Tracking Interacting Objects in Image Sequences

THÈSE N^{O} 6632 (2015)

PRÉSENTÉE LE 3 JUILLET 2015

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE DE VISION PAR ORDINATEUR

PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Xinchao WANG

acceptée sur proposition du jury:

Prof. W. Gerstner, président du jury
Prof. P. Fua, directeur de thèse
Prof. J. Sullivan, rapporteuse
Prof. S. Roth, rapporteur
Prof. P. Dillenbourg, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2015

To my beloved ones and a better myself

Acknowledgements

This thesis concludes my research in the Computer Vision Lab under the supervision of Prof. Pascal Fua. First and foremost, I would like to express my deepest gratitude to Pascal for his professional guidance, numerous discussions and making me a member of the lab, where I've met so many smart and hard-working colleagues. Pascal taught me the principles of solid research: simplicity and elegancy, and showed me that research should be conducted in pursuit of aesthetics. He also demonstrated me how to express my thoughts clearly in academic writing. This thesis would not exist without his effort.

I would like to express my sincere thanks to my thesis jury members, Prof. Wulfram Gerstner, Prof. Josephine Sullivan, Prof. Stefan Roth and Prof. Pierre Dillenbourg for kindly accepting to evaluate this work and providing constructive comments.

During my doctoral studies, I had the opportunity to collaborate with a number of great people who shared their experience and smart ideas with me and contributed to this thesis. I own my sincere thanks to Dr. Francois Fleuret for sharing his expertise on object detection and insightful comments during his regular Wednesday-morning visits. My special thanks to Dr. Engin Turetken for sharing his knowledge on research and career, and for those unforgettable days and nights before the conference deadlines. To Dr. Horesh Ben Shitrit for helping me understand the people tracking pipeline which lays the foundation of this thesis, and for traveling together to Munich and Prague.

I am fortunate to be a member of the Computer Vision Lab and I own my thanks to every member. To our secretary Josiane Gisclon for taking care of everyone of us and being the problem solver in the lab. To Prof. Vincent Lepetit for the constructive comments for our paper submissions. To Alberto Crivellaro for always being ready to help me in French, including the translation of the abstract of this thesis, and being someone I can talk to in office when I feel down. To Carlos Becker for sharing his knowledge in machine learning with me. To Dr. Anne Jorstad for being a patient audience and her valuable comments on presentation skills. To Dr. Yannick Verdie and Dr. Kwang Moo Yi for proofreading my thesis draft. To Timur Bagautdinov for the interesting discussions on the problems that we both encounter. To Bugra Tekin for the

discussions on pose estimation. To Artem Rozantsev, Amos Sironi and my friend Francisco Paulo in Brazil, for the beers and laughter we had together. To Xiaolu Sun for being a good friend ever since my very first day in Lausanne and all the coffees, dinners, hikes and chats. To Bin Fan for being a great friend and all our discussions on research and life.

I would like to take this opportunity to thank our project collaborators for their support and their input for our research. To our industry partner Swisstiming, Neovision and Honeywell for providing the valuable data for our research. Thanks to the CENTAUR project, I spent two months in Prague, where I met colleagues from both industry and academia: Dr. Vit Libal and Dr. Pavel Vacha from Honeywell, Prof. Tomas Pajdla and Prof. Vaclav Hlavac from CTU and Petr Palatka, Ondrej Fisar and Radek Svoboda from Neovision. I owned my thanks to all of them for the interesting discussions and for showing me around in Prague.

My friends, Runwei Zhang, Zhou Xue, Jingge Zhu, Ye Pu, Bailin Deng, Hanjie Pan, Tian Guo, Hao Zhuang, Xifan Tang, Jian Zhang, Juyong Zhang, Yun Tang, Wenqi You, Feng Yang, Hu Xu, Zichong Chen, Jiaqing Du, Xiaowen Dong, Li Pu, Danni Le, Yuxuan Ji, Yun Bai, Yu Chen, Na Li and many more I cannot list here, thank you for making my life in Lausanne colorful.

My special thanks to Prof. Dacheng Tao, for all the endless phone calls across ten time zones and his generous answers to whatever research and career related questions. To Dr. Zhu Li for bringing the fresh air of America and always sharing his valuable resources with me.

I dedicate this thesis to my beloved family members. My infinite gratitude goes to my parents for their boundless, priceless and unconditional love, and to my soulmate and love, Ashley, for her dedication, personal sacrifice and being part of myself.

Lausanne, June 2015 Xinchao Wang

Abstract

Object tracking in image sequences is a key challenge in computer vision. Its goal is to follow objects that move or evolve over time while preserving the identity of each object. However, most existing approaches focus on one class of objects and model only very simple interactions, such as the fact that different objects do not occupy the same spatial location at a given time instance. They ignore that objects may interact in more complex ways. For example, in a parking lot, a person may get in a car and become invisible in the scene.

In this thesis, we focus on tracking interacting objects in image sequences. We show that by exploiting the relationship between different objects, we can achieve more reliable tracking results. We explore a wide range of applications, such as tracking players and the ball in team sports, tracking cars and people in a parking lot and tracking dividing cells in biomedical imagery.

We start by tracking the ball in team sports, which is a very challenging task because the ball is often occluded by the players. We propose a sequential approach that tracks the players first, and then tracks the ball by deciding which player, if any, is in possession of the ball at any given time. This is very different from standard approaches that first attempt to track the ball and only then to assign possession. We show that our method substantially increases performance when applied to long basketball and soccer sequences.

We then focus on simultaneously tracking interacting objects. We achieve this by formulating the tracking problem as a network-flow Mixed Integer Program, and expressing the fact that one object can appear or disappear at locations of another in terms of linear flow constraints. We demonstrate our method on scenes involving cars and passengers, 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.

We finally extend our approach to dividing cells in biomedical imagery. In this case, cells

interact by overlapping with each other and giving birth to daughter cells. We propose a novel approach to automatically detecting and tracking cell populations in time-lapse images. Unlike earlier approaches that rely on linking a predetermined and potentially incomplete set of detections, we generate an overcomplete set of competing detection hypotheses. We then perform detection and tracking simultaneously by solving an integer program to find the optimal and consistent subset. This eliminates the need for heuristics to handle missed detections due to occlusions and complex morphology. We demonstrate the effectiveness of our approach on a range of challenging image sequences consisting of clumped cells and show that it outperforms the state-of-the-art techniques.

Keywords: tracking, detection, interaction, multi-object tracking, network flow programming, mixed integer programming, linear programming, dynamic programming

Résumé

Le tracking d'objets est une tache fondamentale dans le domaine de la Computer Vision. Son but est d'identifier et de tracker sur une séquence d'images des objets qui bougent ou évolvuent dans les temps. La plupart des méthodes existantes peuvent gèrer seulement une classe d'objets et des intéractions très simples, comme par exemple le fait que, à un instant donné, différents objets ne peuvent pas occuper la meme position. Telles méthodes ignorent le fait que les objets peuvent intéragir de façon complexe. Par exemple, une personne peut sortir ou rentrer dans une voiture dans un parking.

L'objet de ce travail de thèse est le tracking d'objets qui intéragissent durant un certain laps de temps dans une video. Nous démonterons dans ce travail de these qu'il est possible d'obtenir des résultats plus stables si on exploite les rélations entre objets. Plusieurs applications sont explorées, tel que le tracking des joueurs et de la balle dans des sports d'équipes, le tracking de personnes et de voitures dans un parking, et le tracking de cellules qui se divisent dans des séquences d'imageries médicales.

Nous aborderons tout d'abord le tracking d'une balle dans des sports d'équipes : il s'agit d'une tache extrememement délicate, puisque la balle est cachée par les joueurs la plupart du temps. Nous proposerons une approche séquentielle, où le tracking des joueurs est d'abord effectué, suivit par le calcul de la position de la balle a traver l'estimation de quel joueur la possède. Notre approche differe des approches standards, qui approche le probleme en commencant par tracker de la balle et ensuite estimer qui en est en possession. Nos résultats montrent que notre méthode améliore sensiblement les performances du tracking appliqué à des longues séquences d'images issues de matchs de football et de basket.

Ensuite, nous considèrerons le tracking simultané d'objets intéragissants entre eux. Nous formulerons le tracking comme un problème de calcul de flux optimal avec contraintes ("network-flow Mixed Integer Program") , et nous modéliserons le fait qu'un objet peut apparaitre ou disparaitre sur la séquence en tant que contraintes linéaires. Nous démontrerons l'efficacité de notre méthode sur des séquences avec des voitures et des passagers, des sacs deposés et recoltés par des individus, et des balles passées d'un joueur à l'autre dans des

sports d'équipes. En particulier, nous montrons qu'en estimant de façon globale et simultanée les trajectoires de différents types d'objets, il est possible de déduire la présence sur la scène d'objets qui ne peuvent etre détectés en se basant seulement sur l'appérance de l'image.

Finalement, nous présenterons une extension de notre methode pour la détections de cellules durant leur processus de division dans des séquences d'imagerie médicale. Pour cette application, les cellules intéragissent en se superposent pour créer une nouvelle cellule. Nous proposerons une nouvelle approche pour la détection et le tracking automatique de populations de cellules dans des séquences d'images en time-lapse. Nous génererons un ensemble overcomplete d'hypothèses de détections. Ensuite, nous efecturons les détections et le tracking simultanément en resolvant un problème de integer programming, afin de trouver un ensemble consistant et optimal d'objets. Nous démontrerons l'efficacité de notre approche sur de nombreuses séquences des cellules groupées entre elles, en améliorant sensiblement l'état de l'art.

Mots-clés: suivi, détection, interaction, suivi multi-objets, calcul de flux optimal avec contraintes, optimisation linéaire en nombres entiers, optimisation linéaire, programmation dynamique

Contents

List of Figures

List of Tables

1 Introduction

Multi-object tracking is a crucial task of Computer Vision. It has numerous applications, ranging from surveillance to medical imaging analysis. However, it is a challenging task for various reasons. For example, the target objects may occlude one another and thus become difficult to distinguish. It becomes even more challenging if the target objects interact, as the interactions introduce additional ambiguity and complexity. Despite the significant progress in recent years, multi-object trackers remain error-prone. Especially when compared to human ability, the performance of state-of-the-art trackers is much lower.

Most existing trackers focus on one type of object like people or cars, and assume the objects interact in very simple ways, such as different instances not being able to occupy the same spatial location. This assumption severely limits the power of the trackers and therefore when applied to scenarios where objects interact in complex ways, most trackers are prone to errors, such as false negatives, drifts and identity switches.

In this thesis, we attempt to overcome the above limitations of existing trackers and develop robust algorithms for tracking interacting objects. We achieve this by exploiting the relationships between different objects and by explicitly incorporating their interactions in the tracking formulation. We explore a broad range of applications, such as tracking players and the ball in team sports, tracking cars and passengers in a parking lot, and tracking dividing cells that overlap or give birth to others.

We show that by explicitly modeling the interactions between the targets, we can significantly improve tracking results. For example, when tracking the ball in team sports, we exploit the contextual information and track the ball by deciding which player is in possession of the ball. This produces more accurate tracking results as we show in Fig. 1.2. When dealing with

passengers and cars in a parking lot, we track both of them jointly by imposing the constraints that a person can only disappear when he or she enters a car. This approach allows us to correctly track the car despite the detection failure shown in Fig. 1.3. When tracking dividing cells in biomedical imagery, we generate multiple sets of conflicting hypotheses to account for mutual occlusion between the cells, and conduct detection and tracking jointly by solving a network flow program. As shown in Fig. 1.4, this approach yields better results than the state-of-the-art approaches that completely decouple detection and tracking.

In the remainder of this chapter, we start with the definition of tracking interacting objects, and then provide a number of its potential applications. We then discuss the technical challenges, and present the contributions of this thesis. We finally give an outline of the remaining chapters.

1.1 Problem Definition

In this section, we describe the problem we address in this thesis. We first define several important terminologies used throughout this thesis, and then provide a categorization of the general vision-based tracking problem. Based on the categorization, we finally describe the focus of this thesis.

1.1.1 Terminology

In what follows, we define some important terminologies that we use throughout the thesis.

- **Detection**: the operation that determines the state of each object instance of a specific category at a given time, if any are present. Here, we take "object" to mean either a concrete one like a car, or an event like a person entering a car. We take "state" to be either the physical condition of an concrete object, like the location of a car, or the condition of an event, like two people being in a car. For example, in this thesis, the task of detecting a car comprises finding the location and the orientation of a car; the task of detecting a cell division event comprises deciding whether the cell is about to divide into multiple daughter cells or not.
- **Tracking**: the operation that determines the states of objects over time while preserving their identities. Therefore, tracking can be seen as linking detections of different time instances into trajectories. For example, in this thesis, the task of tracking a car comprises finding its location and orientation at each time instance. If the goal is to track more

than one target, it is important to preserve the identity of each target. The error that, the identities of two targets are incorrectly flipped, is known as *Identity Switch*.

- **Interacting**: the process that multiple objects come together and have effects on one another [1]. Interactions between objects are pervasive, ranging from neurons whose sizes are measured in microns, through humans whose sizes are measured in meters, to galaxies whose sizes are measured in thousands of light years. In our daily scenarios, for example, basketball players interact with the ball by throwing or catching it, and passengers interact with a car by getting in or out of it.
- **Tracking Interacting Objects**: the operation that determines the states of multiple objects that interact over time while preserving objects' identities. Tracking interacting objects is an example of the general tracking problem, but it introduces additional tasks: tracking the interactions. For example, the task of tracking players and the basketball comprises finding the locations of the players and the ball as well as reporting the ball possession in each time instance.

The goal of this thesis is to study the problem of tracking interacting objects. Before we summarize our studies in Sec. 1.1.3, we provide a categorization of the general vision-based tracking problem in Sec. 1.1.2,

1.1.2 Categorization of the General Tracking Problem

Tracking is one of the fundamental problems in Computer Vision, and it has been extensively studied by researchers from many aspects for decades. To provide the readers a clear positioning of the focus of this thesis, we categorize the general tracking problem based on four different criteria, and then position this thesis based on the categorization.

Based on the number of targets, the tracking problem can be classified into two categories:

- **Tracking Single Object**
  Single-object tracking aims at tracking a pre-designated target in the scene [2, 3, 4]. If the background is static, meaning that the target is the only moving object in the scene, the tracking problem is relatively easy. Otherwise if the background is dynamic, the problem becomes more challenging due to the additional noise from the background.
- **Tracking Multiple Objects**
  Compared to single-object tracking, multi-object tracking is more difficult because the

targets may move simultaneously in the scene [5, 6, 7]. In addition, the number of targets is usually unknown *a priori*. The tracker will therefore have to infer the correct number of objects and their corresponding trajectories. **Tracking interacting objects** is an example of more general multi-object tracking, but with additional challenges introduced by the interaction between the objects.

Based on the number of cameras used for tracking, the tracking problem can be classified as follows:

- **Tracking with Single Camera**
  This is also known as *monocular* tracking [8, 9, 10]. The camera calibration parameters may or may not be given. If the calibration parameters are not given, tracking is usually conducted on the image space. As a result, the trajectories of the objects are only available on the image plane but not in 3D space. Otherwise, the calibration can be utilized to compute the 3D location of the target objects.
- **Tracking with Multiple Cameras**
  The camera calibration parameters are usually given under this setup, so the 3D location of the target objects can be computed [11, 12, 13]. Compared to the monocular setup, the multi-camera setup can give a more precise localization of the objects in the 3D space. In addition, the multi-camera setup can benefit occlusion reasoning because of the image evidences obtained from cameras at different angles.

Based on the physical state of the camera, the tracking problem can be classified as follows:

- **Tracking with Steady Camera**
  The cameras are physically stable, meaning that all camera parameters remain unchanged during the video capturing process [12, 13, 14].
- **Tracking with Moving Camera**
  The cameras are physically mobile, meaning that the calibration parameters change. As a result, the parameters have to be re-estimated frame by frame if they are to be applied for 3D localization [15, 16, 17].

To tackle the object tracking problems, modern approaches can be broadly classified into two categories: tracking by model evolution and tracking by detection. We briefly introduce them here and provide a more detailed discussion in Sec. 4.1 and Sec. 5.1.

- **Tracking by Model Evolution**
  The tracking problem is formulated with a Bayes filter, whose goal is to estimate an unknown probability density function recursively over time using the given observations [18]. Kalman Filter [19] and Particle Filter [2] are the two representative approaches in this category. Because of the recursive nature, when dealing with multiple targets, these approaches are often prone to errors that are difficult to recover from [20].

- **Tracking by Detection**
  To track the targets, detectors are first applied on each frame to detect the targets. Afterwards, tracking is modeled as a data association problem, whose goal is to link the detections over time into trajectories. The data association is usually formulated as a global optimization, which accounts for the detections over a temporal span [13, 21, 22]. As a result, when dealing with multiple targets, tracking by detection is more robust than tracking by model evolution [20].

1.1.3 Thesis Focus

In this thesis, we focus on **tracking interacting objects** captured by **one** or **multiple steady cameras**. If input images are captured by multiple cameras, we require the cameras to be synchronized and share overlapping fields of view. We aim at developing robust and efficient algorithms that track interacting objects. Our methods fall into the category of **tracking by detection**.

Object interactions are pervasive at all scales and as a result, studying them are vital in many applications. However, most existing tracking approaches model only one class of objects and account for only very simple interactions, such as the fact that objects may repel each other to avoid bumping into each other. They do not account for more complex interactions between the targets. Therefore, when the targets interact, existing trackers do not have mechanisms to deal with the interactions and report them. However, in many scenarios the statistics of the interactions are valuable and of interest to a large audience. For example, basketball fans are more interested in which player is in possession of the ball and for how long, than where the ball is by itself. Furthermore, since existing trackers do not model the interactions, when they occur, the tracking results are prone to errors like false positives and identity switches.

In contrast to most existing approaches, in this thesis, we explicitly incorporate the relationships between different objects in our tracking formulation. As we will show in the following chapters, by doing so we achieve consistently better results compared to state-of-the-art

Figure 1.1 – Our tracking results on different scenarios where objects interact. **Top row** (left to right): (a) Tracking basketball players and the ball, (b) Tracking soccer players and the ball. **Bottom row** (left to right): (c) Tracking pedestrians and their luggage, (d) Tracking cars and passengers, (e) Tracking dividing cells. The identities of the tracked objects are encoded in different colors. In (a) to (d), objects interact in a way that, one class of bigger objects (i.e., containers) can "contain" the other class of smaller objects (i.e., containees). For example, players "contain" the ball, and cars "contain" passengers. We show the container objects in circles and containee ones in dots. The dots inside circles indicate the "invisible" containee objects contained by the containers. In (e), we show the tracking results on dividing cells, where the red dot in the cell centers indicate the ground truths.

trackers. More importantly, we provide mechanisms to track the interactions. For example, when tracking players and the ball in team sports like soccer, we also track the ball possession in each frame; when tracking passengers and cars, we detect the event that a person gets in or gets out of a car, and track the number of passengers inside a car; when tracking cells in microscopy images, we also report a division event when it occurs. We output trajectories of both the targets and the interactions, which allow us to further analyze the behaviors of targets and make predictions.

In this thesis we study several interacting scenarios as shown in Fig. 1.1. These scenarios are representatives of the more general ones. For example, the basketball game is a representative scenario where objects of one type (i.e., the ball) can be heavily occluded by objects of other types (i.e., players); the passenger-car scenario is a representative scenario where objects of one type (i.e., passengers) may be fully enclosed by objects of other types (i.e., cars). Therefore, our methods can be applied directly or extended to much broader range of applications.

1.2 Applications

There are numerous scenarios where objects interact with each other and a robust tracker is needed, ranging from modeling neurons measured in Electron Microscopy images [23], through understanding collaborative learning in a classroom in our daily life [24, 25], to studying trajectories of stars [26, 27].

Based on the obtained trajectories, it is possible to analyze the behaviors of the targets and make predictions. In this section we briefly introduce some applications of tracking interacting objects.

1.2.1 Surveillance and Security

In recent years, there have been a significant amount of resources spent on surveillance [28, 29]. For example, between 2007 and 2011, authorities spent £515 million on installing and maintaining closed-circuit television (CCTV) cameras in UK [30]. However, most existing surveillance systems are passively monitored. In other words they rely on human monitoring [31]. In the case of railway stations, experienced operators need to watch the live videos captured by the surveillance cameras so as to assess crime or pre-crime situations. As a result, active surveillance systems are highly desired. The ultimate goal of such systems is to monitor the area of interest and detect or alert the authorities about abnormal events in real time.

There are many surveillance applications that demand automatic tracking of interacting objects. In a parking lot, a surveillance system that tracks people and cars can detect potential car thefts [32, 33]. In front of an ATM machine, a surveillance system that tracks people and the environment can detect robbery and report to the police [34]. On a busy street, a surveillance system that tracks the cars and pedestrians can detect accidents and potentially save lives [9, 35]. In crowded public places like metro stations and airports, a surveillance system can detect abnormal situations, such as a person abandoning a bag, to prevent potential crime or terrorism [36, 37, 38].

Surveillance systems can also benefit hospitalized patients or assisted living residents [39, 40]. One important feature of such a system is the automatic fall detection, which by itself comprises tracking people and their surroundings. If the monitored person falls, such system generates real-time staff notification. This further enables root-cause analysis of patient falls to prevent potential falls or decrease fall risk [41].

1.2.2 Robotics

A visual tracker is a key component of a robotic system. In the recent effort to develop autonomous cars, visual tracking systems are being integrated to avoid pedestrian collisions [42, 43]. Such systems track pedestrians in real time and predict their future movements to execute automatic braking. In this case, incorporating the interactions between the pedestrians can improve the tracking accuracy [44]. The research of pedestrian tracking is nowadays supported by major car manufacturers [45, 46, 47].

A robust tracker is also crucial for medical robots. In robot-assisted surgery, tracking the instruments and their interactions with the environment may augment the clinician's experience, improve positioning accuracy and ensure minimally invasive surgery [48, 49]. Tracking a patient's interaction with the surroundings helps rehabilitation robotics to better recognize the patient's movement intention to assist therapeutic training [50, 51].

1.2.3 Sports Analytics

Automatic tracking in sports is of tremendous importance to athletes, referees, coaches and fans. In the case of individual sports such as cycling, skiing and running, tracking can be achieved by using accelerometers or Global Positioning System (GPS) sensors, which provide a log of statistics [52]. Nowadays, these sensors are also embedded in cell phones. However, in the case of professional team sports such as basketball and soccer, installing such sensors is usually forbidden. The scouting therefore has to be conducted manually by human experts, which is time and effort consuming.

Vision-based tracking systems provide one solution to this problem [53, 54, 14, 55]. An ideal vision-based tracking system captures the game using cameras and tracks the players, the ball and their interactions. The obtained trajectories can be further used to compute game statistics, which are not only important for player-evaluation purposes in the game, but also in the training stage to help improve player performance [56].

1.2.4 Customer Analytics

A vision-based tracking system can benefit customer behavior analysis. In shopping malls and supermarkets, customers interact with the environment such as goods. A tracking system that collects trajectories of the customers and their interactions with goods, can generate valuable statistics for data mining purposes to increase sales. For example, the trajectories can

be used to cluster customer behaviors into groups and predict future interactions [57]. With such knowledge, shops can optimize their product allocation to assist customers and provide recommendations to them [58, 59].

1.2.5 Biomedical Analytics

Tracking is important for biomedical analytics. Characterizing migration of cells and their interactions with the surrounding environment is crucial to understanding cell behaviors and their implications in normal tissue development and different kinds of diseases, such as cardiac infarction and immunological disorders [60, 61]. Understanding the morphology and behavior of neurons can potentially lead to future treatments to various pathological and neurodegenerative conditions using adult neural stem cells [62, 63].

With the advent of modern acquisition technologies that produce huge amounts of image streams, it is impractical for human experts to label and track all of them manually. As a result, automated vision-based tracking systems are increasingly in demand. To attract Computer Vision researchers' interests in biomedical tracking and to speed up the process, a number of tracking competitions have recently been organized [64, 65].

1.3 Technical Challenges

As an example of the more general multi-object tracking, tracking interacting objects inherits all the challenges of multi-object tracking and brings additional ones.

We categorize the challenges of tracking interacting objects into three classes: data acquisition, physical states and interactions of the targets, as well as model complexity. The first challenge, data acquisition, is inherent to any vision-based tracking task. The second challenge, physical states and interactions of the targets, concerns all multi-object tracking problems, and especially if the objects interact in complex way. To model the possible states of the targets, a large number of variables and constraints are needed, which leads to the third challenge, model complexity. In what follows, we discuss these challenges in more details.

1.3.1 Data Acquisition

The first task of a video-based tracking system is to acquire data, and this itself is a challenge. The cameras should be placed at proper locations, focused and zoomed at the correct distance with appropriate exposure. Cameras at improper locations that cover only small fractions

of the monitored area will severely limit the power of any tracker, whereas those that are not focused or incorrectly zoomed will produce blurred images and downgrade the tracking performance.

In the case of natural scenes, lighting condition also plays an important role, because it influences the appearance of objects. Especially in outdoor scenarios, when the daylight changes, the color and dynamics of the background will be affected accordingly, adding additional complexity to the tracking.

In the case of biomedical imagery, data acquisition may also give rise to challenges. For example, the image acquisition process may generate noise such as speckles that appear very similar to cells. In addition, in 2D microscope imagery, cells of different depths may clump and overlap with each other, making them very difficult to distinguish even for human observers.

1.3.2 Physical States and Interactions of the Targets

Physical states of the targets such as appearance or movement create challenges for tracking. For example, the appearance of multiple targets may be similar and thus it is difficult to tell them apart; the targets may move fast and their movements may be difficult to predict; objects may occlude one another on the image plane, and as a result the image evidence becomes ambiguous and the objects are difficult to distinguish.

The tracking task becomes even more challenging if the targets interact, as the interactions introduce additional complexity: tracking the interactions. In this case, the tracker should not only output the trajectories of the targets but also report interaction in the case of occurrence.

Tracking players and the ball in team sports is a typical example of tracking interacting objects, in which case their interaction is the ball possession. It is a very challenging task, because the ball is often hidden by the players and follows an unpredictable trajectory when it is passed or taken from one player to another. Furthermore, as we show in Fig. 1.2, the amount of image evidence for the ball, even when it is visible, is dwarfed by that of the players.

The same goes for tracking cars and pedestrians in a parking lot, in which case the tracker should not only track all the objects but also report the events that a person gets in or gets out of a car. This task inherits the challenges of general object tracking. For example, as shown in Fig. 1.3, the appearance of the targets may be similar to that of the background, which makes it difficult for the detector to detect the target and further leads to the failure in tracking.

Apart from the aforementioned difficulties, cell tracking in biomedical imagery introduces additional ones that, the cells can either divide into multiple daughter cells or wither away and disappear in mid-sequence. Therefore, it is important for a cell tracker to report such events in the case of occurrence. In addition, complex morphology of cell populations makes it very difficult to tell them apart in a single frame, as shown in Fig. 1.4. Only by considering the image sequence as a whole can a tracker or even a human observer tell the correct configuration. Recently, a number of cell tracking competitions have been organized [64, 65]. These competitions have shown that state-of-the-art methods are still error-prone due to occlusions, imaging noise, and complex cell morphology.

1.3.3 Model Complexity

To formulate the problem of multi-object tracking, a large amount of variables are needed to account for the possible states of all the target objects and their possible transitions. For example, in network-flow tracking algorithms such as K-Shortest Path (KSP) [13], in order to deal with missed detections, all possible spatial locations of the targets are modeled in each time instance together with all possible transitions to the next time instance. This results in a huge state space. In addition, in order to generate physically-plausible tracking results, constraints on the variables are usually needed. They can be imposed as soft ones that are encoded as part of the objective function [66, 22], or as hard ones that form the feasible sets [13, 14]. This also brings additional complexity to tracking.

To deal with interacting objects, apart from the aforementioned complexities, additional variables and constraints are needed to model the interactions. For example, in the case of basketball tracking, one additional class of variables is needed to model the ball possession, in other words which player is holding the ball at a given time instance; additional constraints, such as the fact that there can be at most one ball in the basketball court during the game, also need to be imposed.

1.4 Thesis Contribution

The main goals of this thesis are to study tracking interacting objects by explicitly modeling their interactions and to develop practical and efficient solutions. We explore a wide range of applications, such as tracking players and the ball in team sports, tracking cars and pedestrians in a parking lot and tracking cells in biomedical imagery.

We summarize our contributions as follows:

- **A sequential approach to tracking ball in team sports by focusing on team play**
  We propose a sequential approach that tracks players first, and then tracks the ball by deciding which player is holding the ball. We show that, tracking players benefits tracking the ball (**Chapter 3**).
- **A framework to track two classes of objects simultaneously using intertwined flows**
  We propose a framework to simultaneously track two classes of objects, one of which can "contain" the other, using network flow programming. We show that, by tracking all objects simultaneously, the evidence of objects from one class may help tracking objects from the other class. For example, in our framework, tracking passengers help tracking cars, and in turn, tracking cars also help tracking passengers (**Chapter 4**).
- **An approach to tracking dividing cells using network flows**
  We propose a robust approach to tracking dividing cells by casting the global optimization as a network flow program on an over-complete graph of conflicting detection hypotheses. We show that, our approach yields consistently better results on challenging sequences than state-of-the-art cell trackers (**Chapter 5**).

In the remainder of this section, we describe our three contributions in more details.

1.4.1 Tracking Ball by Focusing on Team Play

Accurate tracking of players and the ball in sports is of tremendous importance. However it is a very challenging task as we have discussed in Sec. 1.3.

In such scenarios, frame-to-frame tracking is extremely unreliable. For example, even a state-of-the-art algorithm [67] that has been shown to be superior to many other state-of-the-art trackers for single object tracking and whose code is publicly available never tracks the ball for more than five consecutive frames in the sequences we present. Modern tracking-by-detection approaches that re-detect the object as often as necessary and aggregate results over several frames increase robustness but can still easily fail if the object is hard to detect in individual frames.

As our first contribution, in Chapter 3, we introduce a novel ball tracking approach, which works in contrast to common approaches that first track the ball and only then decide which player owns it. In our approach, we first track the players, and then track the ball by deciding

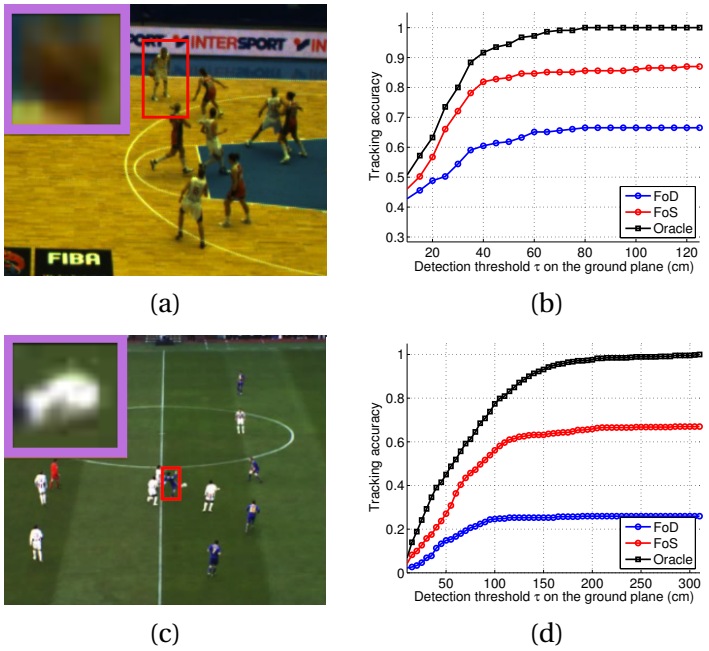
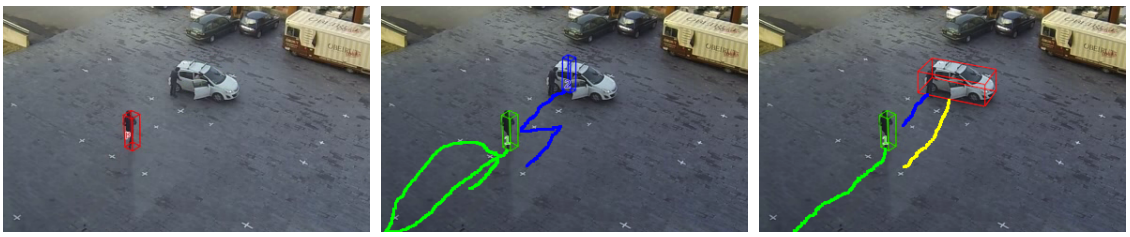


Figure 1.2 – Tracking the ball in team sports, such as basketball (a) and soccer (c) is hard due to the ball's small size, barely 12x12 pixels in the videos we work with as shown in the upper left corner insets. Furthermore, it is subject to prolonged occlusions even when multiple views are available. Our key contribution is to overcome these difficulties by exploiting the contextual relation between the players and the ball, in particular with respect to *ball ownership*. To illustrate this, we plot in (b) and (d) the accuracy we would obtain based on the sole knowledge of ball ownership if it were given to us by an oracle, in this case a person looking at the videos. We also plot the accuracy results of both a state-of-the-art approach that only looks at the ball and our attempt to establish ball ownership without human input. The latter (red FoS curve) decisively outperforms the former (blue FoD curve) and approaches human performance levels (black Oracle curve).

ball possession. We finally use the ball ownership information as a means to achieve reliable ball tracking, which lets us turn unreliable image-based information into dependable trajectories. This is effective because the ball trajectory is intimately linked to that of the players who pass it to each other or steal it from one another. As shown in Fig. 1.2, given perfect knowledge of ball ownership, this simple approach yields a relatively accurate estimate of the ball location and obtaining this knowledge therefore is what must be addressed.

Our tracker yields dependable ball possession and accurate trajectories, which approach the best that can be expected in long sequences with multiple attach phases and timeouts. Furthermore, our formulation is generic and can be potentially extended to other ball games.



(a) POM [11] (b) KSP [13] (c) Ours

Figure 1.3 – Motivation of our approach that tracks two classes of objects simultaneously. (a) Thresholding the detector [11] scores for cars and people produces only one strong detection in this specific frame of a complete video sequence. (b) Linking people detections across frames [13] 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.

1.4.2 Tracking Interacting Objects Using Intertwined Flows

Multi-object tracking is a very challenging task, because the targets may move simultaneously and interact one another. Most tracking 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 [44, 68]. They do not account for more complex interactions between the targets. For example, in a parking lot, people may get in a car and therefore become invisible in the scene. Therefore, when applied to complex scenes where objects interact, most existing trackers are prone to error.

As our second contribution, in Chapter 4, we introduce a Mixed Integer Programming framework that lets us model the more complex relationship between the presence of objects of a certain kind and the appearance or disappearance of objects of another. For example, when tracking people and cars in a parking lot, this enables us to express that people may only appear or disappear either at the edge of the field of view or as they enter or exit cars that have stopped. Similarly, when attempting to check if a bag has been abandoned in a public place, we can express that this can only happen at locations where 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 enforcing the fact that one object can only appear or disappear at locations

where another is can be made possible by imposing linear flow constraints. This results in a Mixed Integer Programming problem, for which the global optimum can be found using standard optimization packages [69]. Since different object types are handled simultaneously, the presence of any one of them can be evidence for the appearance of others. Fig. 1.3 depicts a case where simply thresholding the response of the car detector we use leads to a car being missed by the detector. However, because people are properly detected disappearing at a location in the middle of the parking lot, the algorithm eventually concludes correctly that there must have been a car there which they entered. In this scenario, not only does the presence of a vehicle explain the apparent disappearance of pedestrians, but also the disappearance of pedestrians is evidence of the presence of a vehicle.

Our proposed approach is a mathematically principled and computationally efficient. It accounts for the relationship between flows representing the motions of different object types, especially with regard to their container/containee relationship and appearance/disappearance. Furthermore, our tracklet-based implementation yields near real-time tracking 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 in a ball-game.

1.4.3 Tracking Dividing Cells Using Network Flows

Detecting and tracking cells over time is key to understanding cellular processes including division (mitosis), migration, and death (apoptosis). Modern microscopes produce vast image streams making manual tracking tedious and impractical. High-throughput automated systems are therefore increasingly in demand.

In its generic form, the problem is often formulated as a two-step process that involves first detecting potential objects in individual frames and then linking these detections into complete trajectories. This approach is attractive because spurious detections, such as false positives due to imaging noise and artifacts, can be eliminated by imposing temporal consistency across many frames, which is more difficult to do in recursive approaches.

Many of the most successful algorithms for both people [72, 73, 74, 75] and cell tracking [76, 61, 70] follow this two-step approach by first running an object detector on each frame independently and building a graph whose nodes are the detections and whose edges connect pairs of nodes. These algorithms then find a subgraph that represents object trajectories by considering the whole graph at once. However, to handle missed detections, these methods often rely on heuristic procedures that offer no guarantee of optimality. This is of particular

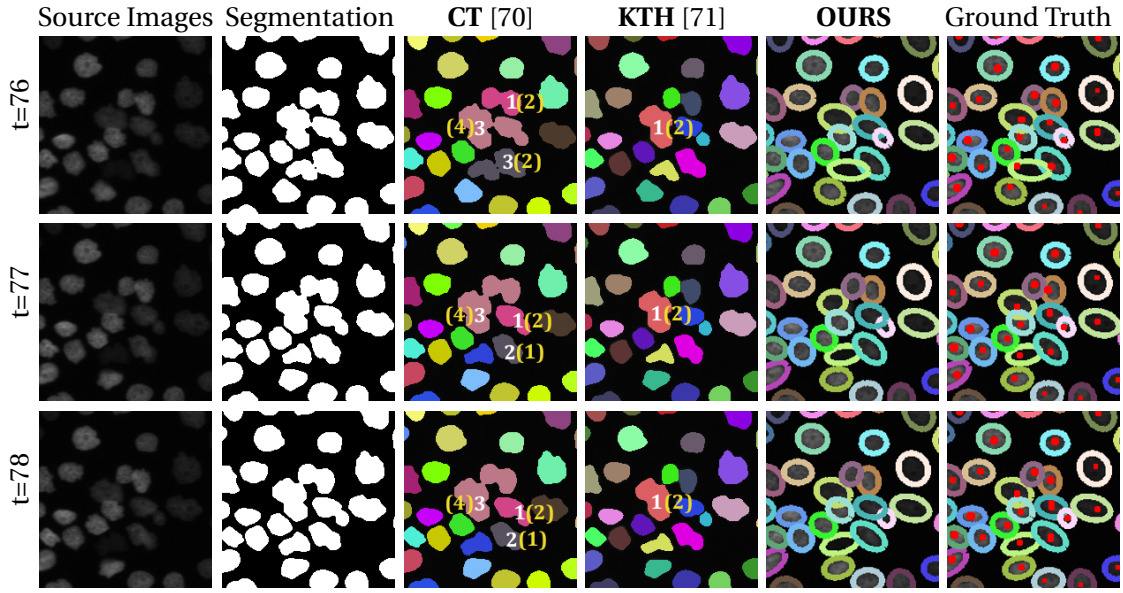


Figure 1.4 – Three images from a typical sequence; the original segmentations produced by a pixel-based classifier; the results of [70](**CT**), [71](**KTH**), and our method (**OURS**); the manually annotated tracking ground truth in red dots and the optimal ellipse-tracks obtained using this ground truth. The track identities are encoded in colors. Our approach correctly tracks the cells in spite of long-term segmentation failures, and produces results that are very similar to the ground truth. In the **KTH** and **CT** columns, the white-colored numbers indicate the tracker-inferred numbers of cells that are contained by the segments beneath them, and the yellow-colored numbers show the ground truth. Best viewed in color.

concern for cell tracking because detections are often unreliable due to the complex morphology of cell populations. For example, in Fig. 1.4, groups of cells that appear clumped together in some frames can only be told apart when considering the sequence as a whole, which current approaches to cell tracking rarely do.

As our third contribution, in Chapter 5, we address the detection and tracking problems simultaneously by casting them as a network flow problem on an over-complete graph of potentially conflicting hypotheses. These hypotheses are competing explanations of the initial segmentations, in which the cells are often under-segmented, meaning that a single foreground region can correspond to many cells. Instead of selecting a single hypothesis for each such region, we generate a hierarchy of spatially overlapping detection hypotheses that accounts for occlusions, which are prevalent in biological data due to insufficient imaging resolution, poor reagent selectivity, or missing depth information in 2D. This is in contrast to most approaches that rely on spatially disjoint superpixels [70, 77], which do not explicitly account for occlusions and may result in missed detections.

Once the competing hypotheses are obtained, the next step involves building a spatio-temporal graph over all of them, and finding the globally optimal explanation by solving a Linear Integer Program (IP) within a small tolerance. This results in large graphs but eliminates the need for using heuristics to handle missed detections, as required by recent approaches [61, 70, 77] that also rely on integer programming. However, unlike these approaches, which involve many variable and constraint types, ours only requires a single set of flow variables and three types of linear constraints that exclude conflicting hypotheses while governing cell division, appearance, and disappearance.

1.5 Outline

The remainder of this thesis is organized as follows.

In Chapter 2, we briefly review two important algorithms that are closely related to this thesis, namely Probability Occupancy Map (POM) [11] and K-Shortest Path (KSP) [13], from which we got our inspiration for this thesis. We then discuss the performance metrics for object tracking evaluation, which we use throughout this thesis.

In Chapter 3, we describe our ball tracking algorithm that focuses on the team play. Our approach first tracks the players and then tracks the ball by deciding which player is in possession of the ball. We will show that this approach yields dependable ball trajectories in long basketball and soccer sequences. The material presented in this section has previously appeared in [78].

In Chapter 4, we describe our tracking algorithm that tracks two classes of objects simultaneously. This is achieved by introducing one type of flow variables for each class of objects, and expressing the fact that one object may appear or disappear at the location of another in terms of linear constraints. We demonstrate this framework in scenes such as people and cars in a parking lot, pedestrians and their luggage in a railway station as well as players and the ball in professional matches. Part of the material in this section has appeared in [79].

In Chapter 5, we extend our network-flow approach to tracking cells in biomedical images. We generate an over-complete set of competing detection hypotheses and then conduct detection and tracking jointly by solving an integer program. We show that this approach yields consistently better results in challenging image sequences. The material presented in this section has appeared in [80].

In Chapter 6, we summarize the contribution of this thesis. We also present some of our

research collaboration, where the algorithms developed in this thesis are applied. Finally, we discuss potential future research directions.

2 Preliminaries

In this chapter, we start by reviewing two important algorithms for multi-object detection and tracking: Probability Occupancy Map (POM) [11] and K-Shortest Path (KSP) [13]. These two algorithms are developed earlier in our lab and they lay the foundation for this thesis. We then present the evaluation metrics for performance measurement, which are used throughout this thesis.

2.1 Related Algorithms

The POM algorithm detects people from calibrated cameras at each time instance. The KSP algorithm takes the detection results of POM as input, and links the detections over time to obtain full trajectories. The POM-KSP workflow falls into the category of tracking by detection as we have discussed in Sec. 1.1, which proves to be effective and is adopted by many state-of-the-art multi-object trackers [14, 22, 81, 7, 82, 83].

However, unlike many other detection and tracking algorithms that apply on the image plane, both POM and KSP algorithms work on the ground plane. To model the spatial locations of the target objects in each frame, the algorithms represent the monitored ground plane as a grid of cells. In the original version of POM, it is assumed that each cell can be occupied by at most one object. In our extension to be discussed in Sec. 4.4, we allow objects of different attributes, like objects of different orientations, to occupy each such spatial cell.

2.1.1 Probability Occupancy Map (POM)

Let **B** denote a set of binary images, where the foreground pixels indicate the potential regions of interest. The binary images can be the results of background subtraction, or more generally, the results of classification. Based on the evidence in the binary images, POM estimates the posterior of an object being present with some specific configuration, for example, the posterior of an object occupying a specific location. The original version of POM is applied for people detection, where each person is modeled as a cylinder and his/her projection is a rectangle in each camera frame. We summarize the original version of the algorithm here and refer the readers to [11] for details.

Let X_i be the random variable denoting state of occupancy at location i at a given time instance, $X_i = 1$ denote the event that the location is occupied, and ρ_i denote the probability of location i being occupied. Given a set of ρ_i, for each camera view, we can obtain a *synthetic average image*, representing the "virtual" scene implied by all the ρ_i. In the middle row of Fig. 2.1, we show an example of a synthetic image for each camera view. These synthetic images are produced based on the estimated ρ_i, shown in the bottom row of Fig. 2.1. The objective of POM is to find a set of ρ_i for each time instance, such that the synthetic average images best approximate the input binary images. POM achieves this by minimizing the Kullback-Leibler divergence between the product law of estimated posterior, and the true conditioned law of X given **B**.

Let $\mathbf{A}_c^1$ and $\mathbf{A}_c^0$ denote the average synthetic images in view c given $X_i = 1$ and $X_i = 0$, respectively. ρ_i is computed as the fixed point of a large system of equations of the form

$$\rho_i = \frac{1}{1 + \exp\left(\lambda_i + \sum_c \psi\left(\mathbf{B}_c, \mathbf{A}_c^1\right) - \psi\left(\mathbf{B}_c, \mathbf{A}_c^0\right)\right)} \ , \tag{2.1}$$

where λ_i is the prior of location i being occupied. ψ computes the dissimilarity between two images and is defined as

$$\psi\left(B, A\right) = \frac{1}{\sigma} \frac{\parallel B \otimes (1 - A) + (1 - B) \otimes A \parallel}{\parallel A \parallel} \ , \tag{2.2}$$

where $\otimes$ denotes the pixel-wise product of two images and σ accounts for the quality of the input binary images. The complete derivation of 2.1 can be found in [11]. Note that, the projections of people are rectangles in each camera. Therefore, the estimation of synthetic images, which is the dominant term, can be done very efficiently using integral image techniques.

To solve the large system of equations, the algorithm iteratively updates each ρ_i until conver-

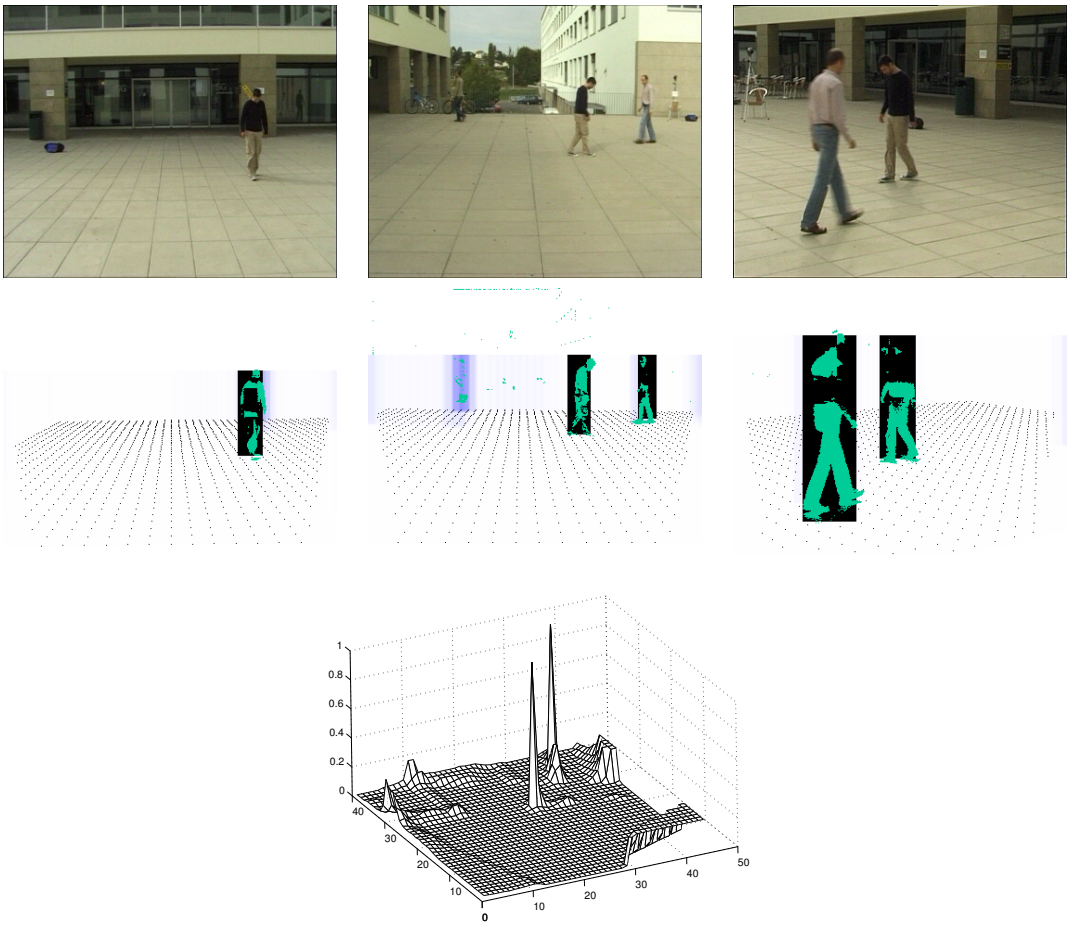


Figure 2.1 – Computing the probability of occupancy from binary input images. Top Row: images captured by synchronized cameras. Middle Row: binary images obtained by background subtraction (shown in green) and synthetic average images obtained by POM after convergence (shown in blue and black). Bottom Row: obtained probabilities of occupancy for all locations. Images are taken from [11].

gence, which usually takes less than 150 iterations in practice. Fig. 2.1 depicts an example input and output of the POM algorithm, where the first row depicts the source images, the second row depicts the synthetic average images overlaid with background subtractions, and the bottom row depicts the obtained ground-plane occupancy probabilities.

2.1.2 K-Shortest Path (KSP)

Given the probability of occupancy, the next goal is to link the detections over time into full trajectories. The KSP algorithm achieves this by modeling the transition of objects using flow variables and solving a constrained global optimization with respect to the flows. We briefly review the KSP algorithm here and refer the readers to [13] for more details.

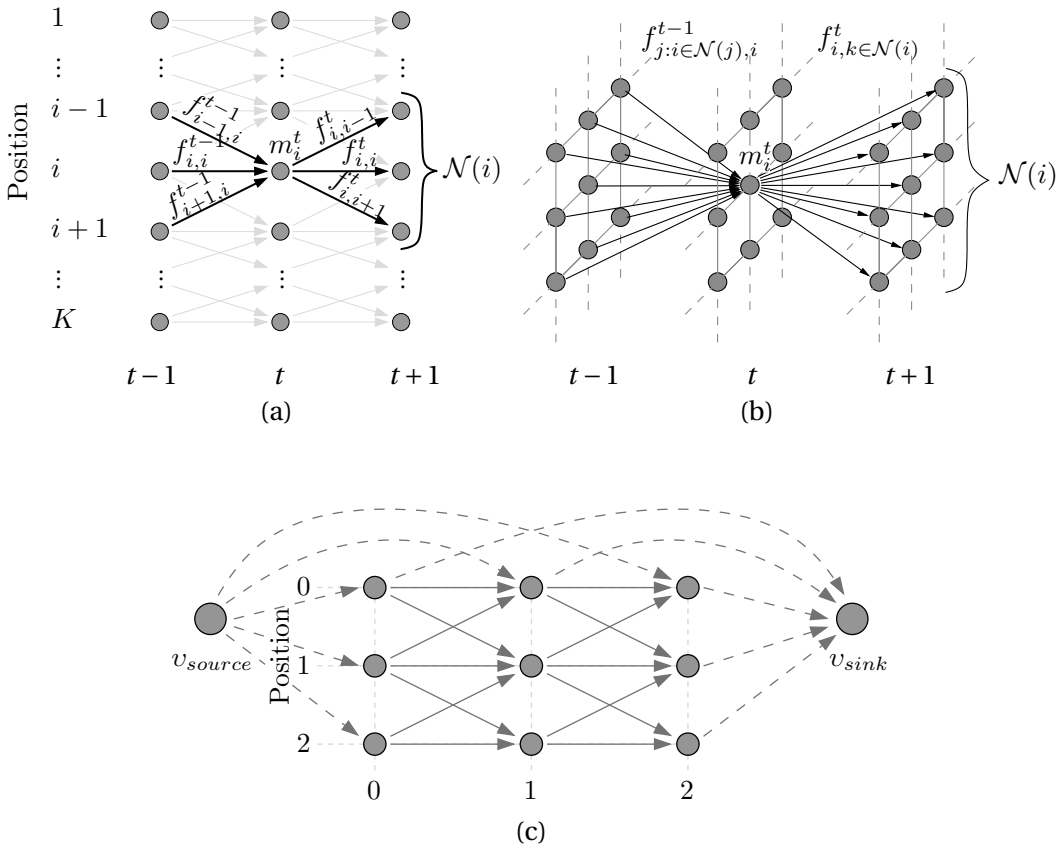


Figure 2.2 – Directed Acyclic Graph (DAG) and flows. We use a circle to denote a position on the ground, and an arrow to denote a flow variable. We show a simplified DAG in (a), where all the positions on the ground are arranged in one dimension, and show in (b) the DAG used for tracking people on 2D grids. We show in (c) a small K-Shortest Path (KSP) graph, where position 0 is a possible entrance and exit point and thus linked to the source and sink node. Images are taken from [13].

Let i index a spatial location on the ground, and K denote the total number of such locations. Also, let $\mathcal{N}(i)$ denote the neighborhood of location i, in other words the locations that an object can reach at time $t+1$ from location i at time t. Note that in this section, the words "location" and "position" are used interchangeably.

The KSP algorithm models the tracking problem using a Directed Acyclic Graph (DAG). To account for all possible trajectories, KSP models each spatial location in each time instance with a vertex in the DAG. This leads to a total of $K \cdot T$ vertices in the DAG. An example of a simplified DAG is depicted in Fig. 2.2(a), where each column of nodes corresponds to all positions of one time instance. In Fig. 2.2(b), we show a 2D visualization of the grids that we use for people tracking.

The edges that connect the vertices, which are shown as arrows, represent allowable transitions. Let $f_{i,j}^t$ denote a binary flow variable from location i at time t to location $j \in \mathcal{N}(i)$ at time $t+1$, and $f_{i,j}^t = 1$ denote such a moving event of an object. The cost of each flow variable is defined based on the associated detection score. To allow an object to enter or to leave the scene, we introduce two virtual vertices, v_{source} and v_{sink}, into the DAG. These two virtual vertices are linked to all the nodes that represent the possible entrances and exits, such as the edges of the monitored area. In addition, the two virtual nodes are also linked to all the nodes in the first and last time instance. Fig. 2.2(c) depicts an example of a complete KSP graph, where the dashed arrows represent the flows from and to the virtual vertices and the solid arrows represent those between the physical locations.

The objective of KSP is to find the globally optimal set of trajectories over the whole temporal span, or equivalently, an optimal set of flow variables that yields the minimal cost. This can be expressed as a linear combination of all the flow variables. However, not all assignments of the flow variables are physically plausible. To remove those impossible configurations, some hard constraints are imposed on the flow variables. For example, to model the fact that different objects cannot occupy the same spatial location at a given time instance, the sum of all incoming flows to a vertex is enforced be less or equal to one. This results in an Integer Programming (IP), which we write as follows

minimize

$$\sum_{t,i} -\log\left(\frac{\rho_i^t}{1-\rho_i^t}\right) \sum_{j\in\mathcal{N}(i)} f_{i,j}^t \tag{2.3}$$

subject to

$$f_{i,j}^t \geq 0 \quad \forall t,i,j, \tag{2.4}$$

$$\sum_{j\in\mathcal{N}(i)} f_{i,j}^t \leq 1 \quad \forall t,i, \tag{2.5}$$

$$\sum_{j\in\mathcal{N}(i)} f_{i,j}^t - \sum_{k:i\in\mathcal{N}(k)} f_{k,i}^{t-1} \leq 0 \quad \forall t,i, \tag{2.6}$$

$$\sum_{j\in\mathcal{N}(v_{source})} f_{v_{source},j} - \sum_{k:v_{sink}\in\mathcal{N}(k)} f_{k,v_{sink}} \leq 0\ . \tag{2.7}$$

The constraint matrix of the above IP has been proven to be *totally unimodular* [13]. That means the integrality constraint on the variables can be relaxed, and the solution of the corresponding Linear Programming (LP) are identical to those of the IP. Standard solvers, such as Gurobi [69], CPLEX [84] and CVX [85], can be applied to solve the LP.

However, solving the relaxed LP does not make use of the specific graph structure and may lead to high complexity. In [13], it is shown that the optimization problem can be reformulated as a *k shortest node-disjoint paths problem* and can be solved very efficiently using the K-Shortest Path algorithm (KSP) [86]. In practice, KSP yields a speed-up factor of up to three orders of magnitude, as compared to solving the general LP directly.

2.2 Evaluation

In this section, we describe the evaluation metrics we use for measuring the tracking performance throughout this thesis. The evaluation of tracking is usually conducted by comparing the tracking results with the ground truth, either in the image plane or the ground plane. We categorize the evaluation into single-object evaluation and multi-object evaluation.

2.2.1 Single-Object Tracking Evaluation

In the case of single-object tracking, we assess the tracker's performance by comparing the tracked object with the ground truth in each frame. This can be done by comparing the overlap

ratio between the tracked object and the ground truth on the image plane, or by computing the distance between them on the ground plane.

We compute the tracking accuracy score (TA), which is defined as

$$\mathrm{TA}(\tau) = \frac{1}{T}\sum_{t=1..T}\mathrm{TP}(t;\tau)\ , \tag{2.8}$$

where τ can be either a Euclidean distance in the world coordinate, or an overlap ratio between the projection of the tracked object and the ground truth segmentation in the image coordinate. We summarize the quality of the tracker as a non-negative score in [0, 1], the higher the score the better. If the output of the tracker is within a distance threshold τ from the annotated ground truth location, or the overlap ratio is larger than a threshold τ, we define the detection as a True Positive $\mathrm{TP}(t;\tau)$ detection. By computing the TA for a range of τ, we obtain a monotonic TA curve, which is also known as the "precision plot" in [87]. With a fixed τ, the True Positive rate over all frames is known as the Success Rate (SR) [67].

To provide a complete summary of tracking performance, we also use the Center Location Error (CLE) metric [87], which is defined as the distance between the tracking result and the ground truth on the ground. A smaller CLE indicates more accurate tracking. The mean of CLE (i.e., MCLE), defined as the average CLE over the whole sequence, is also used in our evaluation.

2.2.2 Multi-Object Tracking Evaluation

The evaluation of multi-object tracking is more complicated as compared to that of single-object tracking, because of the additional data-association complexity. More specifically, a tracker should not only detect the correct locations of the target objects, but also preserve their identities over time. This is achieved by first finding the mapping between the tracking results and the ground truth in each frame, and then computing the false positive and false negative counts. In practice, the mapping can be computed using greedy algorithms or locally-optimal ones such as the Hungarian algorithm.

In this thesis, we use the standard CLEAR [88] metrics for evaluating multi-object detection and tracking. Here, we briefly review the two most important metrics, Multiple Object Detection Accuracy (MODA) and Multiple Object Tracking Accuracy (MOTA), and refer the interested readers to [88, 89, 90] for details.

Multiple Object Detection Accuracy (MODA)

MODA is a metric that assesses the performance of a detector. It penalizes false positive and false negative detections in each frame, and summarizes the overall detection accuracy with a number in the range $(-\infty, 1]$. A higher MODA value indicates higher detection accuracy. As discussed above, in order to compute the false positive and false negative counts, the mapping of the tracked objects to the ground truth must be computed beforehand.

Formally, the MODA score is defined as

$$\text{MODA} = 1 - \frac{\sum_{t=1}^{N} \left(c_m \left(m_t \right) + c_f \left(f_t \right) \right)}{\sum_{t=1}^{N} N_G^t} , \tag{2.9}$$

where N is the total number of frames, N_G^t is the number of ground truths at frame t, m_t and f_t respectively denote the number of false negatives and false positives, and c_m and c_f respectively denote the cost of false negative and false positive. In this thesis, we set an equal importance between the two errors, that is, $c_m = c_f = 1$. Similarly to single-object tracking evaluation, the two errors are defined based on thresholds, which can be the Euclidean distance or overlap ratio between the results and the ground truth segmentations.

Multiple Object Tracking Accuracy (MOTA)

To assess the tracking performance, MOTA accounts for false positives, false negatives and identity switches. The false positive and false negative are defined in the same way as in MODA, and the identity switch is defined to be the number of mismatched identities at time $t+1$ given the mapping at time t. We write

$$\text{MODA} = 1 - \frac{\sum_{t=1}^{N} \left(c_m \left(m_t \right) + c_f \left(f_t \right) + c_s \left(s_t \right) \right)}{\sum_{t=1}^{N} N_G^t} , \tag{2.10}$$

where s_t is the number of identity switches and c_s is the corresponding cost. We treat all the errors equally in this thesis, that is, $c_m = c_f = c_s = 1$.

3 Tracking Ball by Focusing on Team Play

In this chapter, we introduce a novel approach to tracking the ball in team sports. The ball's trajectory and its interaction with the players are very important to fans, coaches and referees. However, ball tracking is a very challenging task due to the ball's small size, unpredictable motion and prolonged occlusions by the players. In contrast to conventional approaches that track the ball first and only then decide ball possession, we first track players and then track the ball by deciding which player is holding the ball.

To this end, we define a novel state space which explicitly accounts for ball ownership, a Conditional Random Field (CRF) [91] model that depends on the ball detections and the players' trajectories, and a practical approach to learning the model parameters from training data. More specifically, we start from video sequences from several synchronized cameras from which the players' trajectories [13, 92] can be reliably extracted using publicly available software [93], and the ballistic trajectory of the ball can be also be extracted in consecutive frames where it is clearly visible. Between such frames, when individual players own the ball, its approximate position can be assumed to be their position up to the reach of their arms or legs. When passed, we take its path in the horizontal plane to be a straight line from on one player to the next. As shown in Fig. 1.2, given perfect knowledge of ball ownership, this simple approach yields a relatively accurate estimate of the ball location and obtaining this knowledge therefore is what must be addressed. We formulate this problem in terms of assigning possession to a player, or to no-one when the ball is being passed, by minimizing a global objective function that takes into account players' positions and image information when the ball is visible.

We will show that this approach produces very reliable ownership assignments and a final accuracy of the ball trajectory that approach the best that can be expected over multiple attack

phases and timeouts. This is in stark contrast to most existing approaches that have only been demonstrated on short sequences.

The rest of this chapter is organized as follows. In Sec. 3.1, we review related ball tracking methods. We present our formulation in Sec. 3.2 and show our implementation in Sec. 3.3. We show comparative tracking results in Sec. 3.4 and conclude the chapter in Sec. 3.5.

3.1 Related Work

This section distinguishes two broad classes of approaches, those whose sole focus is on ball-tracking, and those that exploit spatio-temporal context to track a hard-to-discern target.

3.1.1 Tracking Only the Ball

We first discuss a handful of ball-tracking scenarios where the problem is considered solved, and off-the-shelf commercial solutions exist. Then we discuss the more broad class of problems, which are still beyond the reach of the current state of the art.

Ball Tracking in the Commercial World

Ball tracking has received considerable attention over the years. In some cases such as tennis, reliable commercial software is now available [94]. While the high velocity of the ball poses an engineering challenge, it is brightly colored and rarely occluded. Even then, achieving the required level of reliability and accuracy requires ten high-speed cameras looking at the scene from different angles.

The problem becomes substantially harder in team sports. For example in soccer, the ball can be surrounded by multiple players, who create occlusions in many views. This is probably why the soccer software sold by the company that developed the tennis ball tracking system discussed above [94] only aims at tracking the ball as it is being shot towards the goal, and therefore unoccluded.

In basketball, the problem is even harder because multiple players compete for the possession of the ball, generating even more occlusions. Since the basketball may be guided by hand, its trajectory is also more unpredictable. Lastly, the ball can be of similar color as the players' jerseys, adding yet another level of complexity. Even though some companies advertise the capabilities of post-game basketball trajectory analysis the amount of automation is

unclear [95].

Ball Tracking in the Research World

Ball tracking has also been pursued in the academic world [96, 97] before the advent of the commercial systems. For sports such as tennis or golf, it is now considered a solved problem. The same cannot be said of team sports for the reasons discussed above.

In many team sports, the trajectories of the ball and the players overlap in space-time. Therefore to track the ball one also has to localize the players. However, deriving a unified formulation for tracking the ball and the moving players is challenging. Typical approaches [98, 99] to track the players and the ball independently relied on ad-hoc rules to decide if and when a player handled the ball. The composition and the parameters of these rules were defined manually rather than with respect to a clearly-defined objective function.

The fact that soccer players interact with the ball in a structured manner was exploited in [100]. A set of *ball phases* was defined—*in-possession*, *rolling*, *flying*—and for each such phase a physics-based motion model of the ball was introduced. Once the ball became un-occluded, and its phase was known, the appropriate motion model could be used for tracking. The *in-possession* phase was treated as a means to "initialize other phases", while in our work it is the main focus; thus the two approaches are complementary.

An approach to jointly track the soccer players and the ball was proposed in [101]. First, players were detected in each frame independently. Next, the ball was tracked across frames taking into account player detection. Last, player detections were linked into tracks, while taking into account the location of the soccer ball. Unfortunately, the ball-tracking formulation was not specified, except for a brief mention of an Adaboost classifier for scoring the ball trajectories. The validation was performed on sequences of 150 frames, spanning only a few seconds, making it hard to determine how useful the approach would be in practice.

Summary Ball tracking in team sports in the presence of prolonged occlusions remains an unsolved problem. Ball-tracking approaches designed for individual or one-on-one games [96, 97] only work for long passes and shots on the goal, while approaches that attempt to incorporate player trajectories [102, 101] rely on ad-hoc rules for ball possession. Consequently, none of these approaches work on long video sequences. By contrast, our tracker minimizes a global objective function that not only takes into account image evidence about the ball but also players' location and the phase of the game.

3.1.2 Object-tracking and Context

Oftentimes, a target that needs be detected or tracked is correlated in space and/or in time with its surroundings. We first discuss approaches to model complex individual and group behavior in ball-centric team sports, and then overview approaches that focus on a relation between a target and its local spatio-temporal context.

Exploiting the Context of a Ball-Centric Team Sport

To the best of our knowledge, we are the first to develop an effective approach to exploit the highly complex context of a sports game for ball-tracking. In this section, we mention relevant work on analyzing the high-level state of a ball-centric game without the benefit of automatic ball tracking. We view the relation between the two sets of approaches as complimentary: many of those prior approaches to game analysis rely on a manual labeling of the ball and would therefore benefit from our formulation. Conversely, our end-to-end demonstration system requires an estimate of the phase of the game, which is obtained using the features of [103], and which we describe next.

Finding regions of player convergence We first mention two approaches that rely on players' trajectories to estimate the regions of convergence. In some cases, these ground-plane regions happen to contain the ball.

In [102] the focus was on basketball. It was assumed that in an input video sequence it was known which team was attacking, and that this team was in the vicinity of the opponent's basket. Then, from the velocities of the players on the ground plane, *regions of convergence* were computed. It was assumed that each of these regions would be a candidate for the basketball location, and a Kalman-filter was used to *validate* ball-location candidates over time. Because of the strong assumptions on the content of the input video sequences, the algorithm could only be validated on sequences of up to ten seconds in length. These test sequences are too short to fully understand the algorithm's failure modes and to determine how it could generalize to other basketball games. Furthermore, attempting to localize the basketball without taking into account image evidence cannot yield optimal results.

In [104], velocities of the soccer players in the ground plane were used to compute ground-plane regions where some interesting event could happen in the future. In some cases those regions happened to correspond to the future location of the ball, but in general it was not possible to attribute those regions to known elements of the game. Since the output of the

algorithm could not be used to predict a sport-specific quantity, such as the location of the ball, the authors compared their regions of interests with the field of view selected by the "real camera operators" during professional games.

Recognizing elements of the game from player trajectories A system for analyzing basketball games was described in [103]. The system comprised several components, including tracking, recognizing which team was attacking, which elements of the play were being executed, etc. Notably missing from the scope was tracking the basketball itself, which was explained in the paper by the "state of technology." However, the authors of [103] noted that knowing the ball location would "considerably improve the performance..." of their system.

Systems for analyzing hockey games were described in [105, 106, 107]. Similar to the system of [103], the system of [107] comprised components for player tracking, and the recognition of several elements of the play, e.g., *power play shot*. The location of the hockey puck was essential for their recognition algorithm, but the authors relied on the manual localization of the puck.

The approach to recognizing team-level activity in European handball was proposed in [108]. The set of six manually-specified labels included "slowly going into offense," "offense fast break", etc., and the features were based on players' on the player density function defined over the whole court (obtained by solving a Poisson equation). Notably, the location of the ball was not a part of the feature set, which precluded the recognition of "more complex activity classes."

An approach to recognize elements of a baseball game was presented in [109]. In that approach detections of the players in a video segment were automatically labelled with their roles, e.g., a pitcher, and a causal relations between actions were inferred. Recognition of social roles in field hockey, e.g., attacker, defender was proposed in [110]. The formulation integrated per-player location and role cues into a consistent interpretation via a conditional random field. Inference in such a model was computationally complex due to the combinatorial number of possible assignments, and furthermore a different model needed to be initialized depending on the user's query. The formulation was validated using "ground-truth person locations, as well as those using a simple automated detector"; it is unclear if in the complete system the ground-truth ball locations were utilized or if the ball location was ignored altogether.

Recognition of *plays* in American football was tackled in [111]. Their approach included a *temporal interaction matrix* defined with respect to the trajectories, and a manifold-based

formulation for comparing such interaction matrices. The location of the ball was excluded from the formulation. The experiments were conducted with an assumption of "ground-truth data, where tracks for each player are manually labeled." Such an assumption does not hold true for our problem.

Summary The approaches such as [103, 111, 102, 108, 110, 104] would benefit from automatic ball tracking. An effective formulation for achieving this goal is the main contribution of this chapter.

Exploiting the Context of Objects and Parts

Models that exploit spatial and spatio-temporal context for object detection and tracking have been proposed in [112, 113, 114]. These models require a target to be surrounded by multiple moving objects or image patches. The appearance of these contextual objects should be sufficiently distinct so that they themselves can be reliably detected in novel video frames. To locate the target, each contextual object is employed as a predictor for its center.

In [112], which focused on vehicle tracking in aerial imagery, the model for a motion context captured "...an intuitive observation that the locomotive behavior of an object (e.g. car) provides information about locomotive behaviors of nearby objects (cars)." To identify a motion context among many moving objects, a measure of the chaos in a dynamical system was employed. Given the motion context, the target's location was implemented as a linear regression with respect to the motion of the contextual objects.

In [113] it was assumed that "relative position of feature and target is more or less fixed over short time intervals". The contextual features were derived from frames with "good visibility", and the presence of a feature in a novel frame was probabilistically estimated. Given the contextual features, the prediction of the target's location was implemented as generalized Hough transform.

The context model of [114] was developed for detecting a small body part, such as a hand, in a single image. In this model, the context comprised groups of image features, sequentially ordered into chains. All chains were constrained to originate from a single always-visible body part, such as a person's face.

Summary While the above approaches have demonstrated the importance of context in tracking, they are not applicable to our problem. The initialization assumptions of [113] cannot be satisfied, and the requirement of [114] that an easy-to-detect anchor object is always visible, is inapplicable to our setup. The assumption of [112] and also of [113, 114] that the target's context can be unambiguously re-acquired in a novel video frame does not hold, since at our image resolution body parts of players tend to look alike.

More importantly, the uncertainty in our context is fundamentally different. In our case, the spatio-temporal context is clear from the outset since the ball is always passed among a fixed set of players. The regression function to predict the target from its context that was central to [112, 113], is an identity function since the player who owns the ball and the ball are co-located. The real challenge is in deciding which of contextual elements, i.e., the players, is indeed associated with the ball, and this challenge has not been resolved in prior work.

3.2 Formulation

Our formulation is general and is applicable to ball-centric team sports with a defined notion of ball possession. For the sake of simplicity and concreteness, we use the terminology of the sports where the ball may be passed to a teammate or shot on the goal over the players' heads. These include basketball, rugby, soccer, and many others.

When a ball undergoes ballistic motion in the air, such as during a long-distance pass, the resulting trajectory can, ignoring friction, be approximated by a parabola. This would suggest a second-order motion model for ball-tracking, and indeed, such models have been applied in prior work to track an unoccluded ball.

However, our focus is on a harder problem, that of tracking the ball while it is dribbled, carried, or otherwise *owned* by a player, and also during relatively-short passes in the vicinity of other players. Since a player who owns the ball is likely to make abrupt moves in order to confound the opponents, the second-order model becomes less useful.

In the remainder of this section, we begin by introducing our first-order model in its most generic form and then specialize it so that its individual components can be learned from the limited amount of training data, which is often the case. In Sec. 3.3 we will provide an example of an end-to-end system that can track the ball during long game sequences, and in that system, the second-order motion model will become useful for accurately segmenting long passes.

3.2.1 First Order Model

We use the notation of Table 3.1 to express our model. We represent the state of the ball at time t as $Y_t \in \{1,\ldots,K,\ominus\}$, where K is the number of players. The ball is either *owned* by a player k (*e.g.*, player k is dribbling the ball) or is in state $\ominus$, meaning it is *free* (*e.g.*, the ball is being passed or shot at the goal). For the rest of this chapter, we use the word *owned* and *possessed* interchangeably. Inferring the state of the ball $Y_t = y_t$ at time t enables us to estimate its ground-plane locations $G_t = g_t$ at all times by using a player's location when the ball is owned by that player, and by interpolation when the ball is in-between; this is shown in Fig. 3.1.

Table 3.1 – Notation for our formulation

K	the number of players
$\ominus$	no-one owns the ball; i.e., the ball is *free*
Y	the state of the ball, $Y \in \{1,\ldots,K,\ominus\}$
T	number of frames in a temporal window $t \in [1,\ T]$
y_t	the realization of Y at time t
r_t	auxiliary random variable derived from y_t
G	ground-plane location of the ball on a 2D grid
X	image evidence for all video frames
ψ_1	unary potential of our model
ψ_2	pairwise potential of our model
W	parameters of our model
S	phase of the game, $S \in \{0,1,2\}$
s_t	the realization of S at time t timeout ($s=0$) or one of the two teams attacks
$\rho(\cdot)$	$\rho : k \mapsto \{1,2,3\}$ group membership of actor k: referee ($\rho(k)=3$) or one of the two teams

For a sequence of video frames in the temporal interval $t = 1,\ldots,T$ our goal is to infer the most likely state sequence $(y_1,....,y_T)$ of $Y = (Y_1,....,Y_T)$ given the image evidence $X = (X_1,....,X_T)$. We do so by minimizing a energy function $E(y_1,\ldots,y_T;X,W)$, where W represents a set of learned parameters. We express the energy function E in terms of a Conditional Random Field (CRF) [91] as

$$\sum_{t=1..T} \psi_1(y_t;X,W) + \sum_{t=1..T-1} \psi_2(y_t,y_{t+1};X,W)\ , \tag{3.1}$$

where our image evidence comprises per-view ball detections and the player trajectories.

Given analytical expressions for ψ_1 and ψ_2, as well as appropriate values for W, finding an

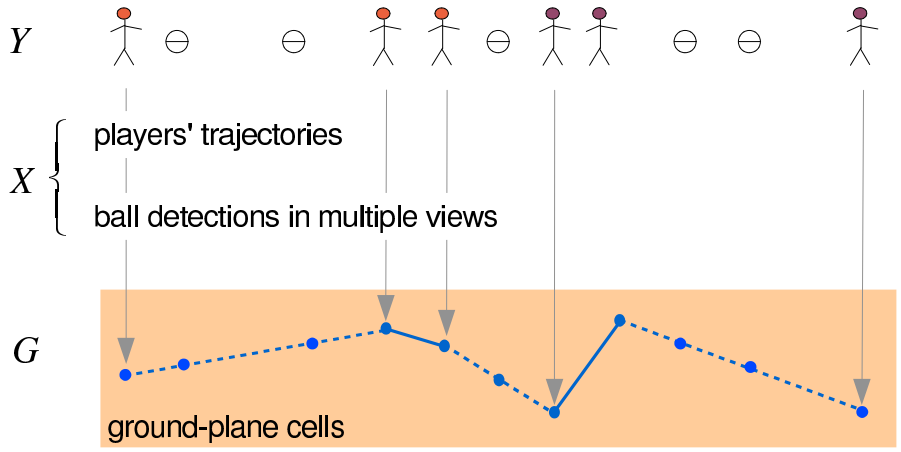


Figure 3.1 – Overview of our ball tracking approach. The ownership of the ball Y is inferred from X; afterwards the ground-plane locations of the ball G are determined either by linking the locations of the corresponding players when $Y \neq \ominus$ (solid lines), and when $Y = \ominus$ by interpolation (dashed lines).

optimal sequence of y's can be accomplished efficiently with dynamic programming. We now turn to deriving and learning these expressions and values.

In our implementation, ψ_1 and ψ_2 take the form of negative log-likelihood:

$$\psi_1(y_t; X, W_1) = -\log p(y_t; X, W_1) \text{ and} \tag{3.2}$$

$$\psi_2(y_t, y_{t+1}; X, W_2) = -\log p(y_{t+1}|y_t; X, W_2), \tag{3.3}$$

where $p(y_t)$ are $p(y_{t+1}|y_t)$ are probability distributions.

A naive learning of the potentials in Eq. 3.2 and Eq. 3.3 would be quite complicated. The reason is that monolithic models for $p(y_t; X, W_1)$ and $p(y_{t+1}|y_t; X, W_2)$ would have to account for all the complexities of a team sport, such as the structure of the game, roles, and skill level. Instead, we factorize these probability densities to arrive at several well-defined learning tasks, as discussed in the following two sections.

3.2.2 Unary Potential

Our unary potential term accounts for the states of the ball at each time instant, namely the ball being with each player or being free. To train a single classifier that estimates the probability of each such state would be complicated, because the classification is a $(K+1)$-class task and the semantic of each player holding the ball is not consistent if the roles of

players are unknown. To tackle this problem, we factorize the unary potential to make the learning task tractable. In this section, we factorize the unary potential in Eq. 3.2, derive the features to evaluate it, and define a method to learn its parameters.

Factorizating the Unary Potential

We first factorize the unary potential with respect to whether or not the ball is free and then with respect to the team that owns the ball. Although in practice, the probability densities derived in the factorizations will all become functions of the image evidence X, our derivations will not require a particular form of X. We will therefore employ a shorthand notation, such as $p(y) \doteq p(y_t; X, W_1)$.

Factorizing ball ownership Frequency and duration of the ball possession tend to be influenced by the players' roles and skill levels. Thus, in theory, the knowledge of a player's identity can be used to design the ball-possession prior. While our formulation does not preclude such a prior, in our current implementation we treat all players as indistinguishable.

When the players' roles are unknown, the states of Y lose their semantics. This influences our supervised learning problem since the label $y = k$ cannot be applied consistently. This is in contrast with, say, learning semantic segmentation, where labels such as $y =$"sky", $y =$"grass" can be defined in a consistent manner.

In principle, one could learn $p(y)$ for $y \in \{1, \ldots, K, \ominus\}$. However, the learning task would have to contend with the fact that the label $y = \ominus$ is semantically different from the remaining K labels, and as noted earlier, the labels $\{1, \ldots, K\}$ have no semantic meaning. We therefore factorize $p(y)$ so that instead of dealing with one complicated learning task we solve several simpler ones.

We introduce an auxiliary random variable $R \in \{0, 1\}$, which depends on Y as follows:

$$r = 1 \Leftrightarrow 1 \leq y \leq K, \quad r = 0 \Leftrightarrow y = \ominus\,. \tag{3.4}$$

Therefore, we can write $p(y = \ominus | r = 1) = 0$, $p(1 \leq y \leq K | r = 0) = 0$, $p(y = \ominus | r = 0) = 1$, and

$$p(y) = \underbrace{p(y|r=1)}_{\text{Eq. 3.6}}\, p(r = 1) + p(y|r = 0)\, p(r = 0)\,, \tag{3.5}$$

where the computation of $p(y|r = 0)$ follows from Eq. 3.4.

Factorizing the phase of the game We now factorize the first term of Eq. 3.5 with respect to the phase of the game. Such a factorization will make it practical to track the ball throughout an entire game period.

Let $s = 0$ denote timeout and $s \in \{1,2\}$ denote which of the two teams is attacking. When $y \neq \ominus$, i.e., $r = 1$ we have:

$$\begin{aligned} p(y = k|r = 1) &= \sum_{s\in\{0,1,2\}} p(y = k|s, r = 1)\, p(s|r = 1) \\ &= p(y = k|s = \rho(k), r = 1)\, p(s = \rho(k)) + \frac{1}{K} p(s = 0), \end{aligned} \tag{3.6}$$

where all the probability densities are functions of X and W_1. In deriving Eq. 3.6 we approximated $p(s|r = 1)$ by $p(s)$ which we found to be appropriate. However, if required, one could include the knowledge of the teams's relative strengths into the estimate of the phase of the game. During the timeouts we model the ball ownership as uniformly distributed among the two teams and the referees, i.e. $p(y = k|s = 0\,, r = 1) = 1/K$.

Summary of Factorization We have reduced the problem of learning $p(y)$ to the problem of learning the distributions $p(r)$ and $p(y = k|s = \rho(k), r = 1)$, the latter being the probability that a player holds the ball given the ball is indeed owned and his team is in possession of the ball. Our factorization also requires learning $p(s)$, which can be accomplished using known approaches; our implementation is presented in Sec. 3.3.1.

Learning the Unary Potential

In deriving the factorized unary potential in Eqs. 3.5 and 3.6, we did not make any assumptions about the particular form of the image evidence X. Therefore, one has the freedom to derive features from X that are compatible with the envisioned application. For the sake of demonstration, we define geometric features based on projections of the monocular ball detections on the ground plane.

We accumulate multi-view evidence for the ball in a sparse ground-plane representation called the Ball Occupancy Map (BOM), and we have one such BOM for each time instant. The details of constructing the BOM are presented in Sec. 3.3.2. Briefly, monocular detections give rise to likely 3D locations of the ball, and these locations are then projected onto the ground plane, yielding peaks in the BOM; the geometric construction is shown in Fig. 3.2.

Given the BOM at time t, let $d_t^{j,k}$ be the distance between peak j and player k, see Fig. 3.3,

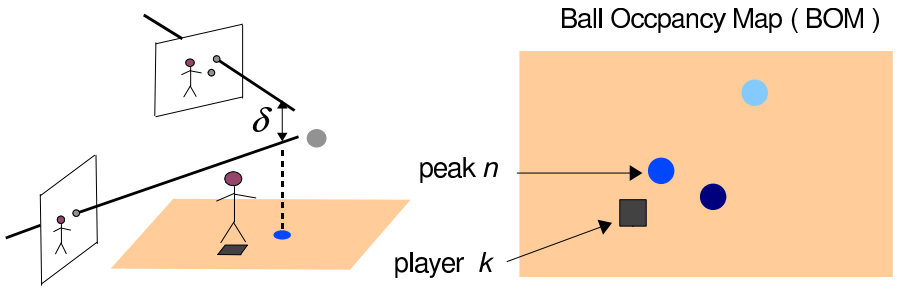


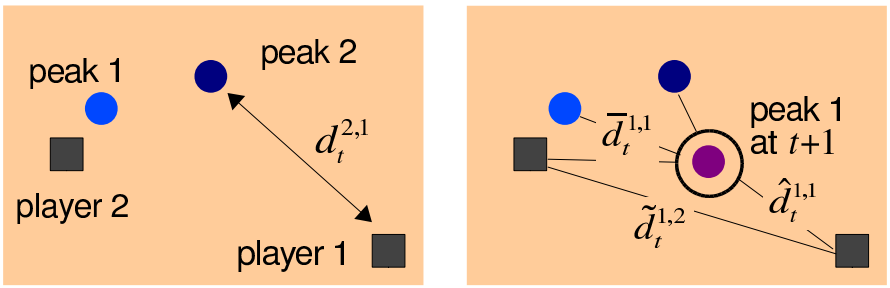
Figure 3.2 – The Ball Occupancy Map (BOM) is a sparse ground-plane representation of the image evidence. (a) For every ball detection in every view a ray is cast from the camera center. Two rays, each from a different view, cross within distance δ. (b) Projections of these crossing on the ground plane define peaks of the BOM. Each BOM peak has a score that is derived from the confidences of its corresponding monocular detections. Three peaks, each with a different score, are shown as circles of different color intensity.

and let c_t^j be the peak's score as defined in Sec. 3.3.2. We will use $d_t^{j,k}$ and c_t^j the primitives for deriving feature vectors, which will make it practical to learn $p(r)$ and $p(y)$.

Learning $p(y = k|s = \rho(k), r = 1)$ For a player k we define a feature vector $\mathbf{x}_{\text{player}}(k;n) \in \mathbb{R}^{2n}$, where n specifies the number of BOM peaks closest to player k that are used to construct this vector. This feature vector takes the form of $\mathbf{x}_{\text{player}}(k;n) = [\mathbf{c}, \mathbf{d}]$. The first component of $\mathbf{x}_{\text{player}}(k;n)$ is $\mathbf{c} \in \mathbb{R}^n$, a vector of sorted BOM scores corresponding to the n peaks. The second component of $\mathbf{x}_{\text{player}}(k;n)$ is $\mathbf{d}$, which comprises a sorted set of distances $d_t^{j,k}$.

We train a probabilistic classifier to predict if a player k has the ball; in our implementation this classifier takes the form of a random forest. Classifier training is accomplished using examples of the form $(\mathbf{x}_{\text{player}}(k;n), \mathbf{1}(y^{\text{gt}} = k))$, where $\mathbf{1}(\cdot)$ is an indicator function, and $y^{\text{gt}} \neq \ominus$ is the ground-truth label. During prediction we obtain K predictions α_k, one for each player, and set $p(y = k|s = \rho(k), r = 1) = \alpha_k \cdot (\sum_{i \in \rho(k)} \alpha_i)^{-1}$.

Learning $p(r)$ We define a feature vector $\mathbf{x}_{\text{own}}(n) \in \mathbb{R}^{n \cdot (K+1)}$, where n specifies the number of BOM peaks used to construct this vector, and which takes the form of $\mathbf{x}_{\text{own}}(n) = [\mathbf{c}, \mathbf{d^1}, ..., \mathbf{d^n}]$. The first component of $\mathbf{x}_{\text{own}}(n)$ is $\mathbf{c} \in \mathbb{R}^n$, a vector of sorted BOM scores. The remaining components of $\mathbf{x}_{\text{own}}(n)$ are n vectors $\mathbf{d^j} \in \mathbb{R}^K$. Each such $\mathbf{d^j}$ comprises a sorted set $\{d_t^{j,k}\}_{k=1}^K$. An example of constructing $\mathbf{x}_{\text{own}}(n)$ for $n = 2$ is shown in Fig. 3.3. Our intuition for this feature is that if one observes a BOM peak with a high score, and that peak is distant from any player on the ground, then it is more likely that the ball is free, i.e., $r = 0$.



(a) (b)

Figure 3.3 – Given the locations of the players and BOM at time t and $t+1$, we define a set of geometric primitives. These primitives will then be used to derive the features for learning ψ_1 and ψ_2. (a) We define $d_t^{j,k}$ as the distance between peak j and player k at time t. (b) We define $\tilde{d}_t^{k,k'}$ as the distance between player k and player k' at time t, $\hat{d}_t^{j,k}$ as the distance between peak j at time $t+1$ and player k at time t, and $\bar{d}_t^{j,j'}$ as the distance between peak j at time t and peak j' at time $t+1$.

In our implementation, $p(r)$ takes the form of a random forest classifier. The classifier is trained in a standard fashion on a set of pairs of the form $(\mathbf{x}_{\text{own}}(n), r)$.

3.2.3 Pairwise Potential

The pairwise potential accounts for the transition of the ball between two consecutive time instants. In this section we factorize the pairwise potential in Eq. 3.3, derive the features to evaluate it, and define a method to learn its parameters.

Factorization of the Pairwise Potential

In the previous section we derived a factorized form of the unary potential with respect to R and S, which in turn made it practical to learn this potential from the training data. We factorize the pairwise term with respect to teams' tactics and spatial context. The tactics defines a team's behavior such as frequency of passes, which is independent of players' locations, while the spatial context models the temporal evolution of the ball conditioned on image evidence. In other words, the pairwise term encodes both empirical knowledge and contextual information. Formally, we write

$$p(y_{t+1}|y_t; X, W_2) = \quad p_{\text{tactics}}(y_{t+1}|y_t; W_2) \underbrace{p_{\text{context}}(y_{t+1}|y_t; X, W_2)}_{\text{Eq. 3.8}}, \tag{3.7}$$

where p_{tactics} and p_{context} model teams' tactics and spatial context respectively. In the next section we show our features derived from geometric primitives defined on the BOM, and how we apply these features to learn Eq. 3.7.

Learning of the Pairwise Potential

We define $\tilde{d}_t^{k,k'}$ as the distance between player k and player k' at time t; $\hat{d}_t^{j,k}$ as the distance between peak j at time $t+1$ and player k at time t; and $\bar{d}_t^{j,j'}$ as the distance between peak j at time t and peak j' at time $t+1$. An example of these primitives is shown in Fig. 3.3.

Learning p_{context} To learn the spatial context term, we apply a truncated zero-mean Laplace distribution $\boldsymbol{\kappa}(\cdot\,;\boldsymbol{\lambda})$ with parameter $\boldsymbol{\lambda} \in \mathbb{R}^2$ specifying its bandwidth and the truncation threshold. Note that, this distribution is a radius basis function whose probability depends only on distances. Since all our features for learning p_{context} are derived based on distances, this simple model serves our purpose well. We write,

$$
\begin{aligned}
&p_{\text{context}}(y_{t+1} = k' | y_t = k) = C_{t+1}^{-1} \\
&\times \boldsymbol{\kappa}(\min_{j,j'}\{|\bar{d}_t^{j,j'} - v_{\text{ball}}|\}_{j,j'}; \boldsymbol{\lambda}_1)^{\mathbf{1}(r_t=0)\cdot\mathbf{1}(r_{t+1}=0)} \\
&\times \boldsymbol{\kappa}(\min_{j}\{|\hat{d}_t^{j,k} - v_{\text{ball}}|\}_{j}; \boldsymbol{\lambda}_2)^{\mathbf{1}(r_t=0)\cdot\mathbf{1}(r_{t+1}=1)} \\
&\times \boldsymbol{\kappa}(\min_{j}\{|\hat{d}_t^{k,j} - v_{\text{ball}}|\}_{j}; \boldsymbol{\lambda}_3)^{\mathbf{1}(r_t=1)\cdot\mathbf{1}(r_{t+1}=0)} \\
&\times \boldsymbol{\kappa}(\tilde{d}_t^{k,k'}; \boldsymbol{\lambda}_4)^{\mathbf{1}(r_t=1)\cdot\mathbf{1}(r_{t+1}=1)},
\end{aligned}
\tag{3.8}
$$

where C_{t+1} is a normalization constant, v_{ball} is the average speed of the ball in flight, $\mathbf{1}(\cdot)$ is an indicator function, and X and W_2 are implicit. Note that, p_{context} is modeled as a product of four zero-mean Laplace distributions, and the indicator variable guarantees that only one distribution can be selected. The four distributions model the following four scenarios:

1. **The ball being free at time t and $t+1$.** In this case, the min operator operates on the distances between BOM peaks at time t and $t+1$, and selects the one that is closest to the average speed of the ball. With the Laplace distribution $\boldsymbol{\kappa}(\cdot\,;\boldsymbol{\lambda})$, the probability is peaked when the distance is equal to the average ball speed. Intuitively, the fact that two ball detections whose distance approximates the average speed of the ball, is an evidence of the ball being free.

2. **Player k receiving the ball at time $t+1$.** With the min operator, we select the BOM peak at time t whose distance to player k at time $t+1$ best approximates the average speed of the ball. When they are equal, it is an evidence of a player receiving the ball, in which case the probability is peaked.

3. **Player k shooting the ball at time t.** It is semantically symmetric to to last case: we select the BOM peak at time $t+1$ whose distance to player k at time t best approximates the average speed of the ball. When they are equal, it is an evidence of a player shooting the ball, in which case the probability is peaked.

4. **Players in possession of the ball at time t and $t+1$.** In this case, the probability decreases as the distance between players increases. This term encodes that, the probability of ball transition between players is high when the players are closed, and vice versa.

Note that, in all the above cases, when the distance is larger than the truncation threshold of $\boldsymbol{\kappa}(\cdot\,;\boldsymbol{\lambda})$, the probability collapses to zero. This corresponds to the fact that, a ball cannot physically transit to a distant place within one frame given sufficient frame rate.

We learn the $\boldsymbol{\lambda}$'s by minimizing the classification error on a training sequence $(y_1^{\text{gt}},\ldots,y_T^{\text{gt}})$. In our experiments we have found it sufficient to use a single $\boldsymbol{\lambda}$ for all the terms in Eq. 3.8, and perform a grid search with fewer than ten values in each dimension.

Learning p_{tactics} We learn $p_{\text{tactics}}(y_{t+1}|y_t)$ by decomposing it into two multinomial distributions with respect to the two possible outcomes for r_t. The first distribution takes the form of $p(\Omega_1|r_t=0)$, where $\Omega_1=\{y_{t+1}=\ominus, y_{t+1}\neq\ominus\}$. The second distribution takes the form of $p(\Omega_2|y_t,r_t=1)$, where $\Omega_2=\{y_{t+1}=\ominus, y_{t+1}=y_t, (y_{t+1}\neq y_t)\wedge(\rho(y_{t+1})\neq\rho(y_t)), (y_{t+1}\neq y_t)\wedge(\rho(y_{t+1})=\rho(y_t))\}$. We learn the parameters of these multinomial distributions using maximum-likelihood estimation. In other words, we learn the following empirical frequencies: 1) the ball being free, 2) the ball received by a player, 3) the ball shot by a player, 4) a player in possession of the ball, 5) the ball being stolen by a player from the other team and 6) the ball given to a player in the same team.

3.3 Complete System

To validate our approach we implemented a complete system for tracking a ball in team sports over an extended period of time. Because the system, summarized in Alg. 1, exploits

knowledge about the scene, we will refer to it as Focus on the Scene (FoS).

The input to our system comprises multi-view video sequences in the temporal interval $t \in [1,\ T]$, and W, the parameters of for the unary and the pairwise potential. In Step 1 players and referees are tracked, and labeled according to their team membership. In Step 2, the phase of the game, i.e., $\mathbb{S} = \{p(s_t)\}$, is estimated. In Step 3, $\mathbb{B} = \{\text{BOM}_t\}$ is obtained for $t \in [1,\ T]$. In Step 4, the algorithm performs temporal segmentation by detecting *long pass* segments $\mathbb{T}_0$ and labelling the rest as $\mathbb{T}_1$; this step is defined in Sec 3.3.2. In Step 5, for every time index $t \in \mathbb{T}_0$ the algorithm assigns y_t the $\ominus$ label. In Step 6, the algorithm finds the y_t's for $t \in \mathbb{T}_1$ that minimizes the energy in Eq. 3.1, given that $y_{t'} = \ominus$ for $t' \in \mathbb{T}_0$. The optimal sequence of ball states $\hat{y}_1, \ldots, \hat{y}_T$ computed in Step 6 is transformed into a sequence of ground-plane cells $\hat{g}_1, \ldots, \hat{g}_T$ by indexing player locations in X when $y \neq \ominus$ and linearly interpolating between player locations for $y = \ominus$.

In the remainder of this section we state the algorithm's main components. We then describe the parameter settings.

Algorithm 1: Focus-on-the-Scene (FoS) ball tracking

Input: Multi-view video sequences for $t \in [1,\ T]$ and
W = {parameters of ψ_1 and ψ_2 (Sec 3.2.2 and 3.2.3)}.

1. track players and assign team labels (Sec. 3.3.1)
2. estimate $\mathbb{S}$, the phase of the game (Sec. 3.3.1)
3. compute $\mathbb{B} = \{\text{BOM}_t\}$ for $t \in [1,\ T]$ (Sec. 3.3.2)
4. segment $[1,\ T] = \mathbb{T}_0 \cup \mathbb{T}_1$, where $\mathbb{T}_0 \cap \mathbb{T}_1 = \emptyset$, and $\mathbb{T}_0$ corresponds to *long passes* (Sec. 3.3.2)
5. set $y_t \leftarrow \ominus$ for $t \in \mathbb{T}_0$
6. let X = (player tracks, $\mathbb{B}$, $\mathbb{S}$) and
   set $\hat{y}_1, \ldots, \hat{y}_T \leftarrow \arg \min_{\substack{y_1,\ldots,y_T \\ t \in \mathbb{T}_1}} E(y_1, \ldots, y_T; X, W)$

Output:
$\hat{g}_1, \ldots, \hat{g}_T \leftarrow$ linearly interpolate $(\hat{y}_1, \ldots, \hat{y}_T)$

3.3.1 Game phase and player trajectories

Our implementation of player-tracking and trajectory-estimation takes advantage of known approaches and publicly-available software.

Inferring the phase of the game

Our set of features $\mathbf{x}_{\text{phase}}$ is inspired by [103], except that in our case player trajectories and team membership are determined automatically, and referees are also tracked. Our feature vector $\mathbf{x}_{\text{phase}}$ comprises: 1. locations of players, sorted along the court length within their own team; 2. velocities of all players, sorted within their own team; 3. locations of centroids of each team; 4. velocities of centroids of each team.

We learn a random forest to infer the phase of the game. Our training set comprises pairs $(\mathbf{x}_{\text{phase}}, s)$.

Player tracking

To track the players on the ground plane, we apply the multi-layer people tracking framework proposed in [92], which relies on the publicly-available software [115, 93]. We obtain people tracks with team memberships from the two teams and the referees.

3.3.2 Ball detection and ballistic trajectory estimation

In this section, we describe the monocular ball detections, the construction of the Ball Occupancy Map (BOM) as well as the estimation of ballistic trajectories.

Monocular ball detection

To detect the ball from monocular view, we first conduct eigen-background subtraction [116] to extract the foreground pixels. We obtain the template color histogram of the ball from our training sequence as follows: for each training image, we obtain the histogram of the Hue, Saturation and Value channels separately, and then we concatenate them into one single histogram; finally we average all such histograms from all training images and obtain the template histogram of the ball. Then we scan all the foreground pixels in a sliding-window manner, obtain the color-histogram of the current window and compute the inner-product between the template histogram and the current histogram. Thus, we get a confidence for

each pixel being the center of the ball. We also apply the hough circle transform to detect round objects. For each detected circle, we increase the confidence by β_1. Moreover, we increase the confidence of pixels that are not part of players by β_2. This is because we conduct detection by background subtraction, moving object that is not part of players is likely to be the ball. In our implementation, we obtain the per-view projections of the players, and then increase the confidence of the ball detections that fall outside of the players' projections.

Constructing the Ball Occupancy Map (BOM)

Monocular image evidence for the ball can be quite noisy due to the small apparent size of the ball and frequent occlusions. Reconciling this noisy information across multiple views is therefore non-trivial. Since our inference is performed in terms of the ground-plane player locations we accumulate evidence for the ball in a sparse ground-plane representation called the Ball Occupancy Map (BOM), which was introduced in Sec. 3.2, which we now define precisely.

Starting with the per-view detection results, we apply triangulation to reconstruct the 3D ball detections in a pair-wise manner. In other words, for any two detections from two views, we cast the lines of sight. Only those pairs of lines that cross within a threshold δ are kept, from where a 3D position of the point closest to both lines is recovered. The score of such 3D detection is set to be the product of per-view detections. In case there are more than two views, we project the obtained 3D detection on other views, and multiply the corresponding scores. Once we compute the scores, we project the 3D intersections onto the ground plane. We refer to these projections as BOM *peaks*, and we set the scores of BOM peaks to be the scores of the corresponding 3D detections. In Fig. 3.2, we show an example of BOM, where players are denoted by squares and the peaks of BOM are denoted by circles.

Temporally segmenting long passes

As mentioned in Sec. 3.2, we consider long passes as evidence for the ball being free. Detecting long pass is easy because, by definition, the ball is rarely occluded. Indeed, the approaches of [100, 117] and others have addressed the problem of finding segments of parabolic trajectories. However, our objective is to only temporally segment the long pass, rather than smoothing those track segments with the second-order model. In the basketball case, our algorithm grows parabolic segments from detections above the average player's height and stops when the *coefficient of determination* is below a threshold; in the soccer case, our algorithm

detects segments whose distance to the nearest player is larger than a distance threshold.

3.3.3 Parameters

Our parameter settings are as follows. For the monocular ball detection, we set $\beta_1 = 0.2$ and $\beta_2 = 0.1$. For long passes detection, we set the coefficient of determination to be 0.95 for temporal segmentation in the basketball case, and distance threshold to be 1.5 m in the soccer case. We use a publicly-available implementation of the random forest classifier [118] with default parameters and 1,000 trees. For the p_{context} term in Eq. 3.8 we set $\boldsymbol{\lambda} = (10^{-3}, 300 \text{ cm})$ and $\boldsymbol{\lambda} = (10^{-3}, 800 \text{ cm})$ for basketball and soccer respectively.

3.4 Experiments

We validate our approach on challenging basketball and soccer video sequences, described below.

3.4.1 Datasets

The FIBAW (women's championship at Karlovy Vary) dataset comprises two multi-view basketball sequences captured at the 2010 women's world championship: 1. Mali vs. Senegal match (FIBAW-1), 8,480 frames; 2. Czech Republic vs. Belarus (FIBAW-2), 2,850 frames. The matches are captured by ten stationary synchronized megapixel 25-frame-per-second cameras. For the two matches, the cameras are placed at different places around the court. We use three cameras for FIBAW-1 and four for FIBAW-2 with overlapping fields-of-view; these cameras are sufficient for having a coverage of at least two different views for the entire court. We have manually annotated one out of every ten frames for FIBAW-1 and 500 consecutive frames for FIBAW-2, and we perform the quantitative comparison using these frames.

The APIDIS dataset is presented in [119]. It comprises a basketball match sequence captured by seven stationary unsynchronized 2-megapixel 22-frames-per-second cameras placed above and around the court. The authors also provide a pseudo-synchronized sequence and we conduct tracking on it. The APIDIS dataset is challenging for ball tracking because the camera locations are not optimized for capturing the ball and the lightning conditions are far from optimal, as there are many direct light sources which are reflected on the court while some other regions are shaded. Especially when the ball is near the basket, the ball is completely merged with the background. We use two cameras with overlapping fields-of-view and choose

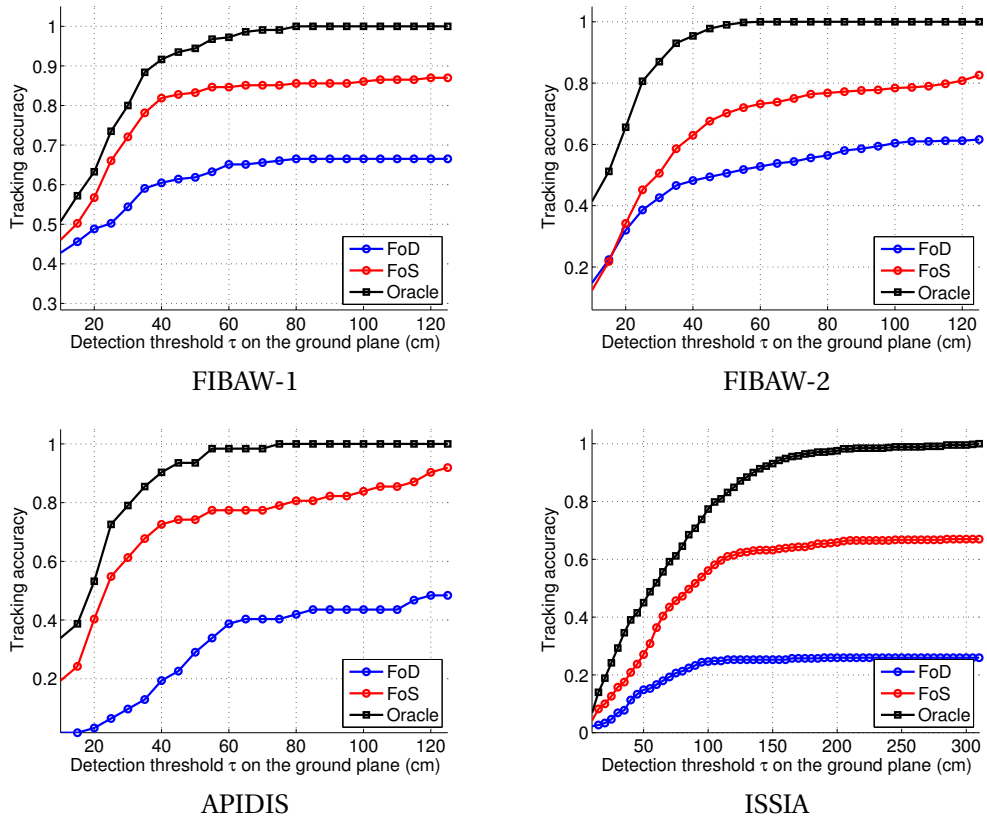


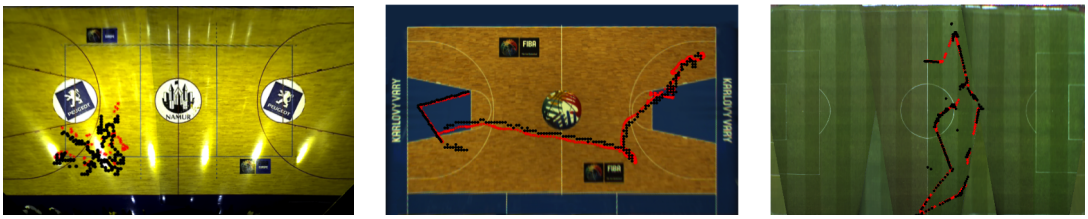
Figure 3.4 – Quantitative ball tracking results of FoD and FoS. The performance of FoS approaches that of the oracle and is decisively better than FoD, particularly in the accuracy range of ≤ 100 cm. The results are obtained by fixing $n = 1$. From left to right: results on FIBAW-1, FIBAW-2, APIDIS and ISSIA dataset.

a sequence of an attacking phase, whose length is much longer than other ball-based work on the same dataset, such as [102].

The ISSIA dataset comprises a multi-view soccer sequence [120] of 3,000 frames captured by six full-HD 25-frames-per-second cameras placed in two sides of the soccer field. We use all the cameras for our experiments. However, the camera settings are designed specifically for player tracking, and thus they do not provide a complete converge of the ball when it is flying in the air. We conduct tracking on a sequence of 2,000 frames where the ball is mostly in sight.

3.4.2 Baseline methods

Because our FoS approach is novel and our testing sequences are challenging, there is no closely related baseline that we can use for comparison. As mentioned in the introduction,



APIDIS FIBAW-2 ISSIA

Figure 3.5 – Projection of FoS tracking results (in black circles) and the ground truth (in red stars) overlaid on an orthographic view of the court. The real and the recovered trajectories are almost superposed. (left) Results on the APIDIS dataset, frames [1 400] (middle) FIBAW-2 dataset, frames [975 1354] (right) ISSIA dataset, frames [880 1399].

the state-of-the-art approach of [67], which has been shown to be successful in single object tracking, never tracks the ball for more than five consecutive frames. We therefore compare our approach both to a detection-linking tracker operating in a batch-smoothing framework and to a version of our approach that does not use the features that we derived from BOM and the player trajectories. We describe the two baselines below.

Focus on Detection (FoD) We implement a detection-linking approach that we call Focus on Detection (FoD). The baseline FoD ball tracker involves three steps: ball detection in each camera view using the same detector that was defined in Sec. 3.3.2, generation of the candidate ball locations in 3D using the same approach as defined in Sec. 3.3.2, but without projecting these candidates into BOM in the ground plane, and linking the 3D detection candidates. Similarly to the FoS tracker, we use two motion models: a second-order model during long passes and a first-order model otherwise.

FoS with Non-Conditioned Potentials We implement ψ_1^* and ψ_2^* that are not conditioned on X. The non-conditioned ψ_1^* follows Eq. 3.5 as does ψ_1, but we now set $p(r=0)$ to a constant learned from the data, and we set $p(y|r=1)$ to be $(1-p(r=0))/K$. The non-conditioned ψ_2^* follows Eq. 3.7, but since now p_{context} does not depend on X, it becomes a uniform distribution.

This baseline is useful in evaluating the contribution of the novel features we derived from BOM and the player trajectories. In particular, we will evaluate the contribution of ψ_1 vs. ψ_1^* and ψ_2 vs. ψ_2^* to the overall accuracy of our algorithm.

3.4.3 Results

For basketball, we train classifiers for the unary potential using the FIBAW dataset, test on the APIDIS dataset, and vice versa. For soccer, due to there being only one sequence available, we are not able to learn a unary potential conditioned on image evidence. Therefore, we apply FoS with a non-conditioned unary potential. We set $n = 1$ for basketball sequences and fix all other parameters as specified in Sec. 3.3.3 for all sequences. We compute the TA curve and the MCLE as we discussed in Sec. 2.2 for each test sequence. If the ball is not detected as part of a long pass and is inferred to be free, we conduct the following processing for better visualization: first, we obtain the BOM peak whose distance is closest to the interpolated location on the ground, then we find the corresponding 3D ball detection, and last we project the 3D ball detection on each camera view.

In Fig. 3.5, we show ball trajectories on an orthogaraphic view of the court. We observe that on all datasets, our obtained trajectories and ground-truth trajectories are almost superposed.

FoS tracking vs. FoD tracking In Fig. 3.4, we show the comparison of tracking performance by TA curve. We compare three approaches: FoD, FoS and oracle, which corresponds to a human observer specifying which player, if any, possesses the ball. Our FoS tracker significantly improves over the baseline FoD tracker and approaches the oracle tracker on all sequences. For the APIDIS dataset, the huge improvement of more than 100 percent, suggests that with such bad lighting conditions and occlusions of the players it is difficult to track the ball relying only on direct ball detections. The same trend is observed on the other datasets: the improvement is around 40 percent for FIBAW-1, 35 percent for FIBAW-2 and 150 percent for ISSIA.

Parameters of FoS In Fig. 3.6, we show the tracking accuracy of our FoS tracker for different settings of n, trained on FIBAW-1 and tested on APIDIS. The obtained results are similar. We have also trained on APIDIS and tested on both FIBAW-1 and FIBAW-2, where the same trend is observed.

Contribution of image evidence As Fig. 3.7 indicates, using our proposed conditioned potentials yields decisive improvement. In particular, when the non-conditioned pairwise potential is used, the accuracy is significantly lower, particularly for $\tau > 60$ cm. In fact, the accuracy curves for APIDIS and FIBAW follow the same trend, flattening at about $\tau = 60$ cm. The effect of the non-conditioned unary potential exhibits the same trend for both datasets,

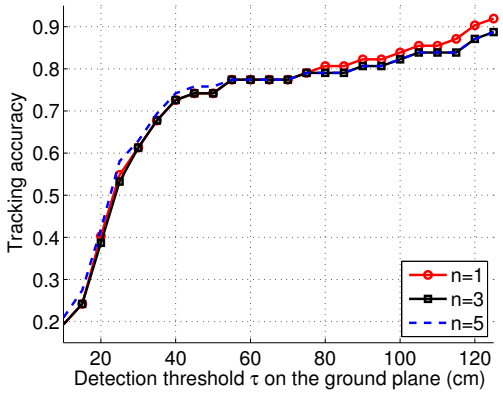
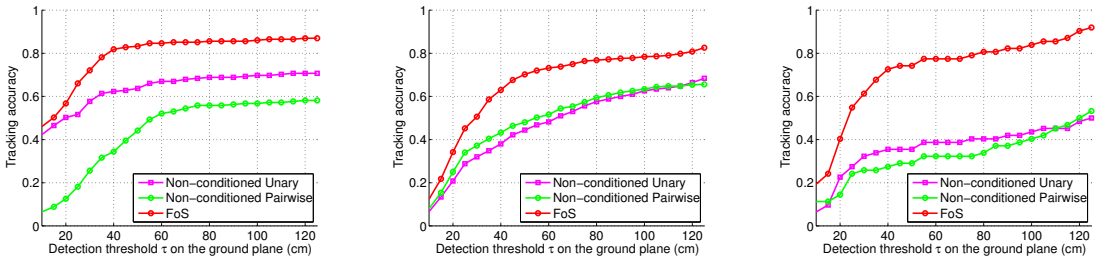


Figure 3.6 – Tracking performance with different settings of n, i.e. the number of BOM peaks used for training the unary potential.



FIBAW-1 FIBAW-2 APIDIS

Figure 3.7 – Contribution of the conditioned unary and pairwise terms. (left) Results on the FIBAW-1 dataset (middle) Results on the FIBAW-2 dataset (right) Results on the APIDIS dataset.

roughly following the curve for the non-conditioned pairwise term, and below the accuracy for the model that uses X in both terms.

In Table 3.2 and Table 3.3 we show the Success Rate (SR) of all algorithms, at distance thresholds of 30 cm and 100 cm respectively. On all the datasets, FoS yields an SR over 50 percent at threshold of 30 cm and over 78 at a threshold of 100 cm, which is significant higher than other trackers.

In Fig. 3.12, we show the performance of different trackers using MCLE. The FoS agains yields significant improvement. On the FIBAW dataset, the MCLE is around 70 cm for FoS and >150 cm for FoD. This means that on average, our tracking result is as close as 70 cm to the ground truth, which is reasonably satisfactory given the size of basketball court is 28 m × 15 m.

Table 3.2 – Success Rate (SR) at the threshold of 30 cm

Algorithm	FIBAW-1	FIBAW-2	APIDIS
FoS	**0.721**	**0.506**	**0.613**
FoD	0.544	0.426	0.097
Non-cond. Unary	0.577	0.320	0.323
Non-cond. Pairwise	0.256	0.372	0.258

Table 3.3 – Success Rate (SR) at the threshold of 100 cm

Algorithm	FIBAW-1	FIBAW-2	APIDIS
FoS	**0.861**	**0.784**	**0.839**
FoD	0.665	0.604	0.436
Non-cond. Unary	0.698	0.626	0.436
Non-cond. Pairwise	0.567	0.634	0.403

The performance is even better on the APIDIS dataset: the MCLE is less than 40 cm for FoS and around 95 cm for FoD. On the ISSIA dataset, the MCLE is 162.7 cm for FoS and 272.3 cm for FoD. Note that the MCLE we obtained for soccer is larger than that of basketball. This can be explained in part by the definition of ball ownership in soccer: when players dribble the soccer ball, they may let it roll on the ground ahead of them. Therefore, the distance between the player and the ball is larger than that of basketball.

We show CLE versus frame and the corresponding tracking results on the APIDIS dataset in Fig. 3.8. We observe that the CLE of FoS is less than 30 cm in most frames and is never larger than 200 cm. From frame 205 to 275, the CLE of FoS is large, as it tracks a player who does not own the ball. FoS makes such errors because the tracked player and the player who indeed owns the ball are of the same team and close to each other. From frame 260 to 400, there is a significant divergence of the non-conditioned unary tracker, as it incorrectly assigns the ball to a player who is far from the ground truth. From frame to frame, the CLE of non-conditioned pairwise tracker fluctuates widely. This is because we ignore X by assuming p_{context} as a uniform distribution, and the distances between players are not encoded in the transition probability. Therefore, the transition between two players who are far apart does not get penalized, which results in the fluctuation in tracking results and the CLE.

Analysis of Failures Our proposed FoS tracking approach decisively improves on the state-of-the-art. However, given the ambiguous image evidence and close proximity of the players in our datasets, the FoS tracker may sometimes yield incorrect results. We show some cases

where the estimated ball location differs from the ground truth in the example tracking results. In Fig. 3.8(c) and Fig. 3.10(d), the tracker assigns incorrect player ownership. In Fig. 3.9(b), because the ball detection is close to a player, FoS assigns the ball to that player, while the FoD maintains the correct 3D location.

One approach to preventing these types of mistakes is to refine the model of ball ownership. In particular, the tactics and the context terms of Eq. 3.7 could be guided by a more nuanced knowledge of the particular sport. In applications where post-processing of the ball trajectories is feasible, such post-processing can be guided by a higher-order model of the tactics.

Summary of Experiments Our FoS approach decisively outperformed the state-of-the-art FoD on all of our datasets as measured by the standard tracking-performance metrics. This outcome was obtained by automatically learning the parameters of our model, and keeping all other parameters of our complete system fixed across all experiments.

In addition to higher accuracy, FoS is more computationally-efficient than FoD. On a Quad-Core 2.50 GHz CPU running MacOS, given pre-computed monocular ball detection, the throughput of FoS is 8-10 frame/sec, while the throughput of FoD is 0.5 frames/sec.

3.5 Conclusion

In this chapter we presented a novel approach to tracking the ball in team sports, which is a challenging task because the ball is often occluded by players. In contrast to conventional approaches that track the ball by itself and only then assign ball possession, our approach first tracks players and then tracks the ball by deciding which player, if any, is in possession of the ball.

This is achieved by introducing a novel state variable for the ball, which allows the ball to be associated with one of the players, or to be free meaning that no player is in possession of the ball. The tracking problem is formulated as a CRF, wherein the unary and pairwise potentials are computed based on image evidence. The unary potentials account for the ball possession and the phase of the game, while the pairwise ones account for tactics and context.

We tested our proposed tracker on challenging basketball and soccer sequences. Our approach consistently works better than the baseline ones in terms of all the metrics we used. We also demonstrated the respective contribution of unary and pairwise potentials.

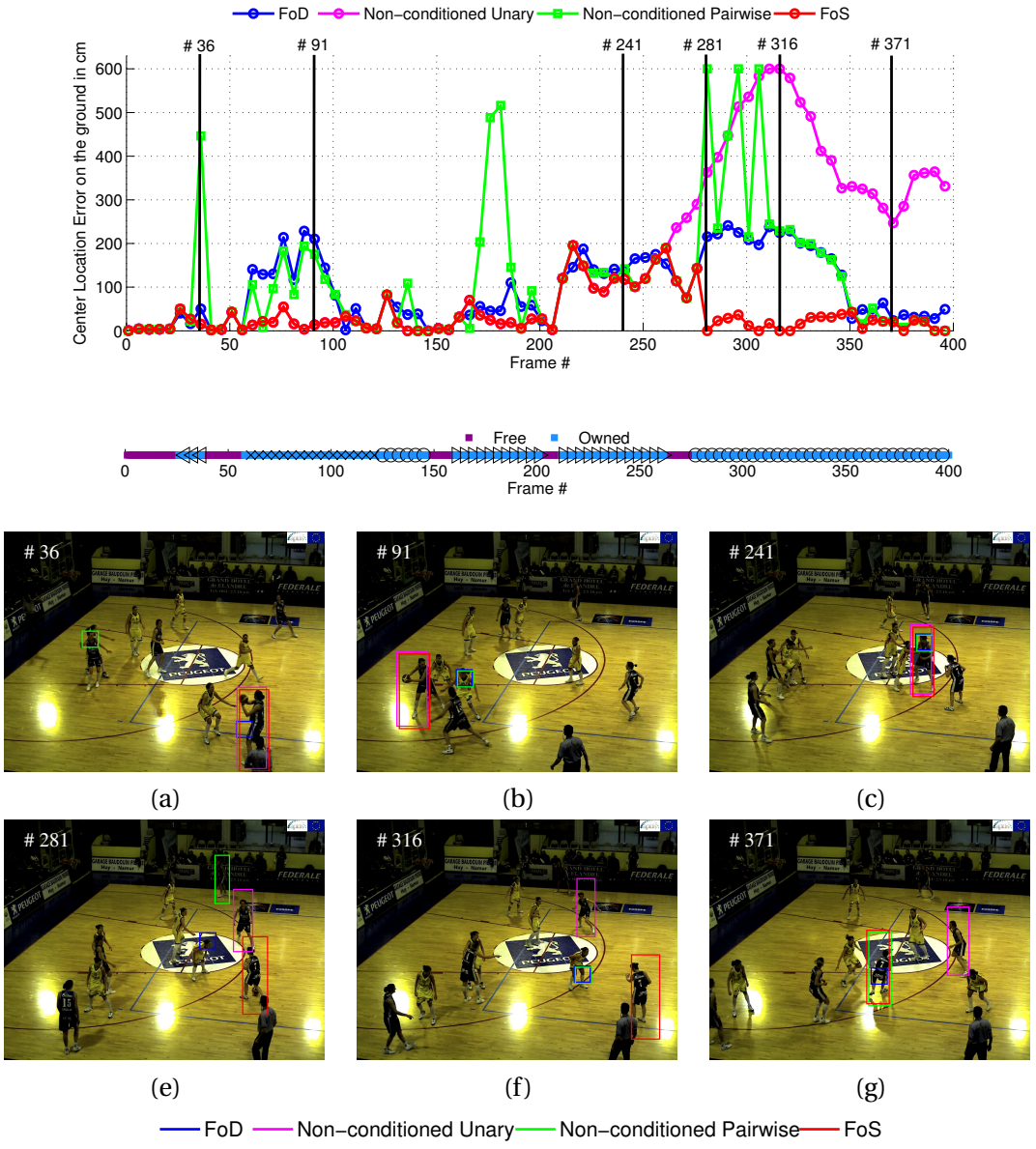


Figure 3.8 – (top) Center Location Error (CLE) versus time, (center) obtained ball states, and (bottom) example tracking results on the APIDIS dataset. The temporal range evaluated is frames [1 400], which is an attacking phase that contains interesting cases such as shot-passes and FoS failures. In the obtained ball states, we show the frames where the ball is predicted to be free by purple, and owned by blue. We use different markers to represent different players owning the ball.

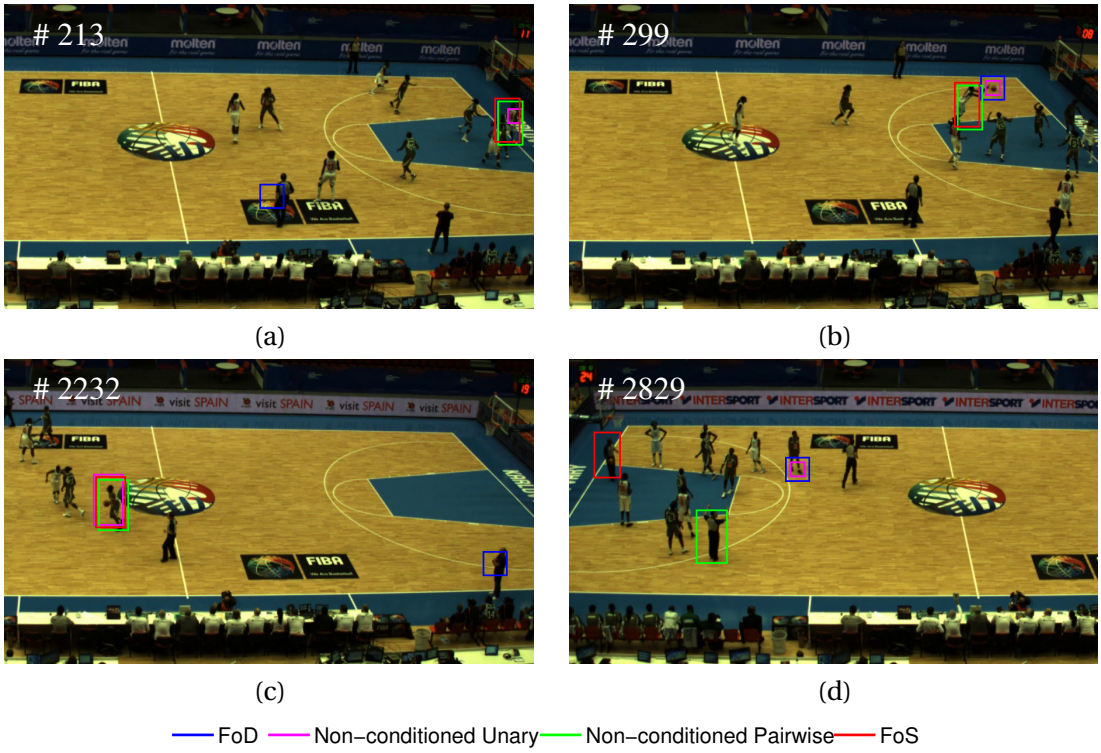


Figure 3.9 – Qualitative tracking results on the FIBAW-1 dataset. FoS assigns the ball to the correct player in (a), (b) and (c) during attacking phases, and in (d) during timeout. In (b), although FoS assigns the ball to the correct player, the tracking is less precise than FoD, which tracks the ball itself.

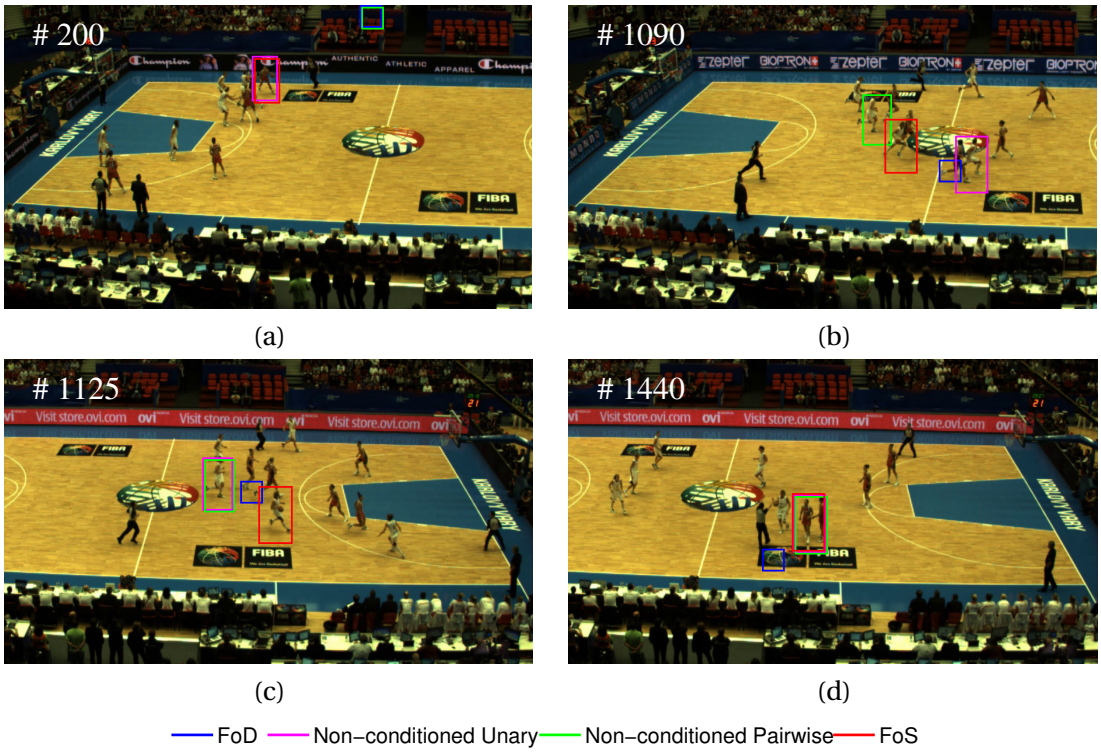


Figure 3.10 – Qualitative tracking results on the FIBAW-2 dataset. FoS assigns the ball to the correct player in (a), (b) and (c) during attacking phases, yet fails in (d) during timeout: the ball is owned by a player in a white jersey, while FoS tracks a nearby player in a red jersey.

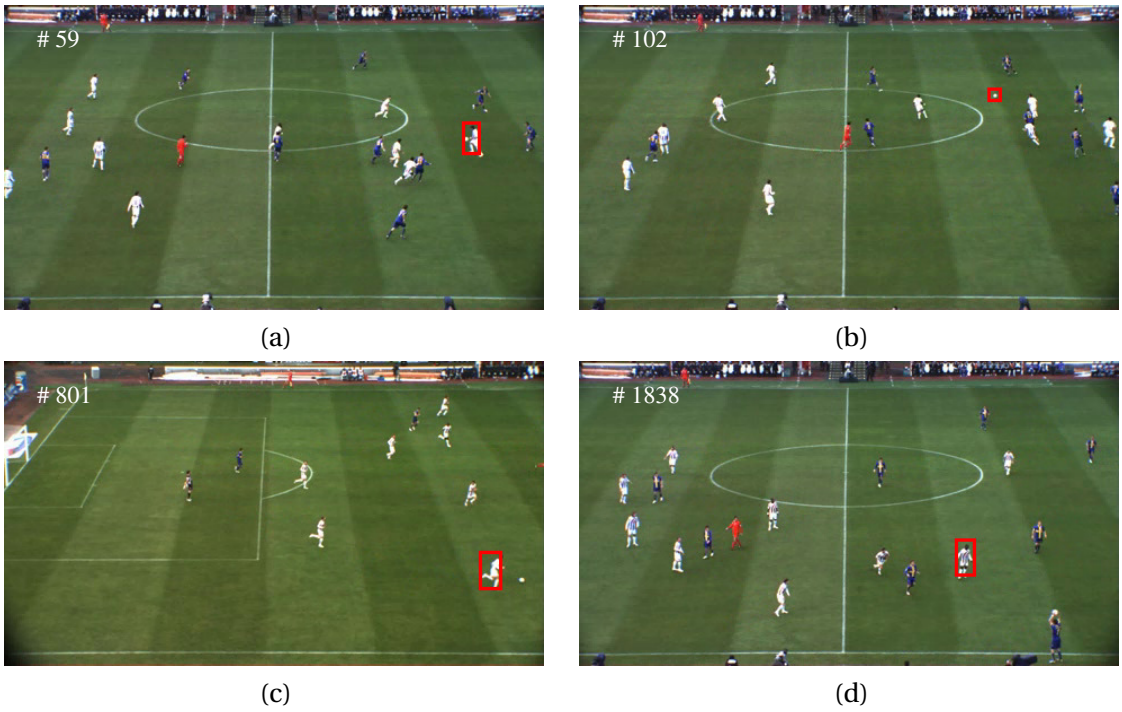


Figure 3.11 – Qualitative tracking results on the ISSIA dataset. FoS tracks the correct player in (a) and (c), but tracks the incorrect one in (d). In (b), the ball is detected as part of a long pass. In (c), the player dribbles the ball by letting it roll. In this case, although the player owns the ball, the distance between the ball and the player is large compared with that of basketball.

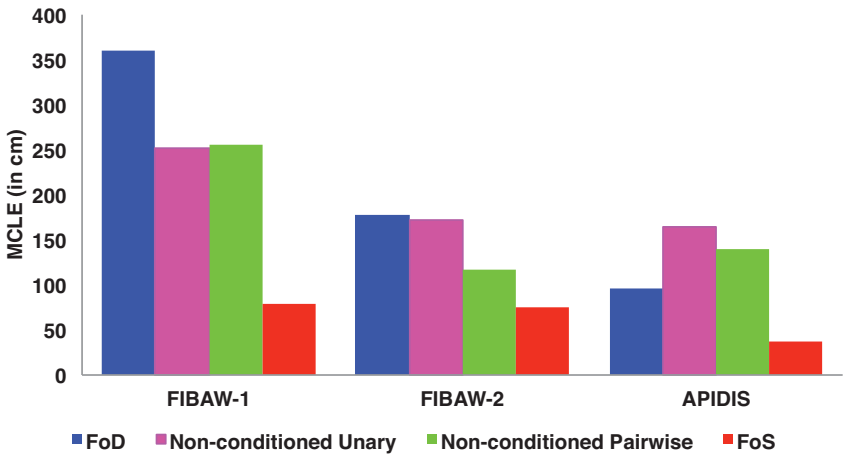


Figure 3.12 – Comparative tracking results using the Mean Center Location Error (MCLE) metric. The MCLE of FoS is lower than other trackers on all datasets.

4 Tracking Interacting Objects Using Intertwined Network Flows

In this chapter, we introduce a network flow programming framework to simultaneously track two classes of interacting objects. Conventional tracking approaches focus on one class of objects and only model very simple interactions, such as two instances cannot occupy the same spatial location at the same time. In contrast, our approach tracks two classes of objects simultaneously and models complex interactions between the presence of one object and the appearance or disappearance of objects of the other class.

We achieve this by modeling the tracking problem as a Mixed Integer Programming (MIP). We model objects of different classes with different flow variables, and express the fact that an object can appear or disappear at the location of another in terms of linear constraints. The objects from both classes are treated symmetrically and the presence of one object may be the evidence of the appearance or disappearance of another. One such example is shown in Fig. 1.3.

Our approach is much more general than what is done in approaches such as [68], in which the appearance of people is used to infer the possible presence of a static entrance. It also goes beyond recent work on interactions between people and objects [121]. 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. Furthermore, we introduce a tracklet-based implementation that yields near real-time tracking performance.

The rest of this chapter is organized as follows. In Sec. 4.1, we review related tracking algorithms. We show our formulation in Sec. 4.2 and describe our optimization techniques in Sec. 4.3. In Sec. 4.4 we show how we compute the probabilities of occupancy. We show

comparative tracking results in Sec. 4.5 and conclude this chapter in Sec. 4.6.

4.1 Related Work

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

4.1.1 Tracking Multiple Objects

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

Tracking by Model Evolution

Early approaches of this category focused on tracking a single object and relied on gating and Kalman filtering, which have later made their way into our community [122, 123]. Because of their recursive nature, they are prone to errors that are difficult to recover from. Particle-based approaches such as [2, 124, 125, 126, 127], among many others, partially address this issue by simultaneously exploring multiple hypotheses. However, they can handle only relatively small batches of temporal frames without their state space becoming unmanageably large, and often require careful parameter setting to converge. [128] uses support vector machine classifiers to track the targets frame by frame. [129] applies AdaBoost to train an ensemble of weak classifiers to distinguish between the foreground and background pixels, while [130] treats the ensemble weight as random variables whose distributions evolve with recursive Bayesian estimation. [131] proposes a semi-supervised boosting approach for on-line tracking. [132] decomposes the tracking tasks into tracking, learning and detection that operate simultaneously. [133] combines matting with tracking to alleviate drifts. [134] proposes to use a pair of bounding boxes to track non-rigid objects. [135] completely ignores the temporal orders of the frames and selects easy-to-track ones first out of the whole span of frames, and [136] extends to an online version. [137] formulates tracking as a binary classification problem that updates in the compressed domain, and [138] extends this framework by considering the locally dense contexts. Despite their success in mostly single object tracking, when dealing with interacting objects that occlude each other such as the players and the basketball in a professional match, by our experiments we found they are still prone to drifts and identity switches and thus not suitable for our purpose.

Tracking by Detection

In recent years, techniques that optimize a global objective function over many frames have emerged as powerful alternatives [13, 73]. They rely on Conditional Random Fields [91, 74], belief Propagation [139, 140], Dynamic or Linear Programming [141]. Among the latter, some operate on graphs whose nodes can either be all the spatial locations of potential people presence [142, 11, 13], only those where a detector has fired [143, 144], or short temporal sequences of consecutive detections that are very likely to correspond to the same person [145, 82, 21, 14, 146].

The K-shortest Path (KSP) algorithm has been successfully applied in multiple people tracking [13]. By working on the graph of all potential locations over all time instances, KSP finds the global optimal solution of multiple object trajectories on the ground. However, KSP assumes that the nodes in the graph are independent therefore it can not handle the poses of the objects or other constraints such as spatial exclusion. Also, KSP works on one type of objects only, therefore, when dealing with two classes of objects such as container and containee, we need to run KSP twice, first on the container objects and then on the containee ones. This approach is however not optimal as the objects are not tracked simultaneously. An example of KSP failure case is shown in Fig. 1.3.

Similar to KSP, the Successive Shortest Path (SSP) links detections using sequential dynamic programming [21]. However, SSP works on sparse graphs and links detections on the image plane. In the same spirit, [147] operates on a hexagonal lattice on the ground and incorporates dynamic models into tracking. [148] and [22] impose mutual exclusion between tracklets and solve continuous energy functions. [81] incorporates second order motion constraints into the min-cost network flow programming and uses Lagrangian relaxation to convert hard constraints into soft ones, however the global optimal solution can not be ensured. Similarly, [149] solves the same problem with dual decomposition. [150] formulates the tracking problem as latent data association, which requires several types of variables and leads to a complicated model. [32] incorporates the whole temporal span and solves the data association for all objects one by one using generalized minimum clique graphs. [151] conducts optimization over the space of detections and data associations in an iterative manner. All these approaches focus on one class of objects, and when applied on interacting objects such as containers and containees, we have to run the algorithms twice and thus they suffer the same problem as KSP.

Many aforementioned approaches obtain object detections using discriminant detectors and

can not handle mutual occlusion directly. To alleviate this problem, [82] adds occlusion hypotheses and solves the network flow programming iteratively. [152] introduces a confidence map based on geometric information to handle missed detections. [66] directly reasons on overlapping objects and maximizes an objective function that favors high confidence detections and penalizes overlapping pairs of detections. [153] explicitly trains people detector on the failure cases of the tracker to exploit the occlusion patterns. In our proposed approach, occlusions are taken care by our generative detector. In the data association stage, our methods can handle objects that are completely hidden during large portions of the interaction. This is in contrast to the above approaches that can only handle short term occlusions.

4.1.2 Tracking Interacting Objects

On average, the more global techniques are more robust than the earlier ones but, especially those that focus on tracking people do not handle complex interactions between them and other scene objects. In papers such as [154], which looks into the behavior of sports players, their trajectories are assumed to be given. In [44, 68], group behavior is considered 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.

In [68], there is also a mechanism for guessing where entrances and exits may be by recording where tracklets start and end. However, this is very different from having entry points that may move, thus allowing objects of a different nature to appear or disappear at varying locations. In [53], a set of game context features are introduced to model player interactions and network flow programming is used to solve the opitimization problem. However this approach is designed specifically for team sports and requires training data to compute the required affinity models. In [155], time-varying spatial patterns are tracked by inferring the optimal state on a mixture of Markov Random Fields. This approach only addresses the problem of tracking objects with regular spatial patterns and is therefore not suitable for our purpose. In [156] the crowd tracking problem is formulated as energy minimization that combines crowd density estimation and people detections, but only mutual occlusions between people are modeled. In [121], person-to-person and person-to-object interactions are exploited to more reliably track all of them. This approach relies on a Bayesian Network model to enforce frame-to-frame temporal coherence, and on training data to learn object types and appearances. Furthermore, it requires the objects to be at least occasionally visible during the interaction. By contrast, we propose a global optimization framework that does not require training and can handle objects that remain invisible during extended periods of time, such as

a person inside a car or a ball being carried and hidden by a player.

4.2 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 the object, as the car does to its occupants, or simply being in possession of and partially hiding it, as a basketball player holding the ball. We then discuss these constraints in more details and show that they result in a Mixed Integer Program (MIP) on a very large graph. We will discuss in the following section our approach to nevertheless solving it fast.

4.2.1 Bayesian Inference

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

Assuming the position of target objects to be completely defined by their ground plane location, we discretize the area of interest into a grid of L square cells, which we will refer to as *spatial locations*. Within each one, we assume that a target object can be in any one of O *poses*. 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 the basketball and soccer, we define the pose space to be the regularly discretized height of the ball.

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

Similar to [13], 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 $V = \{v_k^t\}$ is the set of pose vertex that represent pairs of poses and locations at each time instant. The edges $E = \{e_{kj}^t\}$ represent allowable transitions between them. More specifically, an edge $e_{kj}^t \in E$ connects vertices v_k^t and v_j^{t+1} if and only if $j \in \mathcal{N}(k)$. The number of vertices and edges are therefore roughly equal to OLT and $|\mathcal{N}(.)|OLT$, respectively.

Recall that we are dealing with two kinds of objects, one of which can contain the other. Let

$\mathbf{X} = \{X_k^t\}$ be the vector of binary random variables denoting whether location $l(k)$ is occupied at time t by a *containee* type object with pose $o(k)$, and $\mathbf{x} = \{x_k^t\}$ a realization of it, indicating presence or absence of a *containee* object. Similarly, let $\mathbf{Y} = \{Y_k^t\}$ and $\mathbf{y} = \{y_k^t\}$ respectively be the random occupancy vector and its realization for the *container* object class.

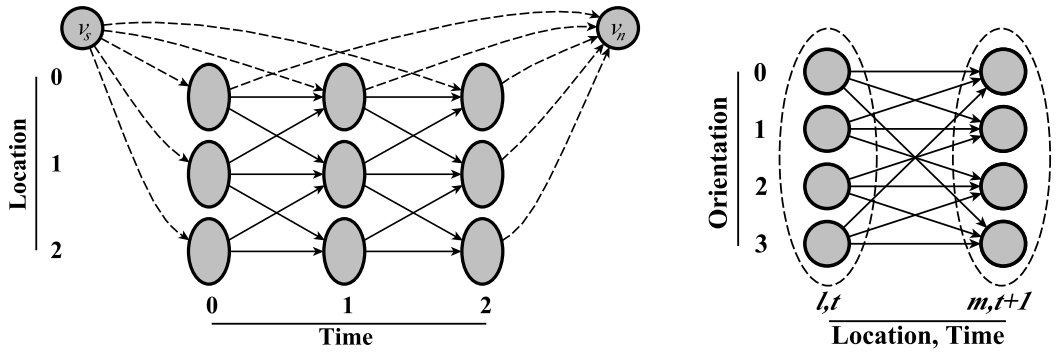
As will be discussed in Sec. 4.4, we can estimate image-based probabilities $\rho_k^t = P(X_k^t = 1 \mid \mathbf{I}^t)$ and $\beta_k^t = P(Y_k^t = 1 \mid \mathbf{I}^t)$ that a containee or container object is present at grid location $l(k)$, with pose $o(k)$, and at time t in such a way that their product over all k and t is a good estimate of the joint probability $P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y} \mid \mathbf{I})$. Among other things, this is done by accounting for objects potentially occluding each other.

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

$$
\begin{aligned}
& (\mathbf{x},\mathbf{y})^* = \underset{(\mathbf{x},\mathbf{y})\in\mathscr{F}}{\operatorname{argmax}} P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y} \mid \mathbf{I}) && (4.1)\\
\approx\ & \underset{(\mathbf{x},\mathbf{y})\in\mathscr{F}}{\operatorname{argmax}} \prod_{t,k} P(X_k^t = x_k^t \mid \mathbf{I}^t) P(Y_k^t = y_k^t \mid \mathbf{I}^t) && (4.2)\\
=\ & \underset{(\mathbf{x},\mathbf{y})\in\mathscr{F}}{\operatorname{argmax}} \sum_{t,k} \log P(X_k^t = x_k^t \mid \mathbf{I}^t) + \log P(Y_k^t = y_k^t \mid \mathbf{I}^t) \\
=\ & \underset{(\mathbf{x},\mathbf{y})\in\mathscr{F}}{\operatorname{argmax}} \sum_{t,k} x_k^t \log \rho_k^t + (1 - x_k^t) \log(1 - \rho_k^t) \ + \ y_k^t \log \beta_k^t + (1 - y_k^t) \log(1 - \beta_k^t) && (4.3)\\
=\ & \underset{(\mathbf{x},\mathbf{y})\in\mathscr{F}}{\operatorname{argmax}} \sum_{t,k} \log\left(\frac{\rho_k^t}{1-\rho_k^t}\right) x_k^t + \log\left(\frac{\beta_k^t}{1-\beta_k^t}\right) y_k^t && (4.4)
\end{aligned}
$$

where $\mathscr{F}$ stands for the set of all feasible solutions as defined in the following section. Eq. 4.2 comes from the aforementioned property that the product of image-based probabilities is close to true posterior of Eq. 4.1, which will be discussed in more details in Sec. 4.4, and from the assumption that all feasible transitions from time t to time $t+1$ are equally likely. Eq. 4.3 is obtained by taking the log of the product of probabilities. Eq. 4.3 is true because both x_k^t and y_k^t are binary variables. Finally, Eq. 4.4 is obtained by dropping constant terms that do not depend on x_k^t or y_k^t. The resulting objective function is therefore a linear combination of these variables.

However, not all assignments of these variables give rise to a plausible tracking result. Therefore, the optimization of Eq. 4.4 must be performed subject to a set of constraints defined by $\mathscr{F}$, which we describe next.



(a) (b)

Figure 4.1 – A graph representing three spatial locations at three consecutive times. (a) Each ellipse corresponds to one spatial location at one time instant. Some are connected to a source and a sink node to allow entrances and exits. (b) Within each ellipse are four nodes, one for each possible pose and the arrows represent possible transitions from one location and pose to those in the neighboring ellipse.

4.2.2 Flow Constraints

To express all the constraints inherent to the tracking problem, we introduce two additional sets of binary indicator variables that describe the flow of objects between pairs of discrete spatial locations and poses at consecutive time instants. More specifically, we introduce flow variables f_{kj}^t and g_{kj}^t, which stand respectively for the number of containee and container type objects moving from pose $o(k)$ and location $l(k)$ at time t to pose $o(j)$ and location $l(j)$ at time $t+1$. The flow variables f_{kj}^t and g_{kj}^t are defined on the edge e_{kj}^t 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 instance, one container object can interact with at most one containee object.

Spatial Exclusion: As detailed in Sec. 4.4.1, we model objects such as cars or people as rectangular cuboids, whose size is usually larger than that of a single grid cell. We impose spatial exclusion constraints to disallow solutions that contain overlapping cuboids in the 3D space. Let $\mathcal{N}_f(k)$ and $\mathcal{N}_g(k)$ denote the spatial exclusion neighborhoods for the containee and container objects respectively. We write

$$\sum_{i:k\in\mathcal{N}(i)} f_{ik}^{t-1} + \sum_{\substack{j\in\mathcal{N}_f(k),\\ i:j\in\mathcal{N}(i)}} f_{ij}^{t-1} \leq 1, \tag{4.5}$$

$$\sum_{i:k\in\mathcal{N}(i)} g_{ik}^{t-1} + \sum_{\substack{j\in\mathcal{N}_g(k),\\ i:j\in\mathcal{N}(i)}} g_{ij}^{t-1} \leq 1, \ \forall t,k\,. \tag{4.6}$$

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

$$y_k^t = \sum_{i:k\in\mathcal{N}(i)} g_{ik}^{t-1} = \sum_{j\in\mathcal{N}(k)} g_{kj}^t, \ \forall t,k\,. \tag{4.7}$$

This ensures that the container objects cannot appear or disappear at locations other than the ones that are explicitly designated as entrances or exits. Graph vertices associated to these entrance and exit points serve respectively as a source and a sink for the flows. To allow this, we introduce two additional vertices v_s and v_n into our graph G, which are linked to all the vertices representing positions through which objects can respectively enter or leave the observed area. Furthermore, we add directed edges from v_s to all the vertices of the first time instant and from all the vertices of the last time instant to v_n, as illustrated by Fig. 4.1.

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

$$\sum_{j\in\mathcal{N}(s)} g_{sj} = \sum_{i:n\in\mathcal{N}(i)} g_{in}\ . \tag{4.8}$$

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

$$-\sum_{\substack{m:l(k)=l(m),\\ i:m\in\mathcal{N}(i)}} g_{im}^{t-1} \le a(t,k) \le \sum_{\substack{m:l(k)=l(m),\\ j\in\mathcal{N}(m)}} g_{mj}^t, \forall t,k \tag{4.9}$$

$$a(t,k) = \sum_{i:k\in\mathcal{N}(i)} f_{ik}^{t-1} - \sum_{j\in\mathcal{N}(k)} f_{kj}^t\,. \tag{4.10}$$

In Eq. 4.9, the total amount of container flow passing through the location k is denoted by the two sums on both sides of the inequality. When they are zero, these constraints impose the conservation of flow for the containee objects at location k. When they are equal to one, a containee object can appear or disappear at k. Note that, here we assume the containee objects never come to interact with the container one at exactly the same moment. For example, at one time instance only one person is allowed to enter the car.

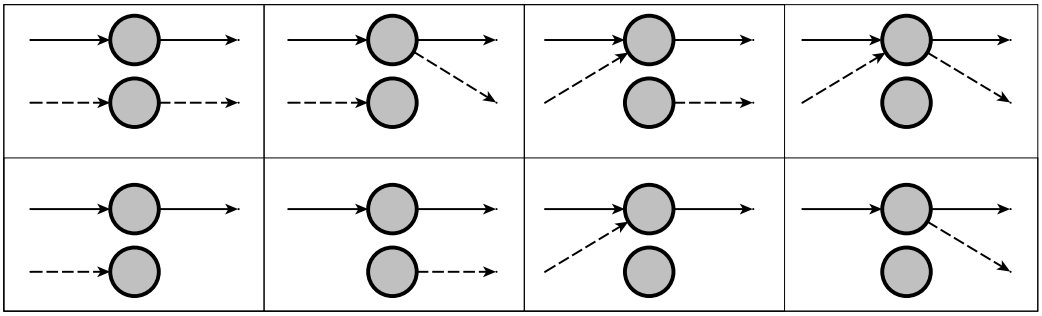


Figure 4.2 – Flow constraints in a two-pose case. In each of the eight examples, the two circles represent two pose nodes at the same spatial location. The solid and the dotted arrows represent respectively non-zero flows g^t_{kj} and f^t_{kj} of the container and of the visible containee objects. **Top Row.** Forbidden configurations, which are all cases where a containee and a container coexist at the same location and at the same time instant without interacting with each other. For example, the configuration on the left could be interpreted as someone jumping in and out of the car at the same time. **Bottom Row**: Feasible configurations.

Note that all four sums in Eqs. 4.9 and 4.10 can be equal to one. As a result, these constraints allow for a container and a containee object to coexist at the same location and at the same time instant. For scenarios such as cars and people, this can give rise to several undesirable results as shown in the top row of Fig. 4.2. 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

$$\sum_{\substack{k:l=l(k),\\ i:k\in\mathcal{N}(i)}} f^{t-1}_{ik} + \sum_{\substack{k:l=l(k)\\ j\in\mathcal{N}(k)}} f^t_{kj} \leq 2 - \sum_{\substack{k:l=l(k)\\ j\in\mathcal{N}(k)}} g^t_{kj}, \quad \forall t, l\,. \tag{4.11}$$

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^t_{kj}, which stand for the number of invisible containees moving from pose $o(k)$ and location $l(k)$ at time t to pose $o(j)$ and location $l(j)$ at time $t+1$. These variables act as counters that are incremented or decremented when a

containee object respectively disappears or appears in the vicinity of a container. We write

$$\sum_{\substack{k:l=l(k)\\ j\in\mathcal{N}(k)}} h_{kj}^t = \sum_{\substack{k:l=l(k),\\ i:k\in\mathcal{N}(i)}} h_{ik}^{t-1} + \sum_{\substack{k:l=l(k),\\ i:k\in\mathcal{N}(i)}} f_{ik}^{t-1} - \sum_{\substack{k:l=l(k)\\ j\in\mathcal{N}(k)}} f_{kj}^t,\ \forall t,l \tag{4.12}$$

$$h_{kj}^t \le c * g_{kj}^t,\ \ \forall t,k,j : j\in\mathcal{N}(k)\,, \tag{4.13}$$

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, in Eq. 4.12, the h_{kj}^t variables are incremented or decremented always by an integer value. Therefore, we allow h_{kj}^t to be continuous in our optimization, except only those that are connected to the source, i.e., h_{sj}, which we restrict to be integers. Our experimental results show that allowing $h_{kj:k\neq s}^t$ 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

$$\sum_{\substack{v_k^t\in V(t),\\ j\in\mathcal{N}(k)}} h_{kj}^t + \sum_{\substack{v_k^t\in V(t),\\ j\in\mathcal{N}(k)}} f_{kj}^t \le 1\,,\ \ \forall t\,, \tag{4.14}$$

where $V(t)$ denotes the set of graph vertices of time instant t. Together with the invisible flow constraints expressed in Eqs. 4.12 and 4.13, these constraints allow us to keep track of where the ball is and who has possession of it even when it is invisible. Another interesting case arises from the fact that a moving vehicle must have a driver inside. We express this as

$$h_{kj}^t \ge g_{kj}^t,\ \ \forall t,k,j : j\in\mathcal{N}(k), l(k)\in\mathcal{N}_m\big(l(j)\big)\,, \tag{4.15}$$

where $\mathcal{N}_m(l)$ denotes the movement neighborhood of the car at location l.

4.2.3 Mixed Integer Programming

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

$$\sum_{\substack{t\in\{1,\cdots,T\},\\ v_k^t\in V(t)}} \sum_{j\in\mathcal{N}(k)} \left(\alpha_k^t\, f_{kj}^t + \gamma_k^t\, g_{kj}^t\right), \tag{4.16}$$

where

$$\alpha_k^t = -\log\left(\frac{\rho_k^t}{1-\rho_k^t}\right) \text{ and } \gamma_k^t = -\log\left(\frac{\beta_k^t}{1-\beta_k^t}\right). \tag{4.17}$$

This objective is to be minimized subject to the constraints introduced in the previous section. Since there is a deterministic relationship between the occupancy variables (x_k^t, y_k^t) and the flow variables (f_{kj}^t, g_{kj}^t), this is equivalent to maximizing the expression of Eq. 4.4.

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.

4.3 Optimization

In most practical situations, the MIP of Eq. 4.16 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 near real-time performance.

4.3.1 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^t and β_k^t 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) [13] 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 [157].

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^t and β_k^t locations that may correspond to missed detections, while being considerably smaller than the original one. As a result, we can solve the MIP of Eq. 4.16, where the variables are flows that link two pose vertices between two consecutive frames. We will refer this approach as Pruned Intertwined Flows (**PIF**) in the remainder of the chapter. We show in appendix that running the MIP on the resulting subgraph yields results very similar to those obtained on the full graph. The number of vertices of the resulting subgraph is up to 10 times larger than that of the ground truth.

4.3.2 Tracklet-Based Intertwined Flows (TIF)

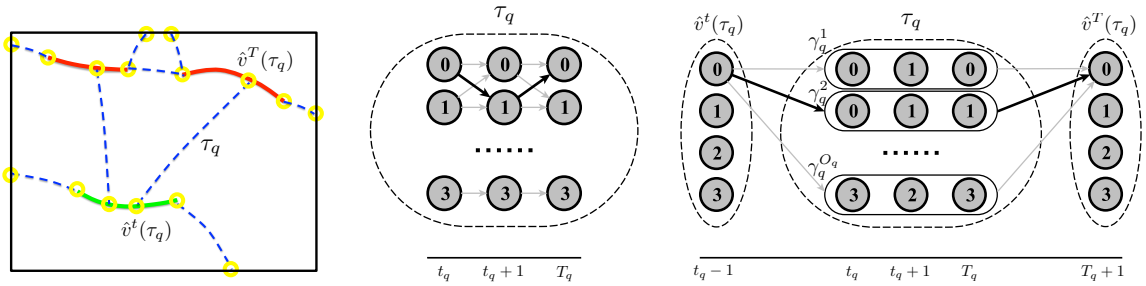
To further reduce the computational complexity while preserving the optimality of the solution on the reduced graph, we introduce a tracklet-based formulation of the optimization problem, which we will refer to as **TIF**. As will be shown in the result section, it yields a speed-up factor of up to three 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. 4.3. We summarize the corresponding workflow below and formalize it in appendix using the notations introduced in Tab. 4.1.

- **Grouping Spatial Vertices into Tracklets** We partition the vertices of the trajectories introduced in Sec. 4.3.1 into two subsets, the shared vertices $\mathscr{S}$ and the non-shared ones $\mathscr{K}$. The vertices in $\mathscr{S}$ are those that are included in more than one trajectory and those at the beginning or end of one. The vertices in $\mathscr{K}$ are the remaining ones and are located between shared vertices. Note that an identity switch or interaction event can

$\mathscr{S}$	set of all shared spatial vertices
$\mathscr{K}$	set of all non-shared spatial tracklets
$\hat{v}_k^t$	spatial vertex at location k at time t
$\mathscr{N}_s(k)$	spatial neighborhood of vertex $\hat{v}_k^t$, i.e., the set of all spatial vertices at time $t+1$ that are can be reached from $\hat{v}_k^t$
τ_q	set of spatial vertices that can be either a spatial tracklet or trajectory
t_q	first time instance of τ_q
T_q	last time instance of τ_q
$\hat{v}^t(\tau_q)$	spatial vertex at time $t_q - 1$ that is in the neighborhood of τ_q
$\hat{v}^T(\tau_q)$	spatial vertex at time $T_q + 1$ that is in the neighborhood of τ_q
γ_q^i	a pose tracklet on τ_q
$c_\rho(\gamma_q^i)$	cost of the pose tracklet γ_q^i of a containee object
$c_\beta(\gamma_q^i)$	cost of the pose tracklet γ_q^i of a container object
O_q	number of poses on the spatial tracklet τ_q
v'_k	pose vertex that represents either a pose on the shared location or a pose tracklet on the non-shared spatial tracklet
e'_{jk}	edge that connects v'_j and v'_k
V'	set of all v'_k
E'	set of all e'_{jk}
G'	the tracklet graph, $G' = (V', E')$
$l'(k)$	spatial location of pose vertex k on G'
$N'(k)$	neighborhood of v'_k on G', i.e., the set of all vertices that is reachable from v'_k on G'
f'_{kj}	binary variable indicating a containee object moving from pose k to pose j on G'
g'_{kj}	binary variable indicating a container object moving from pose k to pose j on G'
h'_{lm}	variable indicating the number of invisible containee objects moving from location l to location m on G'
$\mathscr{N}'_f(k)$	spatial exclusion neighborhood of pose k of a containee object on G'
$\mathscr{N}'_g(k)$	spatial exclusion neighborhood of pose k of a container object on G'

Table 4.1 – Notations for the variables on the pruned graph and those used by the **TIF** approach.



(a) (b) (c)

Figure 4.3 – Construction of the tracklet graph. (a) We obtain a set of paths by the graph size reduction approach as described in Sec. 4.3.1. We use the solid lines to denote the spatial trajectories obtained by the KSP algorithm 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 shared spatial locations. For each spatial tracklet τ_q, there is one unique predecessor spatial vertex and successor spatial vertex, 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 instance of τ_q respectively. We compute pose tracklets on τ_q using dynamic programming. Given the starting pose 0 and ending pose 0, the black arrows denote the pose transitions with the lowest cost and therefore we treat the pose tracklet 0-1-0 as a pose of τ_q. (c) We use γ_q^i to denote a pose of τ_q. Give the starting pose 0 on $\hat{v}^t(\tau_q)$ and the ending pose 0 on $\hat{v}^T(\tau_q)$, the black arrows denote the edge pair with the lowest costs and thus are to be kept, while the gray arrows are to be removed.

occur only at the shared vertices. We therefore collapse non-shared vertices between two shared ones into a single tracklet.

- **Computing the Poses of the Tracklets** Given a spatial tracklet τ_q computed in this way, we take its pose to be a *pose tracklet*, that is, the corresponding set of possible temporally-ordered pose vertices. Since we treat τ_q as a single spatial vertex, its pose is uniquely defined by its starting and ending poses. As a result, we can remove some pose tracklets without loss of optimality. For example, in the case of Fig. 4.3(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. 4.3(b), we would do this $4 \times 4 = 16$ times and retain only 16 pose tracklets.

- **Constructing the Tracklet Graph** We can now construct a new 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 at $\hat{v}^T(\tau_q)$, we only keep one pair of edges that preserve the lowest cost. In the case of Fig. 4.3(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 edge pair with the lost cost and thus should be kept, while the edge pairs in gray can be removed. 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 $\hat{v}^T(\tau_q)$.

Reducing the Number of Counter Variables

The counter variables of Eq. 4.12 are defined on edges connecting pose vertices. However, the constraint of Eq. 4.2 guarantees that there is at most one object moving between two spatial vertices. Therefore, in G' we define the counter variable h' between two spatial vertices instead of two pose vertices. This simple change significantly reduces the number of flow variables. Assuming that the number of neighbors for each pose vertex is N, for a pair of spatial vertices, we reduce the number of counter variables from $N \cdot O$ to 1.

MIP on the Tracklet Graph

Given the notations of Tab. 4.1, our Tracklet-Based Intertwined Flow problem becomes the Mixed Integer Program with respect to the flow variables f', g' and h':

minimize

$$\sum_{v'_k \in V'} \sum_{j \in \mathcal{N}'(k)} \left(c_\rho(v'_k)\ {f'}_{kj} + c_\beta(v'_k)\ {g'}_{kj} \right), \tag{4.18}$$

subject to

$$\sum_{i:k \in \mathcal{N}'(i)} f'_{ik} + \sum_{\substack{j \in \mathcal{N}'_f(k), \\ i:j \in \mathcal{N}'(i)}} f'_{ij} \leq 1, \forall k\,, \tag{4.19}$$

$$\sum_{i:k \in \mathcal{N}'(i)} g'_{ik} + \sum_{\substack{j \in \mathcal{N}'_g(k), \\ i:j \in \mathcal{N}'(i)}} g'_{ij} \leq 1, \forall k\,, \tag{4.20}$$

$$\sum_{i:k \in \mathcal{N}'(i)} g'_{ik} \leq \sum_{j \in \mathcal{N}'(k)} g'_{kj}, \forall k\,, \tag{4.21}$$

$$\sum_{j \in \mathcal{N}'(s)} g'_{sj} \geq \sum_{i:n \in \mathcal{N}'(i)} g'_{in}\,, \tag{4.22}$$

$$-\sum_{\substack{r:l'(k)=l'(r), \\ i:r \in \mathcal{N}'(i)}} g'_{ir} \leq a(k) \leq \sum_{\substack{r:l'(k)=l'(r), \\ j \in \mathcal{N}'(r)}} g'_{rj}, \forall k\,, \tag{4.23}$$

$$a(k) = \sum_{i:k \in \mathcal{N}'(i)} f'_{ik} - \sum_{j \in \mathcal{N}'(k)} f'_{kj}\,, \tag{4.24}$$

$$\sum_{m \in \mathcal{N}_s(l)} h'_{lm} = \sum_{n:l \in \mathcal{N}_s(n)} h'_{nl} + \sum_{\substack{k:l=l'(k), \\ i:k \in \mathcal{N}'(i)}} f'_{ik} - \sum_{\substack{k:l=l'(k) \\ j \in \mathcal{N}'(k)}} f'_{kj},\ \forall l\,, \tag{4.25}$$

$$h'_{lm} \leq \sum_{\substack{k,j:j \in \mathcal{N}'(k) \\ l'(k)=l, l'(j)=m}} c * g'_{kj},\ \ \forall l, m \in \mathcal{N}_s(l)\,, \tag{4.26}$$

$$\sum_{\substack{k:l=l'(k), \\ i:k \in \mathcal{N}'(i)}} f'_{ik} + \sum_{\substack{k:l=l'(k) \\ j \in \mathcal{N}'(k)}} f'_{kj} \leq 2 - \sum_{\substack{k:l=l'(k) \\ j \in \mathcal{N}'(k)}} g'_{kj},\ \ \forall l\,, \tag{4.27}$$

where i, j, k, r index the pose vertices and l, m index spatial vertices. In the car case, we add the constraint that a moving car has at least one driver inside

$$h'_{lm} \geq \sum_{\substack{k:l'(k)=l, \\ l'(j)=m}} g'_{kj},\ \ \forall l, m \in \mathcal{N}_s(l) : m \in \mathcal{N}_m(l) \vee l \in \mathcal{M}, \tag{4.28}$$

where $\mathcal{M}$ denotes the set of all spatial tracklets with movements. In the ball case, we remove constraints Eq. 4.27 to allow the ball and the players to coexist on the same spatial location on

the ground. We replace it by the constraint that there is at most one ball in the court

$$\sum_{\substack{l:t\in T_s(l),\\ m\in\mathcal{N}_s(l)}} h'_{lm} + \sum_{\substack{k:t\in T_p(k),\\ j\in\mathcal{N}'(k)}} f'_{kj} \leq 1\ ,\ \ \forall t\ , \tag{4.29}$$

where $T_s(l)$ and $T_p(k)$ denote the temporal span of spatial vertex l and pose vertex k respectively. Note that we have converted the equalities into inequalities in Eq. 4.21 and Eq. 4.22, which is strictly equivalent so that no additional constraints are needed. This is because Eq. 4.21 ensures that the outgoing flows are greater or equal to incoming flows for all vertices while Eq. 4.21 ensures that sum of flows from the source is greater or equal to those to the sink. Therefore, these two constraints taken together ensure flow conservation just as the equalities do.

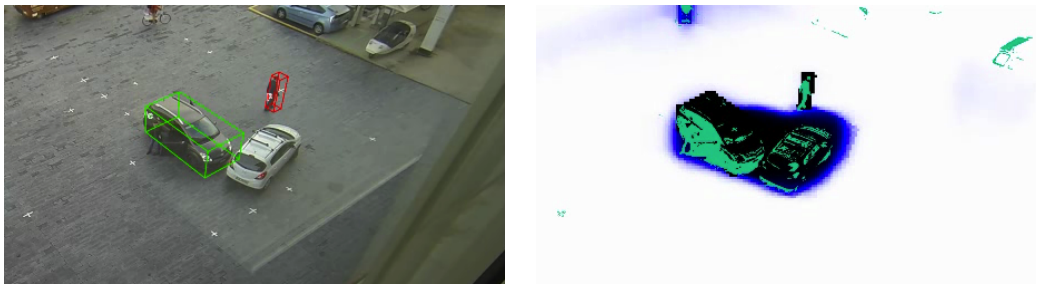
4.3.3 Solving the LP and MIP

We implemented our algorithm in C++ using the Gurobi library V6.0. To solve the MIP, the Gurobi solver first solves a LP and then applies a branch-and-cut procedure to obtain the integer solutions. The LP solution is found by applying a dual simplex algorithm that performs pivots while maintaining the dual feasibility. The branch-and-cut step minimizes the gap between a lower bound obtained from LP relaxations and an upper bound obtained from feasible integer solutions. The algorithm stops when the gap drops below the specified tolerance value. In practice, we set it to $1e^{-3}$, meaning that any solution it finds is very close to the global optimum.

4.4 Estimating Probabilities of Occupancy

Our approach to computing the image-based probabilities of presence ρ_k^t and β_k^t that appear in Eq. 4.3 and Eq. 4.4 is an extension of the one proposed in [11].

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



(a) (b)

Figure 4.4 – Simultaneously detecting people and cars. (a) A person and a car is detected, as indicated by the red and green wireframes. (b) The same boxes are projected and filled as black boxes to create a synthetic image that approximates as closely as possible the background subtraction results, shown in green. Note that the white car is the same as the one that appears in Fig. 1.3. It remains undetected because the background subtraction algorithm fails to extract it.

large system of equations.

Importantly, probabilities computed in this way exhibit the property that allows us to go from Eq. 4.1 to Eq. 4.2 in our derivation of the objective function. We have therefore extended the approach to handling multiple classes of objects simultaneously as follows.

4.4.1 Oriented Objects

To handle oriented objects such as cars or bags, we extend [11] by introducing simple wireframe models to represent them, as shown in Fig. 4.4. The only difficulty is that in the case of cylinders, orientation is irrelevant whereas the projections of our wireframe models depend on it. We solve this by allowing the generative model to model objects of any type at any one of the O regularly spaced orientations. This means that the projections of our 3D models can have arbitrary shapes and that we cannot use the integral image trick of the publicly available software anymore [11]. We therefore use an "integral line" variant, which is comparably efficient. More specifically, we compute an integral image by taking integral of the image values only along the horizontal axis. At detection time, we then take the difference between the left-most and right-most integral pixels of a projected region and sum the resulting differences obtained from each row. This lets us detect objects of different types simultaneously and compute the probabilities of occupancy ρ_k^t and β_k^t introduced in Sec. 4.2.1.

Note however, that the white car in Fig. 4.4 is missed because its color is similar to that of the background used for training, which is taken under direct sunlight. Arguably, we could have used a more powerful car detector but all detectors sometime fail and our contribution is that

our technique can recover from such failures by leveraging information provided by other objects, in this case the people getting out of the car.

4.4.2 Objects off the Ground Plane

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

In theory, this could be handled by treating height over ground as a state variable, much as we do for orientation. However, in the specific case of the basketball competition we show in the result section, when the ball is in the air it is also in front of the spectators, making the background non-constant and the results of [11] unsatisfactory. Therefore, in this specific case, we use a discriminative approach and run a ball detector based on color and roundness in each one of the frames taken at the same time, triangulate the 2D detections to obtain candidate 3D detections. 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. 4.2.

4.5 Experiments

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

4.5.1 Test Sequences

We tested our approach on five datasets featuring four very different scenarios: people and vehicles on a parking lot (Car-People and PETS2000 dataset), people and luggage in a railway station (PETS2006 dataset), basketball players and the ball during a high-level competition (FIBA dataset), and soccer players and the ball in a professional match (ISSIA dataset [120]).

These datasets are either multi-view or monocular, and they all involve multiple people and objects interacting with each other. In Fig. 4.5, we show one image from each dataset with recovered trajectories. We summary the datasets as follows and show more details in Tab. 4.2.

- **Car-People Dataset:** We captured five sequences on a parking lot with two synchronized cameras. The sequences are of about 300 to 5100 temporal frames, and they feature many instances of people getting in and out of the cars. This dataset is challenging because the daylight in the scene changes constantly and the color of the objects are similar to that of the background, which makes the background subtraction prone to errors.

- **PETS2006 Dataset:** We use a 3020-frame sequence acquired by two synchronized cameras that shows people entering and leaving a railway station while carrying bags. Notably, one person brings a backpack into the scene, puts it on the ground, and leaves. The monitored area is relatively small and the pedestrian in the scene heavily occlude each other.

- **PETS2000 Dataset:** We use a 1450-frame monocular sequence showing 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, which makes it hard to detect the ball.

- **ISSIA Dataset:** We use the public available ISSIA dataset [120] that features 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.

4.5.2 Parameters and Baselines

We treat orientation as the pose for cars and luggages, and height as the pose for the ball in the basketball and soccer case. We used twelve regularly distributed orientations for cars and two for luggages. In the basketball and soccer cases, we discretize the height of the ball into 50cm cells. Given the resolution of the cameras, we found our sampling sufficient.

We refer the Pruned Intertwined Flows described in Sec. 4.3.1 as **PIF** and the Tracklet-Based Intertwined Flows in Sec. 4.3.2 as **TIF**. As discussed, **TIF** preserves the completeness of the state space and therefore the Mixed Integer Programs of **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 will compare our **TIF-MIP** approach against the following six methods whose code are public available. These methods have been recently reported to outperform state-of-the-art ones [13, 21, 14].

- **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 Sec. 4.2.2.

- **SSP:** The Successive Shortest Path (SSP) [21] is a algorithm for tracking multiple objects. It first builds a graph by linking pairs of object detections in consecutive temporal frames and then applies Dynamic Programing to find solutions. We run the publicly available SSP code and compared the results with ours.

- **KSP-free:** As discussed in Sec. 4.3.1, the KSP approach of [13] can be used to compute object trajectories for the container and containee objects independently using their occupancy probabilities. We designate all the grid locations as potential entries and exits prior to running the KSP algorithm. As a result, this approach allows objects to appear or disappear at any location at a certain cost value, which we take to be 40.

- **KSP-fixed:** This algorithm is similar to KSP-free, except that we use the original entrances and exits of the scene, such as the edge of the field of view. Therefore, objects can only appear or disappear at these predetermined locations.

- **KSP-sequential:** We first use the KSP-fixed algorithm to track the container objects and designate all the nodes that belong to the resulting trajectories as potential entrances and exits for the containees. We then use the same algorithm to find the containee trajectories, which may emerge from or enter the container ones. In other words, unlike in our approach, the two object classes are *not* treated symmetrically.

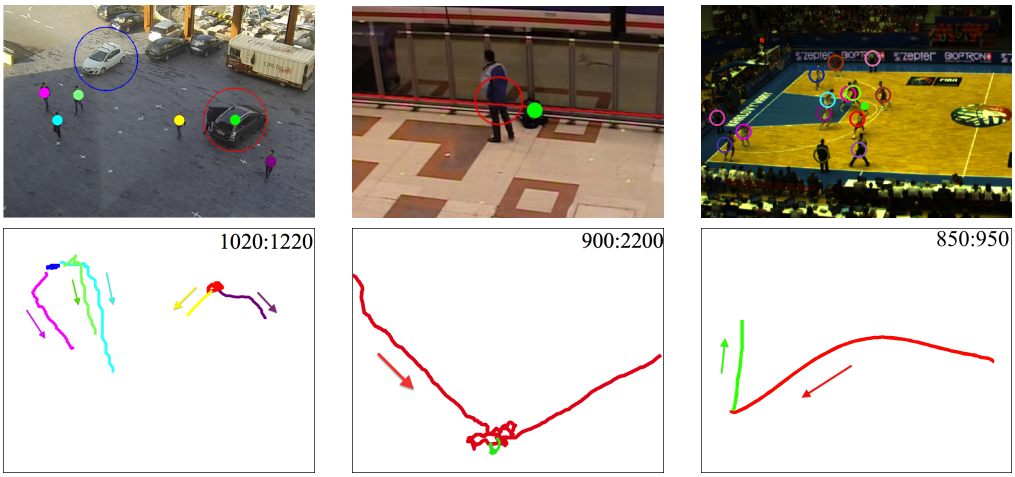
- **TIF-LP:** The Linear Programming approach (LP) solves the problem introduced in Sec. 4.3.2 with the integrality constraints relaxed. The resulting flow variables are then rounded to the nearest integer to obtain the final solution.

The Deformable Part Model (DPM) [158, 159] detects objects on the image plane and does not estimate the object orientations. Therefore, DPM is not suitable for our purpose, as we conduct tracking on the ground plane and express constraints based on orientations of objects.

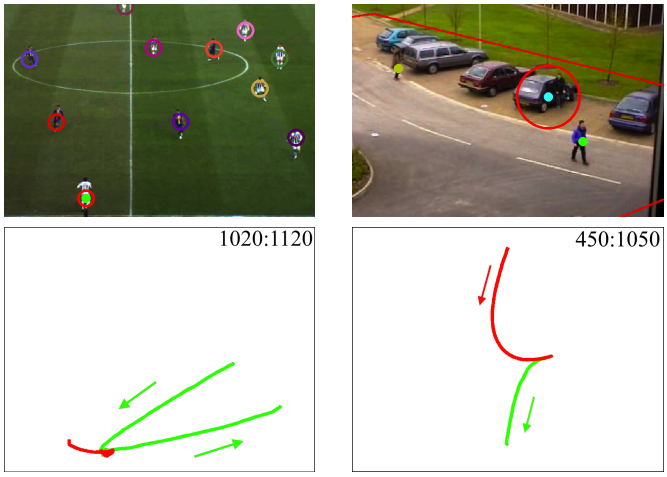
4.5.3 Evaluation Metrics

To quantify the results, we use the Multiple Object Detection Accuracy (MODA) and Multiple Object Tracking Accuracy (MOTA) as we discussed in Sec.2.2. For both MODA and MOTA, False Positive (FP) and False Negative (FN) scores are computed based on a threshold, which is defined to be either the overlap ratio between the detections and the ground truth on the image plane, or the distance to the ground truth on the ground plane. To evaluate oriented objects such as cars, the latter one is not suitable because it does not penalize detections with incorrect orientations, while the previous one accounts for orientation in the image plane. Therefore we used the overlap ratio as the threshold for FP and FN for the Car-People Dataset, PETS2006 Dataset and PETS2000 Dataset. However, for non-oriented objects such as basketball and soccer, the distance to the ground truth on the ground is a good fit because it provides an absolute measurement independent of camera views. We therefore used it as the threshold to compute FP and FN for the FIBA Dataset and ISSIA Dataset.

4.5.4 Results



(a) Car-People (b) PETS2006 (c) FIBA



(d) ISSIA (e) PETS2000

Figure 4.5 – 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. 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.

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	5	1	2
FIBA	6	2850	9728	15	1	1	16
ISSIA	6	1990	34020	25	1	1	17
PETS2000	1	1450	11200	3	3	12	1

Table 4.2 – Summary of our validation sequences. For each sequence, we show the number of cameras, the number of frames, the number of spatial locations, the number of container/containee objects and their number of poses. For the Car-People dataset, PETS2000 and PETS2006 datasets, we take the pose to be the orientation of the targets. For the FIBA and ISSIA dataset, we take the pose to be the height of the ball.

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

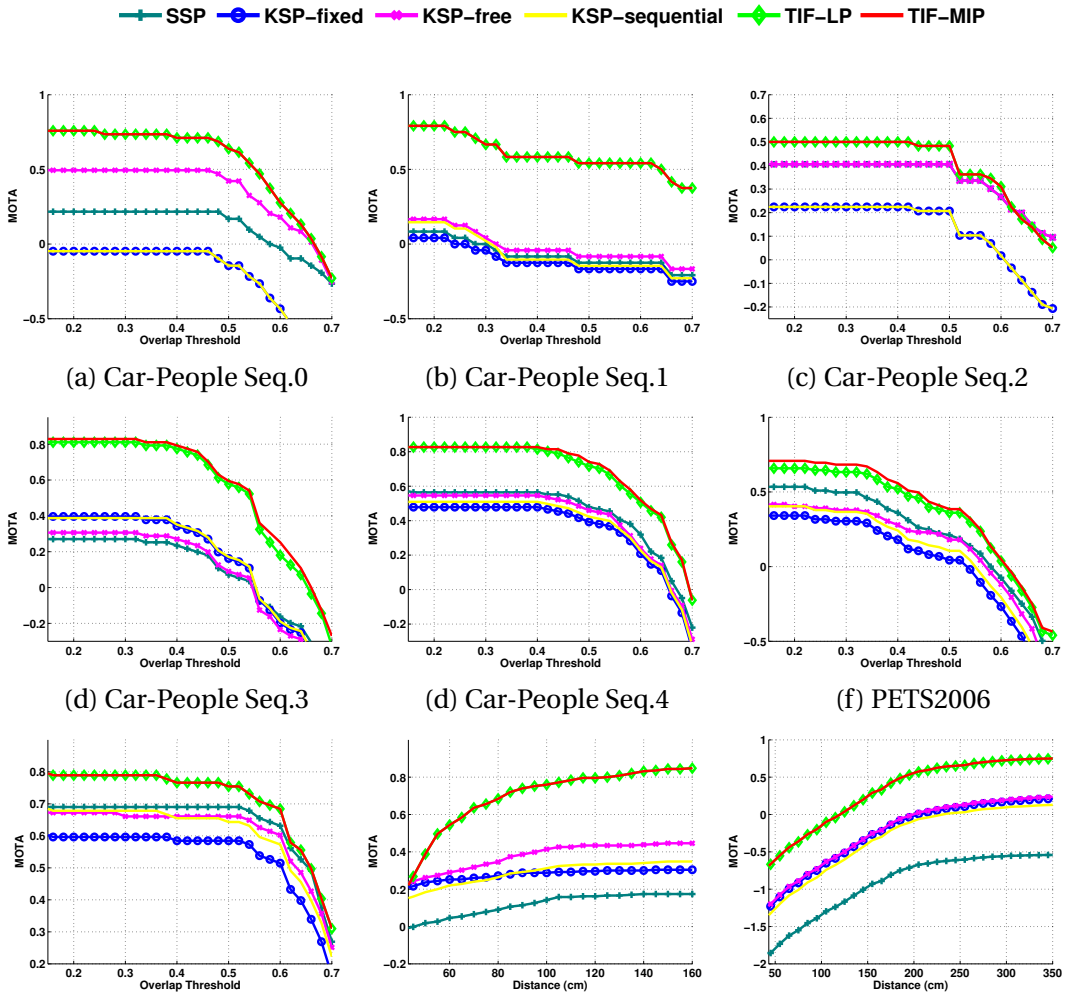
Table 4.3 – Comparison of computational costs for **PIF**, **TIF-LP** and **TIF-MIP**. We show the number of variables, the number of constraints, the total number of non-zero coefficients in the constraint matrix, the time for running the dynamic programming on poses (DP time), the number of iterations, 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 a speed-up factor of a few times 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, and then we use the Hungarian algorithm to glue the batches.

We ran all the baseline algorithms and ours on all the test sequences introduced in Sec. 4.5.1. In Fig. 4.5, we show qualitative tracking results on representative subsequences of each dataset. In Fig. 4.6, we plot MOTA for our approach (**TIF-MIP**) against those of our baselines on all the tested sequences. We use the overlap ratio as the threshold for the Car-People, PETS2006 and PETS2000, and the distance as the threshold for FIBA and ISSIA. We show the results on a range of thresholds as a monotonic curve. In Tab. 4.4, we show the FP rate, FN rate, IDS rate and MODA for all methods at a overlap threshold of 0.5 and distance threshold of 1m for FIBA and 2m for ISSIA.

Quantitative Results (Car-People, PETS2000 and PETS2006)

As we show in Fig. 4.6, our tracker yields a significant improvement 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. 1.3 and the corresponding results are shown in the first column of Fig. 4.6. It involves three people getting into a car stopped at the center of a parking lot. As discussed in Sec. 4.4.1, the POM detector often fails to detect the car due to poor background subtraction. As a result, both KSP-fixed and KSP-sequential yield poor results because they do not create a car track, and hence are forced to explain the people in the scene by hallucinating them entering from the edges of the field of view. SSP and KSP-free do better by allowing the car to appear and disappear as needed but this does not correspond to physically plausible behavior. POM also does better because the people are in fact detected most of the time. **TIF-MIP** approach performs best because the evidence provided by the presence of the people along with the constraint that they can only appear or disappear in the middle of the scene, where there is a stopped car, forces the algorithm to infer that there is one at the right place. As a result, the FN rate is significantly lower than other baselines which further leads to a higher MOTA and MODA score.

The Car-People Seq.1 features two people getting into the first car, staying for a while, and getting out and entering the second one. Here, KSP-sequential and KSP-free do slightly better than KSP-fixed, which needs to hallucinate two false positive tracks to allow for the people emerging from the first car. **TIF** 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 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, for example KSP-sequential, that tracks cars and people separately. The sequences Car-People



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

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

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

Figure 4.6 – 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.

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 jointly.

In the PETS2006 sequence, our approach again performs the best. In particular, a person comes into the scene and puts the bag on the ground, in which case KSP-fixed has to hal-

lucinate a false positive track of the bag from the edge of the monitored area. When the density of people is high in the scene, the baseline algorithms produce solutions that contain overlapping detections in the 3D space. Our tracker enforces the correct constraints that preclude such spurious detections. The same happens for the PETS2000 sequence where our tracker yields better results by imposing the correct constraints and tracking cars and people simultaneously.

Tracking the Invisible (FIBA and ISSIA)

With the help of the counter variables, we are able to track the invisible containee objects when they are contained by containers. In the basketball and soccer case where the ball is often occluded by the players, we utilize the locations of the players to estimate the ball location on the ground. More specifically, when the ball is inferred to be visible, we take its trajectories directly; otherwise, we take the location of the ball to be that of the player who are inferred to be in possession of the ball, or equivalently, locations whose counter variables are non-zero. Since we also impose the constraints that there is at most one ball in the court, in each frame there can be at most one spatial location whose counter variable is one.

In Fig. 4.6.(h) and Fig 4.6.(i), we show the quantitative evaluation of ball-tracking results on the ground. We show the results for the ball only because the people detections are very good and therefore all baseline tracking algorithms perform very similar and the differences would not be visible in print. Because of the weak image evidence, our ball detector gives rises to a large amount of spurious and missing detections, which results in a low MODA score. Among all tested trackers, SSP yields the overall lowest MOTA score, because SSP 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 ground. When the ball is flying fast in the air, KSP-based algorithms enforces 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 of 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. Thanks to the counter variable, we are able to track the ball even when it is occluded by the players.

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

Table 4.4 – 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.

TIF-LP vs. TIF-MIP

Solving the LP problem of Sec. 4.2.3 and subsequently rounding the resulting fractional flow variables as in **TIF-LP** systematically performs either very similarly or 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 constraint 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 sequence, all resulting flows obtained by **TIF-LP** are integers, and therefore the results of **TIF-LP** and **TIF-MIP** are exactly the same. Interestingly, by our experiments we have observed that running **TIF-LP** and **PIF-LP**, that is, running **PIF** without the integrality constraint, in some cases leads to different results, meaning that the LP-relaxed version of **PIF** and **TIF** are not equivalent.

Failure Cases

In the Car-People dataset, we observe a few failure cases where a person gets into the car but the associated counter variable is not incremented. This is because the car is parked on the boundary of the monitored area and the person is detected closer to the boundary than to the car. Therefore the cost of the person disappearing at the boundary is lower and the optimizer prefers the explanation that the person leaves the monitored area than he enters the car. In the FIBA sequence, we observe that in some cases the ball is assigned to the wrong player. This is because there are some spurious ball detections close to players and the optimizer explains it by the playing catching and then throwing the ball.

4.5.5 Computational Cost

We compare the computational costs of **PIF**, **TIF-LP**, and **TIF-MIP** in Tab. 4.3. **TIF** requires much fewer variables and constraints than **PIF**, which yields a speed-up of up 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 iterations needed to solve the LP or MIP, which is independent of the machines we use 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 reduces the number of constraints from 4 million to a quarter. The dynamic programming on poses takes only 0.1 second. The number of iterations drops from 850 thousand to 16

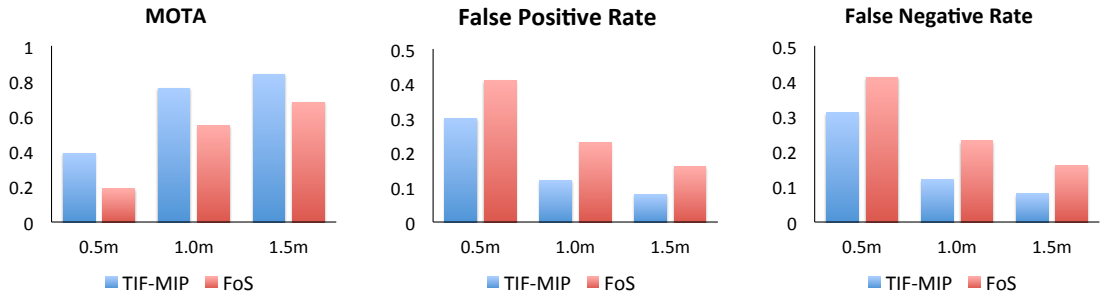


Figure 4.7 – Ball tracking results of **TIF** and FoS on the FIBA dataset. For MOTA, a higher value indicates more accurate tracking; for False Positive (FP) and False Negative (FN) rate, a lower value indicates more accurate tracking.

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 sequence. This can be partially explained by the nature of the match. Because the ball travels fast and there are more container objects on the court as compared to the car sequences, the cost to start or terminate a ball track is lower. Therefore, the optimization is a easier problem to solve as compared to 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**. On the Car-People sequences, all methods run slower but **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 MIP solver introduced in Sec. 4.3.3 on a single thread. We would therefore expect even faster convergence with multi-threads implementations. In any event, despite the large number of variables and constraints, our memory consumption is systematically less than 30GB.

4.5.6 TIF vs. FoS

In Chapter 3, we introduced the FoS approach that tracks the ball by deciding the ball possession. FoS differs with **TIF** in three aspects. First, unlike **TIF** that tracks both players and the ball simultaneously, FoS tracks targets sequentially: it first tracks players and only then tracks the ball. Therefore, in FoS the image evidences of players are used to infer the location of the ball, but not vice versa. Second, FoS operates on a sparse detection graph, while **TIF**

operates on a dense detection graph. In other words, in FoS, we only allow the ball to be at the locations where the ball detector has fired and allow the ball to be with a player, while in **TIF** we discretize the 3D space into a grid of cells and allow the ball to be at any of these cells. Third, **TIF** tracks the ball in the 3D space and accounts for possible transitions of the height of the ball, while FoS tracks the ball on the ground plane. We show the comparative tracking results of **TIF** and FoS in Fig. 4.7. As can be seen, **TIF** yields a better result.

4.6 Conclusion

In this chapter, we introduced a novel framework to simultaneously tracking multiple objects of two different types and accounting for their interactions. This is achieved by modeling the tracking problem as a Mixed Integer Program and expressing the fact that, one object may appear or disappear at the location of another, in terms of linear constraints. This framework also provides an estimate of the location of the objects that are completely hidden or occluded for large portion of the interaction.

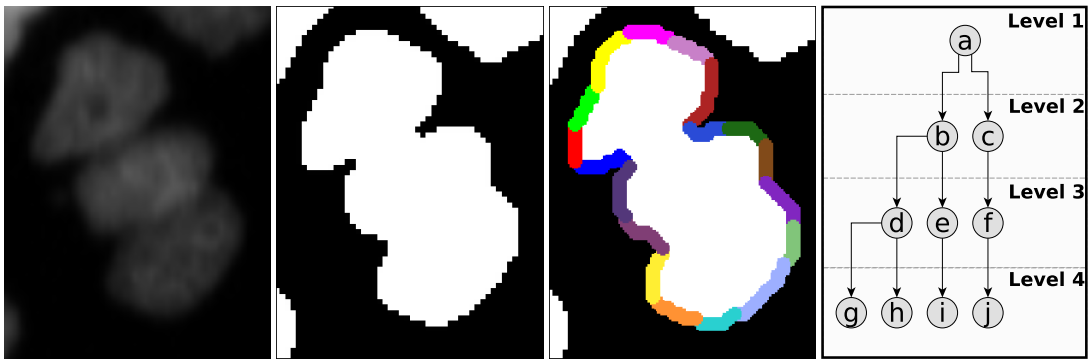
We introduced a tracklet-based implementation that preserves the optimality and achieves near real time performance despite large number of variables. We tested our tracker on a wide range of scenarios, including people getting in and out of cars, pedestrians carrying and dropping luggages, and players passing the ball in professional-level team sports. Furthermore, the framework may be extended to more complex situations, for example, tracking crowds of people where the crowds can be treated as containers and individuals as containees.

5 Tracking Dividing Cells Using Network Flows

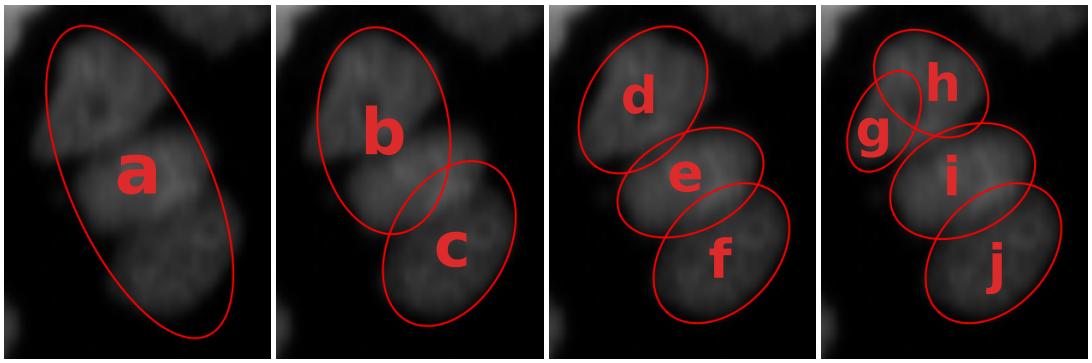
In this chapter, we introduce a novel approach to tracking cells using network flow programming. Unlike common approaches that first detect cells and then link them into full trajectories, we propose a novel approach to generating a hierarchy of competing hypotheses robust to overlaps and occlusions. We then build a spatial-temporal graph of all the hypotheses at all time instances and find the global optimal cell trajectories using Integer Programming.

In Fig. 5.1, we show an example where a foreground region can be explained by ten cell hypotheses from four layers. By considering all such hypotheses from all time instances, our approach computes the optimal cell trajectories on the whole sequence. It eliminates the sub-optimal operations at the detection stage, such as thresholding, non-maximum suppression or adding heuristics to deal with occlusions. Even though we have added another dimension to the tracking problem, our approach requires only one type of flow variables and three sets of constraints that govern appearance/disappearance, division and exclusion of conflicting explanations. We will show that our tracker yields consistently better results as compared to the state-of-the-art cell trackers on challenging sequences.

The rest of this chapter is organized as follows. In Sec. 5.1, we give a brief review on the related algorithms that focus on cell tracking and people tracking. We present our approach in Sec. 5.2 and show comparative tracking results in Sec. 5.3. We conclude this chapter in Sec. 5.4. We show our features used for training the classifier in Appendix C, and show additional comparative tracking results in Appendix D.



(a) Image (b) Segmentation (c) Contourlets (d) Hierarchy



(e) Level 1 (f) Level 2 (g) Level 3 (h) Level 4

Figure 5.1 – Hierarchy of detection hypotheses. (a) An image region containing three HeLa cells clumped together. (b) Applying a pixel classifier results in under-segmentation, in which the three cells appear as a single connected component. (c) Automatically extracted contourlets for this component. Each one is overlaid in a different color. They are used to fit ellipses using a hierarchical agglomerative clustering algorithm. (d) The resulting hierarchy of 10 hypotheses. We show only the first four levels for simplicity. (e-h) Individual levels with one, two, three and four hypotheses. Best viewed in color.

5.1 Related Work

Current cell tracking approaches can be divided into *Tracking by Model Evolution* and *Tracking by Detection* [64]. We briefly discuss state-of-the-art representatives of these two classes below and refer the interested reader to the much more complete recent surveys [160, 64].

5.1.1 Tracking by Model Evolution

Most algorithms in this class simultaneously track and detect objects greedily from frame to frame. This means extrapolating results obtained in earlier frames to process the current one, which can be done at a low computational cost and is therefore fast in practice. Such methods have attracted attention both in the cell tracking field [161, 162, 163, 164] as well as

in the more general object tracking one [138, 4, 165, 166, 167]. Common techniques in the cell tracking category involve evolving appearance or geometry models from one frame to the next, typically done using active contours [161, 162, 163, 164] or Gaussian Mixture Models [168]. Though these methods are attractive and mathematically sound, performance suffers from the fact that they only consider a restricted temporal context and therefore cannot guarantee consistency over a whole sequence.

This limitation has been addressed by more global active contour methods [169, 170] that consider the whole spatio-temporal domain to segment the cells and recover parts of their trajectories. Although this provides improved robustness at the cost of increased computational burden, these approaches do not provide global optimality guarantees either.

5.1.2 Tracking by Detection

Approaches in this class have proved successful at both people [72, 151, 73, 74, 75] and cell tracking [76, 61, 70].

They involve first detecting the target objects in individual frames and then linking these detections to produce full trajectories. This is typically computationally more expensive than Tracking by Model Evolution. However, it also tends to be more robust because trajectories are computed by minimizing a global objective function that enforces consistency of appearance, disappearance, and division over time. This can be seen in the benchmark of [64] in which these methods tended to dominate.

One way to perform tracking-by-detection is to reason in the full spatio-temporal grid formed by stacking up all the spatial locations over time [13, 147, 21]. In the case of non-dividing objects such as pedestrians, this can be done in polynomial time [86]. In spite of the size of the graphs involved, this had made this approach very competitive and real-time implementations have been demonstrated [13]. However, the fact that cells can divide makes the resulting optimization problem NP-Hard, which is mainly why such approaches have not been explored in the cell tracking field. Instead, practical tracking-by-detection approaches typically rely on a small number of strong detections at individual frames to later form complete cell trajectories.

Therefore, the literature in this field has recently focused on solving two main challenges that are specific to cell-tracking, *(a)* cells can divide or die and disappear, and *(b)*, there is no guarantee that individual cells will be detected as separate entities in any given frame because two or more cells can clump together, producing under-segmentation errors, as seen in Fig. 5.1. In the following, we discuss recent efforts to overcome these key challenges.

Division and disappearance. While rule-based approaches have been used to handle division and disappearance, most recent ones formulate tracking as a global integer programming problem [61, 70]. This makes it possible to use priors for cell movement, division and disappearance between adjacent frames. One notable exception is the method of [71], which scored highest in one of the benchmarks [64], which sequentially adds trajectories to a cell lineage tree, using the Viterbi algorithm. Motion, division, and disappearance are encoded through a scoring function that quantifies how well the lineage tree explains the data.

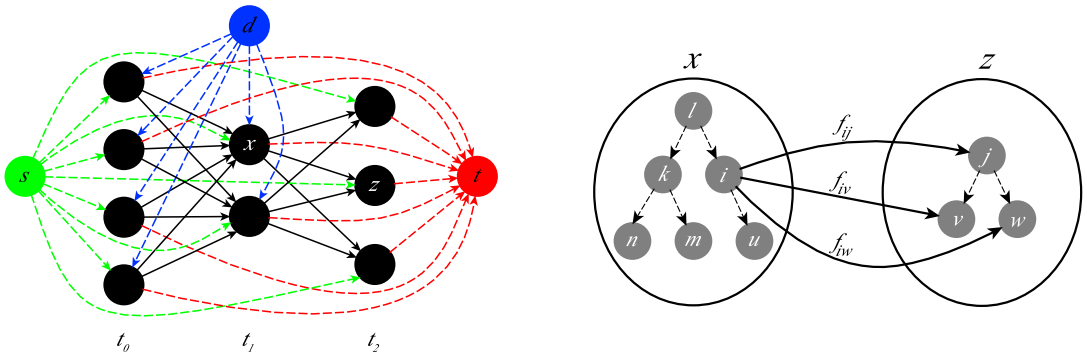
Clumped cells. Similarly, many heuristics have been proposed to solve the problem of clumped cells that cannot easily be told apart. One is to assume that clumped cells are unlikely to happen for a specific modality or that they do not pose a problem for the tracker [76, 61, 71]. While this is appropriate in some cases, it is clearly invalid for certain modalities such as the one shown in Fig. 1.4. A possible solution is to use semi-automated techniques [171] to interactively correct errors made by automated approaches, but this is time consuming for modalities that produce many clumped cells. Other heuristics involve splitting segmentations using the Radon and watershed transforms [161, 64]. Unfortunately, they are still relatively prone to over- and under-segmentation that complicate the tracking step. An ingenious approach is that of [70], which breaks the tracking step into two stages. It first finds trajectories by treating segmentations in each frame as clumps of one or more cells, with the exact number being initially unknown, and then uses a factor graph to resolve this ambiguity. However, because these two steps are performed independently, optimality is not guaranteed.

In the more general case of object detection, tracking groups of objects has been considered as a multiple-hypothesis problem, since there exist many plausible hypotheses to explain what is observed in individual frames. Though multiple-hypothesis tracking has been applied to people tracking [172, 173, 174, 175], to the best of our knowledge, it has not been explored in the context of cell tracking using integer programming.

By contrast, our approach dispenses with such heuristics by allowing multiple competing interpretations of the data, choosing the globally optimal one among them by solving an integer program that enforces consistency over all frames.

5.2 Method

Our approach involves building a spatio-temporal graph of conflicting hypotheses in individual frames, and then finding the most likely trajectories in it. More specifically, we first produce a binary image of the underlying cell populations using a classifier trained on a



(a) (b)

Figure 5.2 – Spatio-temporal graph of hypotheses for three consecutive time frames. (a) Each black circle is a hypervertex corresponding to a connected component of the segmentation and is connected to neighboring ones at the next time step. The special vertices s (source, in green), t (sink, in red) and d (division, in blue) allow respectively for cell appearance, disappearance and division. (b) Each hypervertex, such as x and z, contains a hierarchical set of hypotheses (vertices) such as those depicted by Fig. 5.1. These hypotheses are shown as gray circles and connected to nearby ones in the following frame via directed edges. We only show three of these edges to avoid clutter. Best viewed in color.

few hand-annotated segmentations. For each connected component, we produce multiple detection hypotheses by hierarchically fitting varying number of ellipses to it. This results in a directed graph, such as the one depicted by Fig. 5.2. Its nodes are individual ellipses and its edges connect nearby ones in consecutive frames. Full trajectories can then be obtained by solving an integer program with a small number of constraints that exclude incompatible hypotheses and enforce consistency while allowing for cell-division, migration and death.

In the following, we first describe our approach to segmenting cell images and building hypotheses graphs from them. We then formulate the simultaneous detection and tracking problem on these graphs as a constrained network flow programming problem, and discuss how we compute the various energy terms of its objective function.

5.2.1 Building Hierarchy Graphs

Our algorithm, like those of [76, 61, 70], starts by segmenting cells using local image features. To this end, we first train the binary random forest pixel classifier of [176] for each evaluation dataset on a few partially annotated images. We use four different types of low-level features: pixel intensities, gradient and hessian values, and difference of Gaussians, all of which are computed by first Gaussian smoothing the input image with a range of sigma values.

Applying the resulting classifier to the full image sequences results in segmentations which

often contain groups of clumped cells, such as the ones shown in Fig. 1.4. Therefore, each connected component of the segmentation potentially contains an a priori unknown number of cells.

We produce a hierarchy of conflicting detection hypotheses for each such component by fitting a varying number of ellipses to its contours. More specifically, we first identify all the contour points that are local maxima of curvature magnitude. This is done iteratively by selecting the maximum curvature points and suppressing their local neighborhoods. We then break the contour into short segments at these points, which yields a number of contourlets as shown in Fig 5.1(c). We cluster them in a hierarchical agglomerative fashion and fit ellipses to each resulting cluster using the non-iterative least squares approach of [177]. In all our experiments, we set the size of the suppression neighborhood to seven pixels because this is the minimum number of points required to reliably fit an ellipse using this approach.

Let C denote the set of all contourlet clusters for a connected component. Given a pair of clusters $C_i \in C$ and $C_j \in C$, we define their distance to be

$$\left[\sum_{C_l \in \{C_i \cup C_j\}} h(C_l, e) + \sum_{C_l \in C \setminus \{C_i \cup C_j\}} g(C_l, e) \right] c(e) \sqrt{\frac{1}{1 + ec^2(e)}}\,, \tag{5.1}$$

where e is the ellipse obtained by fitting to the points of $C_i \cup C_j$, and $c(e)$ and $ec(e)$ are its circumference and eccentricity respectively. $h(C_l, e)$ denotes the Hausdorff distance between the points of C_l and the ellipse e. The function $g(C_l, e)$ is defined in a similar way but without considering the points of C_l that are outside e.

The first term in the product captures image evidence along the contours of the entire connected component while the last two ones act as a shape regularizer to prevent implausible ellipse geometries from appearing in the solution. We use this distance measure to compute an ellipse hierarchy, such as the one of Fig. 5.1, for every connected component in the temporal sequence.

In the hierarchy such as the one of Fig. 5.1(d), each node corresponds to an ellipse or a potential cell detection, while each edge defines a conflict relation, meaning that the two incident nodes of the edge are mutually exclusive and at most one can be selected. In practice, we define the conflict based on overlap ratio between ellipses. For example, in the case of Fig. 5.1(d), ellipse f and ellipse j share a significant overlap and thus they conflict each other. To construct the hierarchy, we go through each leaf node and backtrack to the root node, and in this way we obtain a set of conflicting hypotheses for each leaf node. The ellipses in each

such set conflict with one another and at most one ellipse from each set can be selected. Since we go through each leaf node, the number of such sets are equal to the number of the leaf nodes in the hierarchy.

Given these over-complete hierarchy of detections, we then build a graph, whose vertices are the ellipses and the edges link pairs of them that belong to two spatially close connected components in consecutive frames. This is similar in spirit to recent graph-based approaches [61, 70], but with the key difference that our graphs have a hierarchical dimension that allows for finding the globally optimal detection hypotheses and trajectories in a single shot.

5.2.2 Network Flow Formalism

The procedure described above yields a directed graph $G' = (V', E')$, which we then augment with three distinguished vertices; namely the source s, the sink t and the division d as depicted by Fig. 5.2. We connect these three vertices to every other vertex in G' to allow accounting for cell appearance, disappearance and division in mid-sequence.

Let $G = (V, E)$ be the resulting graph obtained after the augmentation. We define a binary flow variable f_{ij} for each edge $e_{ij} \in E$ to indicate the presence of any one of the following cellular events:

- Cell migration from vertex i to vertex j,
- Appearance at vertex j, if $i = s$,
- Division at vertex j, if $i = d$,
- Disappearance at vertex i, if $j = t$.

Let $\mathbf{f}$ be the set of all f_{ij} flow variables and $\mathbf{F} = \{F_{ij}\}$ be the set of all corresponding hidden variables. Given an image sequence $\mathbf{I} = (\mathbf{I}^1, \ldots, \mathbf{I}^T)$ with T temporal frames and the corresponding

graph G, we look for the optimal trajectories $\mathbf{f}^*$ in G as the solution of

$$\begin{aligned}
\mathbf{f}^* &= \underset{\mathbf{f}\in\mathscr{F}}{\operatorname{argmax}} P(\mathbf{F}=\mathbf{f} \mid \mathbf{I}) && (5.2)\\
&\approx \underset{\mathbf{f}\in\mathscr{F}}{\operatorname{argmax}} \prod_{e_{ij}\in E} P(F_{ij}=f_{ij} \mid \mathbf{I}^{t(i)},\mathbf{I}^{t(j)}) && (5.3)\\
&= \underset{\mathbf{f}\in\mathscr{F}}{\operatorname{argmax}} \prod_{e_{ij}\in E} P(F_{ij}=1 \mid \mathbf{I}^{t(i)},\mathbf{I}^{t(j)})^{f_{ij}} \times P(F_{ij}=0 \mid \mathbf{I}^{t(i)},\mathbf{I}^{t(j)})^{(1-f_{ij})} && (5.4)\\
&= \underset{\mathbf{f}\in\mathscr{F}}{\operatorname{argmax}} \sum_{e_{ij}\in E} \log\left(\frac{P(F_{ij}=1 \mid \mathbf{I}^{t(i)},\mathbf{I}^{t(j)})}{P(F_{ij}=0 \mid \mathbf{I}^{t(i)},\mathbf{I}^{t(j)})}\right) f_{ij} && (5.5)\\
&= \underset{\mathbf{f}\in\mathscr{F}}{\operatorname{argmax}} \sum_{e_{ij}\in E'} \log\left(\frac{\rho_{ij}}{1-\rho_{ij}}\right) f_{ij} + \sum_{j\in V} \log\left(\frac{\rho_a}{1-\rho_a}\right) f_{sj} + \sum_{i\in V} \log\left(\frac{\rho_d}{1-\rho_d}\right) f_{it} + \sum_{j\in V} \log\left(\frac{\rho_j}{1-\rho_j}\right) f_{dj} && (5.6)
\end{aligned}$$

where $\mathbf{I}^{t(i)}$ is the temporal frame containing vertex i, and $\mathscr{F}$ denote the set of all feasible cell trajectories, which satisfy linear constraints. In Eq. 5.3, we assume that the flow variables F_{ij} are conditionally independent given the evidence from consecutive frame pairs. Eqs. 5.4 and 5.5 are obtained by using the fact that the flow variables are binary and by taking the logarithm of the product. Finally, in Eq. 5.6, we split the sum into four parts corresponding to the four events mentioned above.

The appearance and disappearance probabilities, ρ_a and ρ_d, are computed simply by finding the relative frequency of these events in the ground truth cell lineages of the training sequences. On the other hand, the migration and the division probabilities, ρ_{ij} and ρ_j, are obtained using a classification approach as described in the next section.

We define three sets of linear constraints to model cell behavior and exclude conflicting detection hypotheses from the solution, which we describe in the following.

Conservation of Flow: We require the sum of the flows incoming to a vertex to be equal to the sum of the outgoing flows. This allows for all the four cellular events while incurring their respective costs given in Eq. 5.6. We write

$$\sum_{e_{ij}\in E'} f_{ij} + f_{sj} + f_{dj} = \sum_{e_{jk}\in E'} f_{jk} + f_{jt}\,, \qquad \forall\, j \in V'\,. \tag{5.7}$$

Prerequisite for Division: We allow division to take place at a vertex $j \in V'$ only if there is a cell at that location. We write this as

$$\sum_{e_{ij} \in E'} f_{ij} + f_{sj} \geq f_{dj} \,, \qquad \forall j \in V' \,. \tag{5.8}$$

Exclusion of Conflicting Hypotheses: Given a hierarchy tree of detections, we define an exclusion set S_l for each terminal vertex $l \in V'$ of this tree. For instance, in the example of Fig. 5.1(d), the four exclusion sets are $\{a, b, d, g\}$, $\{a, b, d, h\}$, $\{a, b, e, i\}$ and $\{a, c, f, j\}$. Let S be the collection of all such sets for all the connected components in the sequence. We disallow more than one vertex from each set to appear in the solution, and express this as

$$\sum_{\substack{j \in S_l, \\ e_{ij} \in E'}} f_{ij} + \sum_{j \in S_l} f_{sj} \leq 1, \qquad \forall S_l \in S \,. \tag{5.9}$$

We solve the resulting integer programs within an optimality tolerance of $1e^{-3}$ using the branch-and-cut algorithm implemented in the Gurobi optimization library [69].

5.2.3 Cell Migration and Division Classifiers

Given two adjacent vertices $i, j \in V'$, and their associated ellipses e_i and e_j in consecutive time frames, we train a classifier to estimate the likelihood that both belong to the same cell. More specifically, we use a Gradient Boosted Tree (GBT) classifier [178] to learn a function $\varphi_{\text{migr}}(e_i, e_j) \in \mathbb{R}$, based on both appearance and geometry features including the distance between the ellipses, their eccentricities and degree of overlap, hierarchical fitting errors and ray features. Once $\varphi_{\text{migr}}(e_i, e_j)$ is learned, we apply Platt scaling [179] to compute the migration probability ρ_{ij} that the two ellipses belong to the same cell, and plug it into Eq. 5.6.

Similarly, the likelihood of ellipse e_j at time t dividing into two ellipses e_k and e_l at $t+1$ is learned with another GBT classifier, trained on features such as the orientation and size differences among the ellipses. We present a detailed list of both the migration and division features in the appendix.

For prediction, we compute the division score for ellipse e_j at time t as

$$\varphi_{\text{div}}(e_j) = \max_{\substack{e_{jk}\in E',\ e_{jl}\in E': \\ k\neq l}} \varphi_{\text{div}}(e_j, e_k, e_l)\,, \tag{5.10}$$

where $\varphi_{\text{div}}(\cdot,\cdot,\cdot)$ is the scoring function learned by the classifier, and (e_k, e_l) is a pair of ellipses corresponding to two potential daughter cells at time $t+1$. We obtain the division probability ρ_j of Eq. 5.6 from $\varphi_{\text{div}}(e_j)$, again using Platt scaling.

5.3 Experiments

In this section, we first introduce the datasets and state-of-the-art baseline methods we use for evaluation purposes. We then demonstrate that our approach significantly outperforms these baselines, especially when the cells divide or are not well separated in the initial segmentations.

5.3.1 Test Sequences

We used 10 image sequences from three datasets of the cell tracking challenge [65]. They involve multiple cells that migrate, appear, disappear, and divide. Difficulties arise from low contrast with the background, complex cell morphology, and significant mutual overlap. We used the leave-one-out training and testing scheme within each dataset to train the classifiers of Section 5.2.3 as well as to learn the appearance and disappearance probabilities of Eq. 5.6.

- **HeLa Dataset:** It comprises two 92-frame sequences from the MitoCheck consortium. Cell divisions are frequent, which produces a dense population with severe occlusions.

- **SIM Dataset:** It comprises six 50- to 100-frame sequences. They simulate migrating and dividing nuclei on a flat surface.

- **GOWT Dataset:** It comprises two 92-frame sequences of mouse stem cells. Their appearance varies widely and some have low contrast against a noisy background.

5.3.2 Baselines

We compared our algorithm (**OURS**) against the following three state-of-the-art methods

- **Gaussian Mixture-based Tracker (GMM) [168]:** We run the Gaussian Mixture Models approach of [168], originally designed to track cell nuclei, whose code is publicly available. We manually tuned its parameters to the ones that yield the best results on each sequence.

- **KTH Cell Tracker (KTH) [71]:** The code is publicly available and has been reported to perform best in the Cell Tracking Challenge [65, 64]. We used the parameter settings optimized for each dataset and provided in the software package. By contrast, our own algorithm does not require any user-defined parameters.

- **Conservation Tracking (CT) [70]:** We run Ilastik V1.1.3 [180] and enabled the C-T functionality that implements the method of [70]. We used the default parameters provided with the tool to handle appearance, disappearance, division and transition weights. The **CT** algorithm, like ours, requires initial segmentations such as the ones shown in the second column of Fig.1.4. We used the same segmentations for both algorithms. We trained the division and the segment count classifiers of **CT** separately for each dataset on manually labeled cells.

To demonstrate the importance of individual components of our approach, we also ran simplified versions of **OURS** with various features turned off:

- **Classifier Only (OURS-CL):** We threshold the output of our migration and division classifiers at a probability of 0.5 and return the resulting ellipse detections.

- **Best Hierarchy Only (OURS-BH):** For each ellipse hierarchy tree, we only keep the level that yields the minimum fitting error, which we define Appendix C. We then run our IP optimization on the resulting graphs, which are smaller than the ones we normally use.

- **Linear Programming Relaxation (OURS-LP):** We relax the integrality constraint on the flow variables and solve the optimization problem of Eq. 5.6 using linear programming. We then round the resulting fractional flows to the nearest integer to produce the final solution.

- **No Conflict Set Constraint (OURS-NC)**: We remove the conflict set constraints of Eq. 5.9 and solve the resulting integer program as before.

- **Fixed Division Cost (OURS-FD)**: We set the division probability to a constant p_d, which we compute by finding the relative frequency of the division event in the training sequences.

		Division			Detection			Migration			MOTA	Time
		Rec	Pre.	F-M.	Rec.	Pre.	F-M.	Rec.	Pre.	F-M.		
HeLa-1	**GMM**	0.56	0.43	0.48	N/A	N/A	N/A	0.92	0.98	0.95	0.82	**44**
	KTH	0.65	0.72	0.68	N/A	N/A	N/A	0.95	**0.99**	0.97	0.91	70
	CT	0.74	**0.79**	0.77	0.56	**0.89**	0.69	0.94	**0.99**	0.97	N/A	74.85
	OURS	**0.92**	**0.79**	**0.85**	**0.96**	0.83	**0.89**	**0.97**	**0.99**	**0.98**	**0.94**	88.25
HeLa-2	**GMM**	0.40	0.18	0.24	N/A	N/A	N/A	0.95	0.98	0.97	0.43	**74**
	KTH	0.65	0.72	0.68	N/A	N/A	N/A	0.94	**0.99**	**0.97**	**0.90**	336
	CT	0.76	0.81	0.78	0.73	0.63	0.67	0.94	**0.99**	0.96	N/A	79.33
	OURS	**0.86**	**0.83**	**0.84**	**0.86**	**0.78**	**0.82**	**0.96**	**0.99**	**0.97**	**0.90**	232.31
GOWT-2	**GMM**	0.0	0.0	0.0	N/A	N/A	N/A	0.16	0.79	0.26	0.02	37
	KTH	0.0	N/A	0.0	N/A	N/A	N/A	0.94	**1.0**	0.97	0.94	16
	CT	**1.0**	0.17	0.29	**1.0**	0.02	0.03	0.95	**1.0**	0.97	N/A	0.81
	OURS	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.96**	**1.0**	**0.98**	**0.96**	**0.22**
SIM-4	**GMM**	0.25	0.33	0.29	N/A	N/A	N/A	0.91	0.94	0.92	0.81	23
	KTH	0.75	0.75	0.75	N/A	N/A	N/A	0.97	0.99	0.98	**0.96**	5
	CT	0.75	0.60	0.67	0.75	0.68	0.72	0.86	0.97	0.92	N/A	**1.69**
	OURS	**1.0**	**0.80**	**0.89**	**1.0**	**0.79**	**0.88**	**0.98**	**1.0**	**0.99**	**0.96**	1.85

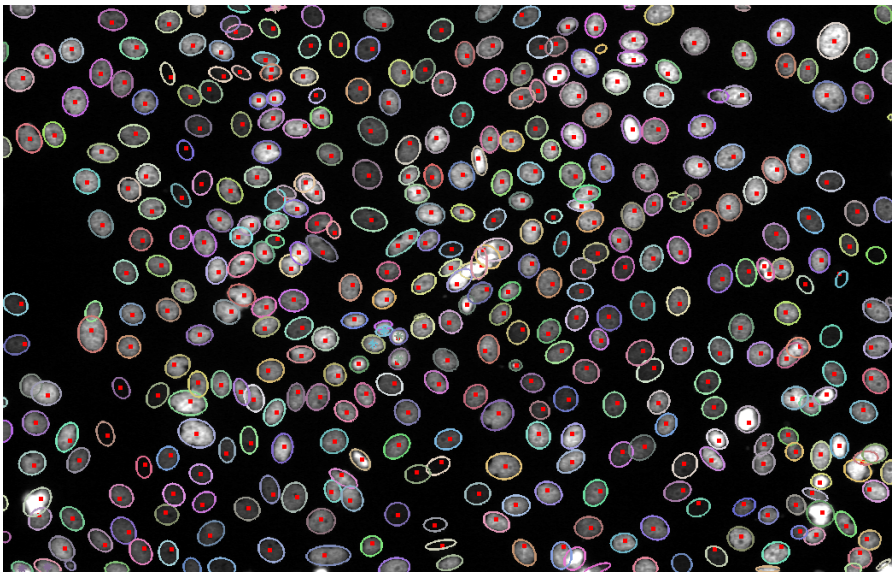
Table 5.1 – Comparison of our algorithm against state-of-the-art cell trackers and running time. It yields a significant improvement on the division and detection accuracies and performs either on par or slightly better on the migration and MOTA scores.

5.3.3 Evaluation Metrics

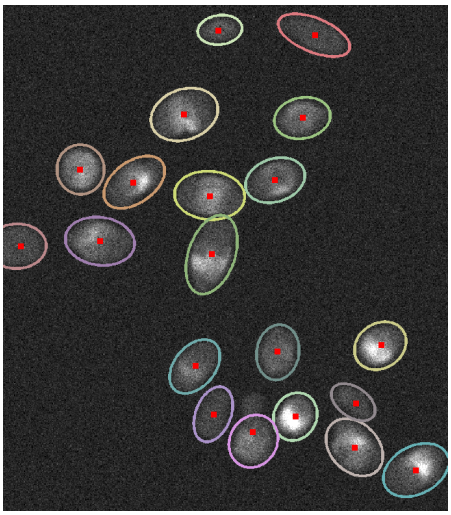
We use precision, recall and the F-Measure, defined as the harmonic mean of precision and recall, to quantify the algorithms' ability to detect cell division, detection, and migration events. We also use the more global Multiple Object Tracking Accuracy (MOTA) metric as we discussed in Sec. 2.2 to evaluate the overall tracking accuracy for each sequence.

We follow the same evaluation methodology described in [70], which uses the connected components of the initial segmentations to compute the division, detection, and migration accuracies. More specifically, we consider a cell migration event to be successfully detected if the two connected components of the cell at t and $t+1$ are correctly identified. Similarly, we consider a division event to be successfully detected if it occurs at the correct time instant with the connected components of the parent and both daughter cells correctly determined. Finally, the detection event is said to be successfully identified if an algorithm infers the right number of cells within a connected component.

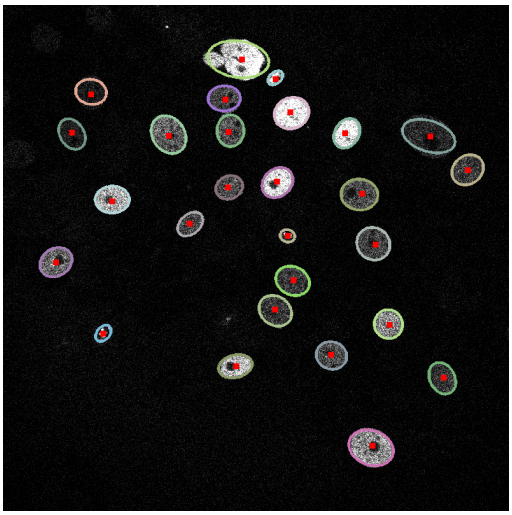
Unlike these measures, the MOTA metric is defined on individual cell tracks rather than connected components that potentially contain multiple clumped cells. It therefore provides a global picture of tracking performance.



(a)



(b)



(c)

Figure 5.3 – Cell Tracking Results on (a) HeLa dataset, (b) SIM dataset and (c) GOWT dataset. Cell indentity is encoded in color and ground truth is denoted by red dot.

5.3.4 Results

Comparing against the Baselines

We ran our algorithm and the baselines discussed above on all the test sequences introduced in Section 5.3.1. Table D.1 summarizes the results for a representative subset and the remainder can be found in Appendix D.

Some numbers are missing because the publicly-available implementation of **CT** [70] does not provide the identities of individual cells in under-segmentation cases. We therefore

cannot extract the complete tracks required to compute the MOTA scores. Similarly, the KTH tracker [71] takes raw images as input and does not use the initial segmentations. Therefore, we cannot compute its accuracy for the detection event.

Table D.1 shows that our tracker consistently yields a significant improvement on the division and detection events. Even on the migration events for which the baselines already perform very well, we do slightly better. However, because the division and detection events are rare compared to migrations, the significant improvements on the first two events only have a small impact on the MOTA scores.

An interesting case is that of **GMM**, which performs poorly on all three events. This could be explained by the fact that **GMM** relies on a simple hand-designed appearance and geometry model to detect individual cells. On the other hand, **KTH** assumes a more flexible appearance model but requires a higher number of parameters to be tuned for each sequence. These differences help explain why it performs better than **GMM**. Finally, **CT** employs a classification approach to segment the cells, resulting in more accurate detections. However, none of the baselines handle the under-segmentations in a global manner, and are outperformed by our approach, particularly for division and detection events.

Our tracker runs relatively fast even though we use a large number of variables. In the SIM-4 case, the number of flow variables is around 1.1 million but the integer programming optimization takes only 2 seconds. In the HeLa-2 case, the number of variables is around 8 million and the optimization takes about 360 seconds.

Evaluating the Importance of Various Components

To produce the results summarized by Table D.1, we used our full approach as described in Section 5.2. In Table D.2, we show what happens when we turn off some of its components to gauge their respective impacts.

OURS-CL relies on local classifier scores and does not impose temporal consistency. That is why, it produces a large number of spurious cell tracks. **OURS-NC** addresses this by imposing temporal consistency globally over the whole sequence but it allows multiple conflicting hypotheses to be active simultaneously. Therefore, it still suffers from spurious detections, which leads to low precision. **OURS-FD** disallows conflicting detections but relies on a fixed division probability, which is why it gives low division performance. **OURS-BH** uses division classifier costs but collapses the hierarchical dimension of our graphs and results in mis-detections. Finally, **OURS-LP** removes the integrality constraints on the flow variables. This

		Division			Detection			Migration			MOTA
		Rec	Pre.	F-M.	Rec.	Pre.	F-M.	Rec.	Pre.	F-M.	
HeLa-1	**OURS-CL**	0.95	0.06	0.11	N/A	N/A	N/A	0.98	0.47	0.64	N/A
	OURS-NC	0.48	0.14	0.22	0.0	0.0	0.0	0.96	0.93	0.94	-0.61
	OURS-FD	0.78	0.81	0.80	0.96	0.85	0.90	0.97	0.99	0.98	0.93
	OURS-BH	0.88	0.73	0.80	0.81	0.87	0.84	0.97	0.99	0.98	0.94
	OURS-LP	0.92	0.80	0.86	0.95	0.81	0.87	0.97	0.99	0.98	0.93
	OURS	0.92	0.79	0.85	0.96	0.83	0.89	0.97	0.99	0.98	0.94
HeLa-2	**OURS-CL**	0.91	0.10	0.18	N/A	N/A	N/A	0.92	0.60	0.72	N/A
	OURS-NC	0.73	0.31	0.43	0.06	0.02	0.02	0.95	0.96	0.95	0.35
	OURS-FD	0.77	0.82	0.80	0.84	0.78	0.81	0.96	0.98	0.97	0.90
	OURS-BH	0.84	0.77	0.81	0.78	0.77	0.77	0.95	0.99	0.97	0.90
	OURS-LP	0.85	0.83	0.84	0.84	0.78	0.81	0.95	0.99	0.97	0.90
	OURS	0.86	0.83	0.84	0.86	0.78	0.82	0.96	0.99	0.97	0.90
GOWT-2	**OURS-CL**	0.0	0.0	0.0	N/A	N/A	N/A	0.94	0.94	0.94	N/A
	OURS-NC	1.0	0.20	0.33	1.0	0.02	0.03	0.96	1.0	0.98	0.93
	OURS-FD	1.0	0.25	0.40	1.0	1.0	1.0	0.96	1.0	0.98	0.96
	OURS-BH	0.0	N/A	0.0	1.0	1.0	1.0	0.91	1.0	0.95	0.91
	OURS-LP	1.0	0.50	0.67	1.0	0.50	0.67	0.96	1.0	0.98	0.96
	OURS	1.0	1.0	1.0	1.0	1.0	1.0	0.96	1.0	0.98	0.96
SIM-4	**OURS-CL**	0.75	0.27	0.40	N/A	N/A	N/A	0.92	0.56	0.70	N/A
	OURS-NC	0.75	0.21	0.33	0.37	0.19	0.25	0.97	0.98	0.98	0.58
	OURS-FD	0.75	0.75	0.75	0.98	0.78	0.87	0.98	1.0	0.99	0.95
	OURS-BH	1.0	0.80	0.89	1.0	0.78	0.87	0.98	1.0	0.99	0.96
	OURS-LP	1.0	0.80	0.89	1.0	0.79	0.88	0.98	1.0	0.99	0.96
	OURS	1.0	0.80	0.89	1.0	0.79	0.88	0.98	1.0	0.99	0.96

Table 5.2 – Tracking results with various features turned off. **OURS-CL** does not impose temporal consistency and suffers from low precision. **OURS-NC** imposes temporal consistency in the optimization but allows multiple conflicting hypotheses to appear in the solution, which yields a low precision. **OURS-FD** eliminates competing hypotheses but uses a fixed cost for the division event. **OURS-BH** performs non-maxima suppression on the hierarchical dimension and hence suffers from mis-detection errors. **OURS-LP** yields fractional flows and the rounding stage eliminates some cell tracks, which leads to a slight drop in performance. With all its features turned on, **OURS** achieves the best overall performance.

gives a similar performance to **OURS** on most of the sequences suggesting that the integrality constraints are seldom helpful. However, in the case of HeLa-2, where division events are frequent, we observed that around 3% of the non-zero flow variables are fractional, which explains the 2% drop in recall compared to **OURS**.

5.4 Conclusion

In this chapter, we introduced a network flow programming approach to detecting and tracking cells in time-lapse images. Our approach generates an over-complete set of conflicting detection hypotheses, and formulates the tracking problem as an Integer Programming whose

optimal solution can be found using standard library. This is in contrast to conventional approaches, which rely on suboptimal heuristics to handle mis-detections due to clumped cells and occlusions. Our approach works better than the state-of-art methods on challenging sequences, especially in detection of mitosis events.

6 Concluding Remarks

In this thesis, we have presented a number of solutions for tracking interacting objects. Unlike conventional methods that focus on one class of objects and model only simple interactions between the targets, our methods incorporate the relationships between the targets in the tracking formulation to achieve dependable tracking. We have demonstrated our methods on a wide range of applications, such as tracking players and the ball in team sports, tracking cars and passengers in a parking lot, tracking people and their luggage in a railway station, as well as tracking dividing cells in biomedical imagery.

In this chapter, we first summarize the contributions of this thesis. We then present some of our research collaborations with other researchers in the field. We end by discussing limitations and future research directions.

6.1 Summary

We began by reviewing the two related algorithms in Chapter 2, namely the Probability Occupancy Map (POM) and K-Shortest Path (KSP), from which we obtained our inspiration for this thesis. The two algorithms are originally intended for tracking people on the ground plane. POM takes binary images as input, and produces estimated probabilities of occupancy on the ground. KSP takes estimated occupancy probabilities of each frame as input, models the transitions of objects using flow variables and solves the data association problem in a global optimal way. However, KSP is applicable only to one class of objects and do not account for complex interactions between objects.

In Chapter 3, we presented a novel approach to tracking the "invisible" ball in team sports. In contrast to the conventional ball-tracking approaches, our approach first tracks players,

decides ball possession and then utilizes players' trajectories to achieve reliable ball tracking. By introducing a novel state space that explicitly accounts for the ball possession, we define our loss function as a Conditional Random Field (CRF), wherein the unary and pairwise potentials are conditioned on image evidence. The unary potential is factorized with respect to the ball possession and the phase of the game, while the pairwise potential is factorized with respect to tactics and context. We applied our proposed tracker on challenging basketball and soccer sequences. Our approach is superior to the state-of-the-art ones using standard evaluation metrics.

In Chapter 4, we introduced a framework to track multiple objects of different types and at the same time accounting for their complex and dynamic interactions. It relies on Mixed Integer Programming (MIP) and ensures convergence to a global optimum using a standard optimizer. Furthermore, it provides an estimate for the *implicit* transport of objects for which the only evidence is the presence of other objects that can contain or carry them. We proposed a tracklet-based implementation that preserves the optimality and yields near real-time performance. We demonstrated our method on real-world sequences including various interactions between people and other objects, such as people getting in and getting out of cars, pedestrians carrying and dropping luggage, and players passing the ball during professional-level team sports.

We extended our methods to cell tracking in Chapter 5. We introduced a novel approach to automatically detecting and tracking cell populations in time-lapse images. Unlike earlier approaches that rely on heuristics to handle mis-detections due to clumped cells and occlusions, our approach simultaneously detects and tracks cells from an over-complete set of competing detection hypotheses by solving a single integer program. As we have shown in Sec. 5.3, our approach produces more accurate trajectories and improves detection of mitosis events.

6.2 Research Collaboration

Human pose estimation and action recognition are widely-studied topics in Computer Vision and they are closely related to multi-object tracking [181, 182, 183, 184, 185]. We collaborated with researchers working on human pose estimation by sharing our people tracking results. We ran our extended POM algorithm as described in Sec. 4.4 and applied the KSP algorithm to obtain the people trajectories on the ground. Our collaborators took our trajectories as input, and conducted body joint detection *only* in the locations where there is a tracked person. As compared to the approach that detects the body joints in the whole image [186], our approach

Figure 6.1 – Pose estimation results on the Shelf Dataset. The body poses are recovered in 3D and then projected onto the four camera views. The identity of each person, as indicated by the number on the top, is obtained by our tracker.

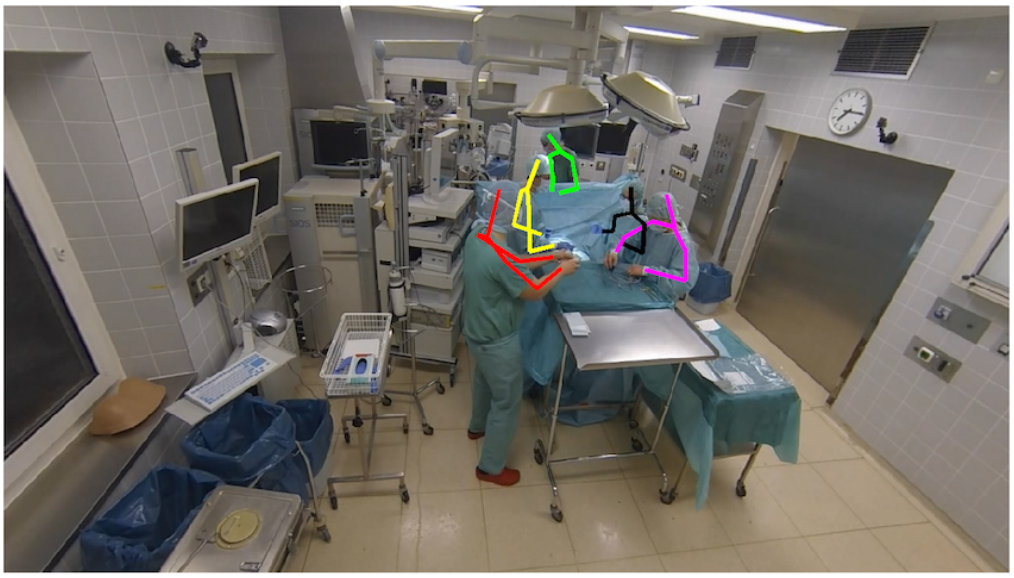


Figure 6.2 – One example frame from the operating room sequence. The 3D poses are shown in color. The similar appearance and prolonged occlusion between the doctors make people tracking and pose estimation very challenging.

significantly reduces the search space and yields better results. In Fig. 6.1, we show the pose estimation results at one time instance from four camera views. In Fig. 6.3, we show the number of recovered 3D body joint candidates versus the number of 2D detection samples. Our collaboration has led to a publication and we refer the interested readers to [187] for details.

Currently, we are applying this approach to sequences captured in an operating room. Our goal is to track the doctors and estimate their poses, which is a very challenging task due to the prolonged occlusions and similar appearance among the doctors. We show an example frame with labeled poses in Fig. 6.2. We aim to unify people tracking and pose estimation into one framework, where the locations and poses of people are estimated simultaneously.

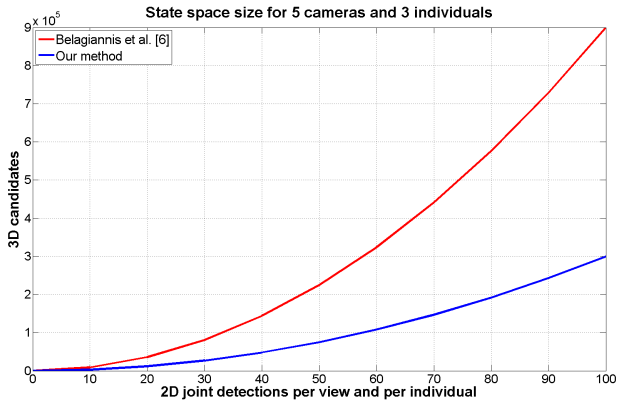


Figure 6.3 – Comparison of the size of the state space. We show the number of recovered 3D detection candidates versus the number of 2D detections. The 3D detection candidates are computed by triangulating 2D detections of all combinations of view pairs.

6.3 Limitations and Future Work

Despite the contributions made in this thesis, our presented work is not without limitations. In this section, we list these limitations and discuss future research directions.

6.3.1 Motion Model

In this thesis, we have used the first-order motion model for tracking interacting objects. In other words, we assume the location of the target object at time $t+1$ only depends on its location at time t. This simplified assumption turns out to be robust on the scenarios we have tested. However, in some cases objects do follow a higher-order motion pattern. In the case of basketball tracking, when the ball is flying in the air, its only acceleration is gravity and therefore its trajectory follows a parabola. In the case of player tracking in team sports, players of the same team work together on defense and offense so their motions are correlated.

Therefore, we suggest incorporating higher-order motion models into the tracking framework in future work [81, 188]. This will however significantly increase the model complexity and demand efficient optimization techniques.

6.3.2 Appearance Model and Re-identification

To handle object tracking in natural scenes, our methods take input from POM directly and do not account for the appearances of the targets. Our methods work consistently better than state-of-the-art methods. Nevertheless, the results can be further improved by introducing

the appearance model of individual target, especially in the basketball scenarios where people usually move close to each other.

Furthermore, introducing the appearance model can boost the tracker's re-identification capability [189]. Consider the case where a person gets into a car, sits in the car for a while and then gets out again. Our proposed method in Sec. 4 does not have the capability of target re-identification and thus will output two independent people trajectories. By introducing the appearance model, the improved tracker should potentially cluster these two trajectories into one class and assign them the same identity.

The method in [14] incorporates the appearance model into network flow programming and solves relaxed linear programming in a global optimal way. For future work, we suggest combining this approach with our framework of tracking interacting objects to improve the identity-preserving capability.

6.3.3 Multi-Class Interaction

In this thesis, we have modeled the interaction relationships of up to two classes of objects, such as people and cars. In the future we are looking forward to extending our methods to more than two classes of interacting objects. This could be potentially achieved by introducing attributes for each flow variable and expressing the interaction constraints in terms of both the flows and their attributes.

6.3.4 Domain-Specific Prior Knowledge

Domain-specific prior knowledge can potentially help to improve the tracking performance. For example, in the case of cell tracking, there is a minimum duration between two division events. Incorporating this knowledge to the tracking formulation could potentially remove false-positive division events. However, this will result in additional constraints in the network flow programming and increase the complexity of the optimization.

6.3.5 Background Subtraction

Background subtraction provides input for the POM algorithm and therefore directly influences the detection and tracking results. In our research we have relied on stable background images captured by static cameras beforehand. As a result, our current system is sensitive to the illuminance change or subtle movement. We aim to replace the static background

subtraction with dynamic methods that learn the background online [190]. This replacement will potentially increase the robustness of our system.

6.3.6 Optimization

In our research we have formulated the problem of tracking interacting objects as a Mixed Integer Programming (MIP). As we have shown in Sec. 4.5 and Sec. 5.3, solving the exact MIP yields better results than solving the relaxed Linear Programming (LP). However, MIP is known to be NP-Hard [191, 192], meaning that it does not scale well to large models. Therefore, we suggest investigating efficient optimization methods to large scale network programming. Combinatorial optimization strategies such as lazy constraint generation [69, 84] and graph partition [193, 194] can potentially speed up the optimization.

Finally, we are looking forward to extending our methods to more scenarios, such as tracking people in crowded scenes. A crowd can be treated as a container object and an individual as a containee object, and thus the method proposed in Chapter 4 can be potentially applied. Furthermore, since the people detections in this case are usually ambiguous due to occlusions and the image data could be explained in terms of multiple and conflicting hypotheses, our method proposed in Chapter 5 can be extended to handle this problem.

A Complete Derivation of the Tracklet Graph

We provide here the procedure to construct the tracklet graph as described in Sec. 4.3.2 using the notations in Tab. 4.1.

A.1 Grouping Spatial Vertices into Tracklets

The graph size reduction step as described in Sec. 4.3.1 produces a set of trajectories $\mathcal{T}$. Each trajectory $\tau_q \in \mathcal{T}$ is a set of spatial vertices

$$\tau_q = \{\hat{v}_i^{t_q}, ..., \hat{v}_j^{T_q}\}, \;\; \hat{v}_k^t \in \hat{V}\,, \tag{A.1}$$

where $\hat{v}_k^t$ denotes a spatial location k at time t, $\hat{V}$ denote the set of all such vertices in the pruned graph, and t_q and T_q denote the first and last time instance of τ_q. We use $\mathcal{N}_s(k)$ and $\mathcal{N}_s(\hat{v}_k^t)$ interchangeably to denote the spatial neighborhood of $\hat{v}_k^t$. 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,

$$\mathcal{N}_s(\tau_q) = \bigcup_{\hat{v}_k^t \in \tau_q} \mathcal{N}_s(k)\,. \tag{A.2}$$

We take the shared 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:

$$\mathcal{S} = \{\hat{v}_k^t \in \hat{V} : \sum_{\tau_q \in \mathcal{T}} \mathbb{1}(\hat{v}_k^t \in \mathcal{N}_s(\tau_q)) > 1 \;\; \vee \exists \tau_q, \;\; t \in \{T_q, t_q\} \wedge \hat{v}_k^t \in \tau_q\}\,. \tag{A.3}$$

Note that the shared spatial vertices comprise the only spatial locations where an interaction event or identity switch can occur. Based on these shared vertices, we partition each trajectory τ_q into multiple segments. We write

$$\tau_q = \bigcup_j \tau_q^j \cup \bigcup_k \hat{v}_k^t, \text{ s.t. } \forall_j\, \tau_q^j \cap \mathscr{S} = \emptyset \;\wedge\; \forall_k\, \hat{v}_k^t \in \mathscr{S} \;\wedge\; \forall_{j,i}\, \tau_q^j \cap \tau_q^i = \emptyset\,. \tag{A.4}$$

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

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

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

$$\begin{aligned} \hat{v}_j^{t_q} &\in \mathscr{N}_s(\hat{v}^t(\tau_q)) \quad \text{s.t.} \quad \hat{v}_j^{t_q} \in \tau_q\,, \\ \hat{v}^T(\tau_q) &\in \mathscr{N}_s(\hat{v}_k^{T_q}) \quad \text{s.t.} \quad \hat{v}_k^{T_q} \in \tau_q\,. \end{aligned} \tag{A.6}$$

A.2 Computing the Poses of the Tracklets

Each non-shared spatial tracklet $\tau_q \in \mathscr{K}$ 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

$$\gamma_q^i = \{v_j^{t_q}, ..., v_k^{T_q}\}, \text{ s.t. } v_{k'}^{t+1} \in \mathscr{N}(v_{j'}^t)\;\; \forall v_{j'}^t, v_{k'}^{t+1} \in \gamma_q^i\,, \tag{A.7}$$

where v_j^t 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

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

Each spatial tracklet τ_q is treated as a single spatial vertex by the MIP solver. Therefore, the pose of τ_q is uniquely defined by the starting pose at its frame time instance and the ending pose at its last time instance. 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 $v_j^{t_q}$ and ending pose $v_k^{T_q}$, 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 poses. For each pair j and k, we compute

$$\gamma_q^i = \underset{\substack{i':\, v_j^{t_q} \in \gamma_q^{i'}, \\ v_k^{Tq} \in \gamma_q^{i'},}}{\arg\min}\, c(\gamma_q^{i'}). \tag{A.9}$$

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:

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

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 shared spatial vertices and non-shared 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 the vertex and edge of G' respectively, where the subscripts k and j denote the global index of pose vertices in G'. We take $t'(k)$ to be the first time instance of the vertex v'_k and 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 shared locations and all the pose tracklets. We write

$$V' = \left\{ v^t_k : \hat{v}^t_{l'(k)} \in \mathcal{S} \right\} \cup \left\{ \gamma^i_q \in \gamma_q : \tau_q \in \mathcal{K} \right\} . \tag{A.11}$$

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

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

where E'_s is

$$E'_s = \left\{ e'_{jk} : k \in \mathcal{N}'(j) \ \wedge \ \hat{v}^{t'(j)}_{l'(j)} \in \mathcal{S} \ \wedge \ \hat{v}^{t'(k)}_{l'(k)} \in \mathcal{S} \right\} , \tag{A.13}$$

To construct the set E'_n, we consider a spatial tracklet τ_q together with its predecessor vertex $\hat{v}^t(\tau_q)$ and successor 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 v'_k at $\hat{v}^T(\tau_q)$, we compute the edge pair that preserve 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 $\left(e'_{ji}, e'_{ik}\right)$. 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:

$$E'_n = \{ e'_{ji} \cup e'_{ik} : i = \underset{i' : i' \in \mathcal{N}'(j), k \in \mathcal{N}'(i')}{\arg\min} c(v'_{i'}) \ \wedge \ \hat{v}^{t'(j)}_{l'(j)} \in \mathcal{S} \ \wedge \ \hat{v}^{t'(k)}_{l'(k)} \in \mathcal{S} \ \wedge \ \hat{v}^{t'(i)}_{l'(i)} \notin \mathcal{S} \} \tag{A.14}$$

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

B Full Graph vs. Pruned Graph

In Sec. 4.3.1, we present an approach to reducing the size of the graph that is used for tracking interacting objects. 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 Sec. 4.3.1, we reduce the number of variables and constraints to a few thousands.

To compare the result 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. B.1. As can be seen, the result on the pruned graph is very similar to that on the full one. In our experiments we found that the optimization on the full graph takes up to 50G of memory and up to 10 hours on a *single* batch of 10 frames, while on the pruned graph it takes only 0.02 second.

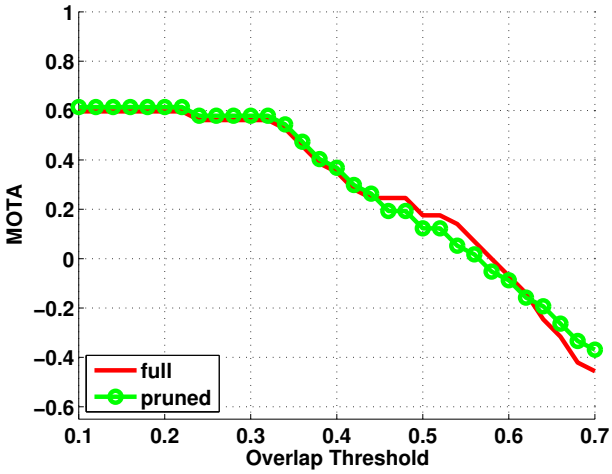


Figure B.1 – Comparison of MOTA scores obtained by running MIP on the full graph and the pruned graph.

C Features for Cell Tracking

In this section, we describe the features for learning the cell migration and division classifiers discussed in Chapter 5. We first define several quantities for individual ellipses and ellipse pairs, and then show how we use them to compute the final feature vectors for each classifier.

C.1 Ellipse Quantities

We compute various geometry and appearance features for individual ellipses and ellipse pairs as summarized in Tables C.1 and C.2 respectively.

We define the fitting error fitErr(e) for ellipse e in terms of the compatibility of its connected component in the segmentation and the set of ellipses in its hierarchy level hLvl(e):

$$\text{fitErr}(e) = \sum_{x \in I} |B(x) - \sum_{e' \in H(e,\text{hLvl}(e))} \mathbb{1}(x, e')\,|\,, \tag{C.1}$$

where x is a pixel in the input image I and $B(x)$ is the binary function that is 1 for pixels in the connected component of e and 0 otherwise. $H(e,\text{hLvl}(e))$ denotes the set of all ellipses at the level hLvl(e).

We define hOverlap(e) as the sum of the normalized overlap areas between e and the remaining ellipses at the same hierarchy level hLvl(e):

$$\text{hOverlap}(e) = \sum_{\substack{e' \in H(e,\text{hLvl}(e)) \\ e' \neq e}} \frac{\textbf{intersect}_{e,e'}}{\text{area}(e)}\,, \tag{C.2}$$

where $H(e,\text{hLvl}(e))$ is defined as before.

Notation	Quantity
ce(e)	Centroid of ellipse e
ec(e)	Eccentricity
ang(e)	Angle of major axis w.r.t. the horizontal axis x
area(e)	Ellipse area
a(e)	Major axis length
b(e)	Minor axis length
pixAvg(e)	Average of pixel values inside ellipse e
pixHist(e)	Histogram (64 bins) of pixel values
hLvl(e) $\in \{1,2,3,\ldots\}$	Hierarchy level of e
fitErr(e)	Fitting error of all the ellipses within the level hLvl(e) of the hierarchy tree of e
hOverlap(e)	Sum of the overlap areas between e and all the other ellipses at the same level hLvl(e)

Table C.1 – List of characteristic quantities for individual ellipses.

Notation	Quantity
$\mathbf{v}_{ij}$	Vector that connects the centroids of ellipses e_i and e_j in consecutive frames
$\mathbf{intersect}_{ij}$	Area of the intersection between ellipse e_i at e_j
$\mathbf{union}_{ij}$	Area of the union between ellipse e_i and e_j

Table C.2 – List of characteristic quantities for ellipse pairs.

C.2 Migration Classifier Features

Table C.3 summarizes the set of migration classifier features we compute using the quantities introduced above. We define them on pairs of ellipses e_i and e_j in two consecutive time frames as described in Sec. 5.2.

C.3 Division Classifier Features

We compute the division classifier features for triplets of ellipses: a parent e_j at time t, and two candidate daughters e_k, e_l at time $t+1$. They include area and eccentricity differences as described in Table C.5.

Feature	Description
$\lvert \mathrm{ec}(e_i) - \mathrm{ec}(e_j) \rvert$	Eccentricity difference
$\frac{\max(\mathrm{ec}(e_i), \mathrm{ec}(e_j))}{\min(\mathrm{ec}(e_i), \mathrm{ec}(e_j))}$	Eccentricity ratio
$\lvert \mathrm{nAng}\big(\mathrm{ang}(e_i) - \mathrm{ang}(e_j)\big) \rvert$	Absolute angular difference of major axes
$\lvert \mathrm{area}(e_i) - \mathrm{area}(e_j) \rvert$	Area difference
$\frac{\max(\mathrm{area}(e_i), \mathrm{area}(e_j))}{\min(\mathrm{area}(e_i), \mathrm{area}(e_j))}$	Area ratio
$\lvert \mathrm{b}(e_i) - \mathrm{b}(e_j) \rvert$	Minor axis difference
$\frac{\max(\mathrm{b}(e_i), \mathrm{b}(e_j))}{\min(\mathrm{b}(e_i), \mathrm{b}(e_j))}$	Minor axis ratio
$\lvert \mathrm{a}(e_i) - \mathrm{a}(e_j) \rvert$	Major axis difference
$\frac{\max(\mathrm{a}(e_i), \mathrm{a}(e_j))}{\min(\mathrm{a}(e_i), \mathrm{a}(e_j))}$	Major axis ratio
$\lvert\lvert \mathrm{ce}(e_i) - \mathrm{ce}(e_j) \rvert\rvert$	Centroid distance
$\mathrm{fitErr}(e_i) + \mathrm{fitErr}(e_j)$	Total hierarchy fit errors
$\frac{\mathbf{intersect}_{ij}}{\mathbf{union}_{ij}}$	Overlap ratio: intersection over union
$\frac{\mathbf{intersect}_{ij}}{\max(\mathrm{area}(e_i), \mathrm{area}(e_j))}$	Overlap ratio: intersection over the larger area
$\frac{\mathbf{intersect}_{ij}}{\min(\mathrm{area}(e_i), \mathrm{area}(e_j))}$	Overlap ratio: intersection over the smaller area
$\chi^2_{\mathrm{dist}}\big(\mathrm{pixHist}(e_i), \mathrm{pixHist}(e_j)\big)$	χ^2 distance between the pixel histograms
$\lvert \mathrm{nAng}\big(\mathrm{ang}(e_i) - \mathrm{ang}(\mathbf{v}_{ij})\big) \rvert$ + $\lvert \mathrm{nAng}\big(\mathrm{ang}(e_j) - \mathrm{ang}(\mathbf{v}_{ij})\big) \rvert$	Total angular differences btw the major axes and $\mathbf{v}_{ij}$
$\mathrm{hOverlap}(e_i) + \mathrm{hOverlap}(e_j)$	Sum of overlap ratios within the hierarchy levels
$\mathrm{area}(e_i) + \mathrm{area}(e_j)$	Sum of the areas

Table C.3 – Migration classifier features for two ellipses e_i, e_j in consecutive frames.

Feature	Description
$\text{area}(e_j)$	Area of ellipse at t
$\frac{1}{2}\big(\text{area}(e_k)+\text{area}(e_l)\big)$	Average area of ellipses at $t+1$
$\min\big(\text{area}(e_k),\text{area}(e_l)\big)$	Minimum area of ellipses at $t+1$
$\max\big(\text{area}(e_k),\text{area}(e_l)\big)$	Maximum area of ellipses at $t+1$
$\text{area}(e_j)-\frac{1}{2}\big(\text{area}(e_k)+\text{area}(e_l)\big)$	Area difference between t and $t+1$
$\text{area}(e_j)-\min\big(\text{area}(e_k),\text{area}(e_l)\big)$	Area difference between t and $t+1$
$\text{area}(e_j)-\max\big(\text{area}(e_k),\text{area}(e_l)\big)$	Area difference between t and $t+1$
$\frac{1}{2}\left(\lvert\lvert\mathbf{v}_{jk}\rvert\rvert+\lvert\lvert\mathbf{v}_{jl}\rvert\rvert\right)$	Average distance from t to $t+1$
$\min\left(\lvert\lvert\mathbf{v}_{jk}\rvert\rvert,\lvert\lvert\mathbf{v}_{jl}\rvert\rvert\right)$	Minimum distance from t to $t+1$
$\max\left(\lvert\lvert\mathbf{v}_{jk}\rvert\rvert,\lvert\lvert\mathbf{v}_{jl}\rvert\rvert\right)$	Maximum distance from t to $t+1$
$\lvert\lvert\mathbf{v}_{jk}-\mathbf{v}_{jl}\rvert\rvert$	Distance between centroids at $t+1$
$\lvert\lvert\mathbf{v}_{jk}-\mathbf{v}_{jl}\rvert\rvert$	Distance between centroids at $t+1$
$\frac{\mathbf{v}_{jl}^T\mathbf{v}_{jk}}{\lvert\lvert\mathbf{v}_{jk}\rvert\rvert\,\lvert\lvert\mathbf{v}_{jl}\rvert\rvert}$	Cosine of angle between the centroid vectors
$\text{ec}(e_j)$	Eccentricity of e_j
$\text{ec}(e_k)+\text{ec}(e_l)$	Eccentricity sum at $t+1$
$\text{ec}(e_j)-\min\big(\text{ec}(e_k),\text{ec}(e_l)\big)$	Minimum eccentricity difference.
$\text{ec}(e_j)-\max\big(\text{ec}(e_k),\text{ec}(e_l)\big)$	Maximum eccentricity difference.
$\lvert\text{ec}(e_k)-\text{ec}(e_l)\rvert$	Eccentricity difference at $t+1$
$\min\big(\text{ec}(e_k),\text{ec}(e_l)\big)\big)$	Minimum eccentricity at $t+1$
$\max\big(\text{ec}(e_k),\text{ec}(e_l)\big)\big)$	Maximum eccentricity at $t+1$

Table C.4 – Division classifier features for an ellipse e_j at time t and two ellipses e_k, e_l at time $t+1$.

Feature	**Description**
$\text{pixAvg}(e_j)$	Average pixel intensity of e_j
$\text{pixAvg}(e_j) + \text{pixAvg}(e_k) + \text{pixAvg}(e_l)$	Total average pixel intensity
$\text{pixAvg}(e_k) + \text{pixAvg}(e_l)$	Average pixel intensity at $t+1$
$\lvert\text{PixAvg}(e_k) - \text{pixAvg}(e_l)\rvert$	Pixel intensity difference at $t+1$
$\text{pixAvg}(e_j) - \frac{1}{2}\left(\text{pixAvg}(e_k) + \text{pixAvg}(e_l)\right)$	Pixel intensity difference between t and $t+1$
$\min\left(\text{PixAvg}(e_k), \text{pixAvg}(e_l)\right)$	Minimum average pixel intensity at $t+1$
$\max\left(\text{PixAvg}(e_k), \text{pixAvg}(e_l)\right)$	Maximum average pixel intensity at $t+1$
$\lvert\text{nAng}\big(\text{ang}(e_k), \text{ang}(e_l)\big)\rvert$	Angle difference absolute value
$\lvert\text{nAng}\big(\text{ang}(e_j) - \text{ang}(e_k)\big)\rvert$ + $\lvert\text{nAng}\big(\text{ang}(e_j) - \text{ang}(e_l)\big)\rvert$	Absolute angle difference sum
$\Big\lvert\lvert\text{nAng}\big(\text{ang}(e_j) - \text{ang}(e_k)\big)\rvert$ - $\lvert\text{nAng}\big(\text{ang}(e_j) - \text{ang}(e_l)\big)\rvert\Big\rvert$	Difference of absolute angle differences
$\text{fitErr}(e_j)$	Hierarchy fit error at t
$\text{fitErr}(e_j) + \text{fitErr}(e_k) + \text{fitErr}(e_l)$	Sum of hierarchy fit errors
$\text{fitErr}(e_k) + \text{fitErr}(e_l)$	Sum of hierarchy fit errors at $t+1$
$\min\left(\text{fitErr}(e_k), \text{fitErr}(e_l)\right)$	Minimum hierarchy fit error at $t+1$
$\max\left(\text{fitErr}(e_k), \text{fitErr}(e_l)\right)$	Maximum hierarchy fit error at $t+1$
$\lvert\text{hLvl}(e_j) - \text{hLvl}(e_k)\rvert + \lvert\text{hLvl}(e_j) - \text{hLvl}(e_l)\rvert$	Sum of hierarchy level differences
$\min\left(\lvert\text{hLvl}(e_j) - \text{hLvl}(e_k)\rvert, \lvert\text{hLvl}(e_j) - \text{hLvl}(e_l)\rvert\right)$	Minimum of hierarchy level differences
$\max\left(\lvert\text{hLvl}(e_j) - \text{hLvl}(e_k)\rvert, \lvert\text{hLvl}(e_j) - \text{hLvl}(e_l)\rvert\right)$	Maximum of hierarchy level differences

Table C.5 – (Cont'd) Division classifier features for an ellipse e_j at time t and two ellipses e_k, e_l at time $t+1$.

D Additional Comparative Results on Cell Tracking

In this section, we provide additional quantitative cell tracking results for the remaining image sequences from the GOWT and the SIM datasets, as we discussed in Sec. 5.3. We present comparative results of our tracker against the three state-of-the-art ones in Table D.1, and against those baselines obtained by removing several key components of our method in Table D.2.

In obtaining the accuracy values given in these tables as well as those presented in Sec.5.3, we discarded the cells that are close to the image boundaries. This is because the ground truth trajectories provided by the cell tracking challenge are not reliable in these regions as also stated by the organizers in the challenge website. That is why, in our evaluation, we used a margin of 50 pixels for the sequences of the SIM and GOWT datasets, and 100 pixels for those of the HeLa dataset.

		Division			**Detection**			**Migration**			**MOTA**	**Time**
		Rec	Pre.	F-M.	Rec.	Pre.	F-M.	Rec.	Pre.	F-M.		
GOWT-1	**GMM**	0.0	0.0	0.0	N/A	N/A	N/A	0.48	0.93	0.63	-0.29	39
	KTH	0.0	N/A	0.0	N/A	N/A	N/A	0.96	**1.0**	0.98	0.95	15
	CT	**0.50**	**1.0**	**0.67**	0.13	**1.0**	0.22	**0.98**	**1.0**	**0.99**	N/A	1.98
	OURS	**0.50**	**1.0**	**0.67**	**0.31**	**1.0**	**0.48**	**0.98**	**1.0**	**0.99**	**0.98**	**0.28**
SIM-1	**GMM**	0.0	0.0	0.0	N/A	N/A	N/A	0.65	0.91	0.76	0.45	14
	KTH	**1.0**	**1.0**	**1.0**	N/A	N/A	N/A	**1.0**	**1.0**	**1.0**	**1.0**	2
	CT	0.50	0.67	0.57	0.0	N/A	0.0	0.99	0.99	0.99	N/A	**0.20**
	OURS	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.99	**1.0**	0.99	1.16
SIM-2	**GMM**	0.0	0.0	0.0	N/A	N/A	N/A	0.76	0.90	0.83	0.56	15
	KTH	0.0	0.0	0.0	N/A	N/A	N/A	**1.0**	0.99	0.99	0.95	5
	CT	**1.0**	**1.0**	**1.0**	**1.0**	0.25	0.40	0.99	**1.0**	**1.0**	N/A	**0.41**
	OURS	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.75
SIM-3	**GMM**	0.0	0.0	0.0	N/A	N/A	N/A	0.77	0.94	0.85	0.66	20
	KTH	0.50	0.40	0.44	N/A	N/A	N/A	0.99	0.98	0.99	0.94	3
	CT	0.25	0.17	0.20	0.71	0.89	0.79	0.96	0.99	0.97	N/A	**0.39**
	OURS	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	1.00
SIM-5	**GMM**	0.0	0.0	0.0	N/A	N/A	N/A	0.77	0.87	0.82	-0.53	31
	KTH	0.0	0.0	0.0	N/A	N/A	N/A	**0.98**	0.99	0.98	0.96	10
	CT	0.50	0.33	0.40	0.93	0.82	0.88	0.96	0.99	0.98	N/A	**8.94**
	OURS	**0.75**	**0.75**	**0.75**	**1.0**	**1.0**	**1.0**	**0.98**	**1.0**	**0.99**	**0.98**	9.56
SIM-6	**GMM**	0.0	0.0	0.0	N/A	N/A	N/A	0.83	0.56	0.67	-0.49	27
	KTH	**1.0**	**1.0**	**1.0**	N/A	N/A	N/A	**0.99**	0.96	**0.97**	**0.95**	6
	CT	0.33	0.25	0.29	0.52	0.81	0.63	0.82	**0.99**	0.90	N/A	**1.28**
	OURS	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.95	**0.99**	**0.97**	**0.95**	5.82

Table D.1 – Comparison of our algorithm against state-of-the-art cell trackers. Our tracker yields a significant improvement on the division and detection accuracies and performs on par or slightly better on the migration and MOTA scores. As discussed in Sec. 5.3.4, the **CT** tracker only outputs the number of cells in each segment but not the final cell trajectories. As a result the optimization problem of **CT** is simpler and therefore in some cases **CT** runs faster than ours.

		Division			Detection			Migration			MOTA
		Rec	Pre.	F-M.	Rec.	Pre.	F-M.	Rec.	Pre.	F-M.	
GOWT-1	**OURS-CL**	0.0	0.0	0.0	N/A	N/A	N/A	0.98	0.89	0.94	N/A
	OURS-NC	0.50	0.50	0.50	1.0	0.29	0.44	0.98	1.0	0.99	0.96
	OURS-FD	0.50	1.0	0.67	0.31	1.0	0.48	0.98	1.0	0.99	0.98
	OURS-BH	0.50	1.0	0.67	0.25	1.0	0.40	0.96	1.0	0.98	0.96
	OURS-LP	0.50	1.0	0.67	0.31	1.0	0.48	0.98	1.0	0.99	0.98
	OURS	0.50	1.0	0.67	0.31	1.0	0.48	0.98	1.0	0.99	0.98
SIM-1	**OURS-CL**	1.0	0.67	0.80	N/A	N/A	N/A	0.99	0.74	0.84	N/A
	OURS-NC	1.0	0.67	0.80	1.0	0.13	0.24	1.0	0.99	1.0	0.92
	OURS-FD	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.99	1.0	0.99
	OURS-BH	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.99	1.0	0.99
	OURS-LP	1.0	1.0	1.0	1.0	1.0	1.0	0.99	0.99	0.99	0.98
	OURS	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.99	1.0	0.99
SIM-2	**OURS-CL**	1.0	1.0	1.0	N/A	N/A	N/A	1.0	0.71	0.83	N/A
	OURS-NC	1.0	0.75	0.86	1.0	0.25	0.40	1.0	1.0	1.0	0.99
	OURS-FD	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	OURS-BH	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	OURS-LP	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	OURS	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SIM-3	**OURS-CL**	1.0	0.67	0.80	N/A	N/A	N/A	0.99	0.45	0.62	N/A
	OURS-NC	0.75	0.33	0.46	0.08	0.02	0.03	0.99	0.97	0.98	0.51
	OURS-FD	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	OURS-BH	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.99	1.0	0.98
	OURS-LP	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	OURS	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SIM-5	**OURS-CL**	0.60	0.75	0.67	N/A	N/A	N/A	0.98	0.61	0.76	N/A
	OURS-NC	0.75	0.27	0.40	0.40	0.07	0.11	0.98	0.98	0.98	0.87
	OURS-FD	0.75	0.75	0.75	1.0	1.0	1.0	0.98	1.0	0.99	0.98
	OURS-BH	0.50	0.67	0.57	1.0	1.0	1.0	0.98	0.99	0.99	0.97
	OURS-LP	0.75	0.75	0.75	1.0	1.0	1.0	0.98	1.0	0.99	0.98
	OURS	0.75	0.75	0.75	1.0	1.0	1.0	0.98	1.0	0.99	0.98
SIM-6	**OURS-CL**	0.75	0.27	0.40	N/A	N/A	N/A	0.98	0.53	0.69	N/A
	OURS-NC	1.0	0.50	0.67	0.21	0.09	0.13	0.95	0.96	0.95	0.42
	OURS-FD	1.0	1.0	1.0	1.0	1.0	1.0	0.95	0.99	0.97	0.95
	OURS-BH	1.0	1.0	1.0	1.0	1.0	1.0	0.95	0.99	0.97	0.95
	OURS-LP	1.0	1.0	1.0	1.0	1.0	1.0	0.95	0.99	0.97	0.95
	OURS	1.0	1.0	1.0	1.0	1.0	1.0	0.95	0.99	0.97	0.95

Table D.2 – Tracking results of our approach and several baselines obtained by turning off some of its features.

Bibliography

[1] Merriam-Webster Dictionary, 2015, http://www.merriam-webster.com/dictionary/interact.

[2] M. Isard and A. Blake, "Condensation - Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, August 1998.

[3] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-Based Object Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–575, 2003.

[4] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 07, July 2012.

[5] D. Koller, J. Weber, and J. Malik, "Robust Multiple Car Tracking with Occlusion Reasoning," Tech. Rep., 1993.

[6] T. Zhao and R. Nevatia, "Tracking multiple humans in complex situations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1208 – 1221, 2004.

[7] P. Nillius, J. Sullivan, and S. Carlsson, "Multi-Target Tracking - Linking Identities Using Bayesian Network Inference," in *Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2187–2194.

[8] D. Koller, K. Daniilidis, and H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *International Journal of Computer Vision*, vol. 10, no. 3, pp. 257 – 281, 1993.

[9] M. Andriluka, S. Roth, and B. Schiele, "People-Tracking-By-Detection and People-Detection-By-Tracking," in *Conference on Computer Vision and Pattern Recognition*, June 2008.

[10] A. Fossati, M. Dimitrijevic, V. Lepetit, and P. Fua, "Bridging the Gap Between Detection and Tracking for 3D Monocular Video-Based Motion Capture," in *Conference on Computer Vision and Pattern Recognition*, June 2007.

[11] 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.

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

[13] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua, "Multiple Object Tracking Using K-Shortest Paths Optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 1806–1819, September 2011.

[14] H. BenShitrit, J. Berclaz, F. Fleuret, and P. Fua, "Multi-Commodity Network Flow for Tracking Multiple People," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1614–1627, 2014.

[15] D. Gavrila and S. Munder, "Multi-cue pedestrian detection and tracking from a moving vehicle," *International Journal of Computer Vision*, vol. 73, no. 1, pp. 41 – 59, 2007.

[16] A. Ess, K. Schindler, B. Leibe, and L. Van Gool, "Improved Multi-Person Tracking with Active Occlusion Handling," in *ICRA Workshop on People Detection and Tracking*, 2009.

[17] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition*, 2012.

[18] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174 – 188, 2002.

[19] R. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, pp. 35 – 45, 1960.

[20] A. Ellis, A. Shahrokni, and J. Ferryman, "Pets 2009 and Winter Pets 2009 Results, a Combined Evaluation," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, December 2009.

[21] H. Pirsiavash, D. Ramanan, and C. Fowlkes, "Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects," in *Conference on Computer Vision and Pattern Recognition*, June 2011.

[22] A. Milan, S. Roth, and K. Schindler, "Continuous Energy Minimization for Multitarget Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 58–72, 2014.

[23] W. Gerstner and W. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[24] P. Dillenbourg, "What do you mean by collaborative learning?" *Collaborative-learning: Cognitive and Computational Approaches*, p. 1 – 19, 1999.

[25] M. Raca, R. Tormey, and P. Dillenbourg, "Sleepers' lag-study on motion and attention," in *International Conference on Learning Analytics And Knowledge*, 2014.

[26] C. Liebe, "Accuracy performance of star trackers - a tutorial," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 587 – 599, 2002.

[27] ——, "Star trackers for attitude determination," *IEEE Aerospace and Electronic Systems Magazine*, pp. 10 – 16, 1995.

[28] F. Porikli et. al, "Video surveillance: past, present, and now the future," *IEEE Signal Processing Magazine*, vol. 30, no. 3, p. 190 – 198, 2013.

[29] C. Norris and M. Mccahill, "CCTV: Beyond Penal Modernism?" *British Journal of Criminology*, vol. 46, no. 1, p. 97 – 118, 2006.

[30] Big Brother Watch, "The Price of Privacy: How local authorities spent £515m on CCTV in four years ," Tech. Rep., 2012.

[31] C. WEBSTER, "CCTV Policy in the UK: Reconsidering the Evidence Base," *Surveillance & Society*, vol. 6, no. 1, pp. 162 – 179, 2009.

[32] A. R. Zamir, A. Dehghan, and M. Shah, "GMCP-Tracker: Global Multi-object Tracking Using Generalized Minimum Clique Graphs," in *European Conference on Computer Vision*, 2012, pp. 343–356.

[33] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah, "Part-based Multiple-Person Tracking with Partial Occlusion Handling," in *Conference on Computer Vision and Pattern Recognition*, 2012.

[34] Video Surveillance for ATMs, 2015, http://www.videosurveillance.com/atm.asp.

[35] K. Fragkiadaki, W. Zhang, G. Zhang, and J. Shi, "Two-Granularity Tracking: Mediating Trajectory and Detection Graphs for Tracking under Occlusions," in *European Conference on Computer Vision*, 2012.

[36] E. Prassler, J. Scholz, and A.Elfes, "Tracking people in a railway station during rush-hour," *Computer Vision Systems*, pp. 162 – 179, 1999.

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

[38] C. Kuo and R. Nevatia, "How does Person Identity Recognition Help Multi-Person Tracking?" in *Conference on Computer Vision and Pattern Recognition*, 2012.

[39] A. Chaaraoui and P. Climent-Pérez and F. Florez-Revuelta, "A review on vision techniques applied to human behaviour analysis for ambient-assisted living," *Expert Systems with Applications*, vol. 39, p. 10873 – 10888, 2012.

[40] J. Villacorta, L. del Val, M. Jimenez and A. Izquierdo , "Security system technologies applied to ambient assisted living," *Knowledge Management, Information Systems, E-Learning, and Sustainability Research*, vol. 111, pp. 389 – 394, 2010.

[41] M. Rantz and T. Banerjee and E. Cattoor and S. Scott and M. Skubic and M. Popescu, "Automated Fall Detection With Quality Improvement "Rewind" to Reduce Falls in Hospital Rooms," *Journal of Gerontological Nursing*, vol. 40, no. 1, p. 13 – 17, 2014.

[42] D. Gavrila and V. Philomin, "Real-Time Object Detection for "smart" Vehicles," in *International Conference on Computer Vision*, 1999, pp. 87–93.

[43] D. Gavrila, J. Giebel, and S. Munder, "Vision-Based Pedestrian Detection: the Protector System," in *Intelligent Vehicles Symposium*, 2004, pp. 13–18.

[44] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll Never Walk Alone: Modeling Social Behavior for Multi-Target Tracking," in *International Conference on Computer Vision*, 2009.

[45] Volvo Pedestrian Detection, 2015, http://www.volvocars.com/uk/top/my_volvo/videos/Pages/Volvo-PedestrianDetection.aspx.

[46] Mercedes-Benz PRE-SAFE Brake, 2015, http://techcenter.mercedes-benz.com/en/pre_safe_brake_2013/detail.html.

[47] Nissan Multi-sensing system with front camera, 2015, http://www.nissan-global.com/EN/TECHNOLOGY/OVERVIEW/front_camera.html.

[48] R. Sznitman, C. Becker, and P. Fua, "Fast Part-Based Classification for Instrument Detection in Minimally Invasive Surgery," in *Conference on Medical Image Computing and Computer Assisted Intervention*, September 2014.

[49] R. Sznitman, K. Ali, R. Richa, R. Taylor, G. Hager, and P. Fua, "Data-Driven Visual Tracking in Retinal Microsurgery," in *Conference on Medical Image Computing and Computer Assisted Intervention*, 2012, pp. 568–575.

[50] D. Novak, A. Nagle, U. Keller, and R. Riener, "Increasing motivation in robot-aided arm rehabilitation with competitive and cooperative gameplay," *Journal of NeuroEngineering and Rehabilitation*, vol. 11, no. 1, pp. 1 – 15, 2014.

[51] Y. Tao, H. Hu, and H. Zhou, "Integration of Vision and Inertial Sensors for 3D Arm Motion Tracking in Home-based Rehabilitation," *The International Journal of Robotics Research*, vol. 26, no. 6, pp. 607 – 624, 2007.

[52] M. McClusky, "The nike experiment: How the shoe giant unleashed the power of personal metrics," *Wired*, 2009.

[53] J. Liu, P. Carr, R. T. Collins, and Y. Liu, "Tracking Sports Players with Context-Conditioned Motion Models," in *Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1830–1837.

[54] W.-L. Lu, J.-A. Ting, K. P. Murphy, and J. J. Little, "Identifying Players in Broadcast Sports Videos Using Conditional Random Fields," in *Conference on Computer Vision and Pattern Recognition*, 2011.

[55] J. Sullivan and S. Carlsson, "Tracking and labelling of interacting multiple targets," in *European Conference on Computer Vision*, 2006.

[56] Performance Analysis in Football, 2015, http://http://www.prozonesports.com/.

[57] T. Kanda, D. Glas, M. Shiomi, H. Ishiguro, and N. Hagita, "Who will be the customer?: a social robot that anticipates people's behavior from their trajectories," in *UbiComp*, 2008, pp. 380 – 389.

[58] V. Uotila and P. Skogster, "Space management in a DIY store analysing consumer shopping paths with data tracking devices," *Facilities*, vol. 25, pp. 363 – 374, 2007.

[59] S. Hui, E. Bradlow, and P. Faderr, "Testing Behavioral Hypotheses Using an Integrated Model of Grocery Store Shopping Path and Purchase Behavior," *Journal of Consumer Research*, vol. 36, no. 3, pp. 478 – 493, 2009.

[60] N. Chenouard et. al, "Objective comparison of particle tracking methods," *Nature Methods*, vol. 11, no. 3, p. 281 – 289, 2014.

[61] B. X. Kausler, M. Schiegg, B. Andres, M. Lindner, U. Koethe, H. Leitte, J. Wittbrodt, L. Hufnagel, and F. A. Hamprecht, "A Discrete Chain Graph Model for 3D+ T Cell Tracking with High Misdetection Robustness," in *European Conference on Computer Vision*, 2012, pp. 144–157.

[62] K. Smith, A. Carleton, and V. Lepetit, "General Constraints for Batch Multiple-Target Tracking Applied to Large-Scale Videomicroscopy," in *Conference on Computer Vision and Pattern Recognition*, 2008.

[63] ——, "Fast Ray Features for Learning Irregular Shapes," in *International Conference on Computer Vision*, 2009, pp. 397–404.

[64] M. Maška, V. Ulman, D. Svoboda, P. Matula, P. Matula, C. Ederra, A. Urbiola, T. España, S. Venkatesan, D. M. Balak *et al.*, "A Benchmark for Comparison of Cell Tracking Algorithms," *Bioinformatics*, vol. 30, no. 11, pp. 1609–1617, 2014.

[65] C. O. Solorzano, M. Kozubek, E. Meijering, and A. M. noz Barrutia, "ISBI Cell Tracking Challenge," 2014. [Online]. Available: http://www.codesolorzano.com/celltrackingchallenge/Cell$_$Tracking$_$Challenge/Description.html

[66] S. Rujikietgumjorn and R. T. Collins, "Optimized Pedestrian Detection for Multiple and Occluded People," in *Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3690–3697.

[67] K. Zhang, L. Zhang, and M. H. Yang, "Real-Time Compressive Tracking," in *European Conference on Computer Vision*, 2012.

[68] B. Yang and R. Nevatia, "Multi-Target Tracking by Online Learning of Non-Linear Motion Patterns and Robust Appearance Models," in *Conference on Computer Vision and Pattern Recognition*, 2012.

[69] Gurobi, "Gurobi Optimizer," 2012, http://www.gurobi.com/.

[70] M. Schiegg, P. Hanslovsky, B. Kausler, L. Hufnagel, and F. Hamprecht, "Conservation Tracking," in *International Conference on Computer Vision*, December 2013, pp. 2928–2935.

[71] K. Magnusson and J. Jalden, "A Batch Algorithm Using Iterative Application of the Viterbi Algorithm to Track Cells and Construct Cell Lineages," in *International Symposium on Biomedical Imaging*, 2012.

[72] W. Ge and R. T. Collins, "Multi-Target Data Association by Tracklets with Unsupervised Parameter Estimation," in *British Machine Vision Conference*, September 2008.

[73] A. Andriyenko, K. Schindler, and S. Roth, "Discrete-Continuous Optimization for Multi-Target Tracking," in *Conference on Computer Vision and Pattern Recognition*, 2012.

[74] B. Yang and R. Nevatia, "An Online Learned CRF Model for Multi-Target Tracking," in *Conference on Computer Vision and Pattern Recognition*, 2012.

[75] C. Wojek, S. Walk, S. Roth, , K. Schindler, and B. Schiele, "Monocular Visual Scene Understanding: Understanding Multi-Object Traffic Scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[76] K. Li, E. D. Miller, M. Chen, T. Kanade, L. E. Weiss, and P. G. Campbell, "Cell Population Tracking and Lineage Construction with Spatiotemporal Context," *Medical Image Analysis*, vol. 12, no. 5, pp. 546–566, 2008.

[77] M. Schiegg, P. Hanslovsky, C. Haubold, U. Koethe, L. Hufnagel, and F. A. Hamprecht, "Graphical model for joint segmentation and tracking of multiple dividing cells," *Bioinformatics*, 2014.

[78] X. Wang, V. Ablavsky, H. BenShitrit, and P. Fua, "Take Your Eyes Off the Ball: Improving Ball-Tracking by Focusing on Team Play," *Computer Vision and Image Understanding*, vol. 119, pp. 102–115, 2014.

[79] X. Wang, E. Turetken, F. Fleuret, and P. Fua, "Tracking Interacting Objects Optimally Using Integer Programming," in *European Conference on Computer Vision*, September 2014.

[80] E. Turetken, X. Wang, C. Becker, and P. Fua, "Detecting and Tracking Cells Using Network Flow Programming," in *arXiv Preprint*, 2015.

[81] A. A. Butt and R. T. Collins, "Multi-target Tracking by Lagrangian Relaxation to Min-cost Network Flow," in *Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1846–1853.

[82] L. Zhang, Y. Li, and R. Nevatia, "Global Data Association for Multi-Object Tracking Using Network Flows," in *Conference on Computer Vision and Pattern Recognition*, 2008.

[83] A. Andriyenko, S. Roth, and K. Schindler, "An Analytical Formulation of Global Occlusion Reasoning for Multi-Target Tracking," in *International Conference on Computer Vision*, 2011.

[84] IBM ILOG CPLEX Optimizer, 2013, http://www.ibm.com/software/integration/optimization/cplex-optimizer/.

[85] C. Grant and S. Boyd, "CVX: Matlab Software for Disciplined Convex Programming," 2013, http://cvxr.com/.

[86] J. W. Suurballe, "Disjoint Paths in a Network," *Networks*, vol. 4, pp. 125–145, 1974.

[87] B. Babenko, M. Yang, and S. Belongie, "Robust Object Tracking with Online Multiple Instance Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1619–1632, 2011.

[88] K. Bernardin and R. Stiefelhagen, "Evaluating Multiple Object Tracking Performance: the Clear Mot Metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, 2008.

[89] 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 Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 319–336, February 2009.

[90] R. Stiefelhagen, K. Bernardin, R. Bowers, R. Rose, M. Michel, and J. Garofolo, "The Clear 2007 Evaluation," 2008.

[91] J. Lafferty, A. Mccallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *International Conference on Machine Learning*, Bellevue, WA, USA, 2001, pp. 282–289.

[92] H. BenShitrit, J. Berclaz, F. Fleuret, and P. Fua, "Tracking Multiple People Under Global Apperance Constraints," in *International Conference on Computer Vision*, 2011.

[93] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua, "KSP: Multiple Object Tracker Using K-Shortest Paths," 2011, http://cvlab.epfl.ch/software/ksp/index.php.

[94] Hawk-Eye tennis officiating system, 2012, http://www.hawkeyeinnovations.co.uk/page/sports-officiating/tennis.

[95] "SportVU Hoops Solutions," 2012, http://www.sportvu.com/basketball.asp.

[96] V. Lepetit, A. Shahrokni, and P. Fua, "Robust Data Association for Online Applications," in *Conference on Computer Vision and Pattern Recognition*, June 2003.

[97] F. Yan, W. Christmas, and J. Kittler, "Layered Data Association Using Graph- Theoretic Formulation with Applications to Tennis Ball Tracking in Monocular Sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1814–1830, 2008.

[98] Y. Ohno, J. Miura, and Y. Shirai, "Tracking Players and Estimation of the 3D Position of a Ball in Soccer Games," in *International Conference on Pattern Recognition*, 2000.

[99] M. Leo, N. Mosca, P. Spagnolo, P. Mazzeo, T. D'Orazio, and A. Distante, "Real-Time Multiview Analysis of Soccer Matches for Understanding Interactions Between Ball and Players," in *Conference on Image and Video Retrieval*, 2008.

[100] J. Ren, J. Orwell, G. A. Jones, and M. Xu, "Real-Time Modeling of 3D Soccer Ball Trajectories from Multiple Fixed Cameras," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 3, pp. 350–362, 2008.

[101] Y. Zhang, H. Lu, and C. Xu, "Collaborate Ball and Player Trajectory Extraction in Broadcast Soccer Video," in *International Conference on Pattern Recognition*, 2008.

[102] F. Poiesi, F. Daniyal, and A. Cavallaro, "Detector-Less Ball Localization Using Context and Motion Flow Analysis," in *International Conference on Image Processing*, 2010.

[103] M. Perše, M. Kristan, S. Kovačič, and J. Perš, "A Trajectory-Based Analysis of Coordinated Team Activity in a Basketball Game," *Computer Vision and Image Understanding*, vol. 113, no. 5, pp. 612–621, 2009.

[104] K. Kim, D. Lee, and I. Essa, "Detecting Regions of Interest in Dynamic Scenes with Camera Motions," in *Conference on Computer Vision and Pattern Recognition*, 2012.

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

[106] W.-L. Lu, K. Okuma, and J. J. Little, "Tracking and Recognizing Actions of Multiple Hockey Players Using the Boosted Particle Filter," *Image and Vision Computing*, vol. 27, pp. 189–205, 2009.

[107] F. Li and R. J. Woodham, "Video Analysis of Hockey Play in Selected Game Situations," *Image and Vision Computing*, vol. 27, pp. 45–58, 2009.

[108] C. Direkoğlu and N. O'Connor, "Team Activity Recognition in Sports," in *European Conference on Computer Vision*, October 2012, pp. 69–83.

[109] A. Gupta, P. Srinivasan, J. Shi, and L. S. Davis, "Understanding Videos, Constructing Plots Learning a Visually Grounded Storyline Model from Annotated Videos," in *Conference on Computer Vision and Pattern Recognition*, 2009, pp. 2012–2019.

[110] T. Lan, L. Sigal, and G. Mori, "Social Roles in Hierarchical Models for Human Activity Recognition," in *Conference on Computer Vision and Pattern Recognition*, July 2012.

[111] R. Li, R. Chellappa, and S. Zhou, "Learning Multi-Modal Densities on Discriminative Temporal Interaction Manifold for Group Activity Recognition," in *Conference on Computer Vision and Pattern Recognition*, 2009.

[112] S. Ali, V. Reilly, and M. Shah, "Motion and Appearance Contexts for Tracking and Re-Acquiring Targets in Aerial Videos," in *Conference on Computer Vision and Pattern Recognition*, 2007.

[113] H. Grabner, J. Matas, L. Van Gool, and P. Cattin, "Tracking the Invisible: Learning Where the Object Might Be," in *Conference on Computer Vision and Pattern Recognition*, 2010.

[114] L. Karlinsky, M. Dinerstein, D. Harari, and S. Ullman, "The Chains Model for Detecting Parts by Their Context," in *Conference on Computer Vision and Pattern Recognition*, 2010.

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

[116] N. Oliver, B. Rosario, and A. Pentland, "A Bayesian Computer Vision System for Modeling Human Interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 831–843, 2000.

[117] P. Parisot and C. D. Vleeschouwer, "Graph-Based Filtering of Ballistic Trajectory," in *International Conference on Multimedia and Expo*, 2011.

[118] "A Matlab Implementation Random Forest Classifier," 2012, http://www.mathworks.com/matlabcentral/fileexchange/31036-random-forest.

[119] APIDIS European Project FP7-ICT-216023, 2008–2010, www.apidis.org.

[120] 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.

[121] T. Baumgartner, D. Mitzel, and B. Leibe, "Tracking People and Their Objects," in *Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3658–3665.

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

[123] A. Mittal and L. Davis, "M2Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene," *International Journal of Computer Vision*, vol. 51(3), pp. 189–203, 2003.

[124] J. Giebel, D. Gavrila, and C. Schnorr, "A Bayesian Framework for Multi-Cue 3D Object Tracking," in *European Conference on Computer Vision*, 2004.

[125] K. Smith, D. Gatica-Perez, and J.-M. Odobez, "Using Particles to Track Varying Numbers of Interacting People," in *Conference on Computer Vision and Pattern Recognition*, 2005.

[126] 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 Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[127] J. Kwon, H. S. Lee, F. C. Park, and K. M. Lee, "A Geometric Particle Filter for Template-Based Visual Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 625–643, 2014.

[128] S. Avidan, "Support vector tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1064–1072, 2004.

[129] ——, "Ensemble Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 261–271, 2007.

[130] Q. Bai, Z. Wu, S. Sclaroff, M. Betke, and C. Monnier, "Randomized Ensemble Tracking ," in *International Conference on Computer Vision*, 2013, pp. 2040–2047.

[131] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised On-Line Boosting for Robust Tracking," in *European Conference on Computer Vision*, 2008, pp. 234–247.

[132] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-N Learning: Bootstrapping Binary Classifiers from Unlabeled Data by Structural Constraints," in *Conference on Computer Vision and Pattern Recognition*, 2010.

[133] J. Fan, X. Shen, and Y. Wu, "Scribble Tracker: A Matting-Based Approach for Robust Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 1633 – 1644, 2012.

[134] J. Kwon, J. Roh, K. M. Lee, and L. V. Gool, "Robust Visual Tracking with Double Bounding Box Model," in *European Conference on Computer Vision*, 2014, pp. 377–392.

[135] S. Hong, S. Kwak, and B. Han, "Orderless Tracking through Model-Averaged Posterior Estimation," in *International Conference on Computer Vision*, 2013, pp. 2296–2303.

[136] H. Nam, S. Hong, and B. Han, "Online Graph-Based Tracking," in *European Conference on Computer Vision*, 2014, pp. 112–126.

[137] K. Zhang, L. Zhang, and M. H. Yang, "Fast Compressive Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 2002–2015, 2014.

[138] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M.-H. Yang, "Fast Tracking via Dense Spatio-Temporal Context Learning," in *European Conference on Computer Vision*, 2014.

[139] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized Belief Propagation," in *Advances in Neural Information Processing Systems*, 2000, pp. 689–695.

[140] W. Choi and S. Savarese, "A Unified Framework for Multi-Target Tracking and Collective Activity Recognition," in *European Conference on Computer Vision*, 2012.

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

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

[143] P. Storms and F. Spieksma, "An Lp-Based Algorithm for the Data Association Problem in Multitarget Tracking," *Computers and Operations Research*, vol. 30, no. 7, pp. 1067–1085, June 2003.

[144] H. Jiang, S. Fels, and J. Little, "A Linear Programming Approach for Multiple Object Tracking," in *Conference on Computer Vision and Pattern Recognition*, 2007.

[145] A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and H. Wensheng, "Multi-Object Tracking through Simultaneous Long Occlusions and Split-Merge Conditions," in *Conference on Computer Vision and Pattern Recognition*, 2006.

[146] C. Huang, Y. Li, and R. Nevatia, "Multiple Target Tracking by Learning-Based Hierarchical Association of Detection Responses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 4, pp. 898–910, 2013.

[147] A. Andriyenko and K. Schindler, "Globally Optimal Multi-Target Tracking on a Hexagonal Lattice," in *European Conference on Computer Vision*, 2010, pp. 466–479.

[148] A. Milan, K. Schindler, and S. Roth, "Detection- and Trajectory-Level Exclusion in Multiple Object Tracking," in *Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3682–3689.

[149] C. Arora and A. Globerson, "Higher Order Matching for Consistent Multiple Target Tracking," in *International Conference on Computer Vision*, 2013, pp. 177–184.

[150] A. V. Segal and I. Reid, "Latent Data Association: Bayesian Model Selection for Multi-target Tracking," in *International Conference on Computer Vision*, 2013, pp. 2904–2911.

[151] R. Collins and P. Carr, "Hybrid Stochastic / Deterministic Optimization for Tracking Sports Players and Pedestrians," in *European Conference on Computer Vision*, 2014.

[152] H. Possegger, T. Mauthner, P. M. Roth, and H. Bischof, "Occlusion Geodesics for Online Multi-object Tracking," in *Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1306–1313.

[153] 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.

[154] P. Lucey, A. Bialkowski, P. Carr, S. Morgan, I. Matthews, and Y. Sheikh, "Representing and Discovering Adversarial Team Behaviors Using Player Roles," in *Conference on Computer Vision and Pattern Recognition*, 2013.

[155] J. Liu and Y. Liu, "Multi-target tracking of time-varying spatial patterns," in *Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1839–1846.

[156] M. Rodriguez, I. Laptev, J. Sivic, and J. Audibert, "Density-aware person detection and tracking in crowds," in *International Conference on Computer Vision*, 2011, pp. 2423–2430.

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

[158] P. Felzenszwalb, D. Mcallester, and D. Ramanan, "A Discriminatively Trained, Multiscale, Deformable Part Model," in *Conference on Computer Vision and Pattern Recognition*, June 2008, pp. 1–8.

[159] 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, 2010.

[160] E. Meijering, O. Dzyubachyk, and I. Smal, "Methods for Cell and Particle Tracking," *Methods in Enzymology*, vol. 504, no. 9, pp. 183–200, 2012.

[161] O. Dzyubachyk, W. V. Cappellen, J. Essers, W. Niessen, and E.Meijering, "Advanced Level-Set-Based Cell Tracking in Time-Lapse Fluorescence Microscopy," *IEEE Transactions on Medical Imaging*, 2010.

[162] A. Dufour, R. Thibeaux, E. Labruyere, N. Guillen, and J.-C. Olivo-Marin, "3D Active Meshes: Fast Discrete Deformable Models for Cell Tracking in 3-D Time-Lapse Microscopy," *IEEE Transactions on Image Processing*, vol. 20, no. 7, pp. 1925–1937, 2011.

[163] R. Delgado-Gonzalo, N. Chenouard, and M. Unser, "Fast Parametric Snakes for 3D Microscopy," in *International Symposium on Biomedical Imaging*, 2012, pp. 852–855.

[164] M. Maška, O. Daněk, S. Garasa, A. Rouzaut, A. Munoz-Barrutia, and C. Ortiz-de Solorzano, "Segmentation and Shape Tracking of Whole Fluorescent Cells Based on the Chan-Vese Model," *TMI*, vol. 32, no. 6, pp. 995–1006, 2013.

[165] S. Oron, A. Bar-hillel, and S. Avidan, "Extended Lucas-Kanade Tracking," in *European Conference on Computer Vision*, 2014.

[166] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: Robust Tracking via Multiple Experts Using Entropy Minimization," in *European Conference on Computer Vision*, 2014.

[167] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky, "Hough Forests for Object Detection, Tracking, and Action Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[168] F. Amat, W. Lemon, D. Mossing, K. McDole, Y. Wan, K. Branson, E. Myers, and P. Keller, "Fast, Accurate Reconstruction of Cell Lineages from Large-Scale Fluorescence Microscopy Data," *Nature methods*, 2014.

[169] H. Li, R. Sumner, and M. Pauly, "Global Correspondence Optimization for Non-Rigid Registration of Depth Scans," in *Symposium on Geometry Processing*, 2008, pp. 1421–1430.

[170] D. Padfield, J. Rittscher, N. Thomas, and B. Roysam, "Spatio-Temporal Cell Cycle Phase Analysis Using Level Sets and Fast Marching Methods," *Medical Image Analysis*, vol. 13, no. 1, pp. 143–155, 2009.

[171] F. Jug, T. Pietzsch, D. Kainmüller, and G. Myers, "Tracking by Assignment Facilitates Data Curation," in *MICCAI IMIC Workshop*, 2014.

[172] M. Hofmann, D. Wolf, and G. Rigoll, "Hypergraphs for Joint Multi-View Reconstruction and Multi-Object Tracking," in *Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3650–3657.

[173] L. Leal-taixe, G. Pons-moll, and B. Rosenhahn, "Branch-And-Price Global Optimization for Multi-View Multi-Target Tracking," in *Conference on Computer Vision and Pattern Recognition*, 2012.

[174] L. Wen, W. Li, J. Yan, Z. Lei, D. Yi, and S. Z. Li, "Multiple Target Tracking Based on Undirected Hierarchical Relation Hypergraph," in *Conference on Computer Vision and Pattern Recognition*, 2014.

[175] M. Liem and D. Gavrila, "Joint Multi-Person Detection and Tracking from Overlapping Cameras," *Computer Vision and Image Understanding*, vol. 128, pp. 36–50, 2014.

[176] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, and B. Schmid, "Fiji: an open-source platform for biological-image analysis," *Nature Methods*, vol. 9, no. 7, pp. 676–682, 2012, code available at http://pacific.mpi-cbg.de.

[177] D. K. Prasad and M. K. H. Leung and C. Quek, "ElliFit: An Unconstrained, Non-Iterative, Least Squares-based Geometric Ellipse Fitting Method," *Pattern Recognition*, vol. 46, no. 5, pp. 1449–1465, 2013.

[178] J. Friedman, "Stochastic Gradient Boosting," *Computational Statistics & Data Analysis*, 2002.

[179] J. Platt, *Advances in Large Margin Classifiers*. MIT Press, 2000, ch. Probabilistic Outputs for SVMs and Comparisons to Regularized Likelihood Methods.

[180] C. Sommer, C. Straehle, U. Koethe, and F. Hamprecht, "ilastik: Interactive Learning and Segmentation Toolkit," in *International Symposium on Biomedical Imaging*, 2011.

[181] J. Sullivan and S. Carlsson, "Recognizing and Tracking Human Action," in *European Conference on Computer Vision*, 2002.

[182] G. Loy, M. Eriksson, J. Sullivan, and S. Carlsson, "Monocular 3D Reconstruction of Human Motion in Long Action Sequences," in *European Conference on Computer Vision*, 2004.

[183] M. Andriluka, S. Roth, and B. Schiele, "Pictorial Structures Revisited: People Detection and Articulated Pose Estimation," in *Conference on Computer Vision and Pattern Recognition*, June 2009.

[184] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-Time Human Pose Recognition in Parts from Single Depth Images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[185] M. Burenius, J. Sullivan, and S. Carlsson, "3D Pictorial Structures for Multiple View Articulated Pose Estimation," in *Conference on Computer Vision and Pattern Recognition*, 2013.

[186] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic, "3D Pictorial Structures for Multiple Human Pose Estimation," in *Conference on Computer Vision and Pattern Recognition*, 2014.

[187] V. Belagiannis, X. Wang, B. Schiele, P. Fua, S. Ilic, and N. Navab, "Multiple Human Pose Estimation with Temporally Consistent 3D Pictorial Structures," in *European Conference on Computer Vision*, September 2014.

[188] R. Collins, "Multitarget data association with higher-order motion models," in *Conference on Computer Vision and Pattern Recognition*, 2012.

[189] S. Gong, M. Cristani, S. Yan, and C. Loy, *Person Re-Identification*. Springer-Verlag London, 2014.

[190] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Conference on Computer Vision and Pattern Recognition*, 1998.

[191] K. Steiglitz and C. Papadimitriou, *Combinatorial optimization : algorithms and complexity* . Prentice-Hall, 1982.

[192] A. Schrijver, *Theory of Linear and Integer Programming*. Wiley, 1998.

[193] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Technical Journal*, vol. 49, no. 2, p. 291 – 307, 1970.

[194] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

Xinchao Wang

Contact Information

EPFL IC ISIM CVLAB
BC 306 Station 14
CH-1015 Lausanne
SWITZERLAND

(+41) 21 69 31283
xinchao.wang@epfl.ch
http://people.epfl.ch/xinchao.wang

Research Interests

Computer Vision, Multimedia, Machine Learning, Image Processing

Education

École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

Ph.D. Candidate, Computer Vision (expected July 2015)

- Dissertation Topic: Tracking Interacting Objects in Image Sequences
- Advisor: Pascal Fua

The Hong Kong Polytechnic University (HKPU), Kowloon, Hong Kong

Hons. in Computing, Aug. 2010

- Highest honors in computing, full GPA (4.0/4.0), Rank #1 in the department
- Most Outstanding Final Year Project Award

Submitted Papers/Papers in preparation

X. Wang*, E. Turetken*, C. Becker and P. Fua, *Anonymous Submission 1,* submitted to International Conference on Computer Vision (**ICCV**) 2015 (* joint first authors).

B. Tekin, X. Sun, **X. Wang**, V. Lepetit and P. Fua, *Anonymous Submission 2,* submitted to International Conference on Computer Vision (**ICCV**) 2015.

X. Wang, E. Turetken, F. Fleuret and P. Fua, *Tracking Interacting Objects Optimally Using Intertwined Flows,* in submitting to IEEE Transactions on Pattern Analysis and Machine Intelligence (**TPAMI**).

V. Belagiannis, **X. Wang**, H. Ben Shitrit, K. Hashimoto, R. Stauder, Y. Aoki, M. Kranzfelder, A. Schneider, S. Ilic, H. Feussner, P. Fua and N. Navab, *Parsing Human Skeletons in the Operating Room,* in submitting to IEEE Transactions on Medical Imaging (**TMI**).

Selected Publications

X. Wang*, E. Turetken*, C. Becker and P. Fua, *Detecting and Tracking Cells using Network Flow Programming,* arXiv, 2015 (* joint first authors).

X. Wang, E. Turetken, F. Fleuret and P. Fua, *Tracking Interacting Objects Optimally Using Integer Programming,* European Conference on Computer Vision (**ECCV**) 2014. **(oral, accepet rate 2.6%)**

V. Belagiannis, **X. Wang,** B. Schiele, P. Fua, S. Ilic and N. Navab, *Multiple Human Pose Estimation with Temporally Consistent 3D Pictorial Structures,* European Conference on Computer Vision (**ECCV**), ChaLearn Looking at People Workshop, 2014.

X. Wang, V. H. Ablavsky, H. Ben Shitrit and P. Fua, *Take your Eyes off the Ball: Improving Ball-Tracking by Focusing on Team Play,* Computer Vision and Image Understanding (**CVIU**). Vol. 119, pp. 102-115, 2014.

X. Wang, W. Bian and D. Tao, *Grassmannian Regularized Structured Multi-View Embedding for Image Classification,* IEEE Transactions on Image Processing (**TIP**).

Vol. 22, pp. 2646-2660, 2013.

X. Wang, Z. Li and D. Tao, *Subspaces Indexing Model on Grassmann Manifold for Image Search,* IEEE Transactions on Image Processing (**TIP**). Vol. 20, pp. 2627-2635, 2011.

Teaching Assistant

Spring 2013/12	Computer Vision (EPFL)
Fall 2013	Pattern Recognition and Machine Learning (EPFL)

Selected Awards and Honors

2010	EPFL EDIC Fellowship
2010	The Hong Kong Government Scholarship
2010	HKPU Most Outstanding Final Year Project Award
2010/09/08	HKPU Post Entry Scholarship
2010/09	HKPU CMA & Donors Scholarship
2010	HKPU Grace Hopper Scholarship

Research Experience

Apr. 2013 - Present	Tracking Interacting Objects from Videos. Advisor: Pascal Fua. EPFL.
July 2014 - Sept. 2014	Tracking Players in a Volleyball Game. Advisor: Tomas Pajdla. Czech Technical University (CTU), Czech Republic.
Mar. 2011 - Mar. 2013	Basketball Tracking from Multiple Cameras. Advisor: Pascal Fua. EPFL.
Jun. 2010 - Aug. 2010	A Unifying Subspace Selection Framework. Advisor: Dacheng Tao. Nanyang Technological University (NTU), Singapore.
May. 2009 - Jun. 2010	Subspace Indexing for Large Scale Visual Identification. Advisor: Zhu Li. HKPU.

Industry Experience

July 2014 - Sept. 2014	Neovision s.r.o, Prague, Czech Republic Computer Vision Researcher
May 2008 - Aug. 2008	Liaoxing Technological Development Corp., China Programmer and database operator

Relevant Skills

Languages:	English, Chinese
Programming:	MATLAB, C/C++, JAVA
Tools:	svn, LaTeX, MS-Office/OpenOffice, Eclipse

Review Activities

IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI), IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision and Image Understanding (CVIU), IEEE Trans. on Image Processing (TIP), Pattern Recognition (PR), IEEE Trans. on Circuits and Systems for Video Technology (TCSVT).

Personal Information

Year of Birth:	1988
Nationality:	Chinese