k) \leq \sum_{\substack{r:l(k)=l(r),\\ j\in\mathcal{N}(r)}} g_{rj}, \ \forall k \tag{10}$$

$$a(k) = \sum_{i:k\in\mathcal{N}(i)} f_{ik} - \sum_{j\in\mathcal{N}(k)} f_{kj} \ . \tag{11}$$

In Eq. 10, the total amount of container flow passing through the location $l(k)$ 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 $l(k)$. When they are equal to one, a containee object can appear or disappear at $l(k)$. Therefore, at the locations other than the entrances and exits, a containee object can appear or disappear only when it interacts with a container object in its vicinity. Note that, here we assume multiple containee objects never interact with a container one at exactly the same moment. For example, at one time instant only one person is allowed to enter the car. Given modern digital cameras recording at more than 25 fps, this assumption imposes at most 1/25 second delay, which is barely noticeable in practice.

Note that all four sums in Eqs. 10 and 11 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. For scenarios such as cars and people, this can give rise to several undesirable results as shown 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. We express this as

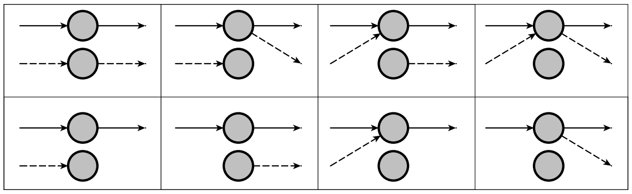


Fig. 3. Flow constraints in a two-pose case. In each of the eight examples, the two circles represent two pose nodes at the same spatial location. The solid and the dotted arrows represent respectively non-zero flows g_{kj} and f_{kj} of the container and of the visible containee objects. **Top Row.** 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.

$$\sum_{\substack{k:l=l(k),\\ i:k\in\mathcal{N}(i)}} f_{ik} + \sum_{\substack{k:l=l(k),\\ j\in\mathcal{N}(k)}} f_{kj} \leq 2 - \sum_{\substack{k:l=l(k),\\ j\in\mathcal{N}(k)}} g_{kj}, \quad \forall l \ . \tag{12}$$

Note that, we do not impose this set of constraints in the basketball and soccer scenarios, where we do allow a flying ball and a player to coexist at the same spatial location.

Tracking the Invisible: We say a containee object is *invisible* when it is carried by a container. The 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_{lm}, which stand for the number of invisible containees moving from spatial location l to spatial location $m \in \mathcal{N}_s(l)$, where $\mathcal{N}_s(l)$ denotes the spatial neighborhood of spatial location l. These variables act as counters that are incremented or decremented when a containee object respectively disappears or appears in the vicinity of a container. We write

$$\sum_{m\in\mathcal{N}_s(l)} h_{lm} = \sum_{n:l\in\mathcal{N}_s(n)} h_{nl} + \sum_{\substack{k:l=l(k),\\ i:k\in\mathcal{N}(i)}} f_{ik} - \sum_{\substack{k:l=l(k),\\ j\in\mathcal{N}(k)}} f_{kj}, \ \forall l \tag{13}$$

$$h_{lm} \leq \sum_{\substack{k:l(k)=l,\\ j:l(j)=m,\\ j\in\mathcal{N}(k)}} c * g_{kj}, \quad \forall l, m \in \mathcal{N}_s(l) \ , \tag{14}$$

where c is an 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 five. Note that, Eq. 13 ensures that the h_{lm} variables are incremented or decremented always by an integer value. Therefore, we allow h_{lm} to be continuous in our optimization, except only those that are connected to the source, i.e., h_{sl}, which we restrict to be integers. Our experimental results show that allowing $h_{lm:l\neq s}$ to be continuous slightly speeds up the optimization, compared to imposing the integrality constraints on them.

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

$$\sum_{\substack{l:t\in T_s(l),\\ m\in\mathcal{N}_s(l)}} h_{lm} + \sum_{\substack{k:t\in T_p(k),\\ j\in\mathcal{N}(k)}} f_{kj} \leq 1 \ , \quad \forall t \ , \tag{15}$$

where $T_s(l)$ and $T_p(k)$ denote the temporal span of spatial location l and state k respectively. Together with the invisible flow constraints in Eqs. 13 and 14, 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

$$h_{lm} \geq \sum_{\substack{k:l(k)=l,\\ j:l(j)=m\\ j\in\mathcal{N}(k)}} g_{kj}, \quad \forall l, m \in \mathcal{N}_m\left(l\right) \ , \tag{16}$$

where $\mathcal{N}_m(l)$ denotes the movement neighborhood of the car at location l. In other words, we say an object moves from l to a different spatial location m, if $m \in \mathcal{N}_m\left(l\right)$ holds.

3.3 Mixed Integer Programming

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

$$\sum_{k,j\in\mathcal{N}(k)} \left(\alpha_k \, f_{kj} + \gamma_k \, g_{kj}\right), \tag{17}$$

where α_k and γ_k are the costs for the flow variables f_{kj} and g_{kj} respectively, and they are defined as follows:

$$\alpha_k = -\log\left(\frac{\rho_k}{1-\rho_k}\right) \text{ and } \gamma_k = -\log\left(\frac{\beta_k}{1-\beta_k}\right). \tag{18}$$

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, y_k) and the flow variables (f_{kj}, g_{kj}), this is equivalent to maximizing the expression of Eq. 5.

Solving the corresponding Linear Program (LP) obtained by relaxing the integrality constraints is usually faster than solving the original MIP but may result in fractional flow values. In the result section, we will compare MIP results against LP results after rounding them to integers.

4 Optimization

In most practical situations, the MIP of Eq. 17 has too many variables to be handled directly by ordinary solvers. In this section, we show how to make the problem more tractable and to achieve real-time tracking performance.

4.1 Pruned Intertwined Flows (PIF)

To reduce the computational time, we first 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 and β_k of being occupied by either a container or containee object 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.

- **Step 1:** 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 trajectories that can start and end anywhere and anytime on the grid.
- **Step 2:** We connect all the resulting trajectories both to each other and to the original entrance and exit locations using the Viterbi algorithm [66].

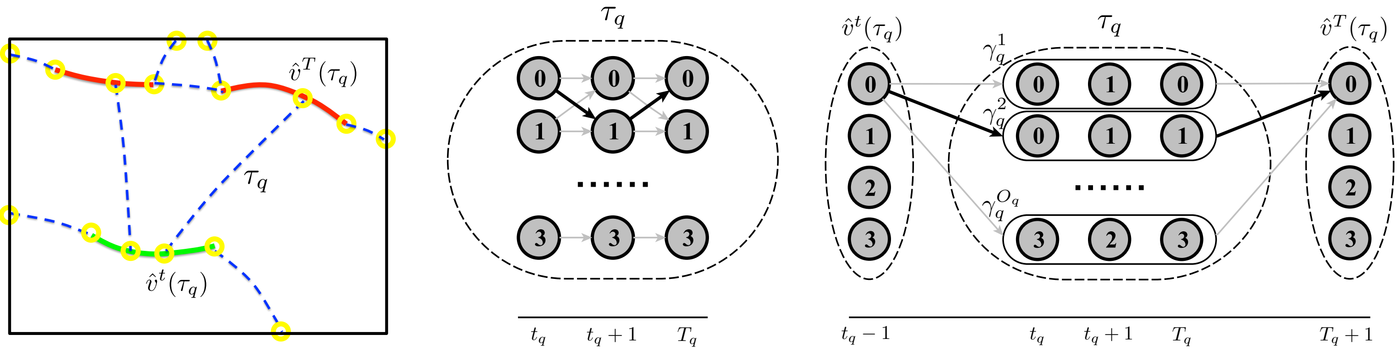
In this way, we obtain a set of spatial trajectories, whose nodes belong either to the trajectories obtained in Step 1, or the paths connecting them obtained in Step 2. The resulting subgraph still contains the low ρ_k and β_k locations that may correspond to missed detections, while being considerably smaller than the original graph. In our experiments, the pruning reduces the number of variables by three orders of magnitude and the number of kept spatial locations is about 10 times as large as that of the ground truth. We solve the MIP of Eq. 17 on the pruned graph, where the variables are flows that link two pose vertices between two consecutive frames, as we did in our original paper [10]. We will refer to this approach as Pruned Intertwined Flows (**PIF**) in the remainder of the paper. Even though **PIF** is an approximation to the MIP optimization on the full graph, in practice both methods yield very similar results as shown in Appendix B.

4.2 Tracklet-Based Intertwined Flows (TIF)

To further reduce the computational complexity while preserving the optimality of the solution on the pruned graph, we introduce a tracklet-based formulation of the optimization problem, which we will refer to as **TIF**. As will be shown in the result section, it yields a speed-up factor of up to three orders of magnitude with respect to the **PIF** approach introduced in the previous section.

We achieve this result by grouping unambiguous spatial vertices into spatial tracklets and then removing redundant pose vertices and edges, as depicted in Fig. 4. We summarize the corresponding workflow below and formalize it in Appendix A.

- **Grouping Spatial Vertices into Tracklets** We partition the spatial vertices of the trajectories introduced in § 4.1 into two subsets, the joint vertices $\mathcal{S}$ and the non-joint ones $\mathcal{K}$. The vertices in $\mathcal{S}$ are those included in the neighborhood of more than one trajectory and those at



(a) (b) (c)

Fig. 4. Construction of the tracklet graph. (a) We obtain a set of paths by the graph pruning approach as described in § 4.1. We use the solid lines to denote the spatial trajectories obtained by the KSP approach described in Step 1, and the dotted lines to denote the paths obtained by dynamic programming in Step 2. We use the yellow circles to denote the joint spatial locations. For each spatial tracklet τ_q, there is a unique predecessor spatial vertex and a unique successor, denoted by $\hat{v}^t(\tau_q)$ and $\hat{v}^T(\tau_q)$ respectively. (b) We use t_q and T_q to denote the first and last time instant of τ_q respectively. We compute pose tracklets on τ_q using dynamic programming. The black arrows denote the pose transitions with the lowest total cost among all transitions that connect the pose 0 at time t_q to the pose 0 at time T_q. Therefore, we treat the shortest path 0-1-0 as a pose tracklet of τ_q and collapse its vertices into a single vertex. (c) We use γ_q^i to denote a pose tracklet of τ_q. The graph can be further simplified by keeping only those edges along the shortest path connecting a pose vertex at time t_q-1 to a pose vertex at time T_q+1. The black arrows highlight the two edges along the shortest path from pose 0 at time t_q-1 to pose 0 at time T_q+1.

the beginning or end of a trajectory. The vertices in $\mathcal{K}$ are the remaining ones and are located between joint vertices. Note that an identity switch or an interaction event can occur only at the joint vertices. We therefore group non-joint vertices between two joint ones into a single spatial tracklet, which we denote by τ_q.

- **Computing the Poses of the Tracklets** Each spatial tracklet τ_q can be treated as a single spatial vertex as shown in Fig. 4(b). The pose of such vertex is a *pose tracklet*, which is a set of possible pose vertices on τ_q. Since τ_q connects to other vertices only through the pose vertices at its first frame t_q and last frame T_q, the pose of τ_q is uniquely defined by its starting and ending pose vertices at t_q and T_q respectively. In other words, among all possible pose tracklets that share the same starting and ending pose vertices, only the one with the lowest cost can be potentially selected. We can therefore remove the majority of pose tracklets of a spatial tracklet without loss of optimality. For example, in the case of Fig. 4(b), the pose tracklet 0-0-0 yields a higher cost than 0-1-0 and will never be selected by the solver. To remove all such pose tracklets while preserving the completeness of the state space, for each τ_q, we run dynamic programming on each pair of starting and ending poses and only keep the best pose tracklet. In the specific case of Fig. 4(b), we would do this $4 \times 4 = 16$ times and retain only 16 pose tracklets, which are treated as the poses of τ_q.
- **Constructing the Tracklet Graph** We can now construct a tracklet graph G' by treating each pose tracklet as a single vertex and taking the edges to be allowable transitions between them. However, some edges can still be removed while preserving optimality. Consider a spatial tracklet τ_q that has only one predecessor and one successor vertex, which we denote by $\hat{v}^t(\tau_q)$ and $\hat{v}^T(\tau_q)$ respectively. Since an interaction event will never take place at τ_q, its incoming and outgoing flows should always be conserved. In other words, if τ_q is selected by the MIP solver, $\hat{v}^t(\tau_q)$ and $\hat{v}^T(\tau_q)$ must also be selected. This means we can collapse all three into one. For each pair of starting pose vertex at $\hat{v}^t(\tau_q)$ and ending pose vertex at $\hat{v}^T(\tau_q)$, we compute the shortest path connecting these two vertices and only keep those edges along the shortest path. In the case of Fig. 4(c), given a starting pose of 0 at $\hat{v}^t(\tau_q)$ and an ending pose of 0 at $\hat{v}^T(\tau_q)$, the black arrows denote the pair of edges along the shortest path and thus should be kept. We go though all combinations and thus keep up to $4 \times 4 = 16$ pairs of edges between $\hat{v}^t(\tau_q)$, τ_q and $\hat{v}^T(\tau_q)$.

4.2.1 MIP on the Tracklet Graph

The MIP of Eq. 17 defined on the original graph G applies directly to the tracklet graph G', where the flow variables f_{kj} and g_{kj} are defined on the edges between pose tracklets. The costs for such flow variables, α_k and γ_k, become the sum of costs for all poses within the pose tracklet k. As a result, the MIP of **TIF** is strictly equivalent to the one of **PIF**.

4.3 Solving the LP and MIP

We implemented our algorithm in C++ using the Gurobi library V6.0 [8]. To solve the MIP, the Gurobi solver uses a branch-and-cut procedure that iterates the steps of solving LPs and applying cuts to obtain integer solutions. The branch-and-cut procedure minimizes the gap between a lower bound obtained from LP relaxations and an upper bound obtained from feasible integer solutions. It stops when the gap drops below a specified tolerance value, set to 0.001 in practice. This means that the solution is very close to the global optimum. To use the Gurobi solver, we chose the dual simplex algorithm

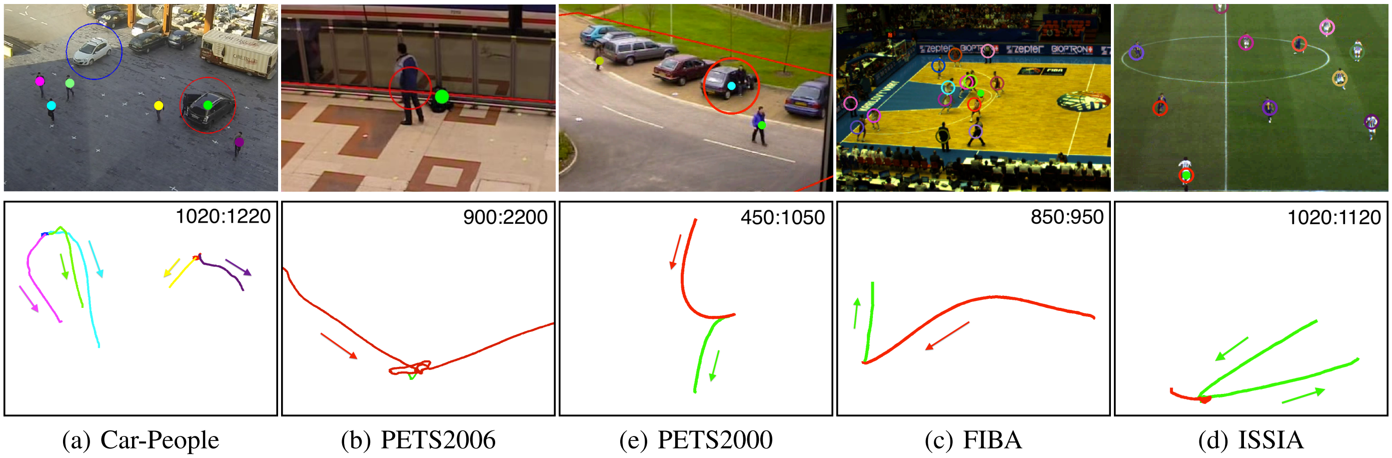


Fig. 5. Tracking results on five representative subsequences taken from our datasets. **Top row.** Sample frames with the detected container objects highlighted with circles and containee ones with dots. **Bottom Row.** Corresponding color-coded top-view trajectories for interacting objects in the scene.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

for solving LPs, which we found to be very efficient, and we set all the other parameters to their default values.

5 Experiments

In this section, we first describe the video sequences and metrics we used for validation purposes. We then introduce several state-of-the-art baseline methods. Finally, we compare our approach to these baselines both in terms of tracking accuracy and computational cost. We will show that our approach outperforms them consistently.

5.1 Test Datasets

We tested our approach on five datasets featuring four very different scenarios: people and vehicles on a parking lot (Car-People and PETS2000 dataset [67]), people and luggage in a railway station (PETS2006 dataset [67]), basketball players and the ball during a high-level competition (FIBA dataset), and soccer players and the ball in a professional match (ISSIA dataset [68]). These datasets are either multi-view or monocular, and 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 describe them below and give more details in Tab. 2.

- **Car-People Dataset:** We captured five sequences on a parking lot with two synchronized cameras. They comprise from 300 to 5100 temporal frames, and they feature many instances of people getting in and out of cars. This dataset is challenging because the lighting in the scene changes constantly and the color of the objects is similar to that of the background, which makes the background subtraction prone to errors.
- **PETS2006 Dataset [67]:** We use a 3020-frame sequence acquired by two synchronized cameras that features 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. The monitored area is relatively small and the pedestrians heavily occlude each other.
- **PETS2000 Dataset [67]:** We use a 1450-frame monocular sequence featuring people and cars entering and leaving a parking lot. This sequence contains multiple car instances and two people getting out of one car, one after the other.
- **FIBA Dataset:** We use a 2850-frame sequence captured by six synchronized cameras at the 2010 FIBA Women World Championship. It features two five-player-teams, three referees and two coaches. This sequence is challenging due to the complex and frequent interactions between the players and the ball, making it hard to detect the ball.
- **ISSIA Dataset [68]:** The dataset contains a sequence featuring two eleven-player-teams and three referees. The sequence is captured by six cameras, three on each side of the court. There is little overlap between the cameras on one side and each target is covered by two cameras only.

5.2 Parameters and Baselines

For car and luggage tracking we take the pose to be the orientation. For ball tracking, we take it to be the height. We use two regularly distributed orientations for luggages and twelve for cars. This allows us to take advantage of the relatively high image resolution to handle occlusions effectively. In the basketball and soccer cases, we discretize the height of the ball into 50cm cells, which we found to be sufficient given the camera resolution.

We will refer the Pruned Intertwined Flows described in § 4.1 as **PIF** and the Tracklet-Based Intertwined Flows in § 4.2 as **TIF**. As discussed, **TIF** preserves the completeness of the state space and therefore the Mixed Integer Programs of

Sequence Name	Cameras	Frames	Locations	Containers	Containees	Container Poses	Containee Poses
Car-People Seq.0	2	350	6848	1	3	12	1
Car-People Seq.1	2	1500	6848	2	3	12	1
Car-People Seq.2	2	296	6848	2	3	12	1
Car-People Seq.3	2	2759	6848	2	8	12	1
Car-People Seq.4	2	5100	6848	4	12	12	1
PETS2006	2	3020	8800	24	1	1	2
PETS2000	1	1450	11200	3	3	12	1
FIBA	6	2850	9728	15	1	1	16
ISSIA	6	1990	34020	25	1	1	17

TABLE 2

Validation sequences. From left to right, we give the number of cameras, frames, spatial locations, objects and distinct poses respectively. For the Car-People, PETS2000 and PETS2006 datasets, we take the pose to be the orientation of the targets, while for the FIBA and ISSIA dataset, we take it to be the height of the ball.

TIF and **PIF** are equivalent. We use **TIF-MIP** to denote the approach that solves the Mixed Integer Program of **TIF**, and **TIF-LP** to denote the approach that solves the Linear Program with the integrality constraints relaxed.

We compare our **TIF-MIP** approach against the following eight state-of-the-art methods whose code are public available. We use the default parameters for all these methods.

- **POM:** We keep those pose 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 Paths (SSP) [3] is an algorithm for tracking multiple objects. It first builds a graph by linking pairs of object detections in consecutive temporal frames and then sequentially applies Dynamic Programming to find solutions. We run the publicly available SSP code and compare the results with ours.
- **LP2D:** LP2D [69] is a multi-object tracking algorithm that yields promising results in the recent Multiple Object Tracking challenge (MOT) [70]. As SSP, it builds a graph whose nodes are detections in the image plane and then minimizes an energy using Linear Programming.
- **LP3D:** The LP3D approach [69] is similar to the LP2D one, except that the graph nodes are detections in the 3D world coordinates.
- **KSP-free:** As discussed in § 4.1, 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.
- **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.
- **TIF-LP:** We relax the integrality constraints of the MIP of Eq. 17 and solve the Linear Program (LP). The resulting flow variables are then rounded to the nearest integer to obtain the final solution.

For all the methods, we use the same detection results obtained by POM, which explicitly accounts for mutual occlusion between the targets. More technical details about this algorithm can be found in [2]. Note that, instead of using POM, we could have relied on the popular Deformable Part Model (DPM) [71]. However, as discussed in Appendix D, it is not as well adapted to our needs because it is not designed to be robust to occlusions.

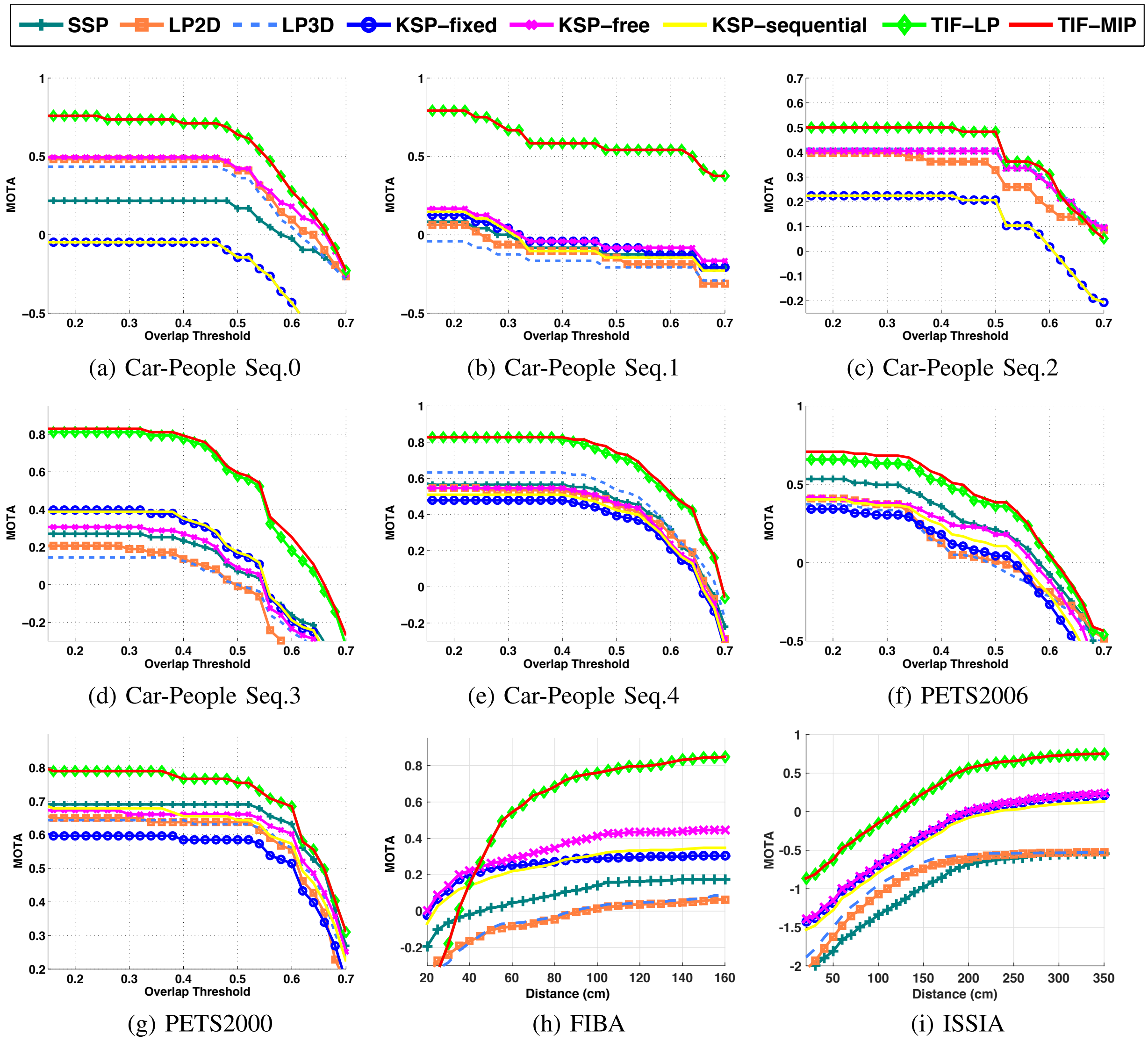
5.3 Evaluation Metrics

To quantify the results, we use the standard CLEAR [72] metrics: Multiple Object Detection Accuracy (MODA) and Multiple Object Tracking Accuracy (MOTA). MODA is a detection-based evaluation metric that penalizes false positive (FP) and false negative (FN) detections, while MOTA is a tracking-based metric that also accounts for identity switches (IDS). To compute MODA and MOTA, we weight all three types of errors equally. We also report the values for the three kinds of errors separately for all tested methods.

The FP and FN scores are computed based on a threshold, which is defined to be either the overlap ratio between a detection and a ground truth on the image plane, or the distance between them on the ground plane. To evaluate oriented object detections, the latter one is not suitable because it does not penalize detections with incorrect orientations, while the former one accounts for orientation on the image plane. Therefore we use the overlap ratio as the threshold for FP and FN for the Car-People, PETS2006 and PETS2000 Dataset. However, for non-oriented objects such as balls, the distance between the detection and the ground truth on the ground is a good fit because it provides an absolute measurement independent of camera views. We therefore use it as the threshold for the FIBA and ISSIA Dataset.

5.4 Results

We ran all the baseline algorithms and ours on all the test sequences introduced in § 5.1. In Fig. 5, we show qualitative tracking results on representative subsequences of each



(a) Car-People Seq.0 (b) Car-People Seq.1 (c) Car-People Seq.2

(d) Car-People Seq.3 (e) Car-People Seq.4 (f) PETS2006

(g) PETS2000 (h) FIBA (i) ISSIA

Fig. 6. Comparing our proposed approach (**TIF-MIP**) against the baselines in terms of the MOTA scores. Our tracker yields a significant improvement on all datasets, thanks to the joint-global optimization on both container and containee objects. (a)-(g) We plot the MOTA curve w.r.t a range of overlap thresholds on the image plane. (h)-(i) We plot the MOTA curve w.r.t a range of distances between the detections and the ground truths on the ground plane.

dataset. In Fig. 6, we plot MOTA curves for all the algorithms on all the tested sequences. We show the results on a range of thresholds as a monotonic curve. In Tab. 3, we show the FP rate, FN rate, IDS rate and MODA for all methods at an overlap threshold of 0.5 and distance threshold of 1m for FIBA and 2m for ISSIA. We provide videos overlaid with tracking results in the supplementary material.

5.4.1 Car-People, PETS2000 and PETS2006 Sequences

As we show in Fig. 6 and Tab. 3, our tracker yields significant improvements over the baseline algorithms on all tested sequences. The sequence Car-People Seq.0 is the one from which we extracted the image shown in Fig. 1. It features three people getting into a car stopped at the center of a parking lot. In this case, the POM detector fails to detect the car in many frames 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, LP2D, LP3D 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 also does better than KSP-fixed and KSP-sequential because the people are in fact detected most of the time. **TIF-MIP** 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. As a result, the FN rate is significantly lower than other baselines which further leads to a higher MOTA and MODA score.

Car-People Seq.1 features two people getting into the first car, staying for a while, getting out and then entering the second one. Here, KSP-free does slightly better than KSP-fixed, which needs to hallucinate two false positive tracks to

Sequence Name	Metric	POM	SSP	LP2D	LP3D	KSP-fixed	KSP-free	KSP-sequential	TIF-LP	TIF-MIP
Car-People Seq.0	**FP**	0.06	**0.04**	0.05	0.05	0.46	0.10	0.46	0.07	0.07
	FN	0.47	0.76	0.48	0.53	0.61	0.41	0.61	**0.25**	**0.25**
	IDS	N/A	**0.04**	0.06	0.06	0.07	0.07	0.07	**0.04**	**0.04**
	MODA	0.47	0.20	0.47	0.42	-0.07	0.49	-0.07	**0.67**	**0.67**
Car-People Seq.1	**FP**	0.98	0.75	0.77	0.75	0.77	0.71	0.75	**0.17**	**0.17**
	FN	0.23	0.25	**0.21**	0.25	0.25	0.25	0.25	0.25	0.25
	IDS	N/A	0.12	0.17	0.21	0.06	0.12	0.15	**0.04**	**0.04**
	MODA	-0.21	0.00	0.02	0.00	-0.02	0.04	0.00	**0.58**	**0.58**
Car-People Seq.2	**FP**	0.03	**0.00**	0.03	**0.00**	0.05	**0.00**	0.05	0.03	0.03
	FN	**0.47**	0.59	0.62	0.58	0.72	0.59	0.72	**0.47**	**0.47**
	IDS	N/A	**0.01**	0.02	**0.01**	0.03	**0.01**	0.03	**0.01**	**0.01**
	MODA	**0.50**	0.41	0.35	0.42	0.23	0.41	0.23	**0.50**	**0.50**
Car-People Seq.3	**FP**	0.59	0.35	0.43	0.27	0.46	0.43	0.43	**0.14**	**0.14**
	FN	**0.17**	0.31	0.23	0.40	0.19	0.23	0.19	0.21	0.21
	IDS	N/A	0.27	0.34	0.33	0.19	0.25	0.21	0.07	**0.05**
	MODA	0.24	0.34	0.34	0.33	0.35	0.34	0.38	**0.65**	**0.65**
Car-People Seq.4	**FP**	0.40	0.19	0.26	0.13	0.32	0.25	0.31	0.08	**0.07**
	FN	**0.15**	0.19	0.16	0.18	0.17	0.17	0.16	0.16	**0.15**
	IDS	N/A	0.14	0.13	0.15	0.12	0.12	0.11	**0.04**	**0.04**
	MODA	0.45	0.62	0.58	0.69	0.51	0.58	0.53	0.76	**0.78**
PETS2006	**FP**	1.15	**0.32**	**0.32**	**0.32**	0.62	0.42	0.56	0.33	0.33
	FN	**0.11**	0.29	0.52	0.55	0.16	0.20	0.16	0.24	0.22
	IDS	N/A	0.18	0.16	0.16	0.17	0.20	0.18	0.07	**0.06**
	MODA	-0.26	0.39	0.16	0.13	0.22	0.38	0.28	0.43	**0.45**
PETS2000	**FP**	0.12	**0.01**	**0.01**	0.02	0.16	0.06	0.11	0.03	0.03
	FN	**0.19**	0.26	0.30	0.30	0.20	0.20	0.20	0.20	0.20
	IDS	N/A	0.04	0.05	0.05	0.05	0.07	0.05	**0.02**	**0.02**
	MODA	0.69	0.73	0.69	0.68	0.64	0.74	0.69	**0.77**	**0.77**
FIBA	**FP**	0.63	0.29	0.32	0.33	0.04	**0.02**	0.10	0.12	0.12
	FN	0.43	0.50	0.57	0.55	0.66	0.56	0.56	**0.12**	**0.12**
	IDS	N/A	0.07	0.09	0.10	**0.00**	0.01	0.03	**0.00**	**0.00**
	MODA	-0.06	0.21	0.11	0.12	0.29	0.42	0.34	**0.76**	**0.76**
ISSIA	**FP**	0.65	1.35	1.35	1.21	0.73	0.71	0.70	0.20	**0.19**
	FN	0.25	0.27	**0.21**	0.27	0.24	0.25	0.22	0.23	0.23
	IDS	N/A	0.07	0.05	0.08	0.04	0.03	0.16	**0.01**	**0.01**
	MODA	0.10	-0.62	-0.56	-0.48	0.03	0.04	0.08	0.57	**0.58**

TABLE 3

Comparison of false positive (FP) rate, false negative (FN) rate and identity switches (IDS) rate between all the methods at overlap ratio of 0.5. For FIBA and ISSIA, the distance is taken to be 1.0 m and 2.0 m respectively.

allow for the people emerging from the first car. **TIF-MIP** yields a better result compared to other baselines because our tracker removes spurious detections that physically overlap each other thanks to spatial exclusion constraints. The case of Car-People Seq.2 is similar to that of Car-People Seq.0, where POM fails to detect the car in many frames but our constraints enforce that there is a car at the location where three people get out. Therefore, our approach works better than other methods such as KSP-sequential that tracks cars and people separately. The sequences Car-People Seq.3 and Car-People Seq.4 feature more instances of people getting in and out of cars, and our approach consistently yields an overall better performance in terms of MOTA and MODA as compared to all baseline methods, thanks to the global optimization that accounts for all objects simultaneously.

In the PETS2006 sequence, a person enters the scene and leaves a bag on the ground. In this case, KSP-fixed hallucinates a false positive track of the bag starting from the edge of the monitored area. Furthermore, when the scene gets crowded, all the baseline algorithms produce solutions that contain overlapping detections in the 3D space. The constraints imposed by our tracker prevent such spurious detections. The same happens for the PETS2000 sequence, where our tracker yields better results by imposing the correct set of constraints while optimizing for both cars and people simultaneously.

5.4.2 FIBA and ISSIA Sequences

As mentioned in § 3.2, our algorithm can keep track of invisible containee objects. In the basketball and soccer cases, where the ball is often occluded by the players, we utilize the locations of the players to estimate the ball's ground-plane location. More specifically, when the ball is inferred as visible, we use its trajectories directly; otherwise, we take the location of the ball to be that of the player who is inferred to be in possession of it, or equivalently, the location whose counter variable is non-zero. Since we also impose the constraints that there is at most one ball in the court, in each frame, there is at most one spatial location whose counter variable is one.

In Figs. 6(h,i), we evaluate quantitatively our ball-tracking results in terms of ground trajectory. We show the results for the ball only because the people detections are very good and therefore all baselines perform similarly. Our ball detector produces many spurious and missing detections because of the weak image evidence, which results in a low MODA score. SSP yields a low MOTA score, because it operates on the sparse detection graphs and links the detections on the image plane. In contrast, all KSP-based algorithms operate on the dense detection graphs and link detections on the

Sequence Name	**Methods**	**Variables**	**Constraints**	**Non-zero Coefficients**	**DP Time (s)**	**Iterations**	**Solving Time (s)**	**Speed (fps)**
Car-People Seq.0	**PIF**	357K	441K	10.8M	N/A	125.3K	225.68	1.6
	TIF-LP	14K	84K	3.5M	0.03	3.3K	1.84	190.2
	TIF-MIP	14K	84K	3.5M	0.03	3.2K	2.28	153.5
Car-People Seq.1	**PIF**	3.0M	4.0M	105.3M	N/A	854.1K	5628.05	0.3
	TIF-LP	334K	994K	37.3M	0.11	16.4K	7.89	190.1
	TIF-MIP	334K	994K	37.3M	0.11	16.3K	13.32	112.6
Car-People Seq.2	**PIF**	149K	293K	5.6M	N/A	44.1K	78.89	3.8
	TIF-LP	10K	83K	5.0M	0.04	3.0K	2.01	147.3
	TIF-MIP	10K	83K	5.0M	0.04	2.9K	3.27	90.5
Car-People Seq.3	**PIF**	3.0M+829K	4.2M+1.2M	91.9M + 26.0M	N/A	1.1M+342.9K	4832.48+286.20	0.6
	TIF-LP	535K+124K	1.2M+371K	36.4M + 14.4M	0.15+0.04	35.3K+10.2K	12.50+2.24	214.3
	TIF-MIP	535K+124K	1.2M+371K	36.4M + 14.4M	0.15+0.04	44.1K+11.8K	39.59+6.50	68.5
Car-People Seq.4	**PIF**	6.0M+7.1M+3.1M	7.7M+9.1M+4.3M	213.1M+249.6M+110.7M	N/A	2.4M+3.8M+2.0M	40.0K+95.1K+24.0K	0.0
	TIF-LP	1.0M+1.2M+539K	1.5M+1.8M+1.0M	75.7M+82.0M+39.6M	0.12+0.06+0.08	41.1K+48.3K+32.3K	36.79+40.22+43.07	49.1
	TIF-MIP	1.0M+1.2M+539K	1.5M+1.8M+1.0M	75.7M+82.0M+39.6M	0.12+0.06+0.08	54.3K+62.9K+43.0K	77.21+76.20+57.81	27.9
PETS2006	**PIF**	1.2M	1.1M	8.1M	N/A	100.5K	57.46	52.6
	TIF-LP	69K	496K	705K	0.06	24.3K	7.05	428.4
	TIF-MIP	69K	496K	705K	0.06	27.0K	17.87	169.0
PETS2000	**PIF**	348K	846K	10.8M	N/A	66K	29.09	49.8
	TIF-LP	50K	273K	8.4M	0.04	6.5K	4.95	292.9
	TIF-MIP	50K	273K	8.4M	0.04	7.6K	10.42	139.2
FIBA	**PIF**	5.1M	1.6M	30.7M	N/A	51.0K	56.60	50.4
	TIF-LP	891K	1.9M	8.4M	0.60	25.0K	8.94	318.8
	TIF-MIP	891K	1.9M	8.4M	0.60	25.0K	14.13	201.7
ISSIA	**PIF**	5.8M	2.1M	34.0M	N/A	33.7K	29.98	66.4
	TIF-LP	1.3M	2.1M	8.3M	1.40	18.3K	5.54	359.2
	TIF-MIP	1.3M	2.1M	8.3M	1.40	18.0K	14.49	137.3

TABLE 4

Comparison of computational costs for **PIF**, **TIF-LP** and **TIF-MIP**. From left to right, we show the total number of variables, the number of constraints, the number of non-zero coefficients in the constraint matrix, the time for running the dynamic programming on poses (DP time), the number of simplex iterations, the solving time by the Guorbi solver and the speed in terms of frames per second (fps). Compared to **PIF**, **TIF** significantly reduces the number of variables and constraints, and yields large speed-up factors up to three orders of magnitude. Due to the large number of variables, we run Car-People Seq.3 and Car-People Seq.4 on batches of 2K frames with 20% overlap for both **PIF** and **TIF** approaches, and then we use the Hungarian algorithm to glue the batches.

ground plane. When the ball is flying fast in the air, KSP-based algorithms enforce physical constraints on the ground plane, for example the maximum speed of the ball, which can not be well accounted for on the image plane by SSP. KSP-sequential yields a poor performance because of the spurious ball detections close to the players. KSP-free eliminates some false positive detections by requiring a cost to be paid for every appearance or disappearance of the ball. Our tracker achieves the best performance by enforcing that there can be at most one ball in the field during the game and reasoning for the players and the ball simultaneously, especially by tracking the players in possession of the ball thanks to the counter variables. In the FIBA sequence, our method outperforms the baselines only for distance thresholds greater than 40cm. This is because it uses discretized player locations to estimate the ball locations when the ball is possessed by a player, which degrades the accuracy.

5.4.3 **TIF-LP** *vs.* **TIF-MIP**

Solving the LP problem of § 3.3 and subsequently rounding the resulting fractional flow variables as in **TIF-LP** systematically performs either very similarly or slightly worse than explicitly imposing the integrality constraints as we do in the **TIF-MIP** approach. We have observed by our experiments that relaxing the integrality constraints in some cases leads to breaks of tracks. In the PETS2006 sequence, where the difference between **TIF-MIP** and **TIF-LP** is visible, the ratio of fractional flows to non-zero flows is about 5.6%. In the Car-People Seq.0, all resulting flows obtained by **TIF-LP** are integers, meaning that the optimal solution of LP lies on one of the vertices of the integral polyhedron. Therefore, the results of **TIF-LP** and **TIF-MIP** are exactly the same. Interestingly, we have observed that running **TIF-LP** and **PIF-LP**, that is, running **PIF** without the integrality constraint, in few cases leads to different results, meaning that the LP-relaxed version of **PIF** and **TIF** are not equivalent.

5.4.4 Failure Cases

In the Car-People dataset, we observe a few failure cases where a person gets into a car without the associated counter variable being incremented. This is because the car is parked on the boundary of the monitored area and the person is detected closer to it than to the car. As a result, the optimizer explains the disappearance of the person as him leaving the scene from the boundary instead of entering the car. In the FIBA sequence, we observe that in some cases the ball is assigned to a wrong player. This is because there are some spurious ball detections close to the players and the optimizer explains them by the players catching and then throwing the ball.

5.5 Computational Cost

We compare the computational costs of **PIF**, **TIF-LP**, and **TIF-MIP** in Tab. 4. **TIF** requires far fewer variables and constraints than **PIF** and yields a speed-up of up to three orders of magnitude. The only overhead is the dynamic programming on poses, which requires much less time than solving the MIP or LP. We also show the number of simplex iterations, which is the number of pivot operations needed to solve MIP or LP. It is independent of hardwares and thus provides an objective measurement of the computational performance.

For Car-People Seq.1, we reduce the number of variables from 3 million to 300 thousand and the number of constraints from 4 million to one quarter. In this case, the dynamic programming on poses takes only 0.1 second. The number of iterations drops from 850 thousand to 16 thousand and solving time drops from 5600 seconds to 13 seconds for the **TIF-MIP** and 8 seconds for the **TIF-LP**. To process the whole Car-People Seq.5 sequence, **PIF** takes more than 159 thousand seconds (44 hours), while **TIF** takes only about 210 seconds. Despite the large number of variables and constraints, both **PIF** and **TIF** run fast on the sports sequences. This can be partially explained by the fact that our player detections are very good in such sequences. As a result, the optimization is an easier problem to solve as compared to those of the car sequences. **TIF** in such cases runs 2-3 times faster than **PIF**. The same happens for PETS2000 and PETS2006, where **TIF** yields a speed-up factor of a few times compared to **PIF**. All methods run slower on the Car-People sequences, however **PIF** yields a significant performance drop, because of the large size of the model as indicated by the number of variables and non-zero coefficients.

Note that we ran the Gurobi dual simplex algorithm introduced in § 4.3 on a single thread. We would therefore expect even faster convergence with multi-thread implementations.

5.6 Limitations and Future Work

Our approach to tracking interacting objects is generic and can be applied to a wide range of scenarios. However, it has two main limitations. First, since it relies on probabilities of occupancy in 3D space, we require the cameras to be calibrated. Second, it does not preserve identities throughout interactions. For example, if someone gets into a car and then gets out, our tracker will assign two different track identities to that person. A simple solution to this problem would be to post-process the results, to cluster the obtained trajectories, and to assign identities to them. However, this would be suboptimal because the tracking step and the identity recovery step are decoupled.

In our future work, we will therefore attempt to unify tracking interacting objects and re-identification in one global optimization so that they can benefit each other. We will also seek to replace POM by an occlusion-resistant people detector.

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 network-flow 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. Our tracklet-based implementation preserves the optimality and yields real-time tracking performance.

We demonstrated our method on real-world sequences that feature people getting in and out of cars, carrying and dropping luggages, and passing the ball during professional-level team sports. Our method yields a significant improvement over the state-of-the-art multi-object trackers.

References

[1] A. Mittal and L. Davis, “M2Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene,” *Int. J. Comput. Vision*, vol. 51(3), pp. 189–203, 2003.

[2] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, “Multi-Camera People Tracking with a Probabilistic Occupancy Map,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 267–282, February 2008.

[3] H. Pirsiavash, D. Ramanan, and C. Fowlkes, “Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects,” in *CVPR*, June 2011.

[4] H. Jiang, S. Fels, and J. Little, “A Linear Programming Approach for Multiple Object Tracking,” in *CVPR*, 2007.

[5] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua, “Multiple Object Tracking Using K-Shortest Paths Optimization,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 11, pp. 1806–1819, September 2011.

[6] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, “You’ll Never Walk Alone: Modeling Social Behavior for Multi-Target Tracking,” in *ICCV*, 2009.

[7] B. Yang and R. Nevatia, “Multi-Target Tracking by Online Learning of Non-Linear Motion Patterns and Robust Appearance Models,” in *CVPR*, 2012.

[8] Gurobi, “Gurobi Optimizer,” 2012, http://www.gurobi.com/.

[9] T. Baumgartner, D. Mitzel, and B. Leibe, “Tracking People and Their Objects,” in *CVPR*, 2013, pp. 3658–3665.

[10] X. Wang, E. Turetken, F. Fleuret, and P. Fua, “Tracking Interacting Objects Optimally Using Integer Programming,” in *ECCV*, September 2014.

[11] M. Isard and A. Blake, “Condensation - Conditional Density Propagation for Visual Tracking,” *Int. J. Comput. Vision*, vol. 29, no. 1, pp. 5–28, August 1998.

[12] J. Giebel, D. Gavrila, and C. Schnorr, “A Bayesian Framework for Multi-Cue 3D Object Tracking,” in *ECCV*, 2004.

[13] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, “Online Multi-Person Tracking-By-Detection from a Single Uncalibrated Camera,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010.

[14] J. Kwon, H. S. Lee, F. C. Park, and K. M. Lee, “A Geometric Particle Filter for Template-Based Visual Tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, pp. 625–643, 2014.

[15] S. Avidan, “Support Vector Tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, pp. 1064–1072, 2004.

[16] Q. Bai, Z. Wu, S. Sclaroff, M. Betke, and C. Monnier, “Randomized Ensemble Tracking,” in *ICCV*, 2013, pp. 2040–2047.

[17] H. Grabner, C. Leistner, and H. Bischof, “Semi-Supervised On-Line Boosting for Robust Tracking,” in *ECCV*, 2008, pp. 234–247.

[18] K. Zhang, L. Zhang, and M. H. Yang, “Fast Compressive Tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, pp. 2002–2015, 2014.

[19] Z. Kalal, J. Matas, and K. Mikolajczyk, “P-N Learning: Bootstrapping Binary Classifiers from Unlabeled Data by Structural Constraints,” in *CVPR*, 2010.

[20] J. Fan, X. Shen, and Y. Wu, “Scribble Tracker: A Matting-Based Approach for Robust Tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, pp. 1633–1644, 2012.

[21] S. Hong, S. Kwak, and B. Han, “Orderless Tracking through Model-Averaged Posterior Estimation,” in *ICCV*, 2013, pp. 2296–2303.

[22] J. Lafferty, A. Mccallum, and F. Pereira, “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data,” in *ICML*, 2001, pp. 282–289.

[23] B. Yang and R. Nevatia, “An Online Learned CRF Model for Multi-Target Tracking,” in *CVPR*, 2012.

[24] A. Milan, K. Schindler, and S. Roth, “Detection- And Trajectory-Level Exclusion in Multiple Object Tracking,” in *CVPR*, 2013, pp. 3682–3689.

[25] J. S. Yedidia, W. T. Freeman, and Y. Weiss, “Generalized Belief Propagation,” in *NIPS*, 2000, pp. 689–695.

[26] W. Choi and S. Savarese, “A Unified Framework for Multi-Target Tracking and Collective Activity Recognition,” in *ECCV*, 2012.

[27] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[28] A. V. Segal and I. Reid, “Latent Data Association: Bayesian Model Selection for Multi-Target Tracking,” in *ICCV*, 2013, pp. 2904–2911.

[29] R. Ahuja, T. Magnanti, and J. Orlin, *Network flows: theory, algorithms, and applications*. Prentice-Hall, 1993.

[30] A. Dehghan, Y. Tian, P. Torr, and M. Shah, "Target Identity-Aware Network Flow for Online Multiple Target Tracking," in *CVPR*, 2015, pp. 1146–1154.

[31] H. BenShitrit, J. Berclaz, F. Fleuret, and P. Fua, "Multi-Commodity Network Flow for Tracking Multiple People," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 8, pp. 1614–1627, 2014.

[32] S. Tang, B. Andres, M. Andriluka, and B. Schiele, "Subgraph Decomposition for Multi-Target Tracking," in *CVPR*, 2015, pp. 5033–5041.

[33] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah, "Part-Based Multiple-Person Tracking with Partial Occlusion Handling," in *CVPR*, 2012.

[34] B. Benfold and I. Reid, "Stable Multi-Target Tracking in Real-Time Surveillance Video," in *CVPR*, 2011.

[35] A. Milan, L. Leal-taixe, K. Schindler, and I. Reid, "Joint Tracking and Segmentation of Multiple Targets," in *CVPR*, 2015.

[36] S. Chen, A. Fern, and S. Todorovic, "Multi-Object Tracking via Constrained Sequential Labeling," in *CVPR*, 2014.

[37] K. Fragkiadaki, W. Zhang, G. Zhang, and J. Shi, "Two-Granularity Tracking: Mediating Trajectory and Detection Graphs for Tracking Under Occlusions," in *ECCV*, 2012.

[38] S. Kwak, M. Cho, I. Laptev, J. Ponce, and C. Schmid, "Unsupervised Object Discovery and Tracking in Video Collections," *arXiv*, 2015.

[39] M. Andriluka, S. Roth, and B. Schiele, "People-Tracking-By-Detection and People-Detection-By-Tracking," in *CVPR*, June 2008.

[40] L. Wen, D. Du, Z. Lei, S. Z. Li, and M. Yang, "JOTS: Joint Online Tracking and Segmentation," in *CVPR*, 2015.

[41] A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and H. Wensheng, "Multi-Object Tracking through Simultaneous Long Occlusions and Split-Merge Conditions," in *CVPR*, 2006.

[42] L. Wen, W. Li, J. Yan, Z. Lei, D. Yi, and S. Z. Li, "Multiple Target Tracking Based on Undirected Hierarchical Relation Hypergraph," in *CVPR*, 2014.

[43] C. Huang, B. Wu, and R. Nevatia, "Robust Object Tracking by Hierarchical Association of Detection Responses," in *ECCV*, 2008, pp. 788–801.

[44] J. Sullivan and S. Carlsson, "Tracking and Labelling of Interacting Multiple Targets," in *ECCV*, 2006.

[45] P. Nillius, J. Sullivan, and S. Carlsson, "Multi-Target Tracking - Linking Identities Using Bayesian Network Inference," in *CVPR*, 2006, pp. 2187–2194.

[46] Z. Qin and C. Shelton, "Improving Multi-Target Tracking via Social Grouping," in *CVPR*, 2012.

[47] W. Brendel, M. Amer, and S. Todorovic, "Multiobject Tracking as Maximum Weight Independent Set," in *CVPR*, 2011.

[48] A. Milan, S. Roth, and K. Schindler, "Continuous Energy Minimization for Multitarget Tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, pp. 58–72, 2014.

[49] R. Collins and P. Carr, "Hybrid Stochastic / Deterministic Optimization for Tracking Sports Players and Pedestrians," in *ECCV*, 2014.

[50] A. Andriyenko and K. Schindler, "Globally Optimal Multi-Target Tracking on a Hexagonal Lattice," in *ECCV*, 2010, pp. 466–479.

[51] A. R. Zamir, A. Dehghan, and M. Shah, "Gmcp-Tracker: Global Multi-Object Tracking Using Generalized Minimum Clique Graphs," in *ECCV*, 2012, pp. 343–356.

[52] A. A. Butt and R. T. Collins, "Multi-Target Tracking by Lagrangian Relaxation to Min-Cost Network Flow," in *CVPR*, 2013, pp. 1846–1853.

[53] V. Chari, S. Lacoste-julien, I. Laptev, and J. Sivic, "On Pairwise Costs for Network Flow Multi-Object Tracking," in *CVPR*, 2015.

[54] C. Arora and A. Globerson, "Higher Order Matching for Consistent Multiple Target Tracking," in *ICCV*, 2013, pp. 177–184.

[55] L. Zhang, Y. Li, and R. Nevatia, "Global Data Association for Multi-Object Tracking Using Network Flows," in *CVPR*, 2008.

[56] S. Rujikietgumjorn and R. T. Collins, "Optimized Pedestrian Detection for Multiple and Occluded People," in *CVPR*, 2013, pp. 3690–3697.

[57] C. Wojek, S. Walk, S. Roth, and B. Schiele, "Monocular 3D Scene Understanding with Explicit Occlusion Reasoning," in *CVPR*, 2011.

[58] H. Possegger, T. Mauthner, P. M. Roth, and H. Bischof, "Occlusion Geodesics for Online Multi-Object Tracking," in *CVPR*, 2014, pp. 1306–1313.

[59] S. Tang, M. Andriluka, A. Milan, K. Schindler, S. Roth, and B. Schiele, "Learning People Detectors for Tracking in Crowded Scenes," in *ICCV*, 2013, pp. 1049–1056.

[60] A. Alahi, V. Ramanathan, and L. Fei-Fei, "Socially-Aware Large-Scale Crowd Forecasting," in *CVPR*, 2014.

[61] M. Rodriguez, I. Laptev, J. Sivic, and J. Audibert, "Density-Aware Person Detection and Tracking in Crowds," in *ICCV*, 2011, pp. 2423–2430.

[62] J. F. Henriques, R.Caseiro, and J. Batista, "Globally Optimal Solution to Multi-Object Tracking with Merged Measurements," in *ICCV*, 2011.

[63] P. Lucey, A. Bialkowski, P. Carr, S. Morgan, I. Matthews, and Y. Sheikh, "Representing and Discovering Adversarial Team Behaviors Using Player Roles," in *CVPR*, 2013.

[64] J. Liu, P. Carr, R. T. Collins, and Y. Liu, "Tracking Sports Players with Context-Conditioned Motion Models," in *CVPR*, 2013, pp. 1830–1837.

[65] J. Liu and Y. Liu, "Multi-Target Tracking of Time-Varying Spatial Patterns," in *CVPR*, 2010, pp. 1839–1846.

[66] G. Forney, "The Viterbi Algorithm," in *Proceedings of IEEE*, March 1973, pp. 268–278.

[67] PETS, "Performance Evaluation of Tracking and Surveillance," 2009, http://www.cvg.rdg.ac.uk/slides/pets.html.

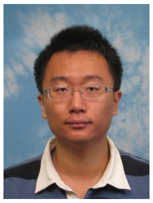
[68] T. D'Orazio, M. Leo, N. Mosca, P. Spagnolo, and P. L. Mazzeo, "A Semi-Automatic System for Ground Truth Generation of Soccer Video Sequences," in *International Conference on Advanced Video and Signal Based Surveillance*, 2009, pp. 559–564.

[69] L. Leal-taixe, M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese, "Learning an Image-Based Motion Context for Multiple People Tracking," in *CVPR*, 2014.

[70] L. Leal-taixe, A. Milan, I. Reid, S. Roth, and K. Schindler, "Motchallenge 2015: Towards a Benchmark for Multi-Target Tracking," in *arXiv*, 2015.

[71] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, 2010.

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



Xinchao Wang received a Ph.D. in Computer Vision in 2015 from EPFL, and a B.Sc. in Computing in 2010 from the Hong Kong Polytechnic University. His research interests include computer vision, applied machine learning and combinatorial optimization.

Engin Türetken received his Ph.D. in Computer Science in 2013 from EPFL. He is currently a postdoctoral researcher at CSEM in Neuchâtel. His research interests include computer vision, microscopic image analysis, graph theory, and combinatorial optimization.

François Fleuret received a Ph.D. in probability from the University of Paris VI in 2000, and the habilitation degree in Applied Mathematics from the University of Paris XIII in 2006. He is the head of the Computer Vision and Learning group at IDIAP, Switzerland. Prior to that, he held positions at the University of Chicago, at INRIA, France, and at EPFL, Switzerland. He is an Associate Editor of the IEEE journal Transactions for Pattern Analysis and Machine Intelligence.

Pascal Fua is a Professor of Computer Science at EPFL. His research interests include shape modeling and motion recovery from images, analysis of microscopy images, and Augmented Reality. He is an IEEE Fellow and has been an Associate Editor of the IEEE journal Transactions for Pattern Analysis and Machine Intelligence.

Tracking Interacting Objects Using Intertwined Flows - Appendices -

Xinchao Wang, Engin Türetken, François Fleuret, and Pascal Fua, *Fellow, IEEE*

In the following appendices, we first provide the complete derivation of the tracklet graph as discussed in § 4.2. Then we show the comparative results obtained on the full graph and on the pruned one. Next, we describe our approach to estimating the probabilities of occupancy. Finally, we discuss why we used this approach rather than the popular Deformable Part Model (DPM) [1].

Appendix A Complete Derivation of the Tracklet Graph

We provide here the procedure to construct the tracklet graph as described in § 4.2 using the notations in our main paper and the ones below.

$\mathcal{T}$	the set of all spatial trajectories obtained from § 4.1
$\hat{v}_l$	the spatial vertex representing an object at location l
$\mathcal{S}$	the set of all joint spatial vertices
$\mathcal{K}$	the set of all non-joint spatial tracklets
$c_\rho(\gamma_q^i)$	the cost of the pose tracklet γ_q^i of a containee object
$c_\beta(\gamma_q^i)$	the cost of the pose tracklet γ_q^i of a container object
O_q	the total number of poses on the spatial tracklet τ_q
v'_k	a vertex in G'
e'_{jk}	an edge in G' that connects v'_j and v'_k
V'	the set of all v'_k
E'	the set of all e'_{jk}
$l'(k)$	the spatial location of v'_k
$N'(k)$	the neighborhood of v'_k on G', i.e., the set of all vertices that can be reached from v'_k on G'

TABLE 1
Notations.

A.1 Grouping Spatial Vertices into Tracklets

The graph size reduction step as described in § 4.1 produces a set of trajectories $\mathcal{T}$. Each trajectory $\tau_q \in \mathcal{T}$ is a set of spatial vertices

$$\tau_q = \{\hat{v}_m, ..., \hat{v}_n\}, \quad \hat{v}_l \in \hat{V} \ , \tag{1}$$

where $\hat{v}_l$ denotes the spatial vertex corresponding to location l, $\hat{V}$ denotes the set of all such spatial vertices in the pruned graph. We use $\mathcal{N}_s(l)$ and $\mathcal{N}_s(\hat{v}_l)$ interchangeably to denote the spatial neighborhood of $\hat{v}_l$. We define the spatial neighborhood of τ_q as the union of all spatial vertices that are reachable from a vertex in τ_q, that is,

$$\mathcal{N}_s(\tau_q) = \bigcup_{\hat{v}_l \in \tau_q} \mathcal{N}_s(l) \ . \tag{2}$$

We take the joint spatial vertices $\mathcal{S}$ to be the spatial vertices included in more than one trajectory as well as those at the beginning or end of a trajectory:

$$\begin{aligned} \mathcal{S} = \{\hat{v}_l \in \hat{V} : \sum_{\tau_q \in \mathcal{T}} \mathbb{1}(\hat{v}_l \in \mathcal{N}_s(\tau_q)) > 1 \ \vee \\ \exists \tau_q, \quad \hat{v}_l \in \tau_q \wedge T_s(l) \in \{T_q, t_q\} \ , \end{aligned} \tag{3}$$

where t_q and T_q denote the first and last time instant of τ_q respectively, and $T_s(l)$ denote the temporal span of location l. Note that $\mathcal{S}$ comprises the only spatial vertices where an interaction event or identity switch can occur. Based on these joint vertices, we partition each trajectory τ_q into multiple segments. We write

$$\begin{aligned} \tau_q = \bigcup_j \tau_q^j \ \cup \ \bigcup_l \hat{v}_l, \ \text{ s.t. } \ \forall_j \ \tau_q^j \cap \mathcal{S} = \emptyset \ \wedge \ \forall_l \ \hat{v}_l \in \mathcal{S} \ \wedge \\ \forall_{j,i} \ \tau_q^j \cap \tau_q^i = \emptyset \ . \end{aligned} \tag{4}$$

In other words, we split each trajectory into two subsets, the joint vertices and the non-joint tracklets. Let $\mathcal{K}$ be the set of all non-joint tracklets. We write

$$\mathcal{K} = \{\tau_q^j \subset \tau_q, \ : \tau_q^j \cap \mathcal{S} = \emptyset\} \ . \tag{5}$$

The construction of $\mathcal{S}$ and $\mathcal{K}$ ensures that for each non-joint tracklet $\tau_q \in \mathcal{K}$, there must be *one and only one* joint spatial vertex in the neighborhood of τ_q at both time $t_q - 1$ and $T_q + 1$. In other words, each non-joint tracklet must have one unique predecessor spatial vertex and successor spatial vertex. Let $\hat{v}^t(\tau_q)$ and $\hat{v}^T(\tau_q)$ respectively denote the predecessor vertex and the successor vertex. We write

$$\begin{aligned} \hat{v}_m \in \mathcal{N}_s(\hat{v}^t(\tau_q)) \quad & \text{s.t.} \quad \hat{v}_m \in \tau_q \ \wedge \ T_s(m) = t_q, \\ \hat{v}^T(\tau_q) \in \mathcal{N}_s(\hat{v}_n) \quad & \text{s.t.} \quad \hat{v}_n \in \tau_q \ \wedge \ T_s(n) = T_q. \end{aligned} \tag{6}$$

A.2 Computing the Poses of the Tracklets

Each non-joint spatial tracklet $\tau_q \in \mathcal{K}$ can be collapsed into a single spatial vertex. We take a pose of such vertices to be a *pose tracklet* γ_q^i, which is a set of temporally-ordered pose vertices in τ_q. We write

$$\gamma_q^i = \{v_j, ..., v_k\}, \tag{7}$$

where v_k denotes a pose vertex. Let $c_\rho(\gamma_q^i)$ and $c_\beta(\gamma_q^i)$ denote the cost of pose tracklet for containee and container respectively, we take them to be

$$\begin{aligned} c_\rho(\gamma_q^i) &= \sum_{k: v_k \in \gamma_q^i} - \log\left(\frac{\rho_k}{1-\rho_k}\right) , \\ c_\beta(\gamma_q^i) &= \sum_{k: v_k \in \gamma_q^i} - \log\left(\frac{\beta_k}{1-\beta_k}\right) . \end{aligned} \tag{8}$$

Since each spatial tracklet τ_q is treated as a single spatial vertex by the MIP solver, the pose of τ_q is uniquely defined by the starting pose at its first time instant and the ending pose at its last time instant. That means among all the pose tracklets of τ_q who share the same starting and ending poses, only the one with the lowest cost can be potentially selected by the MIP solver. As a results, we can remove some pose tracklets with high costs while preserving the optimality. We achieve this using dynamic programming. More specifically, for each pair of starting pose vertex v_j and ending pose vertex v_k, we run Viterbi algorithm to find the pose tracklet with the lowest cost and keep it as the only pose tracklet that links the two pose vertices. For each pair j and k, we compute

$$\gamma_q^i = \mathop{\arg\min}_{\substack{i':\ v_j \in \gamma_q^{i'}, T_p(j)=t_q \\ v_k \in \gamma_q^{i'}, T_p(k)=T_q}} c(\gamma_q^{i'}) \ . \tag{9}$$

We omit the subscript of c as this holds for both the containee and container objects. We compute the pose tracklets for all combinations of starting-ending pose pairs, and we obtain a set of pose tracklets for each τ_q, which we denote as γ_q:

$$\gamma_q = \{\gamma_q^1, ..., \gamma_q^{O_q}\} \tag{10}$$

where O_q denotes the number of kept pose tracklets on τ_q. Since we compute pose tracklet for each starting-ending pose pairs, we have $O_q \leq O^2$.

A.3 Constructing the Tracklet Graph

So far we have defined the poses for the joint spatial vertices and non-joint spatial tracklets. We now construct a new graph G', by treating a pose tracklet as a single vertex in the graph. Let v'_k and e'_{jk} to denote a vertex and an edge of G' respectively, where the subscripts k and j denote the global indices of pose vertices in G'. We take $l'(k)$ to be spatial location of v'_k. We also define $\mathcal{N}'(k)$ to be neighborhood of vertex v'_k in graph G'.

We take the vertex set V' to be the union of all the pose vertices in the joint locations and all the pose tracklets. We write

$$V' = \left\{v_k : \hat{v}_{l'(k)} \in \mathcal{S}\right\} \cup \left\{\gamma_q^i \in \gamma_q : \tau_q \in \mathcal{K}\right\} \ . \tag{11}$$

We take E' to be the set of all edges in G'. We partition E' into two parts, E'_s for the edges only between joint vertices and E'_n for the edges between joint vertices and non-joint tracklets. Note that, the construction of set $\mathcal{S}$ precludes any edges between two non-joint tracklets. We write

$$E' = E'_s \cup E'_n \tag{12}$$

where E'_s is

$$E'_s = \left\{e'_{jk} : k \in \ \mathcal{N}'(j) \ \wedge \ \hat{v}_{l'(j)} \in \mathcal{S} \ \wedge \ \hat{v}_{l'(k)} \in \mathcal{S}\right\} \ . \tag{13}$$

To construct the set E'_n, we consider a spatial tracklet τ_q together with its predecessor spatial vertex $\hat{v}^t(\tau_q)$ and successor spatial vertex $\hat{v}^T(\tau_q)$. These two vertices are the *only* ones in the neighborhood of τ_q. Since an interaction event can never occur on τ_q, its incoming and outgoing flows must be conserved. In other words, if τ_q is selected by the MIP solver, these two vertices must also be selected. We can therefore treat the three vertices as a whole and remove some redundant edges that link $\hat{v}^t(\tau_q)$ to τ_q and τ_q to $\hat{v}^T(\tau_q)$, without loss of optimality. More specifically, for each pair of starting pose vertex v'_j at $\hat{v}^t(\tau_q)$ and ending pose vertex v'_k at $\hat{v}^T(\tau_q)$, we compute the edge pair that preserves the lowest cost. Equivalently, we seek a vertex v'_i with the lowest cost that is in the neighborhood of both v'_j and v'_k, and only keep the edge pair $\left(e'_{ji}, e'_{ik}\right)$ in the tracklet graph G'. We compute such edge pairs for all combinations of starting and ending poses, and define E'_n to be the set of all such edges. We write:

$$\begin{aligned} E'_n = \{e'_{ji} \cup e'_{ik} : i = \mathop{\arg\min}_{i':i' \in \mathcal{N}'(j), k \in \mathcal{N}'(i')} c(v'_{i'}) \ \wedge \\ \hat{v}_{l'(j)} \in \mathcal{S} \ \wedge \ \hat{v}_{l'(k)} \in \mathcal{S} \ \wedge \ \hat{v}_{l'(i)} \notin \mathcal{S}\} \end{aligned} \tag{14}$$

We have now completed the construction of our new tracklet graph $G'(V', E')$.

Appendix B Full Graph vs. Pruned Graph

The full graph contains all the possible states of all objects and thus results in a huge number of variables in the MIP model. For example, on a 10-frame sequence of PETS06 whose monitored area is relatively small, the number of variables reaches about 12 million and the number of constrains reaches about 4.5 million. With the graph size reduction step as described in § 4.1, we reduce the number of variables and constraints to a few thousands.

To compare the results obtained on the full graph with that on the pruned one, we use a 750-frame PETS2006 sequence featuring people dropping a bag. Due to the huge size of the full graph, we run optimization on both the full graph and the pruned graph in batches of 10 frames with 20% overlap. Then we use Hungarian algorithm to glue the results into full trajectories. We show the comparison of MOTA scores in Fig. 1. As can be seen, the result on the pruned graph is very similar to that on the full one. Notably, we found that the optimization on the full graph takes up to 50GB memory and up to 10 hours on a *single* batch of 10 frames, while the **TIF** approach only uses less than 10MB memory and 0.02 second.

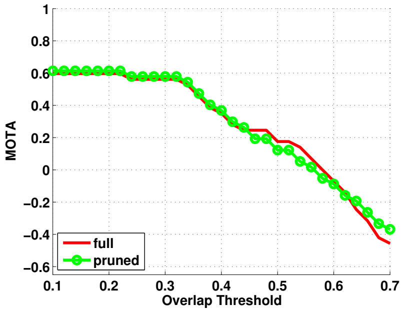


Fig. 1. Comparison of MOTA scores obtained by running MIP on the full graph and the pruned graph.

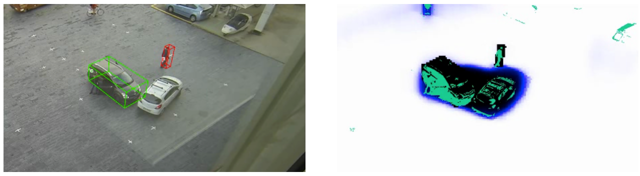
On the whole 3020-frame PETS2006 sequence that features 25 pedestrians and 1 dropped backpack in a crowded scene, **TIF** consumes less than 6.5GB memory.

Appendix C Probabilities of Occupancy

Our approach to computing the image-based probabilities of presence ρ_k^t and β_k^t that appear in Eq. 4 and Eq. 5 is an extension of the one proposed in [2]. 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 pedestrians as cylinders and projects them into the images to create synthetic ideal images that we would observe if the pedestrians 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.

C.1 Oriented Objects



(a) (b)

Fig. 2. 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 in our introduction section of the main paper. It remains undetected because the background subtraction algorithm fails to extract it.

To handle oriented objects such as cars or bags, we extend [2] by introducing simple wireframe models to represent them, as shown in Fig. 2. 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 thus we cannot use the integral image trick of the publicly available software anymore [2]. We therefore use an "integral line" variant, which is comparably efficient. More specifically, we compute the integral of the image values only along the horizontal axis, and at detection time, we 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. 2 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 in the car.

C.2 Objects off the Ground Plane

In [2], 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 game we show in the result section, when the ball is in the air it is also in front of the spectators, making the background non-constant and the results of [2] 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. 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 presence of people and ball given the images, which means we can still multiply the probabilities as required for the derivation of Eq. 2.

Appendix D POM vs. DPM

As mentioned in § 5.2, we could have used a Deformable Part Model (DPM) algorithm [1] instead of POM to compute the probabilities of occupancy. However, DPMs operate in the image plane and do not explicitly take orientation into account. Since we need probabilities expressed in the ground plane, a non-trivial conversion from one to the other would be required. Furthermore, the DPM approach does not explicitly account for mutual occlusions between target objects, while the POM one does [2]. In Tab. 3, we show the detection results of POM and DPM with different threshold settings. As can be seen, when a background subtraction algorithm can be used to provide useful information, the POM approach performs better in terms of MODA scores.

In our future work, we will investigate on 3D object detectors that explore richer image features. For example, if we are to use DPM for this purpose, we will require the probability conversion between ground plane and image plane to account for factors like orientation and mutual occlusion. In particular, the ground-plane probability should be normalized to a proper scale for the MIP solver to produce reasonable results.

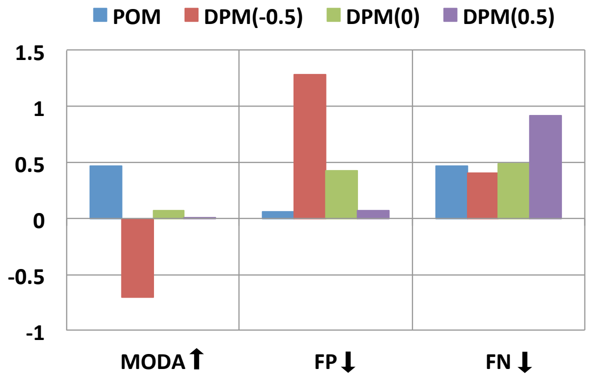


Fig. 3. Detection results of POM and DPM with different threshold settings (-0.5, 0, 0.5). A higher MODA score indicates more accurate detection; a lower False Positive rate (FP) or False Negative rate (FN) indates more accurate detection.

References

[1] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[2] 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*, vol. 30, no. 2, pp. 267–282, February 2008.