

Training Algorithms for Multiple Object Tracking

Thèse N° 9203

Présentée le 30 janvier 2019
à la Faculté informatique et communications
Laboratoire de vision par ordinateur
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

ANDRII MAKSAI

Acceptée sur proposition du jury
Prof. P. Dillenbourg, président du jury
Prof. P. Fua, directeur de thèse
Prof. A. Geiger, rapporteur
Dr M. Andriluka, rapporteur
Prof. A. Alahi, rapporteur

2019

To my family

Acknowledgements

Firstly, I would like to thank my supervisor, Prof. Pascal Fua.

I am immensely grateful to Pascal for hiring me into the lab and allowing to work with him and other smart people. His professionalism, scientific curiosity, honesty and approachability not only immensely helped my work, but are also highly motivating for me. Our discussions helped me realize the importance of clarity, simplicity, and interpretability in research.

I would like to express my gratitude to my thesis jury members, Prof. Alexandre Alahi, Prof. Andreas Geiger, Dr. Mykhaylo Andriluka, and Prof. Pierre Dillenbourg for graciously taking time to evaluate this thesis.

I want to thank Prof. Xinchao Wang, who was my first collaborator as I came into the lab. Our frequent talks, combined with his honest feedback, allowed for more smooth transition into the lab, and provided much-needed understanding of the general pipeline of people tracking.

I am very grateful to Prof. François Fleuret, who frequently visited the lab and found the time to discuss with me details of my work. His very inquisitive questions often helped to clarify many topics, and his expertise and comments were highly valuable.

My thanks goes to Pierre Baqué, Tatjana Chavdarova, Cijo Jose, Louis Lettry, and Timur Bagautdinov, the members of the "Tracking in the Wild" project, with whom we have frequently discussed our work on the topic, and had a very nice time in Zürich recording the people tracking dataset.

I also wanted to thank Prof. Tomas Pajdla, for our discussions and the freedom in selecting the topic during my stay in Prague.

Being a member of Computer Vision Lab, I would like to thank its every member. Both Josiane Gisclon and Ariane Staudenmann, secretaries of the lab, for their help and ability to solve any problem that comes along the way. To Dr. Mathieu Salzmann for his extremely attentive feedback and comments on some of my work. To my officemates Artem Rozantsev, Timur Bagautdinov, and Jan Bednarik, for the ability to relax between the work and our quick chats. To Róger Bermúdez, Isinsu Katircioglu, Ksenia Konyushkova, Agata Mosinska, Dr. Eduard Trulls, Dr. Helge Rhodin, Prof. Kwang Moo Yi, and many others, for pleasant interactions, and, of course,

Acknowledgements

for proof-reading my papers and providing comments in the crazy times of conference deadlines.

I would like to thank numerous EPFL students, for their company during hiking outings, and to my flatmates Arseniy Zaostrovnykh and Dmitrii Ustiugov.

Finally, greatest thanks to my parents, my brother, and my life partner, Anastasia, for their love and support. Anastasia, without you, none of this would have been possible.

My research was supported in part by the Swiss National Science Foundation grant CRSII2-147693 "Tracking in the Wild".

Lausanne, November 2018

Andrii Maksai

Abstract

Multiple object tracking is a crucial Computer Vision Task. It aims at locating objects of interest in the image sequences, maintaining their identities, and identifying their trajectories over time. A large portion of current research focuses on tracking pedestrians, and other types of objects, that often exhibit predictable behaviours, that allow us, as humans, to track those objects. Nevertheless, most existing approaches rely solely on simple affinity or appearance cues to maintain the identities of the tracked objects, ignoring their behaviour. This presents a challenge when objects of interest are invisible or indistinguishable for a long period of time.

In this thesis, we focus on enhancing the quality of multiple object trackers by learning and exploiting the long ranging models of object behaviour. Such behaviours come in different forms, be it a physical model of the ball motion, model of interaction between the ball and the players in sports or motion patterns of pedestrians or cars, that is specific to a particular scene.

In the first part of the thesis, we begin with the task of tracking the ball and the players in team sports. We propose a model that tracks both types of objects simultaneously, while respecting the physical laws of ball motion when in free fall, and interaction constraints that appear when players are in the possession of the ball. We show that both the presence of the behaviour models and the simultaneous solution of both tasks aids the performance of tracking, in basketball, volleyball, and soccer.

In the second part of the thesis, we focus on motion models of pedestrian and car behaviour that emerge in the outdoor scenes. Such motion models are inherently global, as they determine where people starting from one location tend to end up much later in time. Imposing such global constraints while keeping the tracking problem tractable presents a challenge, which is why many approaches rely on local affinity measures. We formulate a problem of simultaneously tracking the objects and learning their behaviour patterns. We show that our approach, when applied in conjunction with a number of state-of-the-art trackers, improves their performance, by forcing their output to follow the learned motion patterns of the scene.

In the last part of the thesis, we study a new emerging class of models for multiple object tracking, that appeared recently due to availability of large scale datasets - sequence models for multiple object tracking. While such models could potentially learn arbitrarily long ranging behaviours, training them presents several challenges. We propose a training scheme and a loss function

Acknowledgements

that allows to significantly improve the quality of training of such models. We demonstrate that simply using our training scheme and loss allows to learn scoring function for trajectories, which enables us to outperform state-of-the-art methods on several tracking benchmarks.

Keywords: multi-object tracking, behaviour modelling, tracking, detection, interaction, mixed integer programming

Résumé

Le suivi de plusieurs objets est une tâche cruciale de vision par ordinateur. Il vise à localiser les objets d'intérêt dans les séquences d'images, en maintenant leur identité, et identifier leurs trajectoires au fil du temps. Une grande partie de la recherche actuelle porte sur le suivi des piétons et d'autres types d'objets, qui présentent souvent des comportements prévisibles, nous permettant, en tant qu'êtres humains, de suivre ces objets. Néanmoins, la plupart des approches existantes reposent uniquement sur de simples indices d'affinité ou d'apparence pour conserver les identités des objets suivis, en ignorant leur comportement. Cela représente un défi lorsque les objets d'intérêt sont invisibles ou impossibles à distinguer pendant une longue période.

Dans cette thèse, nous nous concentrons sur l'amélioration de la qualité de plusieurs méthodes de suivi d'objets en apprenant et en exploitant les modèles de longue portée du comportement des objets. Ces comportements se présentent sous différentes formes, qu'il s'agisse d'un modèle physique du mouvement du ballon, d'un modèle d'interaction entre le ballon et les joueurs dans le sport ou des mouvements de piétons ou de voitures spécifiques à une scène donnée.

Dans la première partie de la thèse, nous commençons par la tâche de suivre le ballon et les joueurs dans les sports d'équipe. Nous proposons un modèle qui suit simultanément les deux types d'objets, tout en respectant les lois physiques du mouvement de la balle en chute libre, ainsi que les contraintes d'interaction qui apparaissent lorsque les joueurs sont en possession du ballon. Nous montrons que la présence des modèles de comportement et la solution simultanée des deux tâches contribuent à la performance du pistage, au basketball, au volleyball et au football.

Dans la deuxième partie de la thèse, nous nous concentrons sur les modèles de mouvement du comportement des piétons et des voitures qui émergent dans les scènes en extérieur. De tels modèles de mouvement sont intrinsèquement globaux, car ils régissent les endroits où les personnes partant d'un endroit ont tendance à se retrouver beaucoup plus tard. Imposer de telles contraintes globales tout en gardant le problème de suivi solvable constitue un défi, raison pour laquelle de nombreuses approches s'appuient sur des mesures d'affinité locales. Nous formulons un problème de suivi simultané des objets et d'apprentissage de leurs comportements. Nous montrons que notre approche, lorsqu'elle est appliquée avec un certain nombre de trackers à la pointe de la technologie, améliore leurs performances en forçant leur sortie à suivre les schémas de mouvement appris de la scène.

Résumé

Dans la dernière partie de la thèse, nous étudions une nouvelle classe de modèles émergents pour le suivi d'objets multiples, apparus récemment en raison de la disponibilité de données à grande échelle - des modèles de séquence pour le suivi d'objets multiples. Bien que de tels modèles puissent potentiellement apprendre des comportements arbitrairement longs, l'entraînement de tels modèles présente plusieurs défis. Nous proposons un programme d'entraînement et une fonction de coût permettant d'améliorer considérablement la qualité d'entraînement de tels modèles. Nous démontrons que le simple fait d'utiliser notre programme d'entraînement et de coût permet d'apprendre la fonction de score pour les trajectoires, ce qui nous permet de surpasser l'état de l'art actuel sur plusieurs benchmark de suivi d'objets.

Mots-clés : suivi multi-objet, modélisation du comportement, suivi, détection, interaction, programmation mixte à nombres entiers

Contents

Acknowledgements	v
Abstract	vii
List of figures	xiii
List of tables	xvi
1 Introduction	1
2 Background	7
2.1 Notations	7
2.2 Definitions	7
2.3 Problem formulation and optimization	9
2.4 Datasets	11
2.5 Metrics	12
3 Physically constrained interaction modelling	15
3.1 Introduction	15
3.2 Related work	17
3.2.1 Fitting Trajectory Segments	17
3.2.2 Global Energy Minimization	17
3.3 Problem Formulation	18
3.3.1 Graphical Model for Ball Tracking	19
3.3.2 Integer Program Formulation	20
3.3.3 Mixed Integer Program Formulation	21
3.4 Learning the Potentials	23
3.5 Implementation details	25
3.5.1 Player Graph	25
3.5.2 Ball Graph	25
3.5.3 Ball states	26
3.5.4 State classifier	29
3.6 Experiments	29
3.6.1 Datasets	30

Contents

3.6.2	Baselines	31
3.6.3	Metrics	32
3.6.4	Comparative Results	33
3.6.5	Computational efficiency	36
3.7	Conclusion	36
4	Non-Markovian globally consistent multi-object tracking	39
4.1	Introduction	39
4.2	Related Work	41
4.2.1	MOT as Data Association	41
4.2.2	Using Behavioral Models	41
4.2.3	Quantifying Identity Switches	42
4.3	Formulation	43
4.3.1	Detection Graph	44
4.3.2	Building the Graph	45
4.3.3	Objective Function	45
4.4	Computing Trajectories and Patterns	47
4.4.1	Trajectories	47
4.4.2	Patterns	48
4.5	Non-Markovian Multiple Object Tracking	49
4.6	Evaluation	50
4.6.1	Datasets	50
4.6.2	Baselines	51
4.6.3	Experimental Protocol	52
4.6.4	Results	54
4.7	Conclusion	60
5	Eliminating exposure bias and loss-evaluation mismatch in MOT	61
5.1	Introduction	61
5.2	Related work	63
5.2.1	Modeling Longer Sequences	63
5.2.2	Reducing Bias and Loss-Evaluation Mismatch	64
5.3	Method	65
5.3.1	Tracking Formalization	65
5.3.2	Tracking	65
5.3.3	Generating Candidate Trajectories	66
5.3.4	Defining the Scoring Function	66
5.3.5	Training Procedure	69
5.4	Results	70
5.4.1	Datasets	70
5.4.2	Implementation Details	71
5.4.3	Baselines	72
5.4.4	Comparative Performance	74

5.4.5 Ablation study	75
5.4.6 Computational effort	79
5.5 Conclusion	80
6 Conclusions	81
6.1 Summary	81
6.2 Future work	82
6.2.1 Future location prediction	82
6.2.2 Role understanding	82
6.2.3 Tracking with segmentation/pose estimation	82
6.2.4 Reducing exposure bias	82
6.2.5 Better data association	83
A An appendix	85
A.1 Cost function definition from Chapter 4	85
A.2 Additional results for Chapter 4	90
A.3 Additional results for Chapter 5	95
Bibliography	111
Curriculum Vitae	113

List of Figures

1.1	Scenarios, where behaviour can significantly aid tracking	2
2.1	Network-flow based formulation for MOT	11
3.1	Importance of simultaneously modeling interactions and imposing physical constraints for tracking in sports.	16
3.2	Graphical models for ball tracking.	18
3.3	Example of spatio-temporal graph constructed on the ball detections.	26
3.4	Tracking accuracy as a function of training data size.	30
3.5	Effect of smoothing on tracking accuracy. Example of the learned classifier. . . .	31
3.6	Comparative results for ball tracking	34
3.7	Comparative results for game state estimation.	35
3.8	Result sequences for ball tracking and state estimation in several sports.	37
4.1	Training and inference procedures for learning global motion patterns and tracking using the patterns.	40
4.2	Objects of interest in our problem formulation: trajectories, patterns, association between the two.	42
4.3	Effect of identity switches on the tracking metrics.	43
4.4	Definitions of functions required to compute the match between a motion pattern and a trajectory.	46
4.5	Improvement in IDF₁ and MOTA metrics brought by applying our approach on Towncentre dataset.	53
4.6	IDF₁ and MOTA scores on the Rene dataset	54
4.7	Examples of learned motion patterns and how their use improves the tracking results of several methods.	54
4.8	Example of unsupervised optimization, that infers both trajectories and motion patterns of the scene.	55
4.9	Examples of the motion patterns learned on DukeMTMC dataset.	55
4.10	Comparison between learned arbitrary and straight motion patterns	58
4.11	Computational burden analysis	59
5.1	Keeping track in difficult situation thanks to well-trained sequence model.	62
5.2	Network architecture for tracking with sequence model.	67

List of Figures

5.3	Candidate pruning phase for multiple hypothesis tracking.	67
5.4	Tracking results on DukeMTMC dataset	75
5.5	Tracking results on MOT17 dataset	76
5.6	Tracking accuracy vs. inference speed.	79
A.1	Example of computing cost function for a particular trajectory and pattern.	86

List of Tables

2.1	Notations	7
2.2	Dataset statistics	12
3.1	Notations specific to the problem of ball and player tracking in sports. The rest of the variables used in this chapter are defined in 2.1.	19
3.2	Physical models associated with different states of the ball.	27
3.3	Importance of simultaneous tracking of the ball and the players.	36
3.4	Running time of player and ball tracking.	36
4.1	IDF₁ and MOTA improvement brought by using motion patterns, averaged across datasets.	56
4.2	Effect of learned motion patterns vs straight line patterns vs motion smoothness term	57
4.3	Evaluation using ground truth detections.	58
4.4	Optimization problem size and run time.	58
5.1	Benchmark results on DukeMTMC dataset.	74
5.2	Benchmark results on MOT15 dataset.	74
5.3	Benchmark results on MOT17 dataset.	74
5.4	Ablation study: variations of training procedure, loss, and training data	77
A.1	Computation of cost function for trajectory and patterns, normal case.	87
A.2	Computation of cost function for trajectory and pattern, corner cases	88
A.3	Computation of cost function for trajectory and absent pattern.	89
A.4	Full results on Towncentre dataset.	91
A.5	Full results on Rene, Station, Hotel, ETH datasets.	92
A.6	Full results on DukeMTMC dataset.	93
A.7	Full results when using ground truth detections.	94
A.8	Benchmark tracking results on the WILDTRACK dataset.	94
A.9	Metrics description.	95
A.10	Full benchmark results on Easy set of sequences of DukeMTMC dataset.	96
A.11	Full benchmark results on Hard set of sequences of DukeMTMC dataset.	96
A.12	Full benchmark results on MOT15 dataset.	96

List of Tables

A.13 Comparison to SORT method on the validation data for DukeMTMC dataset, IDF/MOTA	96
A.14 Full benchmark results on MOT17 dataset.	96

1 Introduction

Tracking multiple objects is one of the key Computer Vision tasks. It has a long tradition for applications such as radar tracking, video surveillance and automatic sport statistics. More recently it has also been a topic of interest for virtual and augmented reality, autonomous navigation, and human computer interaction.

Multiple Object Tracking (MOT) aims at locating objects of interest, maintaining their identities, and identifying their trajectories over time. Objects of interest span pedestrians [183] and vehicles [54], sport players [112], animals [53], and cells [107]. More than 70% of the current research targets pedestrians [115], due to a number of high-level computer vision tasks related to people tracking, and commercial potential of the technology. It frequently serves as the basis required to perform many other tasks such as semantic segmentation, anomaly detection, role understanding, scene understanding, and many others.

While MOT has progressed much in the recent years to the point where tracking methods now rival that of humans, in some cases, challenging scenarios remain. They include tracking in extremely dense crowds, in situations where it is difficult to distinguish between multiple targets due to their small size or non-distinct appearance, and in the cases when tracked objects are occluded for really long periods of time, where humans rely on their knowledge of the world, rather than solely on their vision, to track the objects.

One of the main reasons of frequent tracking failures for such scenarios is the overly local nature of many tracking methods. They produce trajectories of the objects of interest based on grouping object detections in a small temporal window, frequently relying on the appearance similarity and adjacency in space. While this approach is very generic and often works, it overlooks one of the key properties that governs people: human behaviours, that can be complex and difficult to predict.

Indeed, it is easy for humans to understand that if some time a basketball ball was in the hands of one player, and at some other point in the hands of another one, then in between it must have been passed from one to the other when they were close to each other, even if the ball was never visible

between these two points in time. This requires understanding the player interactions. We can also infer how the ball was flying across the net of volleyball court, even if it was too fast for us to see after it was served - because we know about the laws of physics governing the ball's motion. Similarly, if we have observed that pedestrians tend to always turn right at some corner of the crosswalk, we would expect the next one to do the same, even if we never saw that particular person because of the passing car or because we shifted our gaze away for a bit. However, we should also be prepared that once in a while, someone might turn left instead. That is our way of understanding a particular scene. Examples of such scenarios are depicted in Fig. 1.1.



Figure 1.1 – Examples of scenarios where tracking requires understanding of human behaviour. **(a)** Ball is in possession of the volleyball players and is often not seen for prolonged periods of time; **(b)** Small size and large number of people in New York Central station makes it hard to rely only on visual information for tracking, but layout of the scene helps; **(c)** Understanding how humans behave can help track people in this particularly crowded scene, where most people can not be fully seen

Making inferences such as those described above requires us humans to model long term behaviours, which are not limited only to what we have directly seen. In this thesis, we tackle the problem of learning models of long-term behaviours. We first do that for sports players and the ball in sports games, injecting both physical models and interaction behaviour understanding when tracking both players and a ball. Next, we expand this by proposing a generic framework that enables us to use non-Markovian, global constraints for multiple object tracking, without sacrificing the global optimality of the final solution. Finally, we embed these ideas into a promising class of models for multiple object tracking that has appeared in the last couple of years thanks to the appearance of large scale datasets, namely recurrent neural network-based methods. Such methods could potentially account for arbitrarily long term and complex behaviours, but are difficult to train well. Two main reasons for that are the mismatch between the loss function for training and inference, and the mismatch between the data observed during training and inference. We therefore propose a training method for such models, and a loss function, that significantly boost their performance, by tackling these two problems.

This dissertation is based on and uses parts of the following papers:

1. A. MAKSAI, X. WANG, P. FUA : What Players Do with the Ball: A physically constrained interaction modeling. (*Proc. of the 2016 IEEE Computer Vision and Pattern Recognition*)

Conference).

2. A. MAKSAI, X. WANG, F. FLEURET, P. FUA : Non-Markovian Globally Consistent Multi-Object Tracking. (*Proc. of the 2017 IEEE International Conference on Computer Vision*).
3. A. MAKSAI, P. FUA : Eliminating Exposure Bias and Loss-Evaluation Mismatch in Multiple Object Tracking. (*In review for the 2019 IEEE Computer Vision and Pattern Recognition Conference*).

Additional results are presented from the following paper:

- T. CHAVDAROVA, P. BAQUÉ, S. BOUQUET, A. MAKSAI, C. JOSE, T. BAGAUT-DINOV, L. LETTRY, P. FUA, L. VAN GOOL, AND F. FLEURET : WILDTRACK: A Multi-camera HD Dataset for Dense Unscripted Pedestrian Detection (*Proc. of the 2018 IEEE Computer Vision and Pattern Recognition Conference*).

Some tracking results and explanatory videos from these papers are available ^{1,2}.

In summary, the contributions of this work are:

- **Model for tracking interacting objects under physical constraints.** We propose a model for tracking interacting objects, namely ball and players in sports games. In ball games, ball is often in possession of a player, and not seen by any cameras. Therefore, an interaction model is required for robust tracking. When the ball is passed between the players, it often moves very fast and is also hard to see, but a physical model of its motion allows for precise tracking in this scenario. Our main contribution is a model that enables us to combine these two seemingly very different tasks in one single problem, that is solved jointly. Since switching between two modes of tracking also requires understanding of the phase of the game that is currently happening, an additional contribution of this work is estimating the state of the game while tracking. To our knowledge, this is first approach ever that combines tracking the ball under physical constraints, tracking the players and their possession of the ball, and state of the game, jointly. We show that it significantly outperforms methods that solve these tasks independently, on several different sports.
- **Method for imposing global non-Markovian constraints on MOT.** We address the issue of effectively imposing global constraints on the trajectory of an individual. While most current approaches use pairwise affinity measures that ensure correct tracking during a short time span, imposing global constraints remains difficult optimization-wise. We show how a traditional network-flow based approach for multiple object tracking can be extended to enforce such constraints, while keeping optimization computationally feasible.

¹<https://cvlab.epfl.ch/research/research-surv/research-balltracking/>

²https://drive.google.com/open?id=1Udt4Q3UCXNU1ry27j1Vy3w_FC3O-ieBM

- **A joint model for MOT and motion pattern estimation.** Using our proposed approach for imposing global constraints for multiple object tracking, we show how motion patterns can be used to better track objects in a scene. Our approach simultaneously tracks objects and determines a behaviour, or motion pattern, that specific object follows. We show that this synergy between the two tasks improves a number of state-of-the-art object trackers by simply modifying their output to agree with the scene motion patterns, that is, to conform to global constraints of the scene. Additionally, our approach can serve as a way to extract prevalent motion patterns, given the object tracking results, and detect anomalous motion.
- **A training procedure for MOT that removes biases.** In this work, we tackle the problem of learning sequence-based affinity measures for multiple object tracking. With the appearance of large scale datasets, a number of methods for training sequence-based models has emerged. However, these methods tend to use ground truth information to predict correct data association between the detections, which results in exposure bias when during the actual tracking errors lead to situations never seen in training. Our training procedure confronts the learning algorithm with its own errors to make sure that all possible scenarios are observed during training. We also propose a loss function that forces our model to optimize an approximation of the tracking quality metric, removing loss-evaluation mismatch between training and inference. We show that our proposed method for constructing the training dataset helps to largely improve the quality of the learned models and allows us to achieve state-of-the-art tracking results on several challenging tracking benchmarks.

In Chapter 2, we provide background information and necessary preliminaries. Sec. 2.1 describes frequently used notations, followed by Sec. 2.2 where we provide definitions in context of multiple object tracking. Sec. 2.3 discusses problem formulation and optimization methods for multiple object tracking. Commonly used benchmarks for multiple object tracking are discussed in Sec. 2.4, which describes datasets, and Sec. 2.5, which describes metrics.

In Chapter 3, we describe our model for simultaneously tracking the ball and the players under physical constraints. We describe our formalization of the problem of simultaneous tracking of the ball and the players in Sec. 3.3. After that Sec. 3.4 describes how to learn the learnable parameters of our method, and details of the implementation are given in Sec. 3.5. Results of our experiments on ball and player tracking in multiple sports are given in Sec. 3.6.

In Chapter 4, we present a non-Markovian model for multiple object tracking, and how we use it to track pedestrians and cars under constraints of patterns of the scene. Sec. 4.3 provides a formal description of the joint task of finding the behaviours of the pedestrians and their tracks, and gives a description of the motion pattern that we use for our work. Sec. 4.4 describes how tracks could be found assuming that behavioural patterns are known, and vice versa. Sec. 4.5 combines these two results in a scheme which allows to perform both tasks either given ground truth tracking in a particular scene, or when no ground truth is available, in a completely unsupervised fashion. Results are presented in Sec. 4.6.

In Chapter 5, we discuss a training procedure and loss function that significantly boosts the performance of sequence-based models for multiple object tracking. We introduce our tracking approach in Sec. 5.3, which we center around learning good scoring function for trajectories. After that, we describe our exact form of the loss function, and training procedure, that allows to improve sequence models. Implementation details and full breakdown of results follow in Sec. 5.4.

Conclusions and future work follow in chapter 6.

2 Background

We present in this chapter common notations and definitions used through the thesis. After that, we provide background on several ways of representing a problem of multiple object tracking, and corresponding optimization methods. Finally, we describe benchmark datasets and metrics, commonly used in multiple object tracking. More comprehensive review of tasks, applications, approaches, datasets and metrics related to multiple object tracking can be found in [115, 102, 101, 149, 170].

2.1 Notations

MOT	Multiple object tracking
MOTA	Multiple object tracking accuracy (metric)
IDF₁	Identification F1 score (metric)
IoU	Intersection over union
N	Number of temporal frames in the tracking problem
K	Number of possible states for a tracked object
I^t	Image evidence at time t
X^t	Location of the tracked object at time t
\mathcal{G}	Graph in the network flow-based formulation for tracking
\mathcal{V}	Set of nodes in the tracking graph
\mathcal{E}	Set of edges in the tracking graph
T	Set of transitions in the tracking graph that form trajectories
v_{in}, v_{out}	Source and sink nodes in tracking graph

Table 2.1 – Notations

2.2 Definitions

Here we put the task of multiple object tracking in context of other related tasks and provide definitions frequently used throughout the thesis.

Chapter 2. Background

Multiple object tracking (MOT) aims at locating multiple objects of interest, inferring their trajectories and maintaining their identities in a video sequence. This is in contrast to **single object tracking**, where trajectory of only one object needs to be inferred, and therefore identity switches between multiple objects are not possible.

Trajectory of a tracked object is defined as location of object in multiple frames. In each frame, object is usually defined by its bounding box. Auxiliary feedback can be provided in a form of pixel mask of the tracked object (instance-based semantic segmentation) or the locations of keypoints on the object (pose estimation).

Tracking by detection is performed by first locating the candidate detections of the objects of interest in each frame separately, and then associating identities to the detections to form the trajectories of the objects. This is in contrast to **tracking by model evolution**, or **detection-free tracking**, which searches for the candidate detection for each target in each subsequent frame. While first tracking attempts often relied on tracking by model evolution, recursive nature of such approaches often results in identity switches and trajectory fragmentations, which are difficult to recover from.

Batch processing is used to find the tracking given a sequence of frames, with long sequence possibly broken down into several batches, processed sequentially. This is in contrast to **online processing**, which forms trajectories with each new observed frame, allowing for real time performance, but sacrificing the benefits of observing several frames before making the decision about tracking. In between the two are methods that rely on **near-online processing**. Those methods feature small batches and real-time performance for results that are constantly delayed by a small amount of time compared to the input sequence of frames.

Single-camera tracking uses a single input source to obtain tracking results. Tracking results are usually defined in the camera plane. **Multiple-camera tracking** uses input from multiple, usually overlapping cameras, which allows to estimate the trajectory of each tracked object in the world coordinates, either in 2D on the ground plane, or in 3D. Multiple non-overlapping cameras can be used to track and re-identify objects of interest across multiple cameras.

Static camera tracking assumes the use of stationary camera. This often allows to use properties of a given fixed scene to aid the tracking. **Moving camera tracking** is more challenging due to both motion of the camera and absence of a fixed scene.

Interacting objects tracking assumes tracking several types of objects that may interact in certain ways, affecting one another, ie. cars and people getting in and out of them, or players, and

ball that they can possess, in a sports game. This is in contrast to a scenario where motions of different types of objects are assumed to be independent of each other and are tracked separately.

The results presented in this thesis relate to **multiple object tracking**, with the aim of estimating **trajectories** of the tracked objects. All presented methods use the **tracking by detection** framework and rely on **batch processing**. Results are presented both for single, multiple overlapping, and multiple non-overlapping cameras, both static and moving cameras, and both for interacting and non-interacting objects.

In the next section we present common problem formulations for multiple object tracking, and optimization techniques used. We focus on methods that use tracking by detection and rely on batch processing.

2.3 Problem formulation and optimization

Generally, multiple object tracking aims to find sequence of states $X_{1:N}$ of each of the tracked objects in N consecutive moments in time, that maximize a-posteriori probability given the observations $I_{1:N}$. In tracking by detection framework, observations are typically detections \mathcal{D} in each individual frame, and states of the tracked objects are represented by the coordinates of such detections.

Approaches to multiple object tracking can be roughly divided into two big groups, namely probabilistic inference and deterministic optimization.

Probabilistic inference approaches rely on various filtering techniques such as (extended) Kalman filtering [130, 153] or particle filtering [25, 186], and are often coupled together with tracking by model evolution. They are more frequently used for online tracking, since they require only past and current observations. Optimization typically consists of two parts, namely prediction part, which finds the most likely state given a sequence of observations before the current moment $P(X_t|I_{1:t-1})$, and update phase, which updates the predictions $P(X_t|I_{1:t}) \propto P(I_t|X_t)P(X_t|I_{1:t-1})$ based on the observations of the current moment.

Deterministic optimization methods try to find a sequence of states that maximize the likelihood by optimizing the value of energy function $E(X_{1:N}|I_{1:N})$, which typically factorizes over some small groups of states and observations. Depending on these factors of the energy function, several ways to optimize energy function are available.

Arbitrary high-order potentials don't place any assumptions on the factors of the energy function. For example, they could span whole width of the trajectory to ensure consistent appearance throughout the whole trajectory, or have potentials that combine multiple detections in the same frame to model pairwise exclusion, occlusion or social behaviour. In general, finding optimal sequence of states is not possible in polynomial time. Therefore, existing solutions either

Chapter 2. Background

rely on solving the problem optimally in a very small time window by using inference with conditional random fields [36], gradient descent with jumps in the search space to avoid bad local minima [128, 140], or genetic algorithms [140].

Pairwise potentials allow only factors that connect two observations. These factors typically model either homogeneity within a single trajectory (such as consistency of appearance of the object), or pairwise exclusion constraints over a pair of detections within single frame. Having pairwise potentials allows to represent all observations as a **spatio-temporal graph**, where each observation is a node, and each pairwise potential is represented by an edge between two detections, with weight of the edge corresponding to the value of the factor. Such graph formulation allows to find the tracking using several approaches from graph theory or operations research, such as maximum weight independent set / maximum weight clique / maximum weight multi-cut. These approaches represent each trajectory as a set of nodes, densely connected by pairwise edges, and look for a solution that maximizes the weights within each trajectory and/or minimizes the weights in between different trajectories. Such problems can either be solved optimally in non-polynomial time, or there exist efficient heuristics that allow to find near-optimal solution quickly, with one of the latest results being a fast approximate multi-cut algorithm of [161]. Other similar formulations could also be solved by generic integer programming.

Notable differences between the above formulations include the following:

- Maximum weight independent set allows to define potentials for a group of observations belonging to different objects, effectively differentiating them.
- Maximum weight clique and maximum weight multi-cut approaches allow to define a set of observations belonging to the same person across multiple frames, thus becoming more difficult optimization problem as the length of the tracking interval grows, but providing more robust tracking.
- Maximum weight clique is usually defined for a graph of detections, while multi-cut can be defined on different types of edges, allowing to track not only people but also their body parts, connecting different body parts in the same frame and same body parts in the different frames, for example, as in [161].

Pairwise potentials adjacent in time assumes factors are a functions of two detections, adjacent in time, with potential representing similarity or dissimilarity of two detections. Under this formulation, it is possible to represent trajectories as units of flow flowing through the edges of such network, and find the tracking as a solution of a linear program, or maximum-cost maximum-flow problem, which can be done in polynomial time [196, 17]. Such network \mathcal{G} can be represented by a set of nodes and a set of edges between the nodes. A set of nodes $\mathcal{V} = \mathcal{D} \cup \{v_{in}, v_{out}\}$ with v_{in} and v_{out} being so-called source and sink nodes, which are the

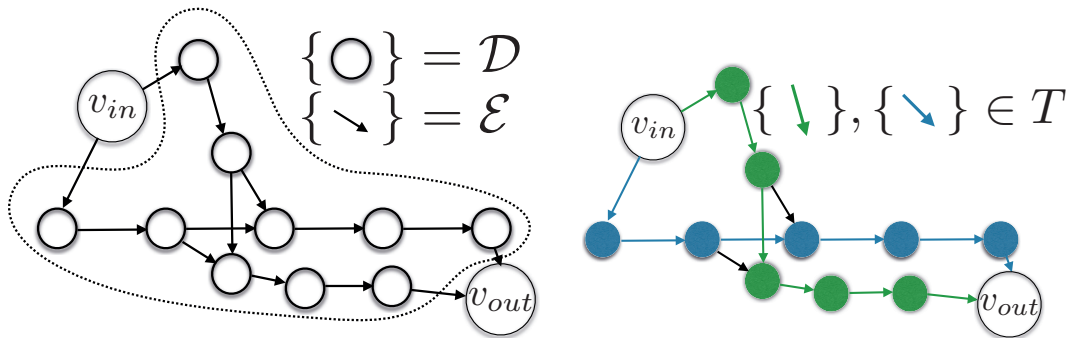


Figure 2.1 – Test

(a) Given a set of detections \mathcal{D} , and a set of allowed transitions \mathcal{E} , we seek to find: (b) trajectories of the objects, represented by transitions from \mathcal{T} .

starting and ending nodes for any trajectory. A set of edges $\mathcal{E} \subset \mathcal{V}^2$ defines possible transitions between the detections, and a choice of such set could be governed by maximum allowed distance in time between two adjacent detections in one trajectory, appearance similarity, or defined in any other way. Optimal trajectories, represented by the units of flows through such a network can then be defined by a set of transitions $\mathcal{T} \subset \mathcal{E}$. An example is provided in Fig. 2.1.

2.4 Datasets

Over years of research in multiple object tracking, it became a standard practice to compare tracking approaches on a benchmark set of sequences, reporting the obtained tracking results. More recently, this practice was further improved by supplying the benchmark sequences with benchmark set of detections. While comparing methods starting purely from sequences helps to identify the methods that achieve best tracking *overall*, it could often be hard to understand, whether the performance of the method comes from a better detector, or from a better data association that preserves the identities between frames. Thus, introducing a benchmark set of detections allows to compare tracking approaches independently of detector, facilitating progress by the ability to combine best detectors with best tracking approaches.

One of the first and most widely popular datasets is **PETS09**, a dataset featuring both single and multiple camera results in the scripted outdoor scene. Other datasets include **KITTI** [58], recorded from a camera on a moving car, **TUD** [4, 5], **Towncentre** [12], and **ETH** [46] and **Hotel** [140], to name a few. They feature outdoor scenes with both static and moving cameras. In, 2015, several sequences from these and a number of other datasets were combined in the **MOT15** [101] benchmark, which became a de-facto standard for evaluating performance on multiple object tracking algorithms, due to a large variety of scenarios (indoor and outdoor, moving and static camera), and presence of training-testing sequence pairs which feature same statistics. This benchmark was further improved by a set of **MOT16** [125] sequences, with more precise annotations and several object categories, and **MOT17** [125], which features the same set of sequences as **MOT16**, but 3 different detector types, allowing to compare tracking methods

Chapter 2. Background

Name	Annotated length, s	FPS	Trajectories	In/outdoor	Unscripted
PETS09 [51]	100	7	40	N/Y	-
ETH [46]	360	4.16	352	N/Y	+
TUD [4, 5]	18	25	31	N/Y	+
Hotel [140]	390	2.5	175	N/Y	+
Towncentre [12]	180	2.5	246	N/Y	+
KITTI [58]	4100	10	—	N/Y	+
MOT15 [101]	50% of 996	2.5-30	1221	N/Y	+
MOT16,MOT17 [125]	50% of 463	14-30	1276	Y/Y	+
Station [201]	3900	1.25	12362	Y/N	+
DukeMTMC [149]	5100	60	7000+	N/Y	+
WILDTRACK [31]	200 of 1800	60	313	Y/N	+
PathTrack [123]	10320	14-30	16287	Y/Y	+
PoseTrack [3]	550 of 2212	—	—	Y/Y	+
JTA [47]	15360	30	10752	—	-

Table 2.2 – Dataset statistics. Length and number of trajectories are summed up for all sequences of each dataset.

over a large variety of detectors.

In several recent years, introduction of much more large scale datasets allowed using more data-demanding methods for object tracking. In particular, **DukeMTMC** [149] dataset features more than an hour long recordings from 8 cameras in different spots on university campus, allowing to explore models of particular scenes for better tracking, as well as training across-camera re-identification models. **PathTrack** [123] dataset features several hundreds of short clips in a wide range of scenarios, and has been shown to be a significant aid in training learnable models for multiple object tracking. **Station** [201] dataset features almost an hour long recording from New York central station, with main focus on the group dynamics of the individuals. **WILDTRACK** [31] dataset features more than 30 minutes of recordings from 8 overlapping cameras for the purpose of multi-camera detection and tracking. **JTA** [47] dataset presents almost 500000 frames in a simulated environment, with annotations for multiple person tracking and pose estimation. **PoseTrack** [3] contains more than 500 short clips of multiple people in diverse settings with annotated human poses. We provide some statistics about the described datasets in Table 2.2. More detailed overview of datasets is available in [115, 101].

2.5 Metrics

Each of the metrics for multiple object tracking relies on the definition of whether a detection from a given trajectory reported by the tracker matches certain detection in the ground truth set of detections. The definition of such match is borrowed from the area of detection: in the image plane detection reported in the tracker and detection in the ground truth are assumed to match if their intersection-over-union (IoU) is greater than 0.5. In the ground plane or in the world

coordinates, detection and ground truth match if the world distance between the two is below 1 meter.

When a detection reported by the tracker is not matched by any ground truth detection, such a detection is considered to be a false positive (FP) detection. Conversely, when a detection is present in the ground truth, but not in the tracker output, it is considered to be a false negative (FN) detection. When two adjacent detections belong to the same trajectory in the ground truth, but to different trajectories in the tracker output, or vice versa, this situation is typically named identity switch (ID). Tracker output without false positives, false negatives, and identity switches is by definition equivalent to the ground truth (up to the precision of matching detections), but the way different metrics combine these errors are different and described below.

Historically, two of the most frequently used sets of metrics are a set of track quality measures [177] and a set of CLEAR MOT [18]. Recently, a set of identity-aware metrics has been proposed in [149] and was quickly adopted into main benchmark datasets. We describe each of the set of metrics below. Sec. 4.2 presents a concrete example of direct comparison between metrics from CLEAR MOT and identity-aware set of metrics, outlining why identity-aware metrics are more suitable for identity-preserving tracking.

Track quality measures Track quality measures identify the percentage of ground truth trajectories, that are mostly tracked, partially tracked, or mostly lost in the tracked output. Each trajectory in the ground truth, for which at least 80% of its detections are matched to some detections in the tracks reported by the tracker, is considered mostly tracked. If this number is below 20%, it is considered mostly lost, and partially tracked otherwise. Note that this measure does not take into account identity switches or false positives, but mostly concentrates on minimizing false negatives. To incorporate identity switches, this set of metrics is sometimes augmented with the number of fragmentations (FM) - number of times when one out of two adjacent detections in the ground truth is matched, and the other is not, or vice versa.

CLEAR MOT set of metrics features multiple object tracking accuracy (**MOTA**) and multiple object tracking precision (MOTP), with **MOTA** typically being a metric, by which most approaches are compared. **MOTA** is calculated as

$$1 - \frac{\sum_{t=1}^N FP_t + FN_t + ID_s_t}{\sum_{t=1}^N GT_t},$$

where GT_t is a number detections in the ground truth trajectories in frame t , and N is the total number of frames. In other words, **MOTA** ranges from 1 for perfectly recovered set of trajectories, to arbitrarily small negative number as the number of false negatives increases. **MOTA** is sensitive

Chapter 2. Background

to false positives, false negatives, and identity switches, but does not differentiate between a ground truth trajectory, which was identified as two equally long trajectories in the tracker output, and a ground truth trajectory, which was almost perfectly tracked, with one identity switch in the last frame. Such insensitivity to identity preservation was the main reason for the introduction of a set of identity aware metrics, which we describe below.

MOTP is calculated as an average misalignment error between ground truth and reported detection, for all matched pairs of detections. Misalignment can be defined as average IoU for tracking in the image plane, and average distance in the world coordinates otherwise. This metric is not sensitive to false negatives, false positives, or identity switches, and serves as a measure of localization error.

Identity-aware metrics Identity-aware metrics addresses the shortcoming of MOTA metric when it comes to identity preservation along the ground truth trajectory. This set of metrics includes identity-aware precision (IDP), identity-aware recall (IDR), and identity-aware F1 (IDF_1), with IDF_1 being the metric by which the trackers are ranked. At the basis of this metric is a computation of matching between ground truth and reported trajectories, that maximizes the number of matched detections. Once such a matching has been computed, IDP is defined as a total number of matched detections, normalized by a total number of detections in the ground truth trajectories, IDR is defined as a total number of matched detections, normalizes by a total number of detections in the reported trajectories, and IDF_1 is twice the total number of the matched detections, normalized by a total number of detections in ground truth plus reported trajectories. This metric is sensitive to false positives, false negatives, and identity preservation. The mathematical definition of this metric follows below:

$$\frac{2 * IDTP}{2 * IDTP + IDFP + IDFN},$$

where IDTP is the number of detections matched to ground truth after computing the assignment of reported trajectories to ground truth that maximizes this value, IDFP is the number of false positives in such an assignment, and IDFN is the number of false negatives in it. As can be seen, the denominator is actually a total number of detections in both trajectories and ground truth, because IDTP and IDFP comprise all of detections in the reported trajectories, and IDTP with IDFN comprise all of detections in the ground truth.

3 Physically constrained interaction modelling

Abstract

Tracking the ball is critical for video-based analysis of team sports. However, it is difficult, especially in low-resolution images, due to the small size of the ball, its speed that creates motion blur, and its often being occluded by players.

In this chapter, we propose a generic and principled approach to modeling the interaction between the ball and the players while also imposing appropriate physical constraints on the ball's trajectory.

We show that our approach, formulated in terms of a Mixed Integer Program, is more robust and more accurate than several state-of-the-art approaches on real-life volleyball, basketball, and soccer sequences.

3.1 Introduction

Tracking the ball accurately is critically important to analyze and understand the action in sports ranging from tennis to soccer, basketball, volleyball, to name but a few. While commercial video-based systems exist for the first, automation remains elusive for the others. This is largely attributable to the interaction between the ball and the players, which often results in the ball being either hard to detect because someone is handling it or even completely hidden from view. Furthermore, since the players often kick it or throw it in ways designed to surprise their opponents, its trajectory is largely unpredictable.

There is a substantial body of literature about dealing with these issues, but almost always using heuristics that are specific to a particular sport such as soccer [198], volleyball [60], or basketball [32]. A few more generic approaches explicitly account for the interaction between the players and the ball [172] while others impose physics-based constraints on ball motion [139]. However, neither of these things alone suffices in difficult cases, such as the one depicted by

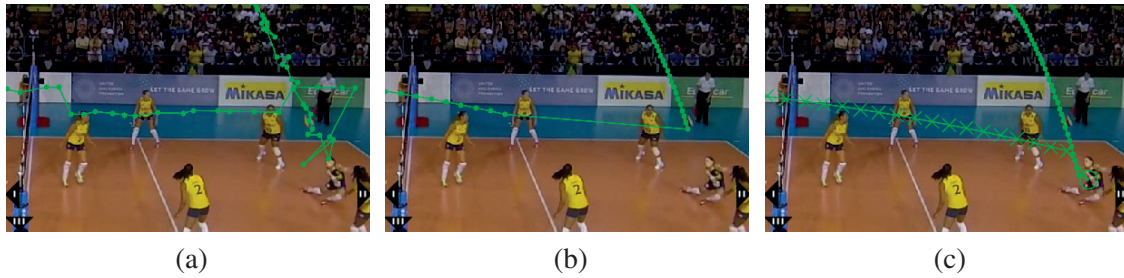


Figure 3.1 – Importance of simultaneously modeling interactions and imposing physical constraints. For most of this 70-frame volleyball sequence depicting the ball crossing the net and being bumped by a defending player and viewed by 3 cameras, the defending player is on the ground. As a result, she was not detected by the person detector we use [52] because it only finds people standing up. Furthermore, while the ball was near the player, it was occluded in the views of 2 of the 3 cameras, and, therefore, not detected as a 3D object. **(a)** Tracking the players and the ball simultaneously without imposing motion constraints as in [172] produces physically impossible trajectories. **(b)** Imposing motion constraints but tracking the players and the ball separately as in [139] does not properly capture the ball and player interaction. **(c)** Our approach to both imposing constraints and modeling the interaction gives a better overall result. The crosses denote the fact that the ball is in the “strike” state until being bumped and in the “flying” one after that. Transitions between these states can only result from interacting with a player, which encourages the optimizer to find one in spite of the weak evidence. Best viewed in color.

Fig. 3.1.

In this chapter, we, therefore, introduce an approach to simultaneously accounting for ball/player interactions and imposing appropriate physics-based constraints. Our approach is generic and applicable to many team sports. It involves formulating the ball tracking problem in terms of a Mixed Integer Program (MIP) in which we account for the motion of both the players and the ball as well as the fact the ball moves differently and has different visibility properties in flight, in possession of a player, or while rolling on the ground. We model the ball locations in \mathbb{R}^3 and impose first and second-order constraints where appropriate. The resulting MIP describes the ball behaviour better than previous approaches [172, 139] and yields superior performance, both in terms of tracking accuracy and robustness to occlusions. Fig. 3.1(c) depicts the improvement resulting from doing this rather than only modeling the interactions or only imposing the physics-based constraints.

In short, our contribution is a principled and generic formulation of the ball tracking problem and related physical constraints in terms of a MIP. We will demonstrate that it outperforms state-of-the-art approaches [171, 172, 139, 60] in soccer, volleyball, and basketball.

3.2 Related work

While there are approaches to game understanding, such as [98, 112, 114, 61, 42, 92], which rely on the structured nature of the data without any explicit reference to the location of the ball, most others either take advantages of knowing the ball position or would benefit from being able to [42]. However, while the problem of automated ball tracking can be considered as solved for some sports such as tennis or golf, it remains difficult for team sports. This is particularly true when the image resolution is too low to reliably detect the ball in individual frames in spite of frequent occlusions.

Current approaches to detecting and tracking can be roughly classified as those that build physically plausible trajectory segments on the basis of sets of consecutive detections and those that find a more global trajectory by minimizing an objective function. We briefly review both kinds below.

3.2.1 Fitting Trajectory Segments

Many ball-tracking approaches for soccer [133, 106], basketball [32], and volleyball [33, 60, 29] start with a set of successive detections that obey a physical model. They then greedily extend them and terminate growth based on various heuristics. In [147], Canny-like hysteresis is used to select candidates above a certain confidence level and link them to already hypothesized trajectories. Very recently, RANSAC has been used to segment ballistic trajectories of basketball shots towards the basket [139]. These approaches often rely heavily on domain knowledge, such as audio cues to detect ball hits [33] or model parameters adapted to specific sports [29, 32].

While effective when the initial ball detections are sufficiently reliable, these methods tend to suffer from their greedy nature when the quality of these detections decreases. We will show this by comparing our results to those of [60, 139], for which the code is publicly available and have been shown to be a good representatives of this set of methods.

3.2.2 Global Energy Minimization

One way to increase robustness is to seek the ball trajectory as the minimum of a global objective function. It often includes high-level semantic knowledge such as players' locations [202, 198, 171], state of the game based on ball location, velocity and acceleration [198, 202], goal events [202] or dynamically weighted combination of the features above [155].

In [172, 173], the players *and* the ball are tracked simultaneously and ball possession is explicitly modeled. However, the tracking is performed on a discretized grid and without physics-based constraints, which results in reduced accuracy. It has nevertheless been shown to work well on soccer and basketball data. We selected it as our baseline to represent this class of methods, because of its state-of-the-art results and publicly available implementation.

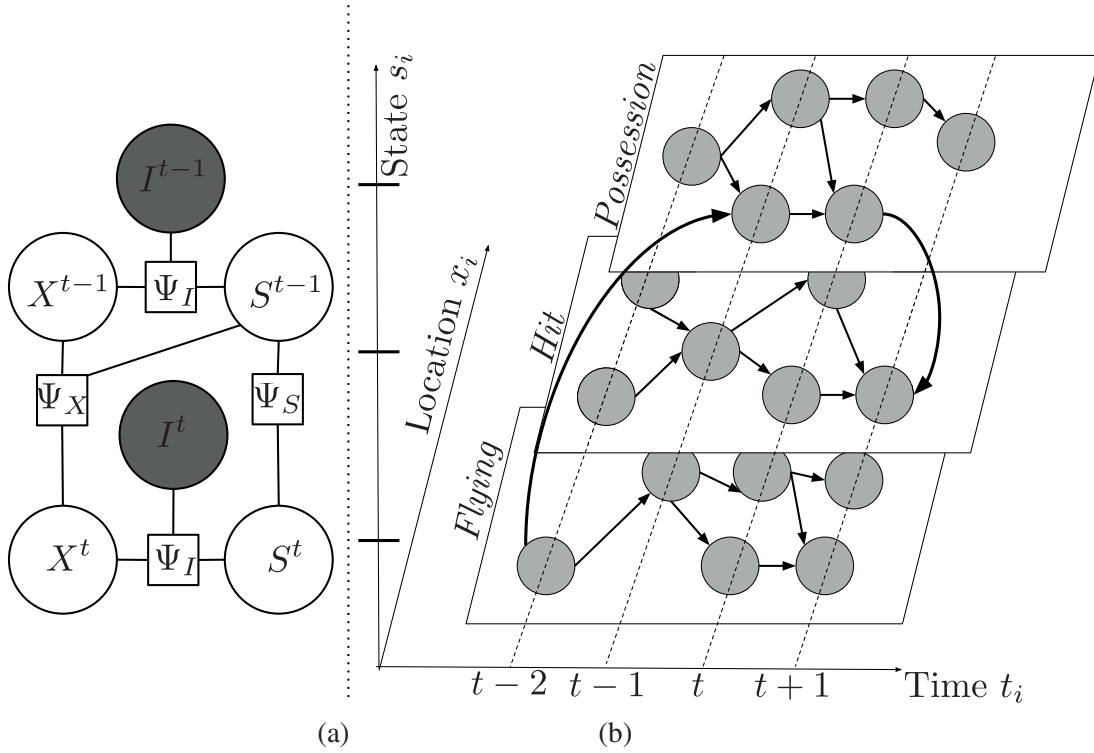


Figure 3.2 – Graphical models. **(a)** Factor graph for ball tracking. At each time instant t , we consider two latent variables, the ball location X^t and state S^t , along with the observed variable - available image evidence I^t . **(b)** Ball graph used to formulate the integer program. To each node i , is associated a location x_i , a state s_i , and a time instant t_i . The relationship between the variables in both graphs is spelled out in Eqs 3.3(d,e).

3.3 Problem Formulation

We consider scenarios where there are several calibrated cameras with overlapping fields of view capturing a substantial portion of the play area, which means that the apparent size of the ball is generally small. In this setting, trajectory growing methods do not yield very good results both because the ball is occluded too often by the players to be detected reliably and because its being kicked or thrown by them result in abrupt and unpredictable trajectory changes.

To remedy this, we explicitly model the interaction between the ball and the players as well as the physical constraints the ball obeys when far away from the players. To this end, we first formulate the ball tracking problem in terms of a maximization of a posteriori probability. We then reformulate it in terms of an integer program. Finally, by adding various constraints, we obtain the final problem formulation that is a Mixed Integer Program.

P^t	3D coordinates of the ball at time t
i, j, k, l	Node indices in the ball or players graph
$\mathcal{V}_b, \mathcal{V}_p$	Sets of nodes in ball and player graphs
$\mathcal{E}_b, \mathcal{E}_p$	Sets of edges in the ball and player graphs
S^t	State of the tracked object at time t
x_i, s_i, t_i	Discrete location, state, and time of node i
S_b	Special node for the ball at $t = 0$
f_i^j, p_i^j	Number of balls and players moving from i to j
c_{bi}^j, c_{pi}^j	Ball and player transition costs from i to j
Ψ_X, Ψ_S, Ψ_I	Position, state, image evidence potentials
ψ	Potential of local image evidence
D_l	Max. permissible distance between X^t and P^t
D_p	Max. permissible distance for ball possession
$A^{s,c}, B^{s,c}, C^{s,c}, F^{c,s}$	Physics-based constants for state s , axis c
$O^{s,c}$	Constraint-free locations for state s and axis c
\mathbb{F}	Permissible ball locations and state sequences

Table 3.1 – Notations specific to the problem of ball and player tracking in sports. The rest of the variables used in this chapter are defined in 2.1.

3.3.1 Graphical Model for Ball Tracking

We model the ball tracking process from one frame to the next in terms of the factor graph depicted by Fig. 3.2(a). We associate to each instant $t \in \{1 \dots N\}$ three variables X^t , S^t , and I^t , which respectively represent the 3D ball position, the state of the ball, and the available image evidence. When the ball is within the capture volume, X^t is a 3D vector and S^t can take values such as *flying* or *in_possession*, which are common to all sports, as well as sport-dependent ones, such as *strike* for volleyball or *pass* for basketball. When the ball is not present, we take X^t and S^t to be ∞ and *not_present* respectively. These notations as well as all the others we use in this chapter are summarized in Table 3.1, which presents notations specific to the tracking of ball and players in sports. General notation is also available in Table 2.1.

Given the conditional independence assumptions implied by the structure of the factor graph of Fig. 3.2(a), we can formulate our tracking problem as one of maximizing the function defined by the product of potentials:

$$\Psi(X, S, I) = \frac{1}{Z} \Psi_I(X^1, S^1, I^1) \prod_{t=2}^N \left[\Psi_X(X^{t-1}, S^{t-1}, X^t) \Psi_S(S^{t-1}, S^t) \Psi_I(X^t, S^t, I^t) \right] \quad (3.1)$$

expressed in terms of products of the following potential functions:

- $\Psi_I(X^t, S^t, I^t)$ encodes the correlation between the ball position, ball state, and the image evidence.

Chapter 3. Physically constrained interaction modelling

- $\Psi_S(S^{t-1}, S^t)$ models the temporal smoothness of states across adjacent frames.
- $\Psi_X(X^{t-1}, S^{t-1}, X^t)$ encodes the correlation between the state of the ball and the change of ball position from one frame to the next one.
- $\Psi_X(X^1, S^1, X^2)$ and $\Psi_S(S^1, S^2)$ include priors on the state and position of the ball in the first frame.

In practice, as will be discussed in Sec. 3.4, the parameters of Ψ functions are learned from training data. Let \mathbb{F} be the set of all possible sequences of ball positions and states. We consider the log of Eq. 3.1 and drop the constant normalization factor $\log Z$. We, therefore, look for the most likely sequence of ball positions and states as

$$(X^*, S^*) = \arg \max_{(X, S) \in \mathbb{F}} \sum_{t=2}^N \left[\log \Psi_X(X^{t-1}, S^{t-1}, X^t) + \log \Psi_S(S^{t-1}, S^t) + \log \Psi_I(X^t, S^t, I^t) \right] + \log \Psi_I(X^1, S^1, I^1). \quad (3.2)$$

In the following subsections, we first reformulate this maximization problem as an integer program and then introduce additional physics-based and *in_possession* constraints.

3.3.2 Integer Program Formulation

To convert the maximization problem of Eq. 3.2 into an Integer Program (IP), we introduce the *ball graph* $\mathcal{G}_b = (\mathcal{V}_b, \mathcal{E}_b)$ depicted by Fig. 3.2(b). \mathcal{V}_b represents its nodes, whose elements each correspond to a location $x_i \in \mathbb{R}^3$, state $s_i \in \{1, \dots, K\}$, and time index $t_i \in \{1, \dots, N\}$. In practice, we instantiate as many as there are possible states at every time step for every actual and potentially missed ball detection. Our approach to hypothesizing such missed detections is described in Sec. 3.5. \mathcal{V}_b also contains an additional node S_b denoting the ball location before the first frame. \mathcal{E}_b represents the edges of \mathcal{G}_b and comprises all pairs of nodes corresponding to consecutive time instants and whose locations are sufficiently close for a transition to be possible.

Let f_i^j is the binary indicator of the presence of the ball moving from i to j and c_{bi}^j denote the corresponding cost. The maximization problem of Eq. 3.2 can be rewritten as

$$\text{maximize } \sum_{(i,j) \in \mathcal{E}_b} f_i^j c_{bi}^j, \quad (3.3)$$

where

$$c_{bi}^j = \log \Psi_X(x_i, s_i, x_j) + \log \Psi_S(s_i, s_j) + \log \Psi_I(x_j, s_j, I^j),$$

subject to

$$\begin{aligned}
 (a) \quad & f_i^j \in \{0, 1\} && \forall (i, j) \in \mathcal{E}_b \\
 (b) \quad & \sum_{(i, j) \in \mathcal{E}_b, t_j=1} f_i^j = 1 \\
 (c) \quad & \sum_{(i, j) \in \mathcal{E}_b} f_i^j = \sum_{(j, k) \in \mathcal{E}_b} f_j^k && \forall j \in \mathcal{V}_b : 0 < t_j < N \\
 (d) \quad & X^t = \sum_{(i, j) \in \mathcal{E}_b, t_j=t} f_i^j x_j && \forall t \in 1, \dots, N \\
 (e) \quad & S^t = \sum_{(i, j) \in \mathcal{E}_b, t_j=t} f_i^j s_j && \forall t \in 1, \dots, N \\
 (f) \quad & (X, S) \in \mathbb{F}
 \end{aligned}$$

We optimize with respect to the f_i^j , which can be considered as flow variables. The flow through the network defines the movement of the ball through the sequence of nodes, each of which corresponds uniquely to the a pair of location and state. The constraints ensure that each possible allowed sequence of nodes in the flow uniquely corresponds to the physically possible sequence of pairs of ball locations and states. More specifically, the constraints of Eqs.3.3(a-c) ensure that at every time frame there exists only one position and one state to which the only ball transitions from the previous frame. The constraints of Eq.3.3(d-e) draw the connection between the flow variables that operate on edges, and ball locations and states X and S defined on the nodes, ensuring that the location and state of the ball corresponds to the node through which the flow goes in the graph. The constraint of Eq.3.3(f) is intended to only allow feasible combinations of locations and states as described by the set \mathbb{F} , which we define below.

3.3.3 Mixed Integer Program Formulation

Some ball states impose first and second order constraints on ball motion, such as zero acceleration for the freely flying ball or zero vertical velocity and limited negative acceleration for the rolling ball. Possession implies that the ball must be near the player.

Unfortunately, imposing the second order constraints requires allowing the location of the ball to be continuous, while the available set of possible ball locations is inherently discrete (discretized at most with the precision of 1 pixel) and therefore fitting a perfect second order model, such as a parabolic motion through these discrete locations is not possible. To alleviate this problem, we will introduce the set of continuous ball locations which will be near the ball detections, but perfectly adhere to the physical model that we will introduce.

In this section, we assume that the players' trajectories are available in the form of a *player graph* $\mathcal{G}_p = (\mathcal{V}_p, \mathcal{E}_p)$ similar to the ball graph of Sec. 3.3.2 and whose nodes comprise locations x_i and time indices t_i . In practice, we compute it using publicly available code as described in Sec. 3.5.1.

Given \mathcal{G}_p , the physics-based and possession constraints can be imposed by introducing auxiliary continuous variables and expanding constraint of Eq. 3.3(f), as follows.

Chapter 3. Physically constrained interaction modelling

Continuous Variables. The x_i represent specific 3D locations where the ball could potentially be, that is, either actual ball detections or hypothesized ones as will be discussed in Sec. 3.5.2. Since they cannot be expected to be totally accurate, let the continuous variables $P^t = (P_x^t, P_y^t, P_z^t)$ denote the true ball position of at time t . We impose

$$\|P^t - X^t\| \leq D_l \quad (3.4)$$

where D_l is a constant that depends on the expected accuracy of the x_i . These continuous variables can then be used to impose ballistic constraints when the ball is in flight or rolling on the ground as follows.

Second-Order Constraints. For each state s and coordinate c of P , we can formulate a second-order constraint of the form

$$A^{s,c}(P_c^t - 2P_c^{t-1} + P_c^{t-2}) + B^{s,c}(P_c^t - P_c^{t-1}) + C^{s,c}P_c^t - F^{s,c} \leq K(3 - M_{s,c}^t - M_{s,c}^{t-1} - M_{s,c}^{t-2}), \quad (3.5)$$

where $M_{s,c}^t = \sum_{(i,j) \in \mathcal{E}_b, t_j=t, s_j=s, x_j \notin O^{s,c}} f_i^j$,

K is a large positive constant and $O^{s,c}$ denotes the locations where there are scene elements with which the ball can collide, such as those near the basketball hoops or close to the ground. Given the constraints of Eq. 3.3, $M_{s,c}^t$, $M_{s,c}^{t-1}$, and $M_{s,c}^{t-2}$ must be zero or one. This implies that right side of the above inequality is either zero if $M_{s,c}^t = M_{s,c}^{t-1} = M_{s,c}^{t-2} = 1$ or a large number otherwise. In other words, the constraint is only effectively active in the first case, that is, when the ball consistently is in a given state. When this is the case, $(A^{s,c}, B^{s,c}, C^{s,c}, F^{s,c})$ model the corresponding physics. For example, when the ball is in the *flying* state, we use $(1, 0, 0, \frac{-g}{f p^2})$ for the z coordinate to model the parabolic motion of an object subject to the sole force of gravity whose intensity is g . In the *rolling* state, we use $(1, 0, 0, 0)$ for both the x and y coordinates to denote a constant speed motion in the xy plane. In both cases, we neglect the effect of friction. Note that we turn off these constraints altogether at locations in $O^{s,c}$.

Possession constraints. While the ball is in possession of a player, we do not impose any physics-based constraints. Instead, we require the presence of someone nearby. The algorithm we use for tracking the players [17] is implemented in terms of people flows that we denote as p_i^j on a player graph $\mathcal{G}_p = (\mathcal{V}_p, \mathcal{E}_p)$ that plays the same role as the ball graph. The p_i^j are taken to be those that

$$\text{maximize } \sum_{(i,j) \in \mathcal{E}_p} p_i^j c_{pi}^j, \quad (3.6)$$

where $c_{pi}^j = \frac{\log P_p(x_i|I^{t_i})}{1 - \log P_p(x_i|I^{t_i})}$,
subject to

$$\begin{aligned} (a) \quad & p_i^j \in \{0, 1\} && \forall (i, j) \in \mathcal{E}_p \\ (b) \quad & \sum_{i:(i,j) \in \mathcal{E}_p} p_i^j \leq 1 && \forall j \in \mathcal{V}_p \setminus \{v_{in}\} \\ (c) \quad & \sum_{(i,j) \in \mathcal{E}_p} p_i^j = \sum_{(j,k) \in \mathcal{E}_p} p_j^k && \forall j \in \mathcal{V}_p \setminus \{v_{in}, v_{out}\}. \end{aligned}$$

Here $P_p(x_i|I^{t_i})$ represents the output of probabilistic people detector at location x_i given image evidence I^{t_i} . $v_{in}, v_{out} \in \mathcal{V}_p$ are the source and sink nodes that serve as starting and finishing points for people trajectories, as in [17]. In practice we use the publicly available code of [52] to compute the probabilities P_p in each grid cell of discretized version of the court.

Given the ball flow variables f_i^j and people flow ones p_i^j , we express the *in_possession* constraints as

$$\sum_{\substack{(k,l) \in \mathcal{E}_p, t_l = t_j, \\ \|x_j - x_l\|_2 \leq D_p}} p_k^l \geq \sum_{i:(i,j) \in \mathcal{E}_b} f_i^j \quad \forall j : s_j \equiv \text{in_possession}, \quad (3.7)$$

where D_p is the maximum possible distance between the player and the ball when location n the player is in control of it, which is sport-specific. We learn this value from the training data, similar to how we learn the maximum permissible speed of the ball in each specific state.

Resulting MIP. Using the physics-based constraints of Eq. 3.4 and 3.5 and the possession constraints of Eq. 3.7 along with the formulation of people tracking from Eq. 3.6 to represent the feasible set of states F of Eq. 3.3(f) yields the MIP

$$\begin{aligned} & \text{maximize} \quad \sum_{(i,j) \in \mathcal{E}_b} f_i^j c_{bi}^j + \sum_{(i,j) \in \mathcal{E}_p} p_i^j c_{pi}^j \\ & \text{subject to the constraints of Eqs. 3.3(a-e), 3.4, 3.5, 3.6(a-c), and 3.7.} \end{aligned} \quad (3.8)$$

In practice, we use the Gurobi [137] solver to perform the optimization. This allows us to solve the formulated integer program globally optimally. Note that we can either consider the people flows as given and optimize only on the ball flows or optimize on both simultaneously. We will show in the results section that the latter is only slightly more expensive but yields improvements in cases such as the one of Fig. 3.1.

3.4 Learning the Potentials

In this section, we define the potentials introduced in Eq. 3.2 and discuss how their parameters are learned from training data. They are computed on the nodes of the ball graph G_b and are used to compute the cost of the edges, according to Eq. 3.3. We discuss its construction in Sec. 3.5.2. Note that we do a piece-wise training to learn parameters of each potential separately. We do not perform any end-to-end training, and do not define any weights for the potentials we use in

Chapter 3. Physically constrained interaction modelling

the cost of the edges. The main reason for that is that potentials Ψ_X and Ψ_S act as strict cutoff, forbidding transitions between detections that are too far apart, or impossible transitions between states. As such, there is not need to learn any weights for them, and the main informative part of the cost function comes from the image evidence potential Ψ_I , described below.

Image evidence potential Ψ_I . It models the agreement between location, state, and the image evidence. We write

$$\begin{aligned}\Psi_I(x_i, s_i, I) &= \psi(x_i, s_i, I) \prod_{\substack{j \in \mathcal{J}_b: t_j = t, \\ (x_j, s_j) \neq (x_i, s_i)}} \left(1 - \psi(x_j, s_j, I)\right), \\ \psi(x, s, I) &= \sigma_s(P_b(x|I)P_c(s|x, I)), \\ \sigma_s(y) &= \frac{1}{1 + e^{-\theta_{s0} - \theta_{s1}y}},\end{aligned}\tag{3.9}$$

where $P_b(x)$ represents the output of a ball detector for location x , $P_c(s|x, I)$ the output of multiclass classifier that predicts the state s given the position and the local image evidence. $\psi(x, s, I)$ is close to one when the ball is likely to be located at x in state s with great certainty based on image evidence only and its value decreases as the uncertainty of either estimates increases.

In practice, we train a Random Forest classifier [24] to estimate $P_c(s|x, I)$. As features, it uses the 3D location of the ball. Additionally, when the player trajectories are given, it uses the number of people in its vicinity as a feature. When simultaneously tracking the players and the ball, we instead use the integrated outputs of the people detector in the vicinity of the ball. We give additional details in the appendix.

The parameters θ_{s0}, θ_{s1} of the logistic function σ_s are learned from training data for each state s . Given the specific ball detector we rely on, we use true and false detections in the training data as positive and negative examples to perform a logistic regression.

State transition potential Ψ_S . We define it as the transition probability between states, which we learn from the training data by counting, that is:

$$\Psi_S(s_i, s_j) = P(S^t = s_i | S^{t-1} = s_j).\tag{3.10}$$

As noted in Sec. 3.3.1, potential for the first time frame has a special form $P(S^2 = s_i | S^1 = s_j)P(S^1 = s_j)$, where $P(S^1 = s_j)$ is the probability of the ball being in state s_j at arbitrary time instant; it is learned from the training data.

Location change potential Ψ_X . It models the transition of the ball between two time instants. Let D^s denote the maximum speed of the ball when in state s . We write it as

$$\Psi_X(x_i, s_i, x_j) = \mathbb{1}(\|x_i - x_j\|_2 \leq D^{s_i}). \quad (3.11)$$

For the *not_present* state, we only allow transitions between the node representing the absent ball and the nodes near the border of the tracking area. For the first frame the potential has an additional factor of $P(X^1 = x_i)$, ball location prior, which we assume to be uniform inside of the tracking area.

3.5 Implementation details

Recall from Sections 3.3.2 and 3.3.3, that our algorithm operates on a ball and player graph. We describe the procedure of building them in Sec. 3.5.1 and Sec. 3.5.2. We provide details of how we handle different ball states in Sec. 3.5.3, and details of learning the potentials from Sec. 3.4 in Sec. 3.5.4. In both sections we describe the used detectors, since the output of the ball detections is used when computing the image evidence potential Ψ_I , as described earlier in this section, and the people detector output is used when constructing the cost of transition between people detections, as described in 3.6.

3.5.1 Player Graph

To detect the players, we first compute a Probability Occupancy Map on a discretized version of the court or field using the algorithm of [52]. We then follow the promising approach of [172]. We use the K-Shortest-Path (KSP) [17] algorithm to produce tracklets, which are short trajectories with high confidence detections. To hypothesize the missed detections, we use the Viterbi algorithm on the discretized grid to connect the tracklets. Each individual location in a tracklet or path connecting tracklets becomes a node of the player graph G_p , it is then connected by an edge to the next location in the tracklet or path.

3.5.2 Ball Graph

To detect the ball, we use a SVM [64] to classify image patches in each camera view based on Histograms of Oriented Gradients, HSV color histograms, and motion histograms. We then triangulate these detections to generate candidate 3D locations and perform non-maximum suppression to remove duplicates. We then aggregate features from all camera view for each remaining candidate and train a second SVM to only retain the best.

Given these high-confidence detections, we use KSP tracker to produce ball tracklets, as we did for people. However, we can no longer use the Viterbi algorithm to connect them as the resulting

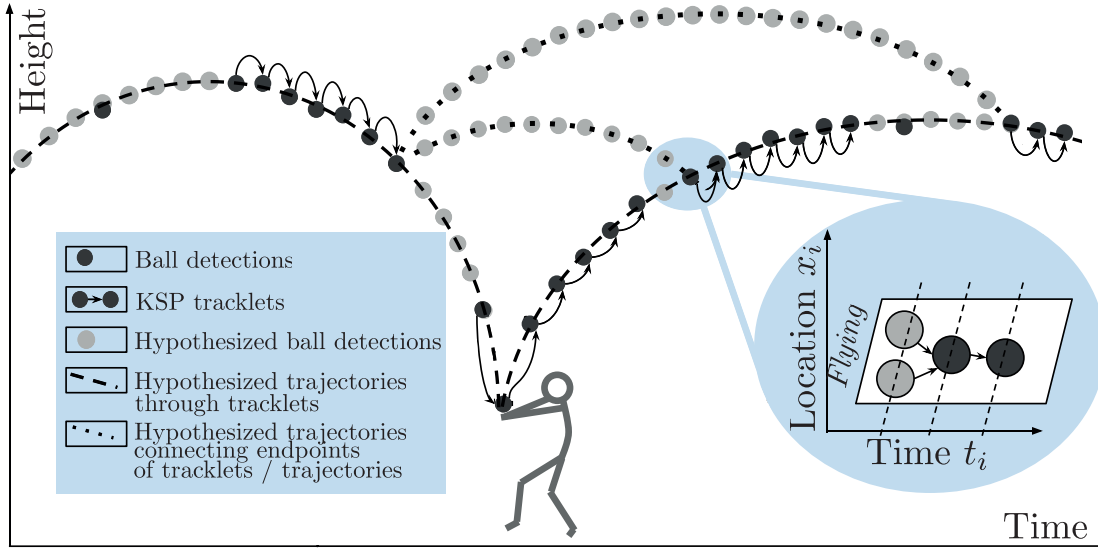


Figure 3.3 – An example of ball detections, hypothesized ball locations when it is missed, and graph construction.

connections may not obey the required physical constraints.

To model the ball states associated to a physical model, we grow the trajectories from each tracklet based on the physical model, and then join the end points of the tracklets and grown trajectories, by fitting the physical model. An example of such procedure is shown in Fig. 3.3. To model the state *in_possession*, we create a copy of each node and edge in the players graph. To model the state *not_present*, we create one node in each time instant and connect it to the node in the next time instant, and nodes for all other states in the vicinity of the tracking area border. Finally, we add edges between pairs of nodes with different states, as long as they are in the vicinity of each other (bold in Fig. 3.2(b)).

3.5.3 Ball states

Here we describe the physical models associated with the different ball states. We use different states for the ball in different sports. We use some states - *flying*, *in_possession* for all sports, and others only for some. For volleyball we add *strike*, and for basketball *pass*. For both soccer and volleyball we add *rolling*, and for basketball our sequences did not feature the ball rolling on the ground. Tab. 3.2 describes the physical constants for all states, introduced in Eq. 3.5. Note that in all cases we limit the absolute value of acceleration / speed / location of the ball, which means that we actually have 2 constraints for each row in the table, with exactly opposite coefficients

$$\begin{aligned}
 A^{s,c}(P_c^t - 2P_c^{t-1} + P_c^{t-2}) + B^{s,c}(P_c^t - P_c^{t-1}) + C^{s,c}P_c^t - F^{s,c} &\leq K(3 - M_{s,c}^t - M_{s,c}^{t-1} - M_{s,c}^{t-2}), \\
 -A^{s,c}(P_c^t - 2P_c^{t-1} + P_c^{t-2}) - B^{s,c}(P_c^t - P_c^{t-1}) - C^{s,c}P_c^t + F^{s,c} &\leq K(3 - M_{s,c}^t - M_{s,c}^{t-1} - M_{s,c}^{t-2}).
 \end{aligned}$$

3.5. Implementation details

They limit the values from above and from below, respectively. Notation used is the same notation we use in Eq. 3.5. Note that physical models for states *flying*, *strike* and *pass* are identical. We differentiate between those states using our state classifier.

State(s) s	Axis c	$A^{s,c}$	$B^{s,c}$	$C^{s,c}$	$F^{s,c}$	Explanation
<i>flying</i> , <i>strike</i> , <i>pass</i> , <i>rolling</i>	X	1	0	0	0	Constant speed in ground plane
<i>flying</i> , <i>strike</i> , <i>pass</i> , <i>rolling</i>	Y	1	0	0	0	Constant speed in ground plane
<i>flying</i> , <i>strike</i> , <i>pass</i>	Z	1	0	0	$\frac{-g}{fps^2}$	Negative g acceleration in vertical plane
<i>rolling</i>	Z	0	0	1	0	Constant height of the ball
<i>flying</i> , <i>pass</i> , <i>rolling</i>	X, Y, Z	0	1	0	$\frac{20000(mm)}{fps}$	Maximum speed limitation
<i>strike</i>	X, Y, Z	0	1	0	$\frac{35000(mm)}{fps}$	Maximum speed limitation

Table 3.2 – Physical models associated with different states of the ball. $g = 9810(\frac{mm}{s^2})$ denotes the free fall acceleration, fps denotes frame rate of the video sequence. Coefficients shown only for constraints that limit from above. Coefficients for limiting from below have the same magnitude as those in the table but are negative.

States with physical model We compute tracklets by joining detections using the K-Shortest-Paths algorithm. Then we create trajectories that go through these tracklets, and additional trajectories that join tracklets and previously built trajectories. The trajectories of the first type represent hypothesized ball locations where the detector has been unable to find the ball. The trajectories of the second type represent hypothesized ball locations when the whole trajectory is missing.

To create ball trajectories that go through the tracklets, we use the following procedure:

1. We start from every pair of consecutive detections in the tracklet. We fit a trajectory that obeys the physical model and goes through these two points. Note that there is only one straight line (for *rolling*) and only one parabola (for *flying*, *strike*, *pass*) that goes through two given points. Uniqueness of parabola is due to fixed force of gravity.
2. We grow the trajectory. At each next frame, if there is indeed a detection within the distance D_l of the trajectory, we add it to the trajectory, and recompute the best model fit through a new set of detections. If there are several detections, we pick the one with the highest confidence. We continue growing the trajectory until it leaves the tracking area
3. From a set of trajectories, we keep only those that are associated with the maximal set of detections. In other words, if we have a trajectory with a set of detections, that is a subset of detections of some other trajectory, we discard the former.

Chapter 3. Physically constrained interaction modelling

D_l represents the distance between the detection and continuous location of the ball, as defined by Eq. 4 of the chapter. We use $D_l = 25cm$ for basketball and volleyball, and $D_l = 75cm$ for football. The value is larger for football both because the cameras are further away from the players and because players often spin the ball so that it follows a curved, rather than straight trajectory. Furthermore, friction can be quite high for the rolling ball, violating the constant velocity constraint.

To join the tracklets and previously built trajectories, we consider all starting and ending points of trajectories and tracklets. We link every pair that are at most 4 seconds apart. We have empirically found that linking those further apart does not improve matters in terms of accuracy, but increases the computational burden.

In_possession state For the *in_possession* state, we create a copy of every node and every edge in the players graph. We associate with each node a detection with the highest confidence at the distance of D_p from the players possible location. We use $D_p = 1m$ for basketball and volleyball, and $D_p = 2.5m$ for football. The value is larger for football because players can bounce the ball further away from themselves when they run with it.

Not_present state For the *not_present* state, we have one node associated with this state in each time frame. We connect such node by an edge to the node with *not_present* state in the next time frame, as well as to all nodes of all states close to the border of the tracking area. More formally, we connect it to all nodes that are within distance D^{max} , where D^{max} is the maximum distance the ball can travel in any of the states within one frame time. As shown in Tab. 1, we take $D^{max} = \frac{35000(mm)}{fps}$.

Edges between states With the exception of *not_present* state, discussed above, we follow one rule for all edges that connect nodes of different states: we join those in the neighbouring frames which are within D^{max} distance of each other.

Post-processing We applied post-processing in the form of smoothing to the results for better appearance of videos. Reported results are without smoothing. Furthermore, as we show below, smoothing does not significantly affect tracking accuracy. Smoothing of the following form was used:

- State smoothing. Ball that left possession of the player and returned within 0.1 seconds is assumed to have never left.
- State smoothing. Ball that was assigned to possession, but returned to flying withing 0.1 seconds and did not change course, is assumed to have stayed on course during this period.

- Ground collision smoothing. We assume the ball to be in contact with the ground in the case of collision if it is below a certain threshold. If there are several such points, we assume them all to have equal zero height.

As Fig. 3.5, left, shows, tracking accuracy with and without smoothing does not differ much. It was obtained on Volley-1 dataset, and we saw similar results on all other datasets.

3.5.4 State classifier

Here we describe the state classifier, introduced in Sec. 3.4. For each sport, we learn a multi-class Random Forest classifier to predict the ball state. We use the 3D ball location and the number of people around it as features. We use neighbourhoods of sizes 1000mm / 500mm and 2000mm / 2500mm for volleyball / basketball / soccer respectively.

When we are simultaneously tracking the ball and the players, the ground truth positions of the players are unknown. In this case, we substitute the number of people by the predicted number of people, which we compute by integrating the Probabilistic Occupancy Map near the ball location. To improve the performance of the classifier given limited amount of available data we take advantage of the court symmetry of the field with respect to the 180 degrees rotation around the center. We treat all data points as if they were located on one side of the field.

Fig. 3.4 presents a partial evaluation of tracking accuracy as a function of the number of frames in the training data to address the question of whether the training data we use is sufficient. Clearly, the more the better, but above 1000 frames the further improvement becomes small.

Fig. 3.5, right, depicts the output of our classifier for volleyball data. Near the ground, probability of having a freely flying ball is low, and most predictions correspond to *flying* and *in_possession*. At 2.5 meters height, predictions are mixed, with *flying* predictions at the ends of the field, that correspond to locations from which the players serve the ball. At the height of 3.5m predicted state is mostly *flying*, except for a stripe in the middle, where players often strike the ball after the jump. Possession by players at such height is not likely. Higher than 4 meters classifier predicts *flying* for all locations. We have checked that cross-validation classification error was under 7% for all our datasets.

3.6 Experiments

In this section, we compare our results to those of several state-of-the-art multi-view ball-tracking algorithms [171, 172, 139], a monocular one [60], as well as two tracking methods that could easily be adapted for this purpose [195, 17].

We first describe the datasets we use for evaluation purposes. We then briefly introduce the methods we compare against and finally present our results.

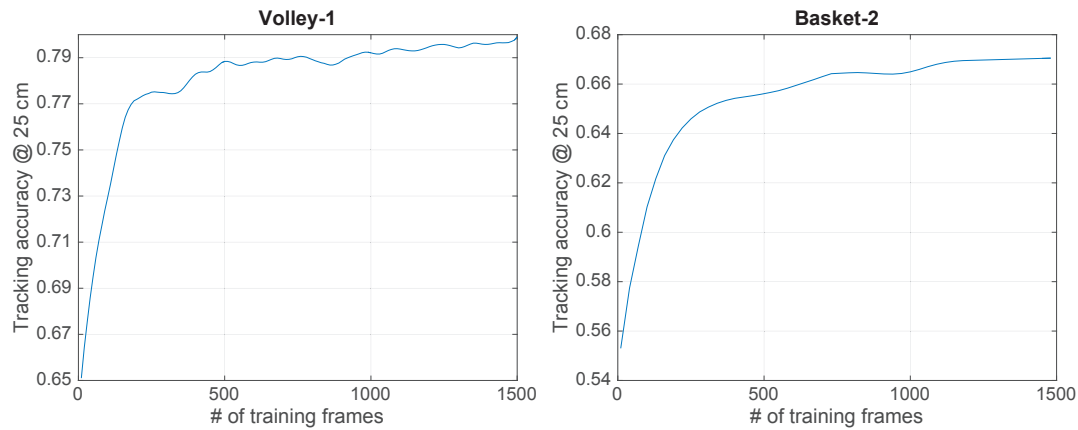


Figure 3.4 – Tracking accuracy at 25 cm for volleyball and basketball as a function of the number of training frames. For consistency, we increased the number of training frames for Basket-2 to 1500.

3.6.1 Datasets

We use two volleyball, three basketball, and one soccer sequences, which we detail below.

Basket-1 and Basket-2 comprise a 4000- and a 3000-frame basketball sequences captured by 6 and 7 cameras, respectively. These synchronized 25-frame-per-second cameras are placed around the court. We manually annotated each 10th frame of Basket-1 and 500 consecutive frames of Basket-2 that feature flying ball, passed ball, possessed ball and ball out of play. We used the Basket-1 annotations to train our classifiers and the Basket-2 ones to evaluate the quality of our results, and vice versa.

Basket-APIDIS is also a basketball dataset [166] captured by seven unsynchronized 22-frame-per-second cameras. A pseudo-synchronized 25-frame-per-second version of the dataset is also available and this is what we use. The dataset is challenging because the camera locations are not good for ball tracking and lighting conditions are difficult. We use 1500 frames with manually labeled ball locations provided by [139] to train the ball detector, and Basket-1 sequence to train the state classifier. We report our results on another 1500 frames that were annotated manually in [166].

Volley-1 and Volley-2 comprise a 10000- and a 19500-frame volleyball sequences captured by three synchronized 60-frame-per-second cameras placed at both ends of the court and in the middle. Detecting the ball is often difficult both because on either side of the court the ball can be seen by at most two cameras and because, after a strike, the ball moves so fast that it is blurred in middle camera images. We manually labeled each third frame in 1500-frame segments of both sequences. As before, we used one for training and the other for evaluation.

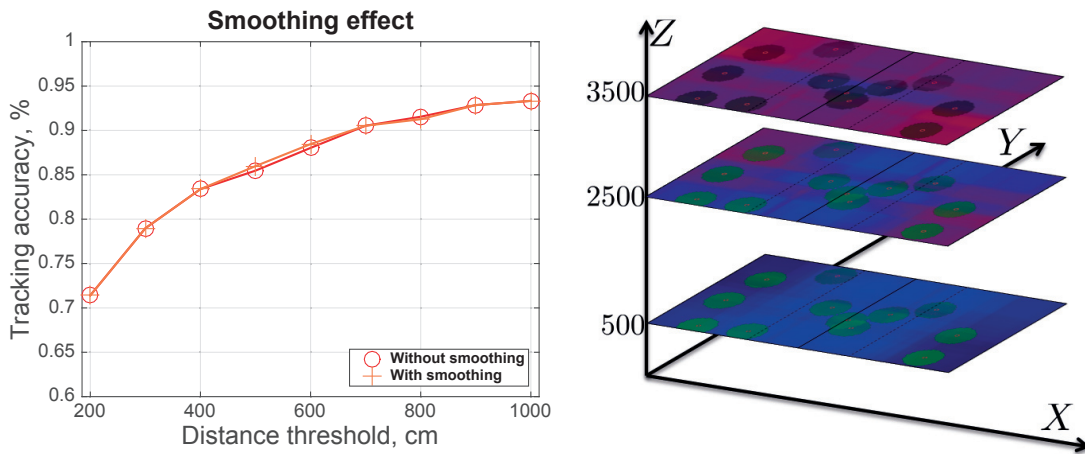


Figure 3.5 – **Left:** Tracking accuracy curve for results with and without smoothing on Volley-1 dataset. **Right:** Example of the classifier output on the volleyball data. Input to the classifier is the 3D location of the ball and the number of people in the vicinity. Picture shows the output of classifier for different xy-locations of the ball, and different heights. Ground-plane positions of the players are denoted by little red circles. R,G,B components of the color indicate the output of the classifier, probability of states *flying*, *in_possession*, and *strike*, respectively.

Soccer-ISSIA is a soccer dataset [45] captured by six synchronized 25-frame-per-second cameras located on both sides of the field. As it is designed for player tracking, the ball is often out of the field of view when flying. We train on the 1000 frames and report results on another 1000.

In all datasets, the apparent size of the ball is so small that state-of-the-art monocular object tracker [195] was unable to track the ball reliably for more than several seconds.

3.6.2 Baselines

We use several recent multi-camera ball tracking algorithms as baselines. To ensure a fair comparison, we ran all publicly available approaches with the same set of detections, which were produced by the ball detector described in Sec. 3.5.2. We briefly describe these algorithms below.

- **InterTrack** [172] introduces an Integer Programming approach to tracking two types of interacting objects, one of which can contain another. Modeling the ball as being “contained” by the player in possession of it was demonstrated as a potential application. In [173], this approach is shown to outperform several multi-target tracking approaches [143, 100] for ball tracking task.
- **RANSAC** [139] focuses on segmenting ballistic trajectories of the ball and was originally proposed to track it in the Basket-APIDIS dataset. This approach is shown to outperform the earlier graph-based filtering technique of [138]. We found that it also performs well

Chapter 3. Physically constrained interaction modelling

in our volleyball datasets that feature many ballistic trajectories. For the Soccer-ISSIA dataset, we modified the code to produce linear rather than ballistic trajectories.

- **FoS** [171] focuses on modeling the interaction between the ball and the players, assuming that long passes are already segmented. In the absence of a publicly available code, we use the numbers reported in the article for Basket-1-2-APIDIS and on Soccer-ISSIA.
- **Growth** [60] greedily grows the trajectories instantiated from points in consecutive frames. Heuristics are used to terminate trajectories, extend them and link neighbouring ones. It is based on the approach of [33] and shown to outperform approaches based on the Hough transform. Unlike the other approaches, it is monocular and we used as input our 3D detections reprojected into the camera frame.

To refine our analysis and test the influence of specific element of our approach, we used the following approaches.

- **MaxDetection**. To demonstrate the importance of tracking the ball, we give the results obtained by simply choosing the detection with maximum confidence.
- **KSP** [17]. To demonstrate the importance of modeling interactions between the ball and the players, we use the publicly available KSP tracker to track only the ball, while ignoring the players.
- **OUR-No-Physics**. To demonstrate the importance of second-order constraints of Eq. 3.5, we turn them off.
- **OUR-Two-States**. To demonstrate the impact of keeping track of many ball states, we assume that the ball can only be in possession and free motion.

3.6.3 Metrics

Our method tracks the ball and estimates its state. We use a different metric for each of these two tasks.

Tracking accuracy at distance d is defined as the percent of frames in which the location of the tracked ball is closer than d to the ground truth location.

The curve obtained by varying d is known as the “precision plot” [7]. When the ball is *in_possession*, its location is assumed to be that of the player possessing it. If the ball is reported to be *not_present* while it really is present, or vice versa, the distance is taken to be infinite.

Event accuracy measures how well we estimate the state of the ball. We take an **event** to be a maximal sequence of consecutive frames with identical ball states. Two events are said to match if there are not more than 5 frames during which one occurs and not the other, which we have empirically found to be enough to account for small offsets in the reported state value, but not large enough to match events that are actually different. Event accuracy then is a symmetric measure we obtain by counting recovered events that matched ground truth ones, as well as the ground truth ones that matched the recovered ones, normalized by dividing it by the number of events in both sequences.

3.6.4 Comparative Results

We now compare our approach to the baselines in terms of the above metrics. As mentioned in Sec. 3.3.3, we obtain the players trajectories by first running the code of [52] to compute the player’s probabilities of presence in each separate frame and then that of [17] to compute their trajectories. We first report accuracy results when these are treated as being correct, which amounts to fixing the p_i^j in Eq. 3.8, and show that our approach performs well. We then perform joint optimization, which yields a further improvement. We report the computational efficiency and all the algorithm parameters below. Our approach requires 3 to 40 seconds for the 500-frame sequences we tested. Our code is publicly available ¹.

Tracking and Event Accuracy. As shown in Fig. 3.6(a-f), **OUR** complete approach, outperforms the others on all 6 datasets. Two other methods that explicitly model the ball/player interactions, **OUR-No-Physics** and **InterTrack**, come next. **FoS** also accounts for interactions but does markedly worse for small distances, probably due to the lack of an integrated second order model.

Volleyball. The differences are particularly visible in the Volleyball datasets that feature both interactions with the players and ballistic trajectories. Note that **OUR-Two-States** does considerably worse, which highlights the importance of modeling the different states accurately.

Basketball. The differences are less obvious in the basketball datasets where **OUR-No-Physics** and **InterTrack**, which model the ball/player interactions without imposing global physics-based constraints, also do well. This reflects the fact that the ball is handled much more than in volleyball. As a result, our method’s ability to also impose strong physics-based constraints has less overall impact.

Soccer. On the soccer dataset, the ball is only present in about 75% of the frames and we report our results on those. Since the ball is almost never seen flying, the two states (*in_possession*

¹<http://cvlab.epfl.ch/research/balltracking>

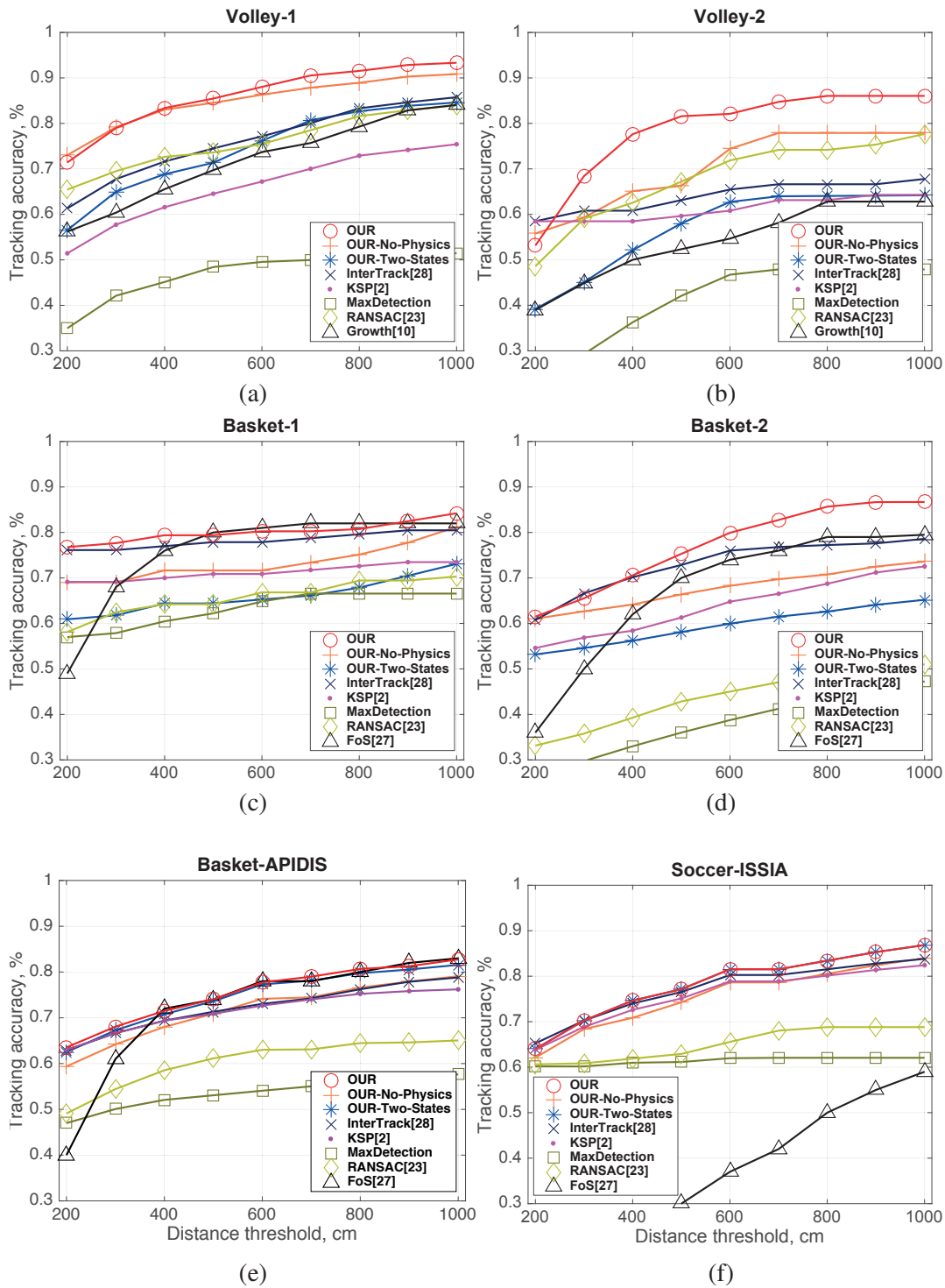


Figure 3.6 – Comparative results for ball tracking. **OUR** outperforms the other approaches in terms of ball accuracy, followed by the other methods that also model ball/player interaction, **OUR-No-Physics**, **InterTrack**, and **FoS** for larger values of d .

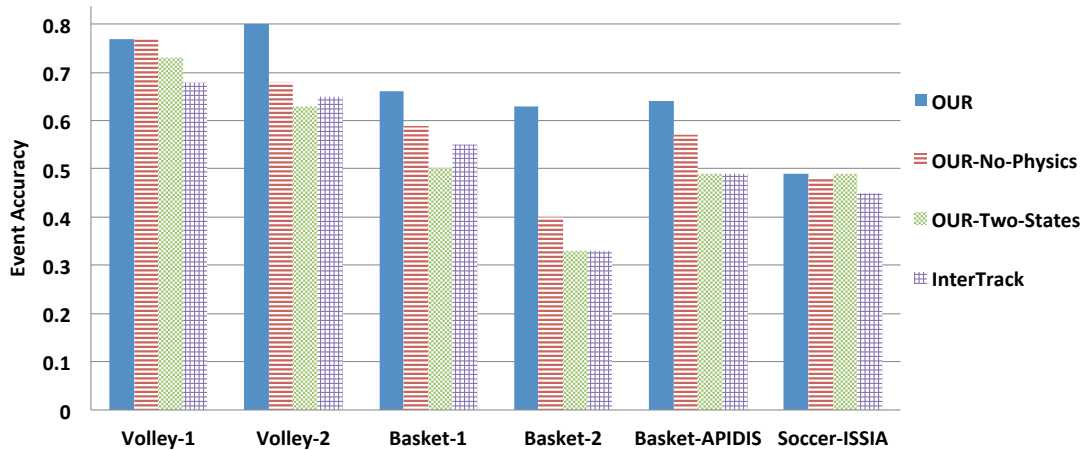


Figure 3.7 – Comparative results for game state estimation. **OUR** performs best in terms of event accuracy.

and *rolling*) suffice, which explains the very similar performance of **OUR** and **OUR-Two-States**. **KSP** also performs well because in soccer occlusions during interactions are less common than in other sports. Therefore, handling them delivers less of a benefit.

Our method also does best in terms of event accuracy, among the methods that report the state of the ball, as shown in Fig. 3.6(g). As can be seen in Fig. 3.8, both the trajectory and the predicted state are typically correct. Most state assignment errors happen when the ball is briefly assigned to be *in_possession* of a player when it actually flies nearby, or when the ball is wrongly assumed to be in free motion, while it is really *in_possession* but clearly visible.

Simultaneous tracking of the ball and players. All the results shown above were obtained by processing sequences of at least 500 frames. In such sequences, the people tracker is very reliable and makes few mistakes. This contributes to the quality of our results at the cost of an inevitable delay in producing the results. Since this could be damaging in the live-broadcast situation, we have experimented with using shorter sequences. We show here that simultaneously tracking the ball and the players can mitigate the loss of reliability of the people tracker, albeit to a small extent. **MODA** or multiple object detection accuracy metric reports average percentage of players detected in each frame, that match the ground truth, penalized for false and missed detections.

As shown in Tab. 3.3 for the Volley-1 dataset, we need 200-long frames to get the best people tracking accuracy when first tracking the people by themselves first, as we did before. As the number of frames decreases, the people tracker becomes less reliable but performing the tracking simultaneously yields a small but noticeable improvement both for the ball and the players. The case of Fig. 3.1 is an example of this. We identified 3 similar cases in 1500 frames of the volleyball sequence used for the experiment.

Batch	MODA [85],%	Tracking acc. @ 25 cm, %
50	94.1 / 93.9 / 0.26	69.2 / 67.2 / 2.03
75	94.5 / 94.2 / 0.31	71.4 / 69.4 / 2.03
100	96.5 / 96.3 / 0.21	72.5 / 71.0 / 1.41
150	97.2 / 97.1 / 0.09	73.8 / 73.0 / 0.82
200	97.3 / 97.4 / 0.00	74.1 / 74.1 / 0.00

Table 3.3 – Tracking the ball given the players’ locations vs. simultaneous tracking of the ball and players. The three numbers in both columns correspond to simultaneous tracking of the players and ball / sequential tracking of the players and then the ball / improvement, as function of the lengths of the sequences.

3.6.5 Computational efficiency

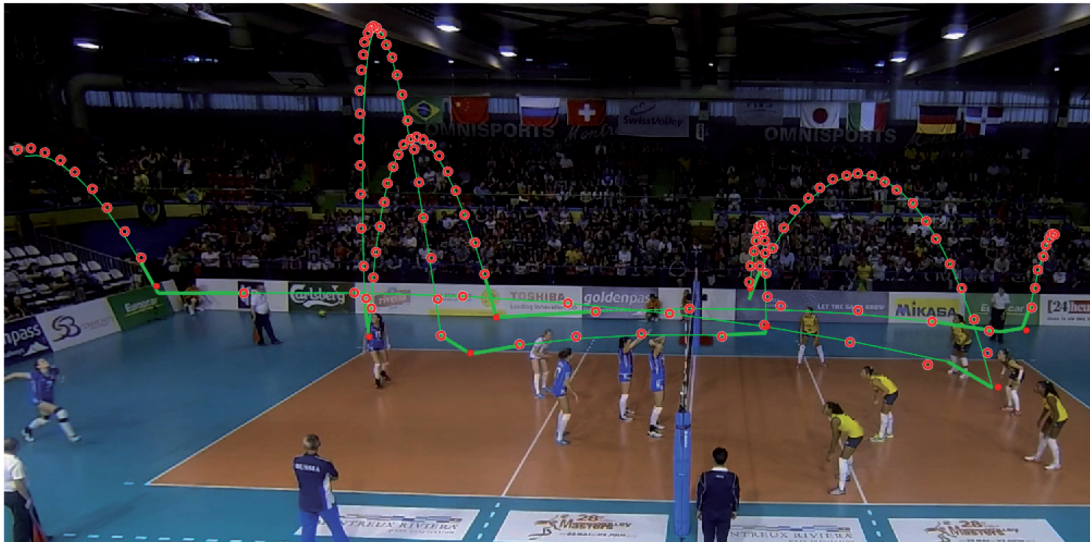
We provide a partial evaluation in Table 3.4. Running on the volleyball and basketball sequences is faster than on the soccer one, because the ball graph of the soccer sequence is larger than the others by an order of magnitude due to the spurious detections and higher edge density. More specifically, D_p , the vicinity in which the ball can be possessed by players, is higher for soccer, as reported in the appendix. This results in more states for the soccer graph.

Dataset	100 frames	250 frames	500 frames
Volley-1,2	0.2/0.4/1	0.5/1.2/2	3/5.3/15
Basket-1,2	0.1/0.3/0.9	5/9.1/12	8/12/38
Basket-APIDIS	0.2/0.4/0.6	1/2.2/3	7/13.1/25
Soccer-ISSIA	3/4/5	16/38/60	794/1072/1350

Table 3.4 – Min/Average/Max processing time (measured in seconds) on batches of different lengths on a 2.5Hz Intel Core i7 processor. Results were computed on non-overlapping intervals covering all test data.

3.7 Conclusion

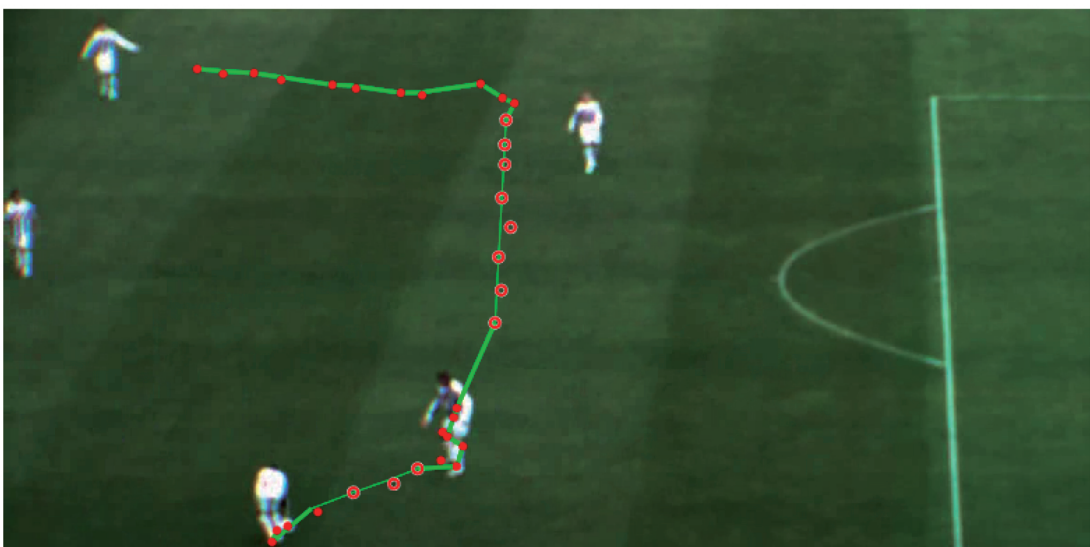
We have introduced an approach to ball tracking and state estimation in team sports. It uses Mixed Integer Program that allows to account for second order motion of the ball, interaction of the ball and the players, and different states that the ball can be in, while ensuring globally optimal solution. We showed our approach on several real-world sequences from multiple team sports. In future, we would like to extend this approach to more complex tasks of activity recognition and event detection. For this purpose, we can treat events as another kind of objects that can be tracked through time, and use interactions between events and other objects to define their state.



Volley-1



Basket-1



Soccer-ISSIA

Figure 3.8 – Visualisation of results on 3 10-second sequences from different sports. Circles indicate true ball location: empty circles correspond to free motion, filled circles indicate ball *in_possession*. Line indicates predicted ball locations: thick when predicted state is *in_possession*, thin otherwise. Best viewed in color.

4 Non-Markovian globally consistent multi-object tracking

Abstract

Many state-of-the-art approaches to multi-object tracking rely on detecting them in each frame independently, grouping detections into short but reliable trajectory segments, and then further grouping them into full trajectories. This grouping typically relies on imposing local smoothness constraints but almost never on enforcing more global ones on the trajectories.

In this chapter, we propose a non-Markovian approach to imposing global consistency by using behavioral patterns to guide the tracking algorithm. When used in conjunction with state-of-the-art tracking algorithms, this further increases their already good performance on multiple challenging datasets. We show significant improvements both in supervised settings where ground truth is available and behavioral patterns can be learned from it, and in completely unsupervised settings.

4.1 Introduction

Multiple Object Tracking (MOT) has a long tradition for applications such as radar tracking [23]. These early approaches gradually made their way into vision community for object tracking purposes. They initially relied on Gating, Kalman Filtering [22, 129, 74, 181, 119] and later on Particle Filtering [59, 159, 134, 88, 184, 124, 26]. Because of their recursive nature, when used to track objects in crowded scenes, they are prone to identity switches and trajectory fragmentations, which are difficult to recover from.

With the recent improvements of object detectors [43, 11], the Tracking-by-Detection paradigm [4] has now become the preferred way to address MOT. In most state-of-the-art approaches [160, 36, 125, 180], this involves detecting objects in each frame independently, grouping detections into short but reliable trajectory segments, or tracklets, and then further grouping those into full trajectories.

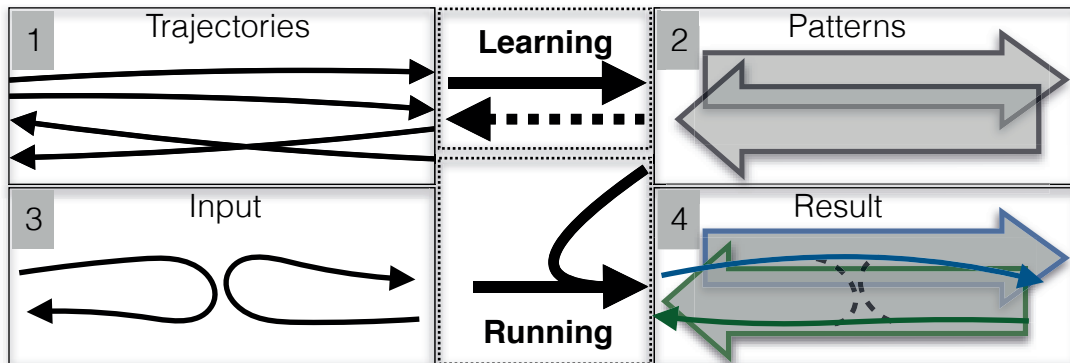


Figure 4.1 – **Top two boxes.** At training time, our procedure alternates between learning global patterns (on the right) from trajectories (on the left) and improving the trajectories on the basis of these patterns. These two actions are shown by a solid and dashed line between the two top boxes. When the initial trajectories come from annotated ground truth data, the patterns are simply learned without further iterations. **Bottom two boxes.** At run time, the learned patterns (from the top right box) are used to improve trajectories produced by state-of-the-art algorithms (from the bottom left box). Obtained results are shown in the bottom right box.

While effective, existing tracklet-based approaches tend to only impose local smoothness constraints on the trajectories. These are Markovian in nature as opposed to being global ones that stem from behavioral patterns. For example, a person entering a building via a particular door can be expected to head to a specific set of rooms. Similarly, a pedestrian emerging on the street from a shop will often turn left or right to follow the sidewalk. Such patterns are of course not absolute because people sometimes do the unexpected but they should nevertheless inform the tracking algorithms. We know of no existing technique that imposes this kind of global non-Markovian constraints in a globally optimal fashion.

Our first contribution is an energy function that relates behavioral patterns to trajectories assigned to them. We use it to infer global patterns and to guide a multi-target tracking algorithm in a non-Markovian fashion.

Our second contribution is an unsupervised training scheme. Given input trajectories from any source, it iterates between learning patterns that maximize our energy function, and improving the trajectories by linking the detections that were the part of the original ones in a potentially different way so as to maximize the same energy. When the original trajectories come from annotated ground truth data, the patterns are simply learned for them without further iterations. The top row of Fig. 4.1 depicts this process. At run-time, previously learned patterns are used to improve the trajectories produced by the original algorithm or any other, as depicted by the bottom row of Fig. 4.1. We show that this approach consistently improves performance on multiple challenging datasets by 7% and 5% on average in supervised and unsupervised fashion respectively. This is mostly attributable to the reduction in identity switches between objects following different patterns. Our code is made publicly available ¹.

¹https://github.com/maksay/ptrack_cpp

4.2 Related Work

We briefly review data association and behavioral modeling techniques and refer the interested reader to [170, 109] for more details. We also discuss the metrics we use for MOT evaluation and their sensitivity to identity switches.

4.2.1 MOT as Data Association

Finding the right trajectories linking the detections, or data association, has been formalized using various models. For real-time performance, data association often relies either on matching locally between existing tracks and new targets [48, 105, 8, 36, 127] or on filtering techniques or using model evolution approaches [132, 152]. The resulting algorithms are fast but often perform less well than batch optimization methods, which use a sequence of frames to associate the data optimally over a whole set of frames, rather than greedily in each following frame.

Batch optimization can be formulated as a shortest path problem [17, 143], network flow problem [196], generic linear programming [78], integer or quadratic programming [104, 27, 167, 148, 40, 191, 122]. A common way to reduce the computational burden is to group reliable detections into short trajectory fragments known as tracklets and then reason on these tracklets instead of individual detections [82, 158, 110, 97, 15].

However, whether or not tracklets are used, making the optimization problem tractable when looking for a global optimum limits the class of possible objective functions. They are usually restricted to functions that can be defined on edges or edge pairs in a graph whose nodes are individual detections or tracklets. In other words, such objective functions can be used only to impose relatively local constraints. To impose global constraints, the objective functions have to involve multiple objects and long time spans. They are optimized using gradient descent with exploratory jumps [128], inference with a dynamic graphical model [36], or iterative groupings of shorter tracklets into longer trajectories [96, 56, 6]. However, this comes at the cost of losing any guarantee of global optimality.

By contrast, our approach is designed for batch optimization and finding the global optimum, while using an objective function that is rich enough to express the relation between global trajectories and non-linear motion patterns. The method of [37] advocates the same philosophy but for the very different activity recognition task.

4.2.2 Using Behavioral Models

A number of works incorporate human behavioral models into tracking algorithms to increase their reliability. For example, the approaches of [141, 1] model collision avoidance behavior to improve tracking, the one of [187] uses behavioral model to predict near future target locations, and the one of [144] encodes local velocities into the affinity matrix of tracklets. These approaches

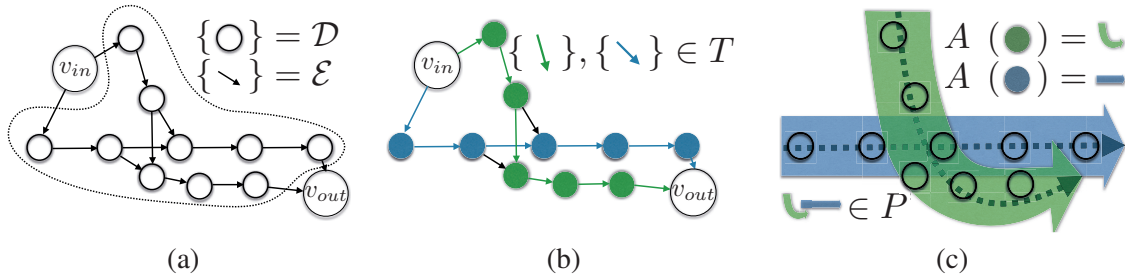


Figure 4.2 – (a) Given a set of high-confidence detections \mathcal{D} , and a set of allowed transitions \mathcal{E} , we seek to find: (b) trajectories of the objects, represented by transitions from T ; (c) a set of behavioural patterns P , which define where objects behaving in a particular way are likely to be found; an assignment A of each individual detection to a pattern, specifying which pattern did the object in this detection follow.

boost the performance but only account for very local interactions, instead of global behaviors that influence the *whole* trajectory.

Many approaches to inferring various forms of global patterns have been proposed over the years [146, 84, 120, 142, 192, 70, 174, 28, 91, 108, 57, ?]. However, the approaches of [16], [2], [94], and [10] are the only ones we know of that attempt to use these global patterns to guide the tracking. The method of [16] is predicated on the idea that behavioral maps describing a distribution over possible individual movements can be learned and plugged into the tracking algorithm to improve it. However, even though the maps are global, they are only used to constrain the motion locally without enforcing behavioral consistency over the whole trajectory. In [10], an E-M-based algorithm is used to model the scene as a Gaussian mixture that represents the expected size and speed of an object at any given location. While the model can detect global motion anomalies and improve object detection, the motion pattern information is not used to improve the tracking explicitly. In [94], modeling the optical flow helps to detect anomalies but only when the crowd is dense enough. In [2], global behavioral patterns are learned as vector fields on the floor. However, when used for tracking in high-density crowds, they are converted to local Markovian transition probabilities, thereby losing their global nature.

Vehicle motion is more structured than the human kind and behavioral models often take into account speed limits or states of the traffic lights [200, 85, 62, 77, 163]. Nevertheless, they retain enough similarities with human motion that we can represent patterns in the same way for both.

4.2.3 Quantifying Identity Switches

In this chapter, we aim for globally consistent tracking by preventing identity switches along reconstructed trajectories, for example when trajectories of different objects are merged into one or when a single trajectory is fragmented into many. We therefore need an appropriate metric to gauge the performance of our algorithms. Here, using Fig. 4.3 we provide a concrete example

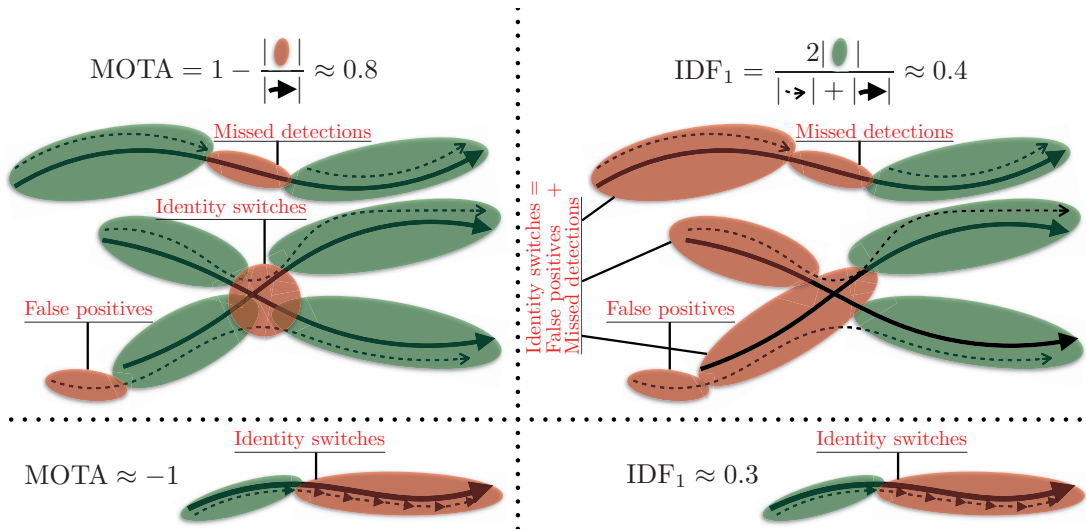


Figure 4.3 – Effect of identity switches on the tracking metrics. The thick lines represent ground-truth trajectories and the thin dotted ones recovered trajectories. The trajectory fragments that count positively are shown in green and those that count negatively in red. The formulas at the top of the figure depict graphically how the **MOTA** and **IDF₁** scores are computed. **Top:** Three ground-truth trajectories, with the bottom two crossing in the middle. The four recovered trajectories feature an identity switch where the two real trajectories intersect, missed detections resulting in a fragmented trajectory and therefore another identity switch at the top, and false detections at the bottom left. When using **MOTA**, the identity switches incur a penalty but only very locally, resulting in a relatively high score. By contrast, **IDF₁** penalizes the recovered trajectories over the *whole* trajectory fragment assigned to the wrong identity, resulting in a much lower score. **Bottom:** The last two thirds of the recovered trajectory are fragmented into individual detections that are not linked. **MOTA** counts each one as an identity switch, resulting in a negative score, while **IDF₁** reports a more intuitive value of 0.3.

of why the **IDF₁** metric is a better tool for our task than the set of CLEAR MOT metrics. We have previously introduced both in Sec. 2.5. In Section 4.6.4, we report results both in terms of **MOTA** and **IDF₁**, to highlight the drop in identity switches our method brings about.

4.3 Formulation

In this section, we formalize the problem of discovering and using behavioral patterns to impose global constraints on a multi-object tracking algorithm. In the following sections we will use it to estimate trajectories given the patterns and to discover the patterns given ground-truth trajectories.

4.3.1 Detection Graph

Given a set of high-confidence detections $\mathcal{D} = \{1, \dots, L\}$ in consecutive images of a video sequence, let $\mathcal{V} = \mathcal{D} \cup \{v_{in}, v_{out}\}$, where v_{in} and v_{out} denote possible trajectory start and end points and each node $v \in \mathcal{D}$ is associated with a set of features that encode location, appearance, or other important properties of a detection. Let $\mathcal{E} \subset \mathcal{V}^2$ be the set of possible transitions between the detections. $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can then be treated as a *detection graph* of which the desired trajectories are subgraphs. As depicted by Fig. 4.2, let

- $\mathcal{P} \subset \mathcal{E}$ be a set of edges defining objects' trajectories.
- P be a set of K patterns, each defining an area where objects behaving in a specific way are likely to be found, plus an empty pattern \emptyset used to describe unusual behaviors. Formally speaking, patterns are functions that associate to a trajectory made of an arbitrary number of edges a score that denotes how likely it is to correspond to that specific pattern, as discussed in Section 4.3.3. In our particular implementation, patterns will be defined by a centerline and width, and we will assume that people following the pattern should go along the centerline within the distance defined by the width - Fig. 4.4 should provide some intuition before we define it more formally in Sec. 4.3.3.
- A be a set of assignments of individual detections in \mathcal{D} into patterns, that is, a mapping $A: \mathcal{D} \rightarrow \{1, \dots, K\}$.

Each trajectory $t \in T$ must go through detections via allowable transitions, begin at v_{in} , and end at v_{out} . Here we abuse the notation $t \in T$ to express that all edges $(v_{in}, t_1), (t_1, t_2), \dots, (t_{|t|}, v_{out})$ from trajectory $t = (t_1, \dots, t_{|t|})$ belong to T . Furthermore, since we only consider high-confidence detections, each one must belong to **exactly** one trajectory. In practice, this means that potential false positives end up being assigned to the empty behavior \emptyset and can be removed as a post-processing step. Whether to do this or not is governed by a binary indicator R_e that is learned. In other words, the edges in T must be such that for each detection there is exactly one selected edge coming in and one going out, which we can write as

$$\forall j \in \mathcal{D}, \exists! i \in \mathcal{V}, k \in \mathcal{V} : (i, j) \in T \wedge (j, k) \in T. \quad (4.1)$$

$\exists!$ denotes the expression "exists and exists only one" in the expression above. Since all detections that are grouped into the same trajectory must be assigned to the same pattern, we must have

$$\forall (i, j) \in T : (i \in \mathcal{D} \wedge j \in \mathcal{D}) \Rightarrow A(i) = A(j). \quad (4.2)$$

In our implementation, each pattern $p \in P \setminus \emptyset$ is defined by a trajectory that serves as a centerline and a width, as depicted by Fig. 4.2(c) and 4.4. However, the optimization schemes we will describe in Sections 4.4.1 and 4.4.2 do not depend on this specific representation and can be replaced by any other.

4.3.2 Building the Graph

To build the graph we use trajectories produced by another algorithm, as input. We want to improve these trajectories, therefore we build a graph so that we can obtain new trajectories and recover from identity switches, fragmentations, and incorrectly merged input trajectories.

We take the set of detections along these input trajectories to be our high-confidence detections \mathcal{D} and therefore the nodes of our graph. We take the edges \mathcal{E} to be pairs of nodes that are either *i*) consecutive in the original trajectories, *ii*) within ground plane distance D_1 of each other in successive frames, *iii*) the endings and beginnings of input trajectories within distance D_2 and within D_t frames, *iv*) or whose first node is v_{in} or second node is v_{out} .

4.3.3 Objective Function

Our goal is to find the most likely trajectories formed by transitions in T^* , patterns P^* , and mapping linking one to the other A^* , given the image information and any *a priori* knowledge we have. In particular, given a set of patterns P^* , we look for the best set of trajectories that match these patterns. Conversely, given a set of known trajectories T^* , we learn a set of patterns, as discussed in Section 4.4.

To formulate these searches in terms of an optimization problem, we introduce an objective function $C(T, P, A)$ that reflects how likely it is to observe the objects moving along the trajectories defined by T , each one corresponding to a pattern from $P = \{p_1 \dots, p_K\}$ given the assignment A . Ideally, C should be the proportion of trajectories that correctly follow the assigned patterns. To compute it in practice, we take our inspiration from the **MOTA** and **IDF₁** scores described in Section 4.2.3. They are written in terms of ratios of the lengths of trajectory fragments that follow the ground truth to total trajectory lengths. We therefore take our objective function to be a similar ratio, but instead of ground truth trajectories we use patterns. More formally:

$$\begin{aligned}
 C(T, P, A) &= \frac{\sum_{t \in T} M(t, p_{A(t_1)})}{\sum_{t \in T} N(t, p_{A(t_1)})}, & (4.3) \\
 N(t, p) &= n(v_{in}, t_1, p) + n(t_{|t|}, v_{out}, p) + \sum_{1 \leq j \leq |t|-1} n(t_j, t_{j+1}, p), \\
 M(t, p) &= m(v_{in}, t_1, p) + m(t_{|t|}, v_{out}, p) + \sum_{1 \leq j \leq |t|-1} m(t_j, t_{j+1}, p),
 \end{aligned}$$

where $n(i, j, p)$ is the sum of the total length of edge (i, j) and of the length of the corresponding pattern centerline, while $m(i, j, p)$ is the sum of lengths of aligned parts of the pattern and the edge. Fig. 4.4 illustrates this computation and we give the mathematical definitions of m and n in the appendix.

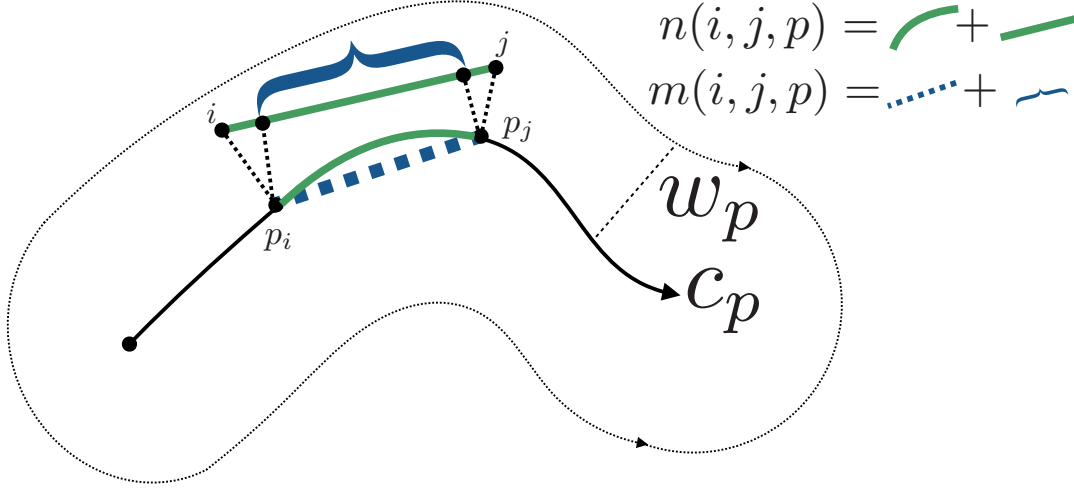


Figure 4.4 – For a pattern p defined by centerline c_p , shown as a thick black line, with width w_p , and an edge (i, j) , we compute functions $n(i, j, p)$ and $m(i, j, p)$ introduced in Section 4.3.3 and shown in green and blue, respectively, as follows: $n(i, j, p)$ is the total length of the edge and the corresponding length of the pattern centerline, measured between the points p_i and p_j , which are the points on the centerline closest to i and j . If both i and j are within the pattern width w_p from the centerline, we take $m(i, j, p)$ to be the sum of two terms: the length in the pattern along the edge, that is, the distance between p_i and p_j , plus the length in the edge along the pattern, that is, the length of the projection of (p_i, p_j) onto the line connecting i and j . Otherwise $m(i, j, p) = 0$ to penalize the deviation from the pattern.

As a result, $N(t, p)$ is the sum of the lengths of trajectory and assigned pattern while $M(t, p)$ measures the length of parts of trajectory and pattern that are aligned with each other. Note that the definition of Eq. (4.3) is very close to that of the metric \mathbf{IDF}_1 introduced in Sec. 4.2.3. It is largest when each person follows a single pattern for as long as possible. This penalizes identity switches because the trajectories that are erroneously merged, fragmented, or jump between objects are unlikely to follow any specific pattern.

Conceptually, we define functions m and n so that, for any trajectory t and pattern p , the value of $n(t, p) = n(v_{in}, t_1) + n(t_{|t|}, v_{out}) + \sum_{j \in \{1, \dots, |t|-1\}} n(t_j, t_{j+1})$ approximates the sum of the total lengths of both pattern and trajectory, while $m(t, p) = m(v_{in}, t_1) + m(t_{|t|}, v_{out}) + \sum_{j \in \{1, \dots, |t|-1\}} m(t_j, t_{j+1})$ approximates the overlap length between pattern and trajectory, as illustrated by Fig. 4.4.

The key properties of the defined function are: *i*) It scores the whole trajectory and even set of trajectories, rather than each independent transition, freeing us from a Markovian assumption of people movement; *ii*) It closely resembles the metrics such as \mathbf{MOTA} and \mathbf{IDF}_1 used to evaluate the quality of people tracking, which are also fractions with the number of matched objects in numerator and total number of objects in denominator; *iii*) As we will see in Sec. 4.4, it can be optimized efficiently. We define m and n more precisely in the appendix.

In Eq. (4.3), we did not explicitly account for the fact that the first vertex i of some edges can be

the special entrance vertex, which is not assigned to any behavior. When this happens we simply use the pattern assigned to the second vertex j . From now on, we will replace $A(i)$ by $A(i, j)$ to denote this behavior. We also adapt the definitions of m and n accordingly to properly handle those special edges.

4.4 Computing Trajectories and Patterns

In this section, we describe how we use the objective function C of Eq. (4.3) to compute trajectories given patterns and patterns given trajectories. The resulting procedures will be the building blocks of our complete MOT algorithm, as described in Section 4.5.

4.4.1 Trajectories

Let us assume that we are given a precomputed set of patterns P^* , then we look for trajectories and corresponding assignment as

$$T^*, A^* = \underset{T, A}{\operatorname{argmax}} C(T, P^*, A). \quad (4.4)$$

To solve this problem, we treat the motion of objects through the detection graph \mathcal{G} introduced in Section 4.3.1 as a flow. Let $o_{ij}^p \in \{0, 1\}$ be the number of objects transitioning from node i to j in a trajectory T assigned to pattern $p \in P^*$. It relates to P^* and T according to:

$$o_{ij}^p = \mathbb{I}(((i, j) \in T) \wedge (P_{A(i, j)}^* = p)). \quad (4.5)$$

Using these new binary variables, we reformulate constraints (4.1) and (4.2) as

$$\begin{aligned} \forall i \in \mathcal{D} \cup \mathcal{O} \quad \sum_{(i, j) \in \mathcal{E}, p \in P^*} o_{ij}^p &= 1, \\ \forall j \in \mathcal{D}, p \in P^* \quad \sum_{(i, j) \in \mathcal{E}} o_{ij}^p &= \sum_{(j, k) \in \mathcal{E}} o_{jk}^p. \end{aligned} \quad (4.6)$$

This lets us rewrite our cost function as

$$C(T, P^*, A) = \frac{\sum_{(i, j) \in T, p \in P^*} m(i, j, p) o_{ij}^p}{\sum_{(i, j) \in T, p \in P^*} n(i, j, p) o_{ij}^p}, \quad (4.7)$$

which we maximize with respect to the flow variables o_{ij}^p subject to the two constraints of Eq. (4.6). This is an integer-fractional program, which could be transformed into a Linear Program [30]. However, solving it would produce non-integer values that would need to be rounded. To avoid this we propose a scheme based on the following observation: Maximizing $\frac{a(x)}{b(x)}$ with respect to x when $b(x)$ is always positive can be achieved by finding the largest α such that an x satisfying $a(x) - \alpha b(x) \geq 0$ can be found. Furthermore, α can be found by binary search.

Chapter 4. Non-Markovian globally consistent multi-object tracking

We therefore take a to be the numerator of Eq. (4.7), b its denominator, and x the vector of o_{ij}^p variables. In practice, given a specific value of α , we do this by running an Integer Linear Program solver [137] until it finds a feasible solution. When α reaches its maximum possible value, that feasible solution is also the optimal one.

During binary search, we fix a particular value of α , and check whether the problem constrained by (4.6) and the following has a feasible point:

$$\sum_{(i,j) \in T, p \in P^*} (m(i,j,p) - \alpha n(i,j,p)) o_{ij}^p \geq 0 \quad (4.8)$$

If a feasible point exists, we pick a value of α to be the lower bound of the best α , for which the problem is feasible, otherwise we pick it as an upper bound. We start with the upper bound of 1 and lower bound of 0, and pick α as an average between the upper and the lower bound (dichotomy). We repeat this process 10 times, allowing us to find the correct value of α with the margin of 2^{-10} , and therefore also finding the values of T and A resulting in the optimal cost function value with the same margin.

4.4.2 Patterns

In the previous section, we assumed the patterns known and used them to compute trajectories. Here, we reverse the roles. Let us assume we are given a set of trajectories T^* . We learn the patterns and corresponding assignments as

$$\begin{aligned} P^*, A^* &= \underset{P, A}{\operatorname{argmax}} C(T^*, P, A), \\ \text{subject to} \quad & P \subset \mathcal{P}, |P| \leq \alpha_p, \sum_{p \in P} M(p) \leq \alpha_c, \end{aligned} \quad (4.9)$$

where α_c, α_p are thresholds and $M: P \rightarrow \mathbb{R}^+$. The purpose of the additional constraints is to limit both the number of patterns being used by α_p and their spatial extent by α_c , to prevent over-fitting. In our implementation, we take $M(p) = l_p w_p$, where l_p is the length of the pattern centerline and w_p is its width. \mathcal{P} is a set of all admissible patterns, which we construct by combining all possible ground-truth trajectories as centerlines with each width from a predefined set of possible pattern widths.

To solve the problem of Eq. (4.9), we look for an assignment between our known ground truth trajectories T^* and all possible patterns \mathcal{P} and retain only patterns associated to at least one trajectory. To this end, we introduce auxiliary variables a_{tp} describing the assignment $A^*: T^* \rightarrow \mathcal{P}$, and variables b_p denoting if at least one trajectory is matched to pattern p .

Formally, this can be written as

$$\begin{aligned}
 a_{tp} &\in \{0, 1\}, \forall t \in T^*, p \in \mathcal{P}, \\
 b_p &\in \{0, 1\}, \forall p \in \mathcal{P}, \\
 \sum_{p \in \mathcal{P}} a_{tp} &= 1, \forall t \in T^*, \\
 a_{tp} &\leq b_p, \forall t \in T^*, p \in \mathcal{P}.
 \end{aligned} \tag{4.10}$$

Given that C is defined as the fraction from Eq. (4.3), we use an optimization scheme similar to the one described at the end of Sec. 4.4.1, where we perform binary search to find the optimal value of α such that there exists a feasible solution for constraints of Eq. (4.10) as well as:

$$\begin{aligned}
 \sum_{t \in T^*} \sum_{p \in \mathcal{P}} (m(t, p) - \alpha n(t, p)) a_{tp} &\geq 0, \\
 \sum_{p \in \mathcal{P}} b_p &\leq \alpha_p, \quad \sum_{p \in \mathcal{P}} b_p M(p) \leq \alpha_c.
 \end{aligned} \tag{4.11}$$

In practice, we do five iterations of binary search, and we obtain the right value of α with precision of 2^{-5} . To create a set of all possible patterns \mathcal{P} we combine the set of all possible trajectories in the current batch (taking only those that start after the beginning of the batch and end before the end of the batch to make sure they represent *full* patterns of movement) with a set of possible lengths.

4.5 Non-Markovian Multiple Object Tracking

Given that we can learn patterns from a set trajectories, we can now enforce long-range behavioral patterns when linking a set of detections. This is in contrast to approaches enforcing local smoothness constraints, that is, Markovian.

If annotated ground-truth trajectories T^* are available, we use them to learn the patterns as described in Sec. 4.4.2. Then, at test time, we use the linking procedure of Sec. 4.4.1.

If no such training data is available, we can run an E-M-style procedure, very similar to the Baum-Welch algorithm [76]: We start from a set of trajectories computed using a standard algorithm, use them to compute a set of patterns, then use the set of patterns to improve trajectories, and iterate. In practice, this yields results that are very similar to the supervised case in terms of accuracy but much slower because we have to run through many iterations. This alternate optimization is the key to making the computation tractable and making its components replaceable.

More specifically, each iteration of our unsupervised approach involves *i*) finding a set of patterns P^i given a set of trajectories T^{i-1} , *ii*) finding a set of trajectories T^i given a set of patterns P^i , as described in Sec. 4.4.2 and 4.4.1.

In practice, for a fixed maximum number of patterns α_c , this scheme converges after few iterations. Since the optimal α_c is unknown *a priori*, we start with a small α_c , perform 5 iterations, increase α_c , and repeat until we reach a predefined maximum number of patterns. To select the best trajectories without reference to ground truth, we define

$$\widetilde{\mathbf{IDF}}_1(T^i) = \frac{1}{2}(C(T_1^i, P_2^i, A_{T_1^i \rightarrow P_2^i}) + C(T_2^i, P_1^i, A_{T_2^i \rightarrow P_1^i})),$$

where T_1^i and T_2^i are time-disjoint subsets of T^i , P_1^i and P_2^i are patterns learned from T_1^i and T_2^i . $A_{T_1^i \rightarrow P_2^i}$ and $A_{T_2^i \rightarrow P_1^i}$ are such assignments of trajectories to the patterns learned on another subset that maximize $\widetilde{\mathbf{IDF}}_1(T^i)$.

In effect, $\widetilde{\mathbf{IDF}}_1$ is a valid proxy for \mathbf{IDF}_1 due to the many similarities between our cost function and \mathbf{IDF}_1 outlined in Sec. 4.3.3. In the end, we select the trajectories that maximize $\widetilde{\mathbf{IDF}}_1$. Using such cross-validation to pick the best solution in E-M models is justified in [68].

4.6 Evaluation

In this section, we demonstrate the effectiveness of our approach on several datasets, using both simple and sophisticated approaches to produce the initial trajectories, which we then improve as discussed in Section 4.5.

In the remainder of this section, we first describe the datasets and the tracking algorithms we rely on to build the initial graphs. We then discuss the experimental protocol. Finally, we present our experimental results.

4.6.1 Datasets

We use **DukeMTMC** [149], **Towncentre** [101, 12], **Station** [201], **MOT16** [125], **ETH** and **Hotel** [140] datasets for people tracking. We use a part of **Rene** [81] for vehicle-tracking, featuring 30 annotated seconds with 27 trajectories. Additional results are reported on **WILDTRACK** dataset [31] in the appendix.

Textual description of the datasets is as follows:

DukeMTMC. A dataset [149] with 8 cameras recording movements of people on various places of Duke university campus at 60fps, containing more than an hour of recordings.

Towncentre. A sequence from the 2DMOT2015 benchmark [101]: a lively street where people walk in different directions.

ETH and **Hotel.** Sequences from the BIWI Walking Pedestrians dataset [140] that were originally used to model social behavior. In these datasets, using image and appearance information for tracking is difficult, due to recordings with an almost vertical viewing angle and low visibility in

the **ETH**.

Station. A one hour-long recording of Grand Central station in New York with several thousands of annotated pedestrian tracks [201]. It was originally used for trajectory prediction in crowds.

MOT16. Sequences from the MOT Challenge 2016 [125]. We used MOT16-01 to evaluate the supervised approach because it features training and testing data recorded at the same place. By contrast, MOT16-08 does not and we used it to evaluate the unsupervised approach. Unfortunately, the other sequences are unsuitable for our current implementation because they involve either a moving camera, meaning there is no fixed scene to learn the patterns from, or only very few trajectories traversing the scene for training purposes.

Rene. A five-minute long sequence of traffic at a street junction [81]. Since only 30 seconds of it are annotated, we ran only the unsupervised approach on the whole sequence and used the annotated frames for evaluation purposes.

WILDTRACK [31]. It contains a sequence recorded by 7 cameras with overlapping fields of view and features a denser crowd than the others. The training and testing sequences are relatively short, but allows to use our method to compare to **KSP** baseline. Results are reported in Tab. A.8 in appendix.

Dataset statistics are shown in Table 2.2. These datasets share several characteristics that make them well suited to test our approach in challenging conditions. First, they feature real-life behaviors as opposed to random and unrealistic motions acquired in lab settings. Second, many of them feature frame rate below 5 frames per second, which is representative of outdoor surveillance setups but makes tracking more difficult.

4.6.2 Baselines

As discussed in Section 4.3.2, we use as input to our system trajectories produced by recent MOT algorithms. In Section 4.6.4, we will show that imposing our pattern constraints systematically results in an improvement over the numerous baselines listed below.

On various datasets we compare to the following approaches: two highest-ranking approaches of **MOT15** [101] with publicly available implementation at the time of writing, namely **MDP** [180] and **SORT** [20]; ECCV 2016 MOT Challenge winner **DM** [160, 161]; various other **MOT15** top scoring methods [36, 157, 168, 87, 185, 89, 169, 189] to which we will refer by the name that appears in the official scoreboard [101]. Finally, we use **RNN** [127] and **KSP** [17] as simple baselines that do not use appearance information, and compare with **BIPCC** [149] as a baseline provided for **DukeMTMC** dataset.

Textual description of the methods is as follows:

Chapter 4. Non-Markovian globally consistent multi-object tracking

MDP [180] formulates MOT as learning Markov Decision Process (MDP) policy and relies on reinforcement learning to do so. At the time of writing, this was the highest-ranking approach on the 2DMOT2015 [101] benchmark with a publicly available implementation.

SORT [20] is a real-time Kalman filter-based MOT approach. At the time of writing, this was the second highest-ranking approach on 2DMOT2015 benchmark with a publicly available implementation.

RNN [127] uses recurrent neural networks to predict the motion of people and perform MOT in real time. It does not require any appearance information, but only the bounding boxes coordinates. In our experiments, it outperformed all other methods that do not use appearance information.

KSP [17] is a simple approach to MOT that formulates the MOT problem as finding K Shortest Paths in spatio-temporal graph, without using appearance information.

DM [160, 161] decomposes the tracking graph into subgraphs and relies on strong matching models. It won the ECCV 2016 Multiple Object Tracking challenge.

BIPCC [149] solves binary integer problem of optimally grouping observations into clusters of detections of similar appearances, and delivers results with moderate recall, but very high precision with few identity switches.

Top scoring methods from the **MOT15** benchmark on the **Towncentre** dataset rely on a people detector that is not always publicly available. We therefore used their output to build the detection graph, and report their results only on **Towncentre**. For all others, the available code accepts a set of detections as input. To compute them, we used the publicly available POM algorithm of [52] to produce probabilities of presence in various ground locations and we kept those with probability greater than 0.5. This proved effective on all our datasets. For comparison purposes, we also tried using SVMs trained on HOG features [39] and deformable part models [50]. While their performance was roughly similarly to that of POM on **Towncentre**, it was much worse when the people are far away or seen from above. For cars, we used background subtraction followed by blob detection.

4.6.3 Experimental Protocol

The data is split one minute long validation and test sequences, and the rest is used for training. Results are averaged for all test intervals which we select in a leave-one-out fashion. We follow

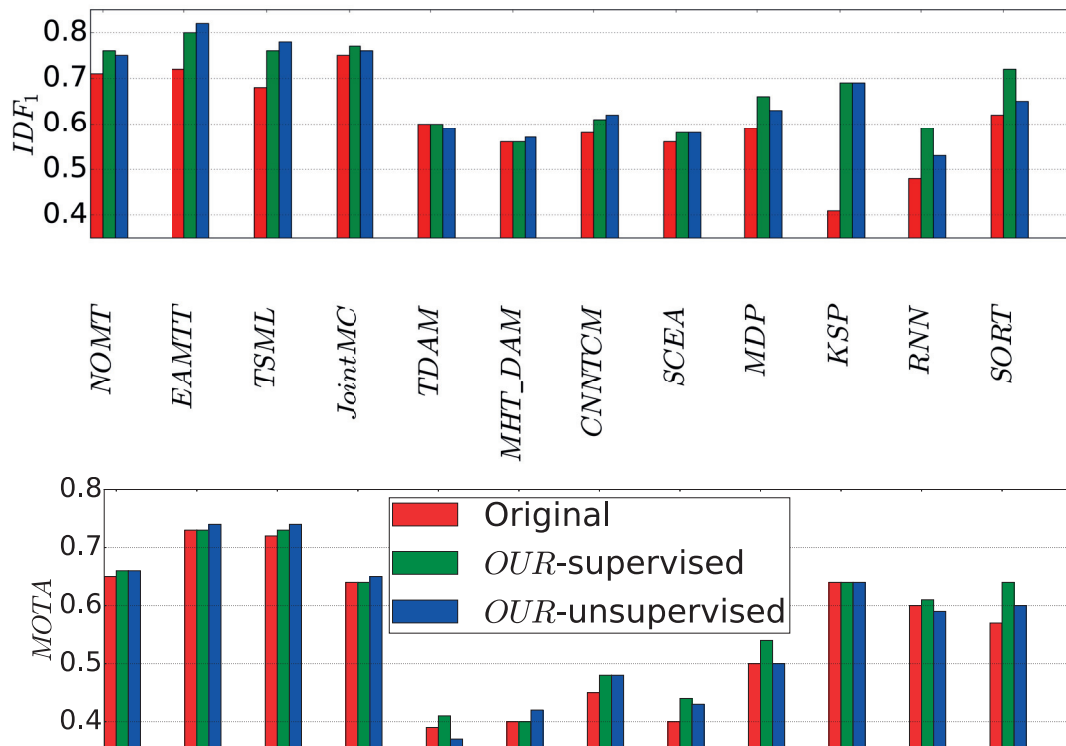


Figure 4.5 – IDF_1 and $MOTA$ scores for various methods on the **Towncentre** dataset. Our approach almost always improves IDF_1 . We provide the actual numbers in appendix.

this protocol for most of the sequences since the shortest sequence is only 3 minutes long. Two exceptions are **DukeMTMC**, in which we trained and validated using provided training data, and evaluated on the whole test sets of 10 and 25 minutes in batch mode to show the ability of our approach to handle long sequences, and **Rene**, in which we had 30 seconds of annotated data. Training data trajectories were used to learn the patterns of Section 4.4.2. Validation data trajectories were used to optimize values of the hyperparameters D_1 , D_2 , D_t , R_e , α_c , α_p introduced in Sections 4.4.1, 4.4.2, using coordinate ascent.

For the sake of fairness, we trained **MDP** and **RNN**, the trainable baselines of Section 4.6.2, similarly and using the same data. However, for **RNN** we obtained better results using the provided model, pre-trained on the 2DMOT2015 training data, and we report these results.

Since for some approaches we only had results in the form of bounding boxes and had to estimate the ground plane location based on that, this resulted in large errors further away from the camera. For this reason, we evaluated $MOTA$ and IDF_1 assuming that a match happens when the reported location is at most at 3 meters from the ground truth location. We also provide results for the traditional 1 meter distance in the appendix and they are similar in terms of method ordering. For

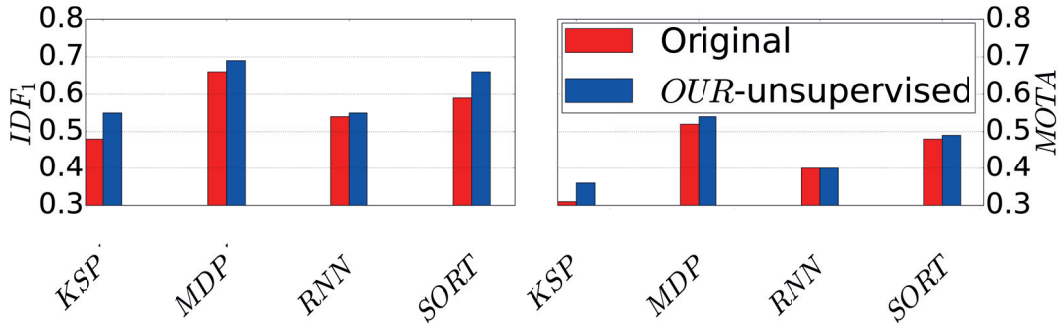


Figure 4.6 – IDF_1 (left) and $MOTA$ (right) scores on the **Rene** dataset.

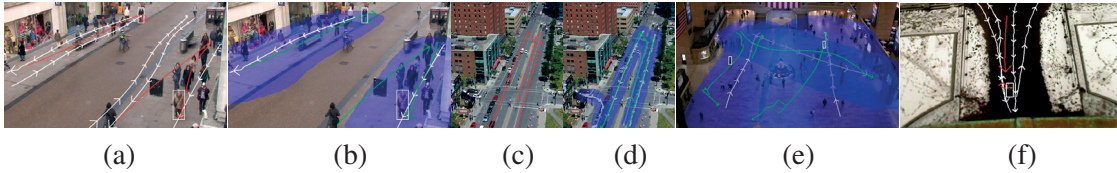


Figure 4.7 – Examples of learned patterns, denoted by their centerline in white, with some erroneous trajectories found by various baselines in red. White bounding boxes for people following the trajectories are shown. Improved trajectories found by our approach in green. Area in blue shown pattern widths, helping understand to which patterns trajectories are assigned. **(a) Towncentre** dataset, **EAMTT [157]** merges trajectories going in opposite directions, but **(b)** correct pattern assignment helps to fix that; **(c)** Using only affinity information, **KSP** is prone to multiple identity switches of cars going in different directions; **(d)** Our approach correctly recovers all trajectories, including one with the turn; **(e)** On **Station** dataset our approach recovers mostly correct trajectories, but trajectories of two different people in the lower left corner going in the same general direction are merged; **(f) ETH** dataset, due to low visibility using flow and feature point tracking is hard, and **MDP** fragments a single trajectory into two, but our approach fixes that (not shown). Best viewed in color.

the **Station** and **Rene** datasets, we did not have the information about the true size of the floor area, as we only estimated the homography between the image and ground plane. That is why we used a distance that is 10% of the size of the tracking area.

For the same reason, for all datasets except **Station** and **Rene**, our set of possible widths of patterns is $\{0.5, 1, 3, 5, 7, 9, 11, 13, 15, 17\}$, while for the **Station** and **Rene** datasets we use $\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ of the tracking area.

4.6.4 Results

IDF₁ and MOTA. Here we report summarized results for multiple approaches and datasets. Comparison on **DukeMTMC** and **MOT16** is also available on MOTChallenge benchmark [101].



Figure 4.8 – Example of unsupervised optimization. **(a)** Four people are tracked using **KSP**. Trajectories are shown as solid black lines, bounding boxes are white. Tracks feature several identity switches. **(b)** First, alternating scheme finds a single pattern, in white, that explains as many trajectories as possible - that is the leftmost trajectory. Given this pattern, next step is the tracking. Trajectories in blue are the ones assigned to this pattern, trajectories in red are assigned to no pattern. One identity switch is fixed. **(c)** After several iterations, we look for the best two patterns. Rightmost trajectory is picked as the second pattern. Fitting trajectories to the best two patterns allows to fix the remaining fragmented trajectory. Trajectories assigned to the second pattern in green.

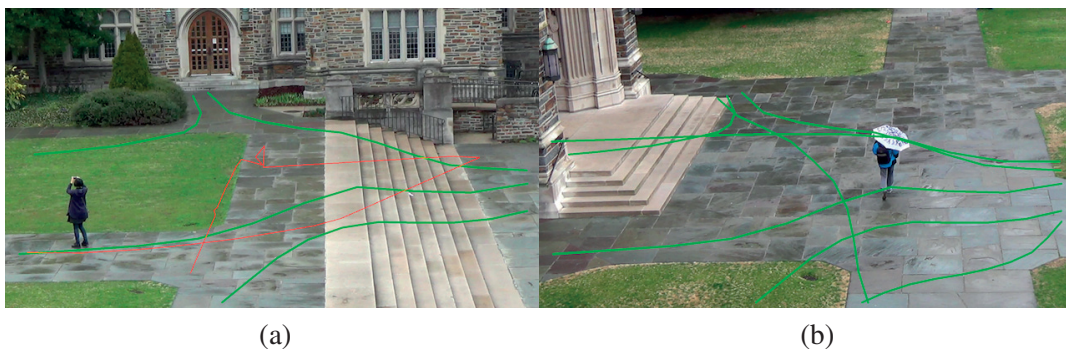


Figure 4.9 – Examples of learned patterns on **DukeMTMC** dataset shown in green. **(a)** Some sequences contain highly non-linear patterns with turns, and our method successfully recovers them. An example of trajectory assigned to no pattern is shown in red. **(b)** A sequence with high number of patterns - each pattern goes in both directions. In such cases our model can incorrectly split an unexpected trajectory into two parts, each of which follows one pattern.

For **DukeMTMC** dataset, our supervised approach achieves +1.1% IDF_1 on all Easy sequences combined, with improvements on 7 out of 8 sequences up to 3.7%, and one drop of 0.5%. It achieves +0.5% IDF_1 on all Hard sequences combined, with improvements on 7 out of 8 sequences up to 8%, and one drop of 0.2%. The unsupervised approach achieves +0.9% IDF_1 on all "trainval-mini" sequences combined, with improvements on 7 out of 8 sequences up to 4.2%, and one drop of 0.1%. Improvements are shown with respect to [149]. Examples of learned

Chapter 4. Non-Markovian globally consistent multi-object tracking

Approach	$\Delta \mathbf{IDF}_1^s$	$\Delta \mathbf{IDF}_1^u$	$\Delta \mathbf{MOTA}^s$	$\Delta \mathbf{MOTA}^u$
KSP	0.16	0.15	-0.01	-0.01
MDP	0.05	0.02	0.03	-0.01
RNN	0.04	0.03	0.00	-0.02
SORT	0.04	0.02	0.06	0.00

Table 4.1 – \mathbf{IDF}_1 and \mathbf{MOTA} improvement, delivered by our approach, averaged over all datasets. The 2nd and 4th columns correspond to the supervised case, the 3rd and 5th to the unsupervised one. Since \mathbf{IDF}_1 scores range from 0 to 1, these represent significant improvements.

patterns are shown in Fig. 4.9.

Fig. 4.5 shows results of methods with published results on the **Towncentre** sequence. For the 4 methods for which there is a publicly available implementation—**KSP**, **MDP**, **RNN**, **SORT**—we computed trajectories on various datasets and evaluated the improvement brought by our approach. These results are reported in Table 4.1 for people and Fig. 4.6 for cars.

As shown in Fig. 4.5, our supervised method improves all the tracking results in \mathbf{IDF}_1 terms on **Towncentre** except one that remains unchanged. The same can be said of the unsupervised version of our method except for one that it degrades by 0.01. Recall that \mathbf{IDF}_1 ranges from 0 to 1. A 0.01 improvement is therefore equivalent to a 1% improvement and our algorithm delivers a significant performance increase. Similarly, Fig. 4.6 depicts original and improved car-tracking results on **Rene**, but only in the unsupervised case owing to the short length of the manually annotated sequence, which we needed for evaluation purposes.

In Tab. 4.1, we average improvement in people-tracking results brought by our approach for four baselines. We observe a consistent improvement in \mathbf{IDF}_1 terms in both the supervised and unsupervised cases. As could be expected, the improvement is much less clear in \mathbf{MOTA} terms because our method modifies the set of input detections minimally while \mathbf{MOTA} is more sensitive to the detection quality than to identity switches. Fig. 4.7 depicts some of the results.

Finally, we used the output of **DM** on the two **MOT16** sequences as input to the supervised and unsupervised versions of our algorithm, as discussed above. We obtained a 37% and 25% drop in identity switches, 4% and 1% drop in number of fragmented trajectories, and 0.1% and 4% increase in \mathbf{MOTA} , compared to the published results. Unfortunately, MOT’16 benchmark does not provide the \mathbf{IDF}_1 numbers which is why we don’t report them for **DM**.

Non-linear learned motion patterns. To evaluate the importance of learning generic patterns such as ours as opposed to simpler ones, or an even simpler smoothness constraint, we introduce two more baselines. In the first, we take patterns to be straight lines crossing the scene in every possible direction. This is still non-Markovian as it forces trajectories to cross the scene completely, starting at one border and going to another. In the second, we find a set of trajectories through our tracking graph that minimizes the second order difference between triplets of

consecutive locations, forcing trajectories to be locally smooth. This is Markovian in nature and does not require trajectories to cross the scene. In the first case, average improvement for all methods drops to (0.11, 0.02, 0.03, 0.02) from (0.16, 0.05, 0.04, and 0.04) as reported in Table 2 of this chapter. In the second case, we observe a steeper drop to (0.07, 0.02, 0.01, 0.00). The difference in results is largest on **Station** dataset where non-linear non-Markovian patterns prevent trajectories from being terminated in stationary crowds, and on the **Hotel** dataset where it is difficult to differentiate between trajectories that traverse the scene and that end in the middle of the scene, entering the hotel. The detailed breakdown is given in Table 4.2.

Note that while using straight line patterns frees us from the learning step, it does not deliver much of a benefit in terms of optimization speed. As shown in the example of Figure 4.10, there are only four learned patterns, but if we define straight line patterns traversing the scene, their number is not known beforehand. This results in a trade-off, where picking too few patterns gives bad tracking results, while having too many of them slows the optimization scheme because of the number of possible patterns trajectories can follow.

Method	Learned patterns (OUR)				Straight line patterns				Markovian smoothness term			
	Town	ETH	Hotel	Station	Town	ETH	Hotel	Station	Town	ETH	Hotel	Station
KSP	0.28	0.17	0.11	0.10	0.19	0.12	0.07	0.05	0.13	0.07	0.04	0.03
MDP	0.07	0.03	0.10	-0.01	0.06	0.02	0.02	-0.02	0.06	0.02	0.02	-0.02
RNN	0.11	0.03	0.00	0.00	0.10	0.02	0.00	0.00	0.05	0.01	-0.01	-0.01
SORT	0.10	0.00	0.06	0.00	0.06	0.00	0.03	0.00	0.04	0.00	-0.01	-0.03

Table 4.2 – IDF_1 improvement for each method and dataset for our method with learned patterns (**left**), for our method with patterns replaced by a pencil of lines, which still forces trajectories to start and end at the borders of the tracking area (**middle**), and for a method where transition cost is based on the local smoothness term, second order difference between coordinates of 3 consecutive detections in a trajectory (**right**). We abbreviate **Towncentre** dataset as **Town**.

Evaluation on Ground Truth Detections. For all baselines that accept a list of detections as input, and for which the code is available, we reran the same experiment using the ground truth detections instead of those computed by the POM algorithm [52] as before. This is a way to evaluate the performance of the linking procedure independently of that of the detections, and can be viewed as an evaluation of this component of our system. It reflects the theoretical maximum that can be reached by all the approaches we compare, including our own. From Table 4.3 we observe that our approach performs very well in such setting.

Computational burden. We also assessed the influence of various terms on our method’s runtime. All people tracking results reported in Figs. 4.5, 4.6 and Tab. 4.1 ran at an average speed of 0.906 fps for the supervised case on a 4 core 2.5Hz machine. The unsupervised computation is much slower, requiring hours for dataset of containing several hundred trajectories. However, this remains practical, as it can be run overnight, and once the patterns have been learned, the system

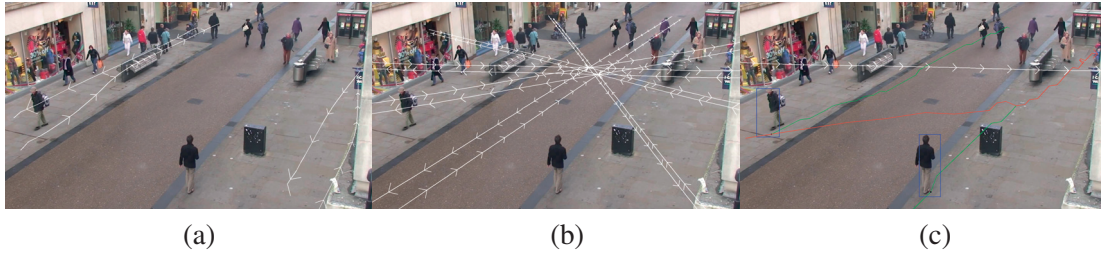


Figure 4.10 – (a) Examples of learned patterns on **Towncentre** dataset. Note that there are only two prevalent directions in which people move. Patterns shown in white. (b) Pencil of lines representing straight line patterns traversing the scene in all directions. Patterns shown in white. (c) Example of an error made when we are using straight line patterns. Two real trajectories (in green) are incorrectly merged via false detections, producing a trajectory (in red) that closely follows one of the patterns (in white). However, in reality this pattern does not exist, but we used it because we didn’t have a learning component. Note, that produced trajectory still starts at the boundary of the image and traverses the scene completely. Even without the learning procedure, our patterns force this. If we replace this non-Markovian constraint by a local smoothness term, errors are numerous, with many trajectories split in the middle.

Metric	IDF ₁					MOTA				
	MDP	RNN	SORT	KSP	OUR	MDP	RNN	SORT	KSP	OUR
Town	0.87	0.65	0.88	0.55	0.93	0.87	0.85	0.90	0.87	0.98
ETH	0.89	0.65	0.93	0.59	0.92	0.85	0.73	0.85	0.70	0.94
Hotel	0.85	0.70	0.88	0.60	0.94	0.84	0.78	0.82	0.74	0.97
Station	0.68	0.40	0.72	0.45	0.70	0.75	0.68	0.70	0.80	0.77

Table 4.3 – IDF₁ (left) and MOTA (right) evaluation results using ground detections. Best score for each dataset and metric in bold. **Towncentre** abbreviated as **Town**.

can run in the supervised mode that can be sped up limiting the density of the graph through parameter D_1 and/or decreasing the number of binary search iterations. Using 5 instead of 10 didn’t affect the IDF₁ by more than 1% in our experiments.

Dataset	Towncentre	ETH	Hotel	Station	Station
Frames	150	227	268	75	75
Trajectories	85	67	47	100	193
Patterns	7	5	4	26	26
Detections	2487	894	1019	1960	3724
Variables	70k	17k	18k	191k	450k
Time, s	26	4	4	160	>3600

Table 4.4 – Optimization problem size and run time of our approach for processing a typical one min batch from each dataset.

As shown in Fig. 4.11, the optimization time depends mostly on the number of possible tran-

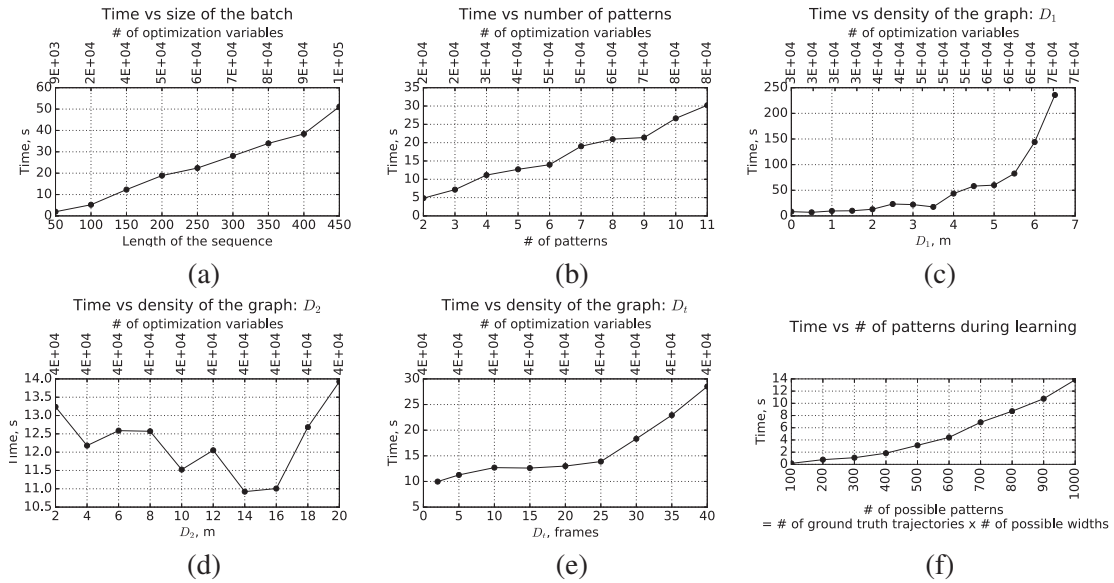


Figure 4.11 – The running time and the number of variables of the optimization for tracking are approximately:

- linear with respect to the number of frames in the batch **(a)**,
- linear with respect to the number of patterns **(b)**,
- superlinear with respect to the maximum distance at which we join the detections in the neighbouring frames D_1 , as it directly affects the density of the tracking graph **(c)**,
- almost independent from the maximum distance in space D_2 and its time D_t at which we join the endings and beginning of the input trajectories D_2 , as it has almost no effect on the density of the tracking graph **(d)**, **(e)**,
- the running time and the number of variables of the optimization for learning patterns grows quadratically with the number of input trajectories, as each of them is both a trajectory that needs to be assigned to a pattern, and a possible centerline of a pattern **(f)**

sitions between people, which is controlled by D_1 . The time for learning the patterns grows approximately quadratically. The number of variables in our optimization problem grows linearly with the length of the batch and number of patterns, and superlinearly with the number of people per frame (as the number of possible connections between people). As shown by Tab. 4.4, for not too crowded datasets without large number of patterns our approach is able to process a minute of input frames under a minute. Pattern fitting scales quadratically with the number of given ground-truth trajectories and runs in less than 10 minutes for all datasets except **Station**. All results above were computed on datasets **ETH**, **Hotel**, **Towncentre**, **Station**, since they shared same experimental protocol. For **DukeMTMC** dataset we ran evaluation for the whole length of sequence in the batch mode. For the sake of completeness, we also measured the running

Chapter 4. Non-Markovian globally consistent multi-object tracking

time. We processed around $300k * 8$ frames in a total of 7853s, ranging from 654s to 1820s per sequence. This was achieved thanks to two reasons. Firstly, since the input tracks are already good, optimal value of hyperparameter D_1 was found to be 0, which enables our approach to merge or split trajectories, but not to intertwine them by splitting and then merging differently. This further reduced the density of the graph. To speed up the approach, we added edges between trajectories not every frame, but every 0.5s, since identity switches are unlikely to happen more often.

4.7 Conclusion

In this work we have proposed an approach to tracking multiple objects under global, non-Markovian behavioral constraints. It allows us to estimate global motion patterns using input trajectories, either annotated ground truth or ones from any sources, to guide tracking and improve upon a wide range of state-of-the-art approaches.

Our optimization scheme is generic and allows for a wide range of definitions for the patterns, beyond the ones we have used here. In the future, we plan to work with more complex patterns, account for appearance, and handle correlations between objects' behavior.

5 Eliminating exposure bias and loss-evaluation mismatch in MOT

Abstract

Identity Switching remains one of the main difficulties Multiple Object Tracking (MOT) algorithms have to deal with. Recently, many approaches started using sequence models to solve this problem. In this chapter, we introduce a new training procedure for sequence learning scenario that confronts the algorithm to its own mistakes while explicitly attempting to minimize the number of switches.

We propose an iterative scheme of building a rich training set and using it to learn a scoring function that is an explicit proxy for the target tracking metric, IDF_1 . Using only simple geometric features, our approach outperforms state-of-the-art in appearance-less tracking. Combining it with appearance features allows us to achieve state-of-the-art results on several MOT benchmarks.

5.1 Introduction

A common concern in many Multi Object Tracking (MOT) approaches is to prevent identity switching, the erroneous merging of trajectories corresponding to different targets into a single one. This is difficult in crowded scenes, especially when the appearance of the individual target objects is not distinctive enough. Many recent approaches rely on tracklets—short trajectory segments—rather than individual detections, to keep track of the target objects. Tracklets can be merged into longer trajectories, which can be split again when an identity switch occurs.

State-of-the-art approaches often started relying on deep networks that can evaluate a whole tracklet, rather than simply compute an affinity measure between two detections, mostly using various RNN architectures. These approaches require training the networks and suffer from one or both of two well-known problems *Exposure bias* and *loss-evaluation mismatch* [145]. Our aim is to overcome them so that we can train our networks better and thus achieve superior performance.

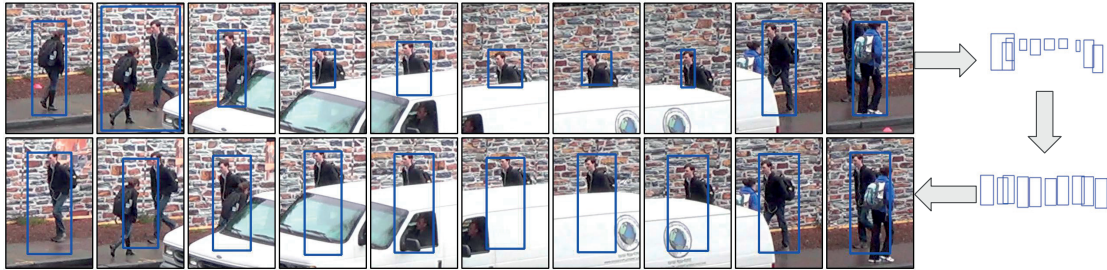


Figure 5.1 – Keeping track in a difficult situation. **Top row:** Because of the occlusion created by the passing car, a tracker can easily return a trajectory that includes several identity switches. The corresponding bounding boxes are shown on the right. **Bottom row:** Our algorithm not only eliminates identity switches but also regresses to a set of much tighter bounding boxes. Note that these results are obtained without use of any appearance information.

- **Loss-evaluation mismatch.** It occurs when training by optimizing during a metric poorly aligned with the actual desired performance during inference, such as when using a classification loss to create trajectories optimal for a tracking-specific metric, such as **MOTA** [18] or **IDF₁** [149]. To eliminate it, we introduce an original way to score tracklets that is an explicit proxy for the **IDF₁** metric and can be computed without the ground truth. We use it to identify how confidently the person is tracked, predict tighter bounding box locations, and estimate how far the real trajectory extends beyond the observed tracklet.
- **Exposure bias.** It stems from the model not being exposed to its own errors during training and results in very different sampling behaviors during training and inference/tracking. We remove this bias by introducing a much more exhaustive, yet computationally feasible, approach to exploiting the training data while training the model than in earlier approaches. To this end, during training, we do not limit ourselves to only using tracklets made of detections of one or two people as in [127, 117, 156]. Instead, we consider any grouping of tracklets produced by the tracking algorithm to be a potential trajectory but prevent a combinatorial explosion by controlling the number of tracklets that start from any given location. This yields a much richer training dataset, solves the exposure bias problem, and enables our algorithm to handle confusing situations in which a tracking algorithm may easily switch from one person to the next or miss someone altogether. Fig. 5.1 depicts one such case. Note that predicting correct trajectory was possible without the use of any appearance information in the depicted scenario.

Our contribution is therefore a solution to these two problems. By integrating it into an algorithm that only uses very simple features—bounding boxes, detector confidence—we outperform state-of-the-art algorithms that do not use appearance features. By also taking advantage of appearance-based features, we similarly outperform those that do. Taking together, this results demonstrate the effectiveness of our training procedure.

The rest of this chapter is organized as follows: Sec. 5.2 contains an overview of the related work on the topics of general MOT, tracking with longer sequences, and combating loss-evaluation

mismatch and exposure bias. In Sec. 5.3 we first describe our tracking approach, which is a variation of the multiple hypothesis tracking, which we center around learning efficient scoring function for tracklets. We then describe the exact form of our scoring function, which combats loss-evaluation mismatch, and our procedure for training it, that tackles exposure bias. Results and implementation details follow in Sec. 5.4.

5.2 Related work

Multiple Object Tracking (MOT) has a long tradition, going back many years for applications such as radar tracking [23]. With the recent improvements of object detectors, the tracking-by-detection paradigm [4] has become a *de facto* standard and has proven effective for many applications such as surveillance or sports player tracking. It involves first detecting the target objects in individual frames, associating these detections into short but reliable trajectories known as tracklets, and then concatenating these tracklets into longer trajectories. They can then be used to solve tasks such as social scene understanding [1, 9], future location prediction [103], or human dynamic modeling [55].

While grouping individual detections into trajectories it is difficult to guarantee that a *single* individual is associated to each trajectory, that is, that there are no identity switches.

Many approaches rely on appearance [65, 99, 193, 199, 35, 111, 150], motion [41], or social cues [69, 140]. They are mostly used to associate pairs of detections, and only account for very short-term correlations. However, since people trajectories are often predictable over many frames once a few have been seen, superior performance could be obtained by modeling behavior over longer time periods [73, 93, 121]. Increasing availability of annotated training data and benchmarks, such as **MOT15**, **MOT16**, **MOT17** [101, 125], **DukeMTMC** [149], **PathTrack** [123], and **WILDTRACK** [31] now makes it possible to learn the data association models required to leverage this knowledge. Since this is what our method does, we briefly review here a few state-of-the-art approaches to achieving this goal.

5.2.1 Modeling Longer Sequences

The work of [136, 135] is one of the first recent approaches to modeling long trajectories using a recurrent neural network. The algorithm estimates ground-plane occupancy, but does not perform explicit data association. Starting with [127], which presented an approach to performing data association without using appearance features by predicting the future location of the target, several MOT approaches have included sequence models to make data association more robust for the purpose of people re-identification [156, 117], learning better social models [1], forecasting future locations [103, 179] or joint detection, tracking, and activity recognition [9].

These models are usually trained on sample trajectories that perfectly match a single person’s trajectory or only marginally deviate from that, making them vulnerable to exposure bias. Fur-

thermore, the loss function is usually designed primarily for localization or identification rather than fidelity to a ground truth trajectory, which introduces a loss-evaluation mismatch with the metric, usually \mathbf{IDF}_1 [149] or \mathbf{MOTA} [18], which reflect more reliably the desirable behavior of the algorithm.

Most recent state-of-the-art approaches that use sequence models rely on one of two optimization techniques: either some form of hierarchical clustering for data association [162, 199, 149, 113, 66, 86], or on multiple hypothesis tracking [190, 90, 34]. The main difference between the two lies in that the latter allow conflicting set of hypotheses to be present before the final solution is presented, while the former usually contains valid groups of observations that don't share common hypotheses. We describe in more details and compare against these and some other state-of-the-art methods in 5.4 section.

Most similar to our approach is [90], which uses a combination of multiple hypothesis tracker and a sequence model for scoring, but performs training with a different loss function and training procedure, which uses mostly ground truth information, and is more subject to exposure bias. Another important comparison is with that of [127], which trains a sequence model for data association simply from geometric features, and is therefore perfect for comparison with our approach, when using only geometric cues.

5.2.2 Reducing Bias and Loss-Evaluation Mismatch

Since exposure bias and loss-evaluation mismatch are also a problem in Natural Language Processing (NLP) [165] and in particular machine translation [178], several methods have been proposed in these fields to reduce it [145, 13]. Most of them, however, operate under the assumption that output sequences can comprise any character from a predefined set. As a result, they typically rely on a beam-search procedure, which itself frequently uses a language model to produce a diverse set of candidates that contains the correct one. More generally, techniques that allow training models making discrete decisions such as policy gradient [176], straight-through estimation [14], and Gumbel-softmax [75] can be seen as methods to reduce exposure bias.

Unfortunately, in the case of MOT, the detections form a spatio-temporal graph in which many nearly identical trajectories can be built, which can easily overwhelm standard beam-search techniques: when limiting oneself to only the top scoring candidates to prevent a combinatorial explosion, it can easily happen that only a set of very similar but spurious trajectories will be considered and the real one ignored. This failure mode has been addressed in the context of single-object tracking and future location prediction in [71, 118] with a tracking policy learned by reinforcement learning and in [38] by introducing a spatio-temporal attention mechanism over a batch of images, thus ensuring that within the batch there is no exposure bias. Instead, the algorithm relies on historical positive samples from already obtained tracks, thus re-introducing it. For MOT, a reinforcement learning-based approach has been proposed [180] to decide whether to create new tracklets or terminate old ones. This is also addressed in [156] but the learning of

sequence models is done independently and is still subject to exposure bias. Approach of [121] attempts to explicitly optimize for the \mathbf{IDF}_1 metric. It does so by refining the output of other tracking methods. This reduces the loss-evaluation mismatch but the sequence scoring model is hard-coded rather than learned and we will show that learning it yields better results.

5.3 Method

In this section, we formalize our approach to tracking, describe the scoring function it requires, and a way to generate training data and train a model to learn such a function.

5.3.1 Tracking Formalization

Let us consider a video sequence made of N frames, in which we run a people detection algorithm on each frame individually. This yields a set \mathcal{D} of people detections $\mathbf{d}_n \in \mathbb{R}^4$, where $n = 1, \dots, N$ is the frame number, and the four elements of \mathbf{d}_n are the coordinates of the corresponding bounding box in the image. We represent a tracklet \mathbf{T} as a $4 \times N$ matrix of the form $[\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_N]$. In practice, tracklets only rarely span the whole sequence. We handle this by setting the \mathbf{d}_n to zero for frames in which the person’s location is unknown. The first non-zero column of a tracklet is therefore its start and the last its end. Two tracklets \mathbf{T}_1 and \mathbf{T}_2 can be merged into a single one \mathbf{T} if there are no frames, in which both tracklets have known detections that differ from each other.

Let us further assume we have defined a “feature” function $\Phi: \mathbb{R}^{4 \times N} \rightarrow \mathbb{R}^{F \times N}$ that assigns a feature vector of dimension F to each column of a tracklet, and from which we can estimate a scoring function $S(\Phi(\mathbf{T}))$ that is maximized when the tracklet represents perfectly a single person’s trajectory and approximates the \mathbf{IDF} score of the tracklet when S is properly trained. Tracking can then be understood as building the set of non-overlapping tracklets \mathbf{T}_j that maximizes the objective function

$$\sum_j S(\Phi(\mathbf{T}_j)). \quad (5.1)$$

5.3.2 Tracking

Our approach to tracking is a variation of multiple hypothesis tracker [89]. We iteratively merge tracklets to create longer candidate trajectories that include the real ones while suppressing many candidates to avoid a combinatorial explosion. We then select an optimal subset greedily. We consider two trajectories to be overlapping if the total number of pixels shared by bounding boxes of the two tracklets, normalized by the minimum of the sum of areas of bounding boxes in each of them, is above a threshold C_{IoU} . We also eliminate tracklets that are either shorter than N - the length of the batch, or whose score is below another threshold C_{score} . C_{IoU} and C_{score} are hyper-parameters that we estimate on a validation set. In short, the two key steps of our approach

are:

1. **Generating the set of candidate trajectories.** It must be rich enough to include all the true trajectories, yet remain small enough to prevent combinatorial explosion.
2. **Scoring the candidates to select the right ones.** The scoring function S from Eq. 5.1 must be learned carefully so that it assigns low scores to the wide range of bad candidate trajectories that can be generated, and high scores to the true trajectories.

Finally, given such a set of tracklets of various lengths, we want to select a compatible subset that maximizes our objective function. To this end we select a subset of hypotheses with the best possible sum of scores, subject to a non-overlapping constraint. We do this greedily, starting with the highest scoring trajectories. As discussed in the ablation study, we also tried a more sophisticated approach that casts it as an integer program solved optimally, and the results are similar.

5.3.3 Generating Candidate Trajectories

In this section, we assume that the scoring function S has been learned and we discuss its use to generate the candidate trajectories among which the final ones can be selected to maximize the objective function of Eq. 5.1. We will discuss the learning of S in Sec. 5.3.4.

Given the initial set of detections \mathcal{D} , we generate an initial tracklet set by linearly interpolating between pairs of detections in different frames. In other words, for each pair $(\mathbf{d}_{n_1}, \mathbf{d}_{n_2}) \in \mathcal{D}$, with $n_1 < n_2$, we take \mathbf{T} to be $[\mathbf{d}_1, \dots, \mathbf{d}_N]$, with $\mathbf{d}_n = \frac{(n-n_1)\mathbf{d}_{n_2} + (n_2-n)\mathbf{d}_{n_1}}{(n_2-n_1)}$ if $n_1 \leq n \leq n_2$ and $\mathbf{0}$ otherwise. We then iterate the following two steps for $n = 2, \dots, N$:

1. **Growing:** merge all pairs of tracklets that span at most n frames and can be merged.
2. **Pruning:** among all pairs of tracklets that we have merged $(\mathbf{T}_1, \mathbf{T}_2)$ keep only one pair for each \mathbf{T}_1 , one with the highest score $S(\Phi(\cdot))$.

This process keeps the number of hypotheses linear with respect to the number of detections. Yet, it retains a candidate for every possible detection. This prevents the algorithm from losing people and terminating trajectories too early even if mistakes are made early in the pruning process. We give an example in Fig. 5.3. In ablation study, we compare this heuristic to several others and show that it is effective at preventing combinatorial explosion without losing valid hypotheses.

5.3.4 Defining the Scoring Function

The algorithm described above relies on the scoring function $S(\Phi(\cdot))$ both to prune the set of tracklets while it is being built (to keep their number in check) and to select the final subset of

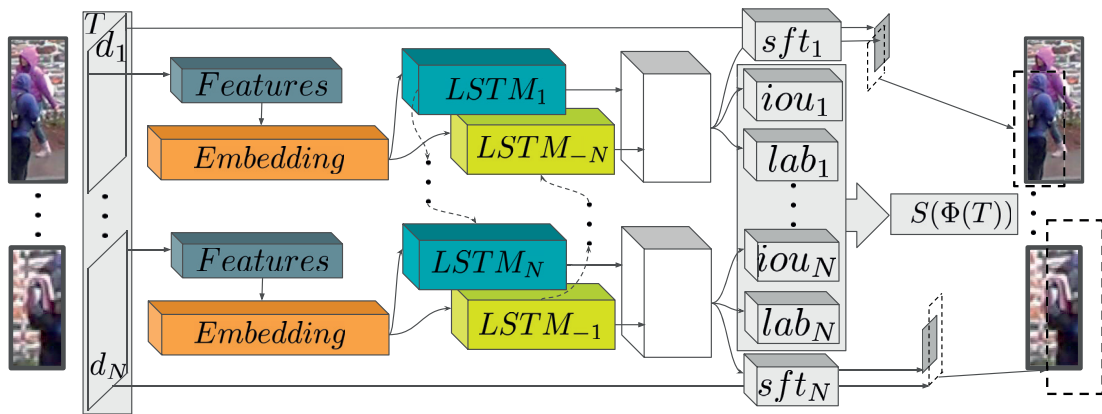


Figure 5.2 – Tracklet features are passed through an embedding layer and then processed using a bi-directional LSTM. Its outputs are used to predict the IoU with ground truth bounding boxes iou , presence of a person in a scene lab , and regress bounding box shift to obtain ground truth bounding boxes sft .

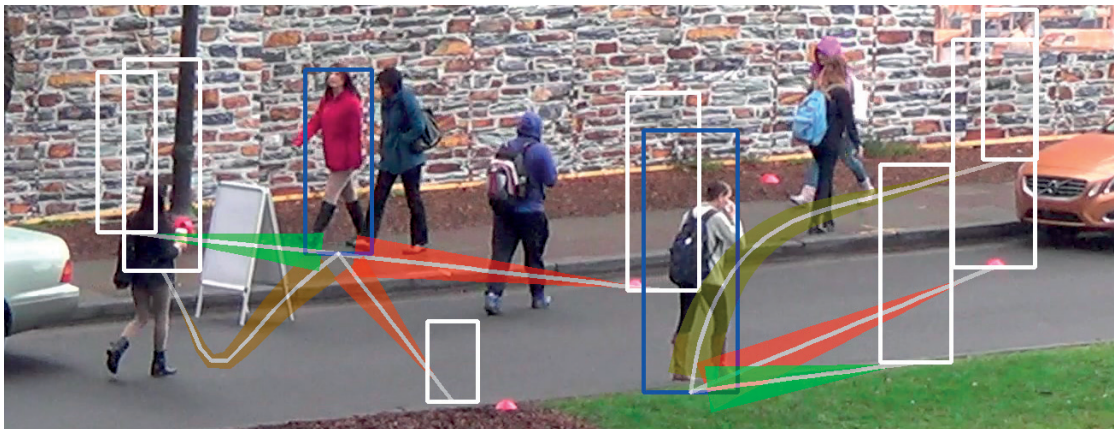


Figure 5.3 – Candidate tracklets starting from two different bounding boxes in blue and ending with bounding boxes in white. In this case, during pruning phase the best ones, shown in green, are assigned the highest score and retained, and all others are eliminated.

trajectories. Since our goal is to build tracklets that describe the trajectory of a single person as well as possible, we try to optimize them in terms of \mathbf{IDF}_1 metric. Alternative metric is \mathbf{MOTA} but \mathbf{IDF}_1 has been shown to be more sensitive to the identity switches [149]).

Ideally, S should return $S(\Phi(\mathbf{T})) \approx \mathbf{IDF}(\mathbf{T}, \mathbf{G})$ for every tracklet \mathbf{T} and the corresponding ground truth trajectory \mathbf{G} . Unfortunately, at inference time, \mathbf{G} is unknown by definition. To overcome this difficulty, recall from [149] that \mathbf{IDF} for tracklet $\mathbf{T} = [\mathbf{d}_1, \dots, \mathbf{d}_n]$ and ground truth trajectory $\mathbf{G} = [\mathbf{g}_1, \dots, \mathbf{g}_n]$ is defined as

$$\mathbf{IDF}_1(\mathbf{T}, \mathbf{G}) = \frac{2 \times \sum_{n: \mathbf{d}_n \neq \mathbf{0}, \mathbf{g}_n \neq \mathbf{0}} \mathbb{1}(IoU(\mathbf{d}_n, \mathbf{g}_n) > 0.5)}{|\{n: \mathbf{d}_n \neq \mathbf{0}\}| + |\{n: \mathbf{g}_n \neq \mathbf{0}\}|}, \quad (5.2)$$

where IoU is the intersection over union of the bounding boxes. To approximate it without knowing \mathbf{G} , we write

$$S(\Phi(\mathbf{T})) = \frac{2 \times \sum_{n: \mathbf{d}_n \neq \mathbf{0}, lab_n > 0.5} iou_n}{|\{n: \mathbf{d}_n \neq \mathbf{0}\}| + |\{n: lab_n > 0.5\}|}, \quad (5.3)$$

assuming that our network (Fig. 5.2 has been trained to regress from \mathbf{T} to

- iou_n : the prediction of intersection over union of the \mathbf{d}_n and \mathbf{g}_n boxes;
- lab_n : the prediction of whether the ground truth trajectory exists in frame n .

We also train our network to predict sft_n , the necessary change to bounding box \mathbf{d}_n to produce the ground truth bounding box \mathbf{g}_n . It is not used to compute S , but can be used during inference to improve the observed bounding boxes for better alignment with the ground truth.

To train the network to predict the lab_n , iou_n , and sft_n values introduced above, we define a loss function L that is the sum of errors between predictions and ground truth:

$$\begin{aligned} L(\mathcal{T}, \mathcal{G}) &= \sum_{n=1}^N L_{lab}(\mathbf{d}_n, \mathbf{g}_n) + \sum_{n: \mathbf{d}_n \neq \mathbf{0}} L_{iou}(\mathbf{d}_n, \mathbf{g}_n) \\ &\quad + \sum_{n: \mathbf{d}_n \neq \mathbf{0}} L_{sft}(\mathbf{d}_n, \mathbf{g}_n), \end{aligned} \quad (5.4)$$

$$\begin{aligned} L_{lab}(\mathbf{d}_n, \mathbf{g}_n) &= \|lab_n - \mathbb{1}(\mathbf{g}_n \neq \mathbf{0})\|_2, \\ L_{iou}(\mathbf{d}_n, \mathbf{g}_n) &= \|iou_n - IoU(\mathbf{d}_n, \mathbf{g}_n)\|_2, \\ L_{sft}(\mathbf{d}_n, \mathbf{g}_n) &= 1 - IoU(\mathbf{d}_n + sft_n, \mathbf{g}_n), \end{aligned}$$

where $\mathbf{d}_n + sft_n$ denotes shifting the bounding box \mathbf{d}_n by sft_n . In practice, we train the network consisting of a fully connected embedding layer, followed by a bi-directional LSTM units, and three fully connected output heads predicting objects of interest on every step. Network is depicted by Fig. 5.2 and is used to predict separately iou_n , lab_n , and sft_n , which we found more effective than regressing to the \mathbf{IDF}_1 directly. In the appendix, we discuss this in more details.

5.3.5 Training Procedure

The key to avoiding exposure bias while training the network to predict $S(\Phi(\cdot))$ is to supply a rich training set. To this end, we alternate between the following two steps:

1. Run the hypothesis generation algorithm of Sec. 5.3.2 with current network weights when evaluating the scoring function S ;
2. Add newly observed tracklets to the training set and perform a single epoch of training.

We are not only learning a scoring function. In effect, we are also improving the quality of the final tracking result: while the tracking procedure makes discrete choices about which hypotheses to pick or discard, which is non-differentiable, we nevertheless steer the tracking procedure towards always selecting the best choice by training the model on all candidates considered during tracking. In other words, our approach makes discrete choices during training, and updates the parameters based on all hypotheses that could have been selected, which is similar in spirit to using a straight-through estimator [14].

While ideologically simple, this training procedure requires a number of tweaks for optimal performance. We describe them below and study their effect in the ablation study.

Stopping criteria We start the process with random network weights and stop it when the training set size increases by less than 5% after iterating the process 10 times. We then fully train the model on the whole resulting training set. This process can be understood as a slow traverse of the search space. It starts with an untrained model that selects random hypotheses. Then, as the training progresses, new hypotheses are added and help the network both to differentiate between good and bad alternatives and to pick the best ones with increasing confidence.

Exploration with probabilistic merges While during inference in the pruning stage we always grow each tracklet by merging it with some other tracklet that provides the best score, to make a more diverse training data, during training we merged tracklet probabilistically with probability proportional to softmax of the score of the merged result multiplied by a weight coefficient. We annealed the weight coefficient during training, so that in the beginning the best pair is always merged, and later more variability is introduced.

Balancing the dataset One potential difficulty is that this procedure may result in an unbalanced training set in terms of the \mathbf{IDF}_1 values which we want to regress. We address this issue by splitting the dataset into 10 groups by \mathbf{IDF}_1 value ($[0.0; 0.1)$, $[0.1; 0.2)$, \dots , $[0.9, 1.0]$), selecting all samples from the smallest group, and then the same number from each other group. This allows us to perform hard-mining: we select $h * K$ samples at random and retain the K that contribute most to the loss.

Adjusting for batch processing We describe how our method could be run in batch mode in Sec. 5.4. Since our approach is trained on detections in a particular interval, without looking into previous detections, it could theoretically be subject to exposure bias during inference. However, one of the appearance features we use (discussed in Sec. 5.4) is a representative appearance of the ground truth trajectory matching to current detections, should there exist such a trajectory. This effectively works as a proxy to the batch processing. We have also tried actually running our approach in batch mode, and then use hypotheses obtained in the middle of the batch processing as the training data, but this reduces the amount of available training data, and has resulted in reduced performance.

5.4 Results

Here we present datasets, baselines we compare to, our results, and ablation study summary.

5.4.1 Datasets

We used the following publicly available datasets to benchmark our approach:

DukeMTMC [149]. It contains 8 sequences, with 50 minutes of training data, and testing sequences of 10 and 25 minutes with hidden ground truth for each camera, at 60fps.

MOT17 [125]. It contains 7 training-testing sequence pairs with similar statistics and hidden ground truth for test sequences, spanning 785 trajectories and both static and moving cameras. Having 3 variants of detections allows to evaluate the quality of the tracking method while ensuring it does not overfit to a particular detector.

MOT15 [101]. It contains 11 training and 11 testing sequences, with moving and stationary cameras in various settings. Ground truth for testing is hidden, and for each testing sequence there is a sequence with approximately similar statistics in the training data.

To showcase that the strength of our training procedure and loss, we performed two evaluations, one with appearance features, and one without. We evaluated on **DukeMTMC** and **MOT17** datasets our model with appearance features, and on **DukeMTMC** and **MOT15** datasets without appearance features. That allowed our model with appearance to compare to the latest state-of-the-art approaches, and our model without appearance to compare to other baselines that do not use appearance on **MOT15** dataset. We also performed ablation study on the training data of **DukeMTMC** dataset, since it is abundant, and the MOTChallenge benchmark that we used does not allow multiple submissions for a purpose of ablation study.

5.4.2 Implementation Details

Batch processing In Sec. 5.3.2 we describe how to obtain tracking results for a particular batch of length N . In practice, we process the whole input sequence in batches, each time shifting the batch by $\frac{1}{3}$ of its length. To ensure consistency with the previous batch, we never suppress hypotheses that must be present (from the previous batch) during the suppression phase, and in when picking the final solution, we make sure to pick at least one candidate that includes each sequence coming from the previous batch. We used batch of length 3s for training, having observed similar to [156] that this is enough for proper learning. During inference, we observed that our model is able to generalize beyond 3s, and having longer batches can be beneficial, which is what we used for some datasets.

Features We show results of our method with two sets of features, one that uses no appearance information, and another that does. This allows us to show that performance of our approach comes from the strength of our training procedure, rather than from having a better appearance or re-identification model. We discuss the effect of different feature types on the tracking quality lower in this section.

Appearance-less features We use simple features that can be computed from the detections without further reference to the images:

- Bounding box coordinates and confidence ($\in \mathbb{R}^5$)
- Bounding box shift with respect to previous and next detection in the tracklet ($\in \mathbb{R}^8$)
- Whether or not the detection was interpolated by the procedure of Sec. 5.3.3 ($\in \mathbb{R}^1$)
- A description of the surrounding in social terms $\in \mathbb{R}^{3 \times M}$. It comprises offsets to the M nearest detections and their respective confidence values. All values are expressed relative to image size for better generalization.

Appearance-based features We use the re-identification model of [67]. To this end, we provide following additional features in our appearance-based model:

- Appearance vector for each bounding box ($\in \mathbb{R}^{128}$)
- Euclidian distance from appearance in the bounding box to the appearance that best represents trajectory so far before the current batch, if one is available ($\in \mathbb{R}^1$). To pick the appearance that best represents trajectory so far, we computed euclidian distances between each pair of appearances in the trajectory, and picked one with the smallest sum of distances to all others.

Chapter 5. Eliminating exposure bias and loss-evaluation mismatch in MOT

- Crowd density feature - distance from the center of current bounding box to the center of nearest 1st, 5th, and 20th detection in the current frame ($\in \mathbb{R}^3$). As we discuss in the ablation study, that feature allowed made impact on the behaviour of our model with appearance in very dense crowd scenarios.

Training and hyperparameters For all datasets and sequences, we set both thresholds C_{IoU} and C_{score} of Sec. 5.3.2 to 0.6 and the hard-mining parameter h of Sec. 5.3.5 to 3. We trained with sequences of length 3 seconds, with input image fps rate ranging from 30 to 2.5 depending on the dataset. For **DukeMTMC**, we selected a validation set of 15'000 frames for each camera, pre-trained the model on data from all cameras simultaneously, and performed a final training on the training data for each individual sequence.

We have trained the model with Adam with the fixed learning rate of 0.001. Our embedding layer consists of a fully connected layer, followed by a batch normalization layer. Size of the hidden state of LSTM were 300. Thanks to abundance of training data, we used fps of 3 for **DukeMTMC** dataset. For **MOT15** and **MOT17** datasets, we trained the model with the maximum frequency every sequence allowed, to increase the amount of training data. During inference, we used batches of length 6s. We used the bounding box shift regression only in combination with the DPM [49] detector, as for other types of detectors it did not prove useful. Nevertheless, we kept L_{sft} as a part of a loss function. We plan to make our implementation (in Python and using Tensorflow) publicly available upon acceptance of the paper.

For each **MOT15** sequence group (KITTI, ADL, etc.), we trained on all sequences excluding the group, using them for validation purposes, and ran inference on the test sequences from the same group. For **MOT17**, we used **PathTrack** for pre-training of the model, and training sequences for validation. We trained re-identification network on **CUHK03** dataset.

To make sure that network makes good use of both geometric and appearance features, we first do pre-training with each type separately, and then do training from scratch, while initializing the embedding layer with the weights obtained by pre-training.

5.4.3 Baselines

We first describe baseline approaches that do not use appearance, to showcase the strength of our training procedure with our method that uses only geometric features. We then describe baseline and state-of-the-art approaches that use appearance model, with which we compare on public benchmarks of **DukeMTMC** and **MOT17** datasets. For fair comparison, we compare to methods that use publicly available set of detections, same as we do.

- **RNN** [127] relies on a recurrent neural network to perform online data association. This method is similar to ours in spirit because it uses RNN for tracking in a straightforward way. However it is trained using a different loss function and approach to create the training

data.

- **LP2D** [101] is the highest-scoring appearance-less original baseline presented with MOT15. Like **KSP**, it formulates tracking in terms of solving a linear program.
- **PTRACK** [121] is an approach to improve results of other methods by refining the trajectories they produce, to maximize an approximation of the **IDF₁** metric. The approximation is hand-designed, and not learned as in our approach.
- **SORT** [20, 131] combines Kalman filtering with a Hungarian algorithm and currently is the fastest one on the MOT15 dataset.
- **MHT** [190] perform multiple hypothesis tracking, aided, among others, by pose features extracted from convolution pose machines [175].
- **CDSC** [162] uses domination set clustering to perform both within- and across-camera tracking. It employs image features from ResNet-50 [63] pre-trained on ImageNet.
- **REID** [199] performs hierarchical clustering of tracklets, and leverages the re-identification model of [197] pre-trained on 7 different datasets.
- **BIPCC** [149] optimally groups observations into clusters of detections of similar appearance, by solving a binary integer problem. This is a baseline appearance method for **DukeMTMC** dataset.
- **DMAN** [83] uses dual attention networks to generate attention that focuses on the relevant image part, as well as relevant temporal fragment, to perform data association.
- **JCC** [86] formulates a joint problem of multiple object tracking and motion segmentation as a joint co-clustering problem, which is solved by local search to jointly group pixels and bounding boxes, to produce tracking and segmentation.
- **MOTDT17** [113] performs a hierarchical data association, grouping detections based on the learned re-identification metric, and exploiting geometric features and Kalman filter in case of failure of the former.
- **MHTBLSTM** [90] is very similar to our approach both in using the multiple hypothesis tracker, and using of sequence model to score the data association. Nevertheless, it is trained only on sequences using ground truth sequence combined with at most one false positive, and possibly some missed detections.
- **EDMT17** [34] solved tracking problem as a multiple hypothesis tracker. Both growing and pruning phase utilizes learned detection-detection and detection-scene association model, which allows to better score detections and hypotheses.
- **FWT** [66] solves a binary quadratic problem to optimally group detections from separately run head and body detectors.

Method	IDF ₁	MOTA	IDs	IDF ₁	MOTA	IDs
Sequence	easy			hard		
OURS	84.0	79.2	169	76.8	65.4	267
MHT	80.3	78.3	406	63.5	59.6	1468
REID	79.2	68.8	449	71.6	60.9	572
CDSC	77.0	70.9	693	65.5	59.6	1637
OURS-geom	76.5	69.3	426	65.5	59.1	972
PTRACK	71.2	59.3	290	65.0	54.4	661
BIPCC	70.1	59.4	300	64.5	54.6	652

 Table 5.1 – Benchmark results on **DukeMTMC** dataset.

Method	IDF ₁	MOTA	IDs
OURS-geom	27.1	22.2	700
RNN	17.1	19.0	1490
SORT	26.8	21.7	1231
LP2D	—	19.8	1649

 Table 5.2 – Benchmark results on **MOT15** dataset.

We will show that we outperform the methods in the first class and, given enough training data, do almost on par with those in the second, even though we use far less image information. This is a testament to the power of our training approach.

5.4.4 Comparative Performance

In Tab. 5.3, we compare our approach that uses appearance to state-of-the-art methods using appearance, on **MOT17** dataset. Our approach is best both in terms of **IDF₁** metric, and the number of identity switches. It does not feature the best **MOTA** score, which is not surprising since our loss function optimizes the proxy to **IDF₁**. Furthermore, we see that the best published

Method	IDF ₁	MOTA	IDs
OURS	57.2	44.2	1529
DMAN	55.7	48.2	2194
JCC	54.5	51.2	1802
MOTDT17	52.7	50.9	2474
MHTBLSTM	51.9	47.5	2069
EDMT17	51.3	50.0	2264
FWT	47.6	51.3	2648

 Table 5.3 – Benchmark results on **MOT17** dataset.



Figure 5.4 – Bounding boxes and the last 6 seconds of tracking, denoted by lines, in dense crowd on **DukeMTMC** dataset.

approach in terms of **MOTA** actually performs worst in terms of **IDF₁** on this dataset.

In Tab. 5.1, we present both the results of our approach that uses appearance, and the one, which does not, on **DukeMTMC** dataset. Our approach with appearance performs best both in Hard and Easy set of sequences in terms of **IDF₁**, **MOTA**, and the number of identity switches. Furthermore, compared to other top scoring methods that use re-identification networks pre-trained on outside dataset, our network uses exclusively training data from **DukeMTMC** dataset. On Easy set of sequences, our approach also outperforms [79], which we did not present in the table since it uses a private set of detections, rather than a public one. Our approach that does not use appearance also fares surprisingly well, outperforming appearance-based baseline of [149], as well as an improvement on the top of it using learned scene patterns of [122]. This shows that the strength of our method comes from the training procedure and loss, rather than from a learned appearance model.

Results from Tab. 5.2 further strengthen this claim. We compare our approach without appearance to other methods that do not use appearance. Our main comparison is with **RNN**, which also uses an RNN to perform data association. Despite the fact that **RNN** uses external data to pre-train their model, and we use only the **MOT15** training data, our approach is able to outperform it with a large margin. Another interesting comparison is with **SORT**, which performs nearly as good as our approach. We additionally run this approach on the validation data we used for **DukeMTMC**. This resulted in a **MOTA** score of 49.9 and **IDF₁** one of 24.9, whereas our method reaches 70.0 and 74.6 on the same data. In other words, because there is much more training data in **DukeMTMC** than **MOT15** and because our method, unlike **SORT**, can take advantage of it, the gap in performance is much larger on the former than the latter. These results, and a full breakdown for other benchmarks, is available in appendix. Some qualitative results are available in Fig. 5.5 and Fig. 5.4.

5.4.5 Ablation study

The last 15'000 frames of training sequences of **DukeMTMC** were used for an ablation study. We varied the three main components of our solution (without appearance) to show their importance

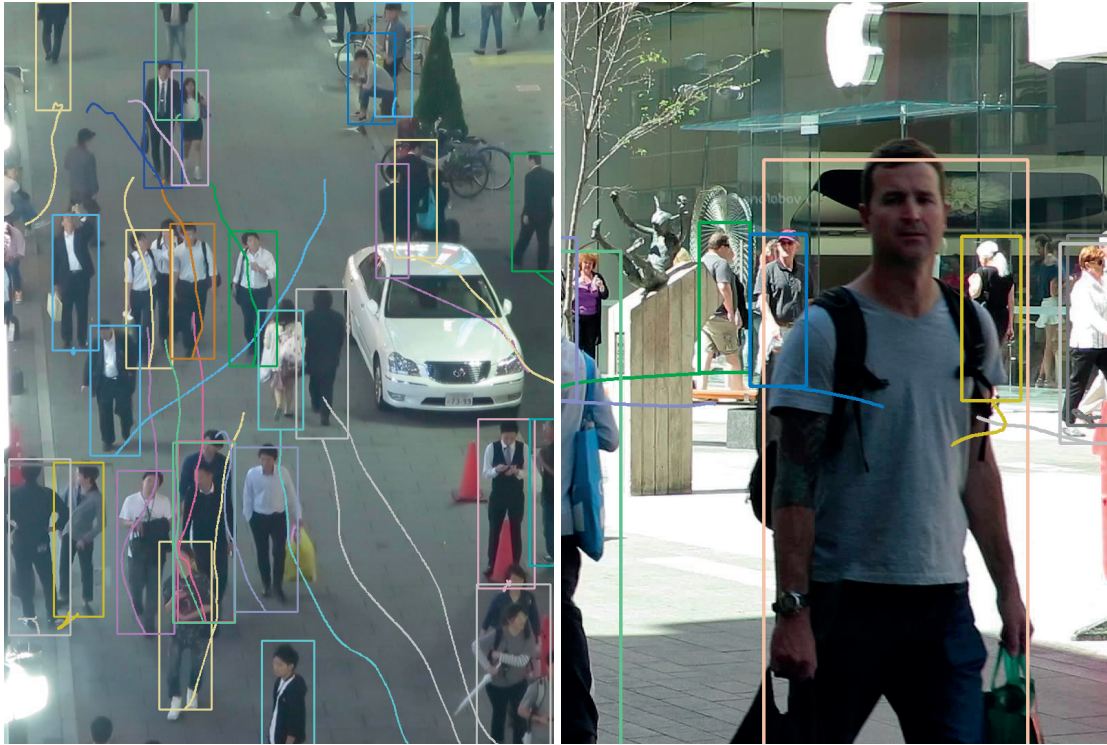


Figure 5.5 – Bounding boxes and last 6 seconds of tracking, denoted by lines, in two sequences of the MOT17 dataset.

for the final tracking accuracy: data composition, scoring function, and training procedure. Creating a fixed training set by considering tracklets with at most one identity switch as in [156] or [117] decreased performance. Pruning hypotheses based on their scores or total count like [190] resulted in either a computational explosion or reduced performance. Computing loss on the prediction of $S(\Phi(T))$, trying to regress **IDF** value directly, not regressing bounding box shifts, or using a standard classification loss as in [156] were equally counter-productive. Not balancing the training set, not using hard-mining, not pre-training the network also adversely affected the results. Selecting the final solution using an Integer Program instead of a greedy algorithm, pre-training model with each type of features separately, or training a deeper network had no significant effect. We detail those changes, and effect of various feature groups, below.

Training dataset generation First 2 columns of Tab. 5.4 show the changes in the training dataset generation procedure: using random tracklets between all pairs of detections, or tracklets obtained by combining two ground truth trajectories; adding not all observed tracklets to the training data, but only those selected into final solution; doing pruning based on the reported score of the tracklet or keeping a fixed number of tracklets with highest scores. Fixing the training set or augmenting it only with tracklets selected into solution yields simply a smaller and less diverse training data, which had a detrimental effect on the results of tracking. Pruning by score did not allow us to train any reasonable model, because of the computational explosion of the

Δ Training	IDF	Δ Dataset	IDF	Δ Loss	IDF
Dataset: all pairs	71.5	Loss on IDF	69.5	-hardmining	72.1
Dataset: mix of two	70.7	Regressing IDF	63.4	-balanced dataset	69.9
Selected only	63.6	-bbox regression	72.4	batch 6	72.6
Pruning by score	—	-bbox loss	66.3	IP solution	73.8
Pruning by count	54.2	classification	41.8	pretraining	71.9

Table 5.4 – Ablation study. Left, middle and right columns show possible changes in dataset creation procedure, loss function, and training procedure, as well as respective values of **IDF** metric with respect to reference solution (**IDF** 74.6). Details about each change are given in Sec. 5.4.5.

trajectories with very similar scores, that were all taken into training data. Pruning by count proved ineffective for the same reason - training data contained many very similar trajectories.

Tracklet scoring function Middle 2 columns of Tab. 5.4 examine the changes in the scoring function: computing loss function on the value of $\|IDF(D, T) - S(\Phi(D))\|_2$, or trying to regress the value of **IDF**₁ directly, without splitting the task into accounting for false positives or false negatives; Not modifying the input detections based on the regression of bounding box shifts, and simply removing L_{sft} from the loss function; Posing task as a classification task, where tracklet belongs to the positive class *iff* all detections overlap with some ground truth trajectory with IoU of at least 0.5. We observed that putting loss on the value of **IDF**₁ did not give any improvement and resulted in small decrease, probably due to the fact that multiple loss components acted as regularizers. Trying to regress **IDF**₁ directly gave even worse results, probably because understanding the behaviour of **IDF**₁ function is much harder than understanding behaviour of false positives and false negatives, which we regress through *lab* and *iou*. Most interesting is the fact that even when we don't modify the input bounding boxes (which, of course, yield improvement, especially in the cases of occlusions, as shown in the table) but simply have L_{sft} as part of the loss function, that improves the results, acting as a regularizer. Posing a classification task doesn't result in a very good trained model due to many overlapping sequences, some of which have IoU greater than 0.5 in every frame, and some don't, and it is hard for the model to distinguish between the two. One change that could have possibly improved the model and that has not been tested is finetuning the loss on **IDF**₁ after training the model with our loss.

Training protocol Last two columns of Tab. 5.4 examine changes to the training protocol: not using hard-mining and not balancing the dataset, using various lengths of batches N in inference, and pre-training the network on data from all cameras. We have observed that our model trained on batches on length 6, can generalize to batch lengths of 12 and 15, but observed saturation in the tracking quality beyond that. Other changes, such as having more layers, pre-training the network for each feature type separately or input dropout, did not have significant effect, probably due to very simple features. We also did not observe significant difference between solving the

IP problem of selecting optimal set of tracklets subject to non-overlapping constraint compared to using a greedy solution because greedy solution frequently found optimal or nearly optimal solutions, while also being faster.

Feature groups We also performed an evaluation of how different features affect the quality of the solution. Since it is hard to evaluate all possible combinations of features, we describe their effect qualitatively below.

- **Appearance features** We have found that among 3 appearance features (appearance of the current bounding box, appearance distance to representative bounding box of the trajectory so far, crowd density around the detection) the appearance distance had the biggest effect. In its absence, appearance of the bounding box was also able to work to a certain effect, but with worse results. Effect of crowd density feature was only visible in crowded scenarios, where our merging procedure preferred to merge detections that are further apart in time, but more similar, compared to less crowded scenarios, where it preferred to merge detections based more on the spatial vicinity. We used the distances to the 1st, 5th, 20th closest bounding box as density feature, and introducing more distances didn't have any significant effect.
- **Social features** Having social features, namely description of the 3 closest bounding boxes, helps our appearance-less model to preserve identities in the absence of visual information, improving IDF_1 from 67.5 to 74.6. Introducing more closest bounding boxes did not improve performance in any meaningful way.
- **Probabilistic merging** had a profound effect on the ability of network to fuse appearance-based and geometry-based features together in our experiments. Without it, picking only the best candidate, resulted in a model that performed merges mostly either based on the appearance information (largely ignoring spatial vicinity), or based on the spatial vicinity and motion information (largely ignoring appearance information).

Other notes We chose not to apply any weight factors to the components of the loss function because its components could be seen as identifying the false positive (when lab should be zero) and false negative (when $iou < 0.5$) errors, and since we wanted to weigh the two equally, we did not use any weight factors to L_{lab} , L_{sft} , L_{iou} .

Additionally, while it may seem logical to use L_{iou} to predict the IoU between the modified bounding box $\mathbf{d}_n + sft_t$ and the ground truth bounding box \mathbf{g}_n , in practice that makes it harder to train the network as it finds an easy solution of regressing empty bounding boxes, which never intersect with the ground truth, thus always making a perfect prediction of L_{iou} . Instead, we use the network during inference in the autocontext mode: we predict the bounding boxes, update the input tracklet with them, and then regress the intersection over union of the new tracklet to compute the value of S .

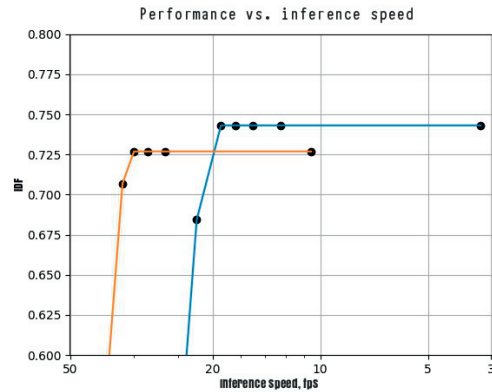


Figure 5.6 – Inference speed, fps, and resulting IDF_1 when we do additional pruning of the hypotheses by the score during inference. Curves are shown for batch lengths 12 and 6. We can reach 30fps without virtually any sacrifice in the quality of the results.

5.4.6 Computational effort

Adding a cutoff on sequence scores in the pruning step of Sec. 5.3.3 allows our python implementation to perform inference at 30fps, at the cost of a very small decrease in performance. Furthermore, most computation occurs in the growing and pruning phases, which could be drastically sped up by re-implementing them in C++ [131].

Computational effort required for our method could be split into 3 groups:

- **Dataset generation.** This requires running the model for several epochs, and keeping all of the training data in memory simultaneously. We have observed that the size of available memory could be a limiting factor for the training procedure, and part of the training data could be saved to disk and restored on each step of the dataset generation. In practice, the process of dataset generation for all datasets took several hours and produced datasets up to 1.5×10^7 samples (**DukeMTMC**, Camera 6), but had to be only performed once. In practice, we have observed that the process of generating dataset oscillates between a slow growth of the dataset when the model is training, combined with sudden jumps where the model discovers new set of possible hypotheses thanks to learning the observed set well enough. This process is repeated several times before saturation, during which growth goes below 5% in 10 epochs, which is when we finish the dataset generation procedure.
- **Model training.** This simply trains the model over already generated data, and we have observed that with our simple features 20-30 epochs are usually enough to reach best performance on the validation data, and took under one hour.
- **Inference.** The most computationally heavy part of the inference is not scoring the trajectories with the learned model, but rather the process of generating possible candidates,

and merging them. While we could not replace our pruning procedure with cutting off hypotheses based on their score for the purpose of dataset generation, having already trained the model, we used this as an optimization to speed up the inference. As shown in Fig. 5.6, by additionally pruning all hypotheses with scores below certain cutoff, we speed up the growing and merging procedure without significant sacrifices in the quality of obtained results, reaching 30 fps.

5.5 Conclusion

We have introduced a training procedure that significantly boosts the performance of sequence models by iteratively building a rich training set. We showed that our full model is able to outperform state-of-the-art approaches on several challenging benchmarks. Our model used only with geometry features outperforms methods relying on the same kind of simple features and are on par with methods using much more sophisticated appearance-based features. This could prove extremely useful to solve problems in which appearance is hard to use, such as cell or animal tracking [127]. We also introduced a sophisticated regression model for target tracking metric **IDF** and showed that using it helps both in training and in tracking. Our data association procedure could also be extended to take more advanced appearance features, such as pose information, into account, which we plan to do in future work.

6 Conclusions

6.1 Summary

In Chapter 3, we presented a model for tracking the ball and the players in volleyball, basketball, and football. Our model accounts for the interaction between the ball and the players, allowing to track the invisible ball while it is possessed by the player. It also accounts for the physical model of the ball motion, allowing to track the ball more precisely while it is in the free fall. To differentiate, which model constraints should be imposed when, our problem formulation also requires to detect the state of the game. We show that simultaneously solving such three tasks, tracking the ball, the players, and estimating game state, brings better performance than individual tracking, across multiple sports.

In Chapter 4, we explored the idea of learning behaviours from the data, when we presented an approach to imposing global, non-Markovian constraints on multiple object tracking, while keeping the tracking problem tractable. As an example of such constraints, we proposed simple motion patterns for pedestrian and car movement, that govern how such objects are likely to enter the scene, traverse it, and exit it. We formulated a joint task of tracking and estimating motion patterns, and showed, how each subtask could be solved: how tracking could be performed given the motion patterns, and how the motion patterns could be estimated given the tracking. We then combined these two approaches in an unsupervised scheme, which iterates between finding the trajectories, and learning the motion patterns. Our experiments showed that such a scheme improves the tracking quality of a number of state-of-the-art trackers.

In Chapter 5, we investigated how learned behaviours could be implicitly incorporated into tracking by learning the sequence models for data association. We discussed that traditionally learning such models is associated with two problems, loss-evaluation mismatch and exposure bias, that hinder the quality of the training. We proposed a training scheme which confronts the model to its own mistakes, to eliminate the exposure bias, and proposed a loss which works as a direct proxy to a tracking metric, reducing the loss-evaluation mismatch. By combining a model trained with our loss and training scheme with a simple multiple hypothesis training framework,

we were able to obtain state-of-the-art results on several tracking benchmarks. Performed ablation study and comparison to a simpler version of the model without appearance information confirmed that our results are a direct consequence of our training scheme and loss, rather than an implication of using a particular appearance model.

6.2 Future work

6.2.1 Future location prediction

One of the benefits of sequence-based models for tracking is the fact that they can be used as generative models to predict future locations of the tracked objects. Combined with relevant benchmarks such as Stanford Drone Dataset [151], this has led to introduction of many approaches that jointly predict motion of multiple targets in pedestrian scenes [156, 182, 1, 118], in traffic scenes [103, 21], and for multiple object tracking with multiple target types [164].

6.2.2 Role understanding

First two of our described approaches relied on the explicit consistency of object behaviour, while the last one relied on learning it implicitly, by learning sequence models. Recently, a number of works used sequence models together with explicit consistency of the object behaviour by performing tracking jointly with role understanding [9, 19]. Given presence of the large-scale datasets, combining explicitly defined human activity labels with its learnable representation should impose activity label consistency and implicitly improve the tracking quality.

6.2.3 Tracking with segmentation/pose estimation

While our work concentrates on long term dependencies in people tracking because short appearance-based interactions can often fail in crowded scenes, the quality of tracking can still be significantly improved by incorporating other short term signals, such as pose consistency of semantic segmentation consistency. Thanks to abundance of data and models for such tasks, it is now becoming possible to jointly perform tracking together with such tasks [126, 80, 72]. However, they usually rely on multiple pre-trained models, which are fused together, rather than models that are trained together to perform such tasks, which possibly limits their performance. Models such as [116] can potentially alleviate such problems.

6.2.4 Reducing exposure bias

Our approach to reducing exposure bias relies on confronting algorithm with its own mistakes, but optimizes the approximation of the tracking metric, which can be obtained on every step of the tracking approach. Another possible approach in this direction relies on the recent

advances in deep reinforcement learning to optimize exactly the tracking metric. This, however, comes with the drawback that it requires to run algorithm on the sequence completely before obtaining the feedback in terms of the tracking metric, which is why many approaches rely on it to optimize the hyperparameters of trackers [1, 44]. In single object tracking, reinforcement learning have recently achieved state-of-the-art results [71, 188, 194]. This topic has been mostly unexplored in multiple object tracking, with the exclusion of [154], which only uses geometric features. Therefore, an important future work direction is the combination of state-of-the-art tracking approaches with reinforcement learning techniques, which could significantly improve performance by tackling exposure bias and loss-evaluation mismatch.

6.2.5 Better data association

In the recent years, the progress in multiple object tracking has been largely driven by better visual models and/or larger datasets. For example, pre-training pairwise data association potentials on PathTrack [123] dataset has been shown to improve the quality of tracking without any changes to the algorithm [36], while approaches using better detections and better trained re-identification model also achieve best results on DukeMTMC dataset [150]. These methods have frequently been combined with very simple data association techniques such as Hungarian algorithm [95] for online tracking, as well as simple network flow based models, hierarchical data association, or multiple hypothesis trackers for batch processing. However, advances in data association techniques are also paramount for getting best tracking quality. [161] could be seen as one of such recent results.

A An appendix

A.1 Cost function definition from Chapter 4

These functions are used to score the edges of a trajectory to compute how likely is it that a particular trajectory follows a particular pattern. As stated in Sec. 4.3.3:

$$C(T, P, A) = \frac{\sum_{t \in T} M(t, p_{A(t_1)})}{\sum_{t \in T} N(t, p_{A(t_1)})}, \quad (\text{A.1})$$

$$N(t, p) = n(v_{in}, t_1, p) + n(t_{|t|}, v_{out}, p) + \sum_{1 \leq j \leq |t|-1} n(t_j, t_{j+1}, p), \quad (\text{A.2})$$

$$M(t, p) = m(v_{in}, t_1, p) + m(t_{|t|}, v_{out}, p) + \sum_{1 \leq j \leq |t|-1} m(t_j, t_{j+1}, p), \quad (\text{A.3})$$

where T is a set of edges of all trajectories, A is the assignment between a trajectory and a pattern, and P is a set of patterns. As shown in (2) and (3), to score a trajectory we score all its edges plus the edges from v_{in} , the node denoting the beginnings of the trajectories, and the ones to v_{out} , the node denoting the ends of trajectories. As mentioned in chapter 3, we want $N(t, p)$ to reflect the full length of the trajectory and the pattern, and $M(t, p)$ to reflect the total length of the aligned trajectory and the pattern. In what follows, we provide definitions of n and m in all cases.

In **Table A.1**, we show how to compute n and m for edges that link two detections and follow some pattern. For n we take the pattern length to be positive or negative depending on whether the projection of the edge to the pattern is positive or negative. For m , we penalize edges far from the pattern and edges going in the direction opposite to the pattern, in two different ways, which gives rise to the three cases shown in the table. In **Table A.2**, we show how to compute n when one of the nodes is v_{in} or v_{out} , denoting the start or the end of a trajectory. A special case arises when a node is in the first or the last frame of an input batch, and a trajectory going through it does not need to follow the pattern completely. This results in a total of two cases

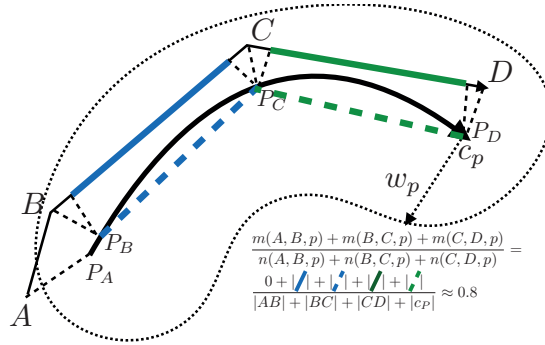


Figure A.1 – Example of computing the cost function C for three consecutive edges (A,B) , (B,C) , (C,D) . Dotted line around the pattern centerline c_p shows the area within the distance w_p to the pattern. The denominator contains the total length of the edges plus the total length of the pattern, while the numerator contains the parts aligned with each other (in green and blue). The edge (A,B) is not counted as aligned, because A is further from the pattern than its width w_p .

we show in the table. In **Table A.3**, we show the two cases when we assign the transition to no pattern \emptyset , one case when we assign a normal edge joining two detections, and the other when we assign edge from v_{in} or to v_{out} , indicating the beginning or the edge of the trajectory.

Case	Explanation	Figure
Normal edge aligned with the pattern: B and C are within distance w_p to the pattern centerline, P_B is earlier on the curve c_p than P_C .	For the edge (B, C) , we find the nearest neighbor of the two endpoints on the pattern, namely P_B and P_C . Formally, we have $P_B = \operatorname{argmin}_{x \in c_p} \ B - x\ $. Then we project P_B and P_C orthogonally back onto (B, C) . This guarantees that $m(B, C, p) \leq n(B, C, p)$ with equality when (B, C) and (P_B, P_C) are two parallel segments of equal length, and also penalizes deviations from the pattern in direction.	<p> $n(B, C, p) = BC + P_B P_C$ $m(B, C, p) = B^1 C^1 + P_B P_C$ </p>
Normal edge aligned with the pattern: B and C are further away than w_p from the pattern centerline, P_B is earlier on the curve c_p than P_C .	$n(B, C, p)$ is calculated in the same way as done in the previous case. To penalize deviations from the pattern in distance, we take $m(B, C, p) = 0$	<p> $n(B, C, p) = BC + P_B P_C$ $m(B, C, p) = 0$ </p>
Normal edge not aligned with the pattern: P_B is later on the curve c_p than P_C .	To keep our rule about N being the sum of lengths of pattern and trajectory, we need to subtract the length of arc from P_B to P_C , as it is pointing in the direction opposite to the pattern. To penalize this behavior, we take $m(B, C, p)$ to be $- P_B P_C $, multiplied by $1 + \epsilon$. In practice, we use $\epsilon = 1$.	<p> $n(B, C, p) = BC - P_B P_C$ $m(B, C, p) = - P_B P_C \times (1 + \epsilon)$ </p>

Table A.1 – Table describing full definitions of n and m in normal cases, when edges between two detections align with a pattern. They all follow naturally from the rule about N being the sum of length of trajectory and the pattern, and M being the sum of aligned lengths.

Appendix A. An appendix

Case	Explanation	Figure
Edge from the source to a normal node I from a normal node to the sink	To keep our rule about N being the sum of lengths of pattern and trajectory, we need to add the length from the beginning of the pattern to the point closest to the node on the centerline I from the point closest to the node on the centerline to the end of the pattern. Since we didn't observe any parts of trajectory aligned with these parts, we take $m = 0$.	
Edge from the source to a normal node in the first frame of the batch I from a normal node in the last frame of the batch to the sink	We assume that our trajectories follow the path completely. However, this might be not true, which we observe from the middle, that is, the ones that begin in the first frame of the batch or end in the last frame. In that case we don't need to add the part of the pattern before I after the current point closest to the node, which is why we take $n = m = 0$.	

Table A.2 – Table describing full definitions of n and m in corner cases when one of the edges go through $I \equiv v_{in}$ or $O \equiv v_{out}$, indicating the beginning or the end of a trajectory. They all follow naturally from the rule about N being the sum of length of trajectory and the pattern, and M being the sum of aligned lengths.

A.1. Cost function definition from Chapter 4

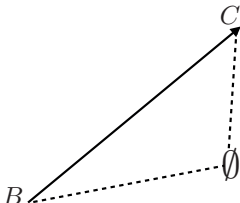
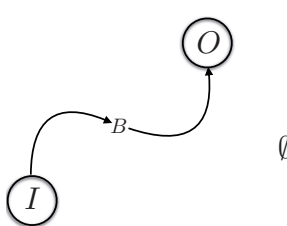
Case	Explanation	Figure
Normal edge aligned to no pattern	To keep our rule about N being the sum of lengths, we take n to be just the length of the trajectory, since we assume the length of empty pattern to be zero. We penalize such assignment by a fixed constant ϵ_\emptyset , taking m to be n multiplied by such constant. In practice, we keep $\epsilon_\emptyset = 0.3$ when training from ground truth, or $\epsilon_\emptyset = -3$ otherwise.	 $n(B, C, p) = BC $ $m(B, C, p) = BC \times (1 + \epsilon_\emptyset)$
Edge from the source $I \equiv v_{in}$ / to the sink $O \equiv v_{out}$, aligned to no pattern	To keep our rule about N , we take both $n = m = 0$.	 $n(I, B, p) = n(B, O, p) = 0$ $m(I, B, p) = m(B, O, p) = 0$

Table A.3 – Table describing full definitions of n and m in corner cases when there is no pattern. They all follow naturally from the rule about N being the sum of length of trajectory and the pattern, and M being the sum of aligned lengths.

A.2 Additional results for Chapter 4

Here we provide the full results of all the methods on all the datasets, after the textual description of datasets and baselines. Tables A.4, A.5 are the full versions of Table 4.1 of chapter 4. Table A.6 reports details on **Duke** dataset in details, as reported on the MOTChallenge website. In Tables A.4, A.5, we compare the original output of the method with the improvements brought by our approach in both supervised and unsupervised manner, denoted "-i" and "-o", respectively. In Table A.7, we compare the methods when using the ground truth set of detections as input. As in chapter 4, we report the results for the matching distances of 3m (0.1 of the tracking area for the **Station** and **Rene** datasets), and for **IDF₁** metric we also show results for 1m to indicate that the ranking of the methods does not change, but the improvement brought by our methods is less visible due to reconstruction errors when we estimate the 3D position of the person from the bounding box. This fact is especially highlighted by the Table A.7, where difference in the metric computed for distances of 3m. and 1m. is especially large.

Specifically, We report the **IDF₁**, identity level precision and recall **IDPR** and **IDRC** defined in [149], as well as **MOTA**, precision and recall **PR** and **RC**, and the number of mostly tracked **MT**, partially tracked **PT** and mostly lost trajectories **ML** defined in [18].

Our evaluation of **DukeMTMC** dataset is available on MOTChallenge website under the name **PT_BIPCC**, and comparison on **MOT16** under the name **PT_JMC**.

Readers may note that often we observe an increase in the number of mostly lost trajectories and drop in recall. One of our optimization parameters controls whether or not to remove trajectories assigned to no pattern during post-processing. Removals reduce the number of false positives, but may discard tracks for some partially tracked people, which don't follow any pattern. This increases the **ML** and lowers the recall, as observed by the reviewer. This happens in large part because both contrast and visibility are low, resulting in poor detection quality, for example on **ETH** dataset.

Finally, additional results of evaluation on **WILDTRACK** datasets are presented in Tab. A.8.

A.2. Additional results for Chapter 4

Method	Dataset	IDF ₁	IDPR	IDRC	MOTA	PR	RC	MT	PT	ML
EAMTT	Town	0.72 (0.59)	0.76	0.68	0.73	0.92	0.82	158	68	20
EAMTT-i	Town	0.80 (0.63)	0.84	0.76	0.73	0.91	0.82	165	59	22
EAMTT-o	Town	0.82 (0.65)	0.83	0.80	0.74	0.89	0.86	182	44	20
JointMC	Town	0.75 (0.63)	0.90	0.65	0.64	0.95	0.68	128	54	64
JointMC-i	Town	0.77 (0.64)	0.91	0.66	0.64	0.95	0.68	129	52	65
JointMC-o	Town	0.76 (0.62)	0.88	0.67	0.65	0.93	0.71	138	50	58
MHT_DAM	Town	0.56 (0.45)	0.82	0.42	0.40	0.90	0.46	55	98	93
MHT_DAM-i	Town	0.56 (0.45)	0.83	0.42	0.40	0.90	0.46	59	90	97
MHT_DAM-o	Town	0.57 (0.45)	0.81	0.44	0.42	0.89	0.48	63	94	89
NOMT	Town	0.71 (0.62)	0.83	0.63	0.65	0.94	0.71	122	76	48
NOMT-i	Town	0.76 (0.65)	0.87	0.68	0.66	0.93	0.72	135	61	50
NOMT-o	Town	0.75 (0.63)	0.83	0.68	0.66	0.91	0.75	144	59	43
SCEA	Town	0.56 (0.43)	0.83	0.42	0.40	0.90	0.46	56	95	95
SCEA-i	Town	0.58 (0.45)	0.87	0.44	0.44	0.95	0.47	62	89	95
SCEA-o	Town	0.58 (0.43)	0.80	0.45	0.43	0.89	0.50	65	94	87
TDAM	Town	0.60 (0.48)	0.71	0.52	0.39	0.78	0.56	70	112	64
TDAM-i	Town	0.60 (0.48)	0.73	0.51	0.41	0.80	0.56	69	110	67
TDAM-o	Town	0.59 (0.45)	0.67	0.54	0.37	0.74	0.60	82	108	56
TSML_CDE	Town	0.68 (0.58)	0.75	0.63	0.72	0.95	0.79	143	79	24
TSML_CDE-i	Town	0.76 (0.62)	0.84	0.70	0.73	0.95	0.79	150	68	28
TSML_CDE-o	Town	0.78 (0.62)	0.82	0.74	0.74	0.92	0.83	161	68	17
CNNTCM	Town	0.58 (0.46)	0.79	0.46	0.45	0.90	0.53	63	110	73
CNNTCM-i	Town	0.61 (0.46)	0.80	0.49	0.48	0.90	0.55	73	96	77
CNNTCM-o	Town	0.62 (0.46)	0.77	0.52	0.48	0.87	0.59	85	95	66
KSP	Town	0.41 (0.26)	0.47	0.36	0.64	0.93	0.73	107	105	34
KSP-i	Town	0.69 (0.42)	0.78	0.61	0.65	0.93	0.73	118	91	37
KSP-o	Town	0.69 (0.42)	0.76	0.63	0.64	0.91	0.75	122	88	36
MDP	Town	0.59 (0.45)	0.65	0.55	0.50	0.81	0.68	103	97	46
MDP-i	Town	0.66 (0.49)	0.72	0.61	0.54	0.83	0.71	116	82	48
MDP-o	Town	0.63 (0.45)	0.66	0.61	0.50	0.79	0.73	113	94	39
RNN	Town	0.48 (0.30)	0.52	0.45	0.60	0.88	0.77	122	103	21
RNN-i	Town	0.59 (0.36)	0.65	0.55	0.61	0.90	0.76	125	98	23
RNN-o	Town	0.53 (0.34)	0.57	0.50	0.59	0.89	0.77	125	99	22
SORT	Town	0.62 (0.46)	0.81	0.50	0.57	0.98	0.61	49	152	45
SORT-i	Town	0.72 (0.47)	0.85	0.62	0.64	0.95	0.69	96	109	41
SORT-o	Town	0.65 (0.46)	0.83	0.60	0.60	0.90	0.65	174	58	14

Table A.4 – Full results for all methods on the **Towncentre** dataset (abbreviated as **Town**), when using our detections as input and using the results of state-of-the-art trackers as input. Number in brackets in **IDF₁** column indicates result for the distance of 1 m.

Appendix A. An appendix

Method	Dataset	IDF₁	IDPR	IDRC	MOTA	PR	RC	MT	PT	ML
KSP	ETH	0.45 (0.15)	0.45	0.45	0.47	0.72	0.71	182	148	22
KSP-i	ETH	0.62 (0.18)	0.71	0.54	0.48	0.75	0.57	134	144	74
KSP-o	ETH	0.57 (0.18)	0.59	0.67	0.49	0.67	0.76	217	121	14
MDP	ETH	0.55 (0.20)	0.63	0.48	0.40	0.79	0.60	113	194	45
MDP-i	ETH	0.58 (0.21)	0.76	0.46	0.41	0.83	0.50	105	143	104
MDP-o	ETH	0.58 (0.21)	0.64	0.62	0.41	0.72	0.69	157	146	49
RNN	ETH	0.51 (0.21)	0.54	0.49	0.48	0.80	0.73	170	162	20
RNN-i	ETH	0.54 (0.21)	0.76	0.39	0.48	0.85	0.44	68	184	100
RNN-o	ETH	0.54 (0.21)	0.40	0.47	0.47	0.64	0.76	205	127	20
SORT	ETH	0.67 (0.29)	0.82	0.57	0.50	0.87	0.61	130	175	47
SORT-i	ETH	0.66 (0.26)	0.84	0.55	0.49	0.86	0.56	136	129	87
SORT-o	ETH	0.67 (0.29)	0.79	0.68	0.49	0.80	0.70	167	148	37
KSP	Hotel	0.44 (0.14)	0.33	0.65	0.32	0.48	0.94	270	40	6
KSP-i	Hotel	0.53 (0.17)	0.38	0.75	0.33	0.47	0.94	273	35	8
KSP-o	Hotel	0.53 (0.17)	0.38	0.77	0.30	0.46	0.94	276	32	8
MDP	Hotel	0.40 (0.12)	0.34	0.46	0.33	0.47	0.64	133	92	91
MDP-i	Hotel	0.50 (0.13)	0.43	0.37	0.38	0.60	0.52	83	110	123
MDP-o	Hotel	0.37 (0.10)	0.28	0.47	0.30	0.40	0.67	143	105	68
RNN	Hotel	0.40 (0.14)	0.30	0.58	0.39	0.46	0.90	252	45	19
RNN-i	Hotel	0.40 (0.14)	0.30	0.59	0.39	0.46	0.90	258	38	20
RNN-o	Hotel	0.39 (0.13)	0.29	0.56	0.38	0.46	0.90	256	41	19
SORT	Hotel	0.54 (0.20)	0.45	0.68	0.37	0.55	0.82	207	87	22
SORT-i	Hotel	0.60 (0.20)	0.46	0.78	0.47	0.52	0.90	240	60	16
SORT-o	Hotel	0.58 (0.20)	0.46	0.78	0.35	0.53	0.88	238	64	14
KSP	Station	0.32	0.27	0.40	0.23	0.61	0.90	10166	1985	211
KSP-i	Station	0.42	0.35	0.52	0.19	0.60	0.91	10296	1879	187
KSP-o	Station	0.40	0.32	0.53	2.27	0.55	0.92	10597	1576	189
MDP	Station	0.48	0.39	0.63	0.51	0.56	0.90	9362	2293	437
MDP-i	Station	0.47	0.36	0.65	0.52	0.51	0.92	10047	1771	544
MDP-o	Station	0.47	0.37	0.66	0.50	0.52	0.92	10010	1930	422
RNN	Station	0.30	0.24	0.37	0.40	0.58	0.90	9826	2333	203
RNN-i	Station	0.30	0.24	0.38	0.41	0.59	0.90	9900	2260	202
RNN-o	Station	0.30	0.25	0.39	0.40	0.57	0.90	9898	2265	199
SORT	Station	0.50	0.50	0.50	0.32	0.71	0.72	5557	6181	624
SORT-i	Station	0.50	0.47	0.54	0.31	0.69	0.78	6996	4882	484
SORT-o	Station	0.52	0.48	0.57	0.31	0.67	0.79	7154	4703	505
KSP	Rene	0.48	0.48	0.49	0.31	0.89	0.38	11	3	13
KSP-i	Rene	0.55	0.69	0.49	0.36	0.65	0.47	15	8	4
MDP	Rene	0.66	0.55	0.63	0.52	0.58	0.51	13	7	7
MDP-i	Rene	0.69	0.57	0.67	0.54	0.58	0.53	17	6	4
RNN	Rene	0.54	0.54	0.52	0.40	0.77	0.43	12	3	12
RNN-i	Rene	0.55	0.54	0.53	0.40	0.77	0.43	14	7	6
SORT	Rene	0.59	0.63	0.27	0.48	0.24	0.61	4	7	16
SORT-i	Rene	0.66	0.77	0.31	0.49	0.24	0.77	14	9	4

Table A.5 – Full results for all methods on all the datasets except **Towncentre** and **DukeMTMC**, when using our detections as input and using the results of state-of-the-art trackers as input. Number in brackets in **IDF₁** column indicates result for the distance of 1 m.

A.2. Additional results for Chapter 4

Method	Sequence	IDF ₁	IDPR	IDRC	MOTA	MOTP	MT	ML	IDs	Frag
BIPCC	Easy-1	57.3	91.2	41.8	43.0	79.0	24	46	39	75
BIPCC-i	Easy-1	57.8	91.9	42.2	42.9	79.0	24	46	41	75
BIPCC	Easy-2	68.2	69.3	67.1	44.8	78.2	133	38	60	184
BIPCC-i	Easy-2	69.2	70.4	68.0	44.7	78.2	133	39	52	172
BIPCC	Easy-3	60.3	78.9	48.8	57.8	77.5	52	22	16	36
BIPCC-i	Easy-3	59.8	78.2	48.4	57.8	77.5	52	22	19	36
BIPCC	Easy-4	73.5	88.7	62.8	63.2	80.2	36	18	7	20
BIPCC-i	Easy-4	76.0	91.7	64.9	63.2	80.2	36	18	9	20
BIPCC	Easy-5	73.2	83.0	65.4	72.8	80.4	107	17	54	139
BIPCC-i	Easy-5	73.3	83.0	65.6	72.6	80.4	107	17	46	132
BIPCC	Easy-6	77.2	87.5	69.1	73.4	80.2	142	27	55	127
BIPCC-i	Easy-6	80.9	91.7	72.4	73.4	80.2	142	27	58	127
BIPCC	Easy-7	80.5	93.6	70.6	71.4	74.7	69	13	23	86
BIPCC-i	Easy-7	80.5	93.6	70.6	71.4	74.7	69	13	23	86
BIPCC	Easy-8	72.4	92.2	59.6	60.7	76.7	102	53	46	134
BIPCC-i	Easy-8	72.7	92.2	60.0	60.9	76.6	103	52	42	135
BIPCC	Hard-1	52.7	92.5	36.8	37.8	78.1	6	34	55	103
BIPCC-i	Hard-1	52.5	91.9	36.7	37.4	78.1	6	35	61	106
BIPCC	Hard-2	60.6	65.7	56.1	47.3	76.5	68	12	194	298
BIPCC-i	Hard-2	61.0	66.0	56.7	46.6	76.5	66	12	194	291
BIPCC	Hard-3	62.7	96.1	46.5	46.7	77.9	24	4	6	12
BIPCC-i	Hard-3	62.7	96.1	46.5	46.7	77.9	24	4	6	12
BIPCC	Hard-4	84.3	86.0	82.7	85.3	81.5	21	0	1	9
BIPCC-i	Hard-4	92.3	93.6	91.0	85.5	81.4	21	0	2	9
BIPCC	Hard-5	81.9	90.1	75.1	78.3	80.7	57	2	13	37
BIPCC-i	Hard-5	81.9	90.1	75.1	78.3	80.7	57	2	13	37
BIPCC	Hard-6	64.1	81.7	52.7	59.4	76.7	85	23	225	369
BIPCC-i	Hard-6	64.7	82.4	53.3	59.4	76.7	85	23	230	369
BIPCC	Hard-7	59.6	81.2	47.1	50.8	73.3	43	23	148	218
BIPCC-i	Hard-7	59.8	81.4	47.2	50.6	73.3	42	23	145	203
BIPCC	Hard-8	82.4	94.9	72.8	73.0	75.9	34	5	10	27
BIPCC-i	Hard-8	82.4	94.9	72.8	73.0	75.9	34	5	10	27

Table A.6 – Full results of comparison on **DukeMTMC** dataset, compared to results of **BIPCC**.

Appendix A. An appendix

Method	Dataset	IDF ₁	IDPR	IDRC	MOTA	PR	RC	MT	PT	ML
KSP	Town	0.56 (0.47)	0.55	0.57	0.87	0.93	0.97	226	8	12
MDP	Town	0.87 (0.84)	0.92	0.82	0.87	0.99	0.89	184	38	24
RNN	Town	0.65 (0.57)	0.65	0.65	0.85	0.95	0.95	222	19	5
SORT	Town	0.88 (0.85)	0.93	0.84	0.90	1.00	0.90	203	34	9
OUR	Town	0.97 (0.92)	0.97	0.97	0.98	1.00	1.00	245	1	0
KSP	ETH	0.59 (0.12)	0.58	0.60	0.70	0.87	0.89	287	56	9
MDP	ETH	0.89 (0.18)	0.91	0.87	0.85	0.95	0.91	300	42	10
RNN	ETH	0.65 (0.16)	0.64	0.65	0.73	0.89	0.90	289	62	1
SORT	ETH	0.93 (0.20)	0.98	0.88	0.85	0.97	0.87	307	31	14
OUR	ETH	0.92 (0.19)	0.92	0.92	0.94	0.98	0.98	347	5	0
KSP	Hotel	0.60 (0.21)	0.61	0.58	0.74	0.90	0.86	217	69	30
MDP	Hotel	0.85 (0.33)	0.87	0.83	0.84	0.95	0.90	249	37	30
RNN	Hotel	0.70 (0.28)	0.69	0.71	0.78	0.91	0.94	284	29	3
SORT	Hotel	0.88 (0.36)	0.97	0.81	0.82	0.99	0.83	191	107	18
OUR	Hotel	0.94 (0.38)	0.94	0.94	0.97	1.00	1.00	314	1	1
KSP	Station	0.45	0.44	0.45	0.80	0.93	0.95	10957	832	573
MDP	Station	0.75	0.70	0.80	0.68	0.81	0.93	464	67	51
RNN	Station	0.40	0.39	0.40	0.68	0.90	0.94	10870	1244	248
SORT	Station	0.72	0.85	0.63	0.70	1.00	0.74	4968	6481	913
OUR	Station	0.70	0.62	0.62	0.77	0.99	0.99	579	3	0

Table A.7 – Full results for all combinations of methods and datasets, when using our set of ground truth detections. Number in brackets in **IDF₁** column indicates result for the distance of 1 m.

Method	IDF1	IDP	IDR	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP
KSP	73.2	83.8	65.0	49	79	43	1,095	7,503	85	92	69.6	61.5
KSP+ptrack	78.4	84.4	73.1	72	74	25	2,007	5,830	103	95	72.2	60.3

Table A.8 – Benchmark tracking results on the **WILDTRACK** dataset.

Measure	Better	Perfect	Description
MOTA	higher	100	Multiple Object Tracking Accuracy [18]. This measure combines three error sources: false positives, missed targets and identity switches.
MOTP	higher	100	Multiple Object Tracking Precision [18]. The misalignment between the annotated and the predicted bounding boxes.
IDF1	higher	100	IDF [149]. The ratio of correctly identified detections over the average number of ground-truth and computed detections.
FAF	lower	0	The average number of false alarms per frame.
MT	higher	100	Mostly tracked targets. The ratio of ground-truth trajectories that are covered by a track hypothesis for at least 80% of their respective life span.
ML	lower	0	Mostly lost targets. The ratio of ground-truth trajectories that are covered by a track hypothesis for at most 20% of their respective life span.
FP	lower	0	The total number of false positives.
FN	lower	0	The total number of false negatives (missed targets).
ID Sw.	lower	0	The total number of identity switches.
Frag.	lower	0	The total number of times a trajectory is fragmented (i.e. interrupted during tracking).
Hz	higher	Inf.	Processing speed (in frames per second excluding the detector) on the benchmark.

Table A.9 – Metrics description.

A.3 Additional results for Chapter 5

Here we present metric description in Tab. A.9 and full results for all benchmarks in Tab. A.10, A.11, A.12, A.14. Legend information and results for DukeMTMC dataset and MOT15 benchmarks were collected from the benchmark website <https://motchallenge.net/> on the 6th of May, 2018 for our appearance-less methods, and on the 31st of October, 2018, for appearance-based methods.. Our tracker results are available there under the name **SAS** and **SAS_full** for DukeMTMC benchmark, **SAS_MOT15** for MOT15 benchmark, and **SAS_MOT17** for MOT17 benchmark.

We also report results of our comparison to **SORT** on the validation data we used for DukeMTMC dataset in Tab. A.13 - 15000 last frames before testing data in each camera view. We tuned the parameters of the method (*max_age*, *min_hits*, detection quality cutoff) on the same data we used for training for ablation study, using grid search.

Appendix A. An appendix

Method	IDF1	IDP	IDR	MOTA	MOTP	FAF	MT	ML	FP	FN	ID Sw.	Frag.
OURS	84.0	89.4	79.2	76.0	76.0	0.09	950	72	66,783	186,974	169	1256
MHT	80.3	87.3	74.4	78.3	78.4	0.05	914	72	35,580	193,253	406	1,116
REID	79.2	89.9	70.7	68.8	77.9	0.07	726	143	52,408	277,762	449	1,060
CDSC	77.0	87.6	68.6	70.9	75.8	0.05	740	110	38,655	268,398	693	4,717
OURS-geom	76.5	83.9	70.3	69.3	74.8	0.10	813	89	76,059	248,224	426	2,081
PTRACK	71.2	84.8	61.4	59.3	78.7	0.09	666	234	68,634	361,589	290	783
BIPCC	70.1	83.6	60.4	59.4	78.7	0.09	665	234	68,147	361,672	300	801

Table A.10 – Full benchmark results on Easy set of sequences of **DukeMTMC** dataset.

Method	IDF1	IDP	IDR	MOTA	MOTP	FAF	MT	ML	FP	FN	ID Sw.	Frag.
OURS	76.8	89.3	67.4	65.4	75.3	0.12	450	87	35,596	210,639	267	977
REID	71.6	85.3	61.7	60.9	76.8	0.14	375	104	40,732	237,974	572	993
OURS-geom	65.5	79.3	55.8	59.1	74.0	0.14	379	102	39,576	251,256	972	1,855
CDSC	65.5	81.4	54.7	59.6	75.4	0.09	348	99	26,643	260,073	1,637	5,024
PTRACK	65.0	81.8	54.0	54.4	77.1	0.14	335	104	40,978	283,704	661	1,054
BIPCC	64.5	81.2	53.5	54.6	77.1	0.14	338	103	39,599	283,376	652	1,073
MHT	63.5	73.9	55.6	59.6	76.7	0.19	400	80	55,038	231,527	1,468	1,801

Table A.11 – Full benchmark results on Hard set of sequences of **DukeMTMC** dataset.

Method	MOTA	IDF1	MT	ML	FP	FN	ID Sw.	Frag.	Hz	Hardware
OURS-geom	22.2	27.2	3.1	61.6	5,591	41,531	700	1,240	8.9	2.5 GHz CPU
SORT	21.7	26.8	3.7	49.1	8,422	38,454	1,231	2,005	1,112.1	1.8 GHz CPU
LP2D	19.8	—	6.7	41.2	11,580	36,045	1,649	1,712	112.1	2.6Hz 16 CPU
RNN	19.0	17.1	5.5	45.6	11,578	36,706	1,490	2,081	165.2	3GHz, CPU

Table A.12 – Full benchmark results on **MOT15** dataset.

Method	Cam1	Cam2	Cam3	Cam4	Cam5	Cam6	Cam7	Cam8	Overall
OURS	82.1/76.8	70.9/67.7	88.5/84.0	70.9/63.0	58.9/49.9	88.5/81.6	71.4/71.6	66.0/66.4	74.6/70.1
SORT	23.7/37.8	27.7/51.2	26.5/53.0	25.7/40.5	28.6/68.0	23.0/54.9	27.6/56.8	16.4/37.0	24.9/49.9

Table A.13 – Comparison to **SORT** method on the validation data for DukeMTMC dataset, **IDF/MOTA**.

Method	MOTA	IDF1	MT%	ML%	FP	FN	IDs	Frag	Hz
OURS	44.2	57.2	16.1	44.3	29,473	283,611	1,529	2,644	4.8
DMAN	48.2	55.7	19.3	38.3	26,218	263,608	2,194	5,378	0.3
JCC	51.2	54.5	20.9	37.0	25,937	247,822	1,802	2,984	1.8
MOTDT17	50.9	52.7	17.5	35.7	24,069	250,768	2,474	5,317	18.3
MHTBLSTM	47.5	51.9	18.2	41.7	25,981	268,042	2,069	3,124	1.9
EDMT17	50.0	51.3	21.6	36.3	32,279	247,297	2,264	3,260	0.6
FWT	51.3	47.6	21.4	35.2	24,101	247,921	2,648	4,279	0.2

Table A.14 – Full benchmark results on **MOT17** dataset.

Bibliography

- [1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-fei, and S. Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *Conference on Computer Vision and Pattern Recognition*, 2014.
- [2] S. Ali and M. Shah. Floor Fields for Tracking in High Density Crowd Scenes. In *European Conference on Computer Vision*, 2008.
- [3] M. Andriluka, U. Iqbal, A. Milan, E. Insafutdinov, L. Pishchulin, J. Gall, and B. Schiele. PoseTrack: A benchmark for human pose estimation and tracking. In *Conference on Computer Vision and Pattern Recognition*, 2018.
- [4] M. Andriluka, S. Roth, and B. Schiele. People-Tracking-By-Detection and People-Detection-By-Tracking. In *CVPR*, June 2008.
- [5] M. Andriluka, S. Roth, and B. Schiele. Monocular 3D Pose Estimation and Tracking by Detection. In *Conference on Computer Vision and Pattern Recognition*, 2010.
- [6] A. Andriyenko, K. Schindler, and S. Roth. Discrete-Continuous Optimization for Multi-Target Tracking. In *Conference on Computer Vision and Pattern Recognition*, pages 1926–1933, June 2012.
- [7] B. Babenko, M.H. Yang, and S. Belongie. Robust Object Tracking with Online Multiple Instance Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011.
- [8] S.-H. Bae and K.-J. Yoon. Robust Online Multi-Object Tracking Based on Tracklet Confidence and Online Discriminative Appearance Learning. In *Conference on Computer Vision and Pattern Recognition*, 2014.
- [9] T. Bagautdinov, A. Alahi, F. Fleuret, P. Fua, and S. Savarese. Social Scene Understanding: End-To-End Multi-Person Action Localization and Collective Activity Recognition. In *Conference on Computer Vision and Pattern Recognition*, 2017.
- [10] A. Basharat, A. Gritai, and M. Shah. Learning Object Motion Patterns for Anomaly Detection and Improved Object Detection. In *Conference on Computer Vision and Pattern Recognition*, 2008.

Bibliography

- [11] R. Benenson, O. Mohamed, J. Hosang, and B. Schiele. Ten Years of Pedestrian Detection, What Have We Learned? In *European Conference on Computer Vision*, pages 613–627, 2014.
- [12] B. Benfold and I. Reid. Guiding visual surveillance by tracking human attention. In *Conference on Computer Vision and Pattern Recognition*, 2011.
- [13] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015.
- [14] Y. Bengio, N. Léonard, and A. Courville. Estimating or Propagating Gradients through Stochastic Neurons for Conditional Computation. *arXiv preprint arXiv:1308.3432*, 2013.
- [15] 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*, 36(8):1614–1627, 2014.
- [16] J. Berclaz, F. Fleuret, and P. Fua. Multi-Camera Tracking and Atypical Motion Detection with Behavioral Maps. In *European Conference on Computer Vision*, October 2008.
- [17] 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*, 33(11):1806–1819, 2011.
- [18] K. Bernardin and R. Stiefelhagen. Evaluating Multiple Object Tracking Performance: the Clear Mot Metrics. *EURASIP Journal on Image and Video Processing*, 2008, 2008.
- [19] G. Bertasius, H. S. Park Soo, X. Y. Stella, and J. Shi. Am i a baller? basketball performance assessment from first-person videos. In *International Conference on Computer Vision*, 2017.
- [20] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Uppcroft. Simple Online and Realtime Tracking. In *International Conference on Image Processing*, 2016.
- [21] A. Bhattacharyya, M. Fritz, and B. Schiele. Long-term on-board prediction of people in traffic scenes under uncertainty. In *Conference on Computer Vision and Pattern Recognition*, 2018.
- [22] J. Black, T.J. Ellis, and P. Rosin. Multi-View Image Surveillance and Tracking. In *IEEE Workshop on Motion and Video Computing*, 2002.
- [23] S.S. Blackman. *Multiple-Target Tracking with Radar Applications*. Artech House, 1986.
- [24] L. Breiman. Random Forests. *Machine Learning*, 2001.
- [25] M.D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Robust Tracking-By-Detection Using a Detector Confidence Particle Filter. In *International Conference on Computer Vision*, pages 1515–1522, 2009.

-
- [26] M.D. 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.
- [27] W. Brendel, M. Amer, and S. Todorovic. Multiobject Tracking as Maximum Weight Independent Set. In *Conference on Computer Vision and Pattern Recognition*, 2011.
- [28] S. Calderara, U. Heinemann, A. Prati, R. Cucchiara, and N. Tishby. Detecting Anomalies in People’s Trajectories Using Spectral Graph Analysis. *Computer Vision and Image Understanding*, 2011.
- [29] B. Chakraborty and S. Meher. A Real-Time Trajectory-Based Ball Detection-And-Tracking Framework for Basketball Video. *Journal of Optics*, 42(2):156–170, 2013.
- [30] A. Charnes and W. Cooper. Programming with Linear Fractional Functionals. *Naval Research logistics quarterly*, 1962.
- [31] T. Chavdarova, P. Baqué, S. Bouquet, A. Maksai, C. Jose, L. Lettry, P. Fua, L. Van Gool, and F. Fleuret. The Wildtrack Multi-Camera Person Dataset. In *Conference on Computer Vision and Pattern Recognition*, 2018.
- [32] H.-T. Chen, M.-C. Tien, Y.-W. Chen, W.-J. Tsai, and S.-Y. Lee. Physics-Based Ball Tracking and 3D Trajectory Reconstruction with Applications to Shooting Location Estimation in Basketball Video. *Journal of Visual Communication and Image Representation*, 20:204–216, 2009.
- [33] H.-T. .-S. Chen and S.-Y. Lee. Physics-Based Ball Tracking in Volleyball Videos with Its Applications to Set Type Recognition and Action Detection. In *International Conference on Acoustics, Speech, and Signal Processing*, 2007.
- [34] J. Chen, H. Sheng, Y. Zhang, and Z. Xiong. Enhancing detection model for multiple hypothesis tracking. In *Conference on Computer Vision and Pattern Recognition Workshops*, 2017.
- [35] L. Chen, H. Ai, C. Shang, Z. Zhuang, and B. Bai. Online multi-object tracking with convolutional neural networks. In *International Conference on Image Processing*, 2017.
- [36] W. Choi. Near-Online Multi-Target Tracking with Aggregated Local Flow Descriptor. In *International Conference on Computer Vision*, 2015.
- [37] W. Choi and S. Savarese. A Unified Framework for Multi-Target Tracking and Collective Activity Recognition. In *European Conference on Computer Vision*, 2012.
- [38] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu. Online Multi-Object Tracking Using CNN-based Single Object Tracker with Spatial-Temporal Attention Mechanism. In *International Conference on Computer Vision*, 2017.

Bibliography

- [39] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.
- [40] A. Dehghan, S. Modiri Assari, and M. Shah. Gmmcp Tracker: Globally Optimal Generalized Maximum Multi Clique Problem for Multiple Object Tracking. In *Conference on Computer Vision and Pattern Recognition*, 2015.
- [41] C. Dicle, O. I Camps, and M. Sznaier. The way they move: Tracking multiple targets with similar appearance. In *International Conference on Computer Vision*, 2013.
- [42] C. Direkoğlu and N.E. O’Connor. Team Activity Recognition in Sports. In *European Conference on Computer Vision*, pages 69–83, October 2012.
- [43] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian Detection: An Evaluation of the State of the Art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.
- [44] X. Dong, J. Shen, W. Wang, Y. Liu, L. Shao, and F. Porikli. Hyperparameter optimization for tracking with continuous deep q-learning. In *Conference on Computer Vision and Pattern Recognition*, 2018.
- [45] 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*, pages 559–564, 2009.
- [46] A. Ess, B. Leibe, K. Schindler, and L. V. Gool. A mobile vision system for robust multi-person tracking. In *Conference on Computer Vision and Pattern Recognition*, 2008.
- [47] M. Fabbri, F. Lanzi, S. Calderara, A. Palazzi, R. Vezzani, and R. Cucchiara. Learning to detect and track visible and occluded body joints in a virtual world. *arXiv preprint arXiv:1803.08319*, 2018.
- [48] L. Fagot-bouquet, R. Audigier, Y. Dhome, and F. Lerasle. Improving Multi-Frame Data Association with Sparse Representations for Robust Near-Online Multi-Object Tracking. In *European Conference on Computer Vision*, pages 774–790, October 2016.
- [49] P. Felzenszwalb, D. Mcallester, and D. Ramanan. A Discriminatively Trained, Multiscale, Deformable Part Model. In *Conference on Computer Vision and Pattern Recognition*, June 2008.
- [50] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.
- [51] J. Ferryman and A. Shahrokni. Pets2009: Dataset and challenge. In *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 1–6. IEEE, 2009.

-
- [52] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multi-Camera People Tracking with a Probabilistic Occupancy Map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):267–282, February 2008.
- [53] E. Fontaine, A. H. Barr, and J.W. Burdick. Model-based tracking of multiple worms and fish. In *International Conference on Computer Vision Workshops*, 2007.
- [54] A. Fossati, P. Schoenmann, and P. Fua. Real-Time Vehicle Tracking for Driving Assistance. *Machine Vision and Applications*, 22(2):439–448, 2011.
- [55] K. Fragkiadaki, S. Levine, Jitendra P. F, and J. Malik. Recurrent Network Models for Human Dynamics. In *International Conference on Computer Vision*, 2015.
- [56] 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.
- [57] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun. 3d traffic scene understanding from movable platforms. 2014.
- [58] 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.
- [59] J. Giebel, D.M. Gavrilu, and C. Schnorr. A Bayesian Framework for Multi-Cue 3D Object Tracking. In *European Conference on Computer Vision*, 2004.
- [60] G. Gomez, P. López, D. Link, and B. Eskofier. Tracking of Ball and Players in Beach Volleyball Videos. *PloS one*, 2014.
- [61] 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*, pages 2012–2019, 2009.
- [62] R. A. Hadi, G. Sulong, and L. E. George. Vehicle Detection and Tracking Techniques: A Concise Review. *arXiv preprint arXiv:1410.5894*, 2014.
- [63] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [64] M.A. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support Vector Machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.
- [65] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, 2016.
- [66] R. Henschel, L. Leal-Taixé, D. Cremers, and B. Rosenhahn. Fusion of head and full-body detectors for multi-object tracking. In *Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

Bibliography

- [67] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.
- [68] T. Hill and P. Lewicki. STATISTICS: Methods and Applications. Technical report, 2007.
- [69] M.. Hu, S. Ali, and M. Shah. Detecting global motion patterns in complex videos. In *International Conference on Pattern Recognition*, 2008.
- [70] W. Hu, T. Tan, L. Wang, and S. Maybank. A Survey on Visual Surveillance of Object Motion and Behaviors. *IEEE Transactions on Systems, Man, and Cybernetics*, 2004.
- [71] J. Supancic III and D. Ramanan. Tracking as Online Decision-Making: Learning a Policy from Streaming Videos with Reinforcement Learning. In *International Conference on Computer Vision*, 2017.
- [72] E. Insafutdinov, M. Andriluka, L. Pishchulin, S. Tang, E. Levinkov, B. Andres, and B. Schiele. Arttrack: Articulated multi-person tracking in the wild. In *Conference on Computer Vision and Pattern Recognition*, 2017.
- [73] U. Iqbal, A. Milan, and J. Gall. Posetrack: Joint Multi-Person Pose Estimation and Tracking. In *Conference on Computer Vision and Pattern Recognition*, 2017.
- [74] S. Iwase and H. Saito. Parallel Tracking of All Soccer Players by Integrating Detected Positions in Multiple View Images. In *International Conference on Pattern Recognition*, pages 751–754, August 2004.
- [75] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.
- [76] F. Jelinek, L. Bahl, and R. Mercer. Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech. *IEEE Transactions on Information Theory*, 1975.
- [77] H. Jeong., Y. Yoo., K.M. Yi, and J.Y. Choi. Two-Stage Online Inference Model for Traffic Pattern Analysis and Anomaly Detection. *Machine vision and applications*, 2014.
- [78] H. Jiang, S. Fels, and J.J. Little. A Linear Programming Approach for Multiple Object Tracking. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.
- [79] N. Jiang, S. Bai, Y. Xu, C. Xing, Z. Zhou, and W. Wu. Online inter-camera trajectory association exploiting person re-identification and camera topology. In *ACM Multimedia Conference*, 2018.
- [80] S. Jin, X. Ma, Z. Han, Y. Wu, W. Yang, W. Liu, C. Qian, and W. Ouyang. Towards multi-person pose tracking: Bottom-up and top-down methods. In *International Conference on Computer Vision Workshops*, 2017.

-
- [81] J.-P. Jodoin, G.-A. Bilodeau, and N. Saunier. Urban Tracker: Multiple Object Tracking in Urban Mixed Traffic. In *IEEE Winter Conference on Applications of Computer Vision*, 2014.
- [82] S. W. Joo and R. Chellappa. A Multiple-Hypothesis Approach for Multiobject Visual Tracking. *IEEE Transactions on Image Processing*, 2007.
- [83] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M. Yang. Online multi-object tracking with dual matching attention networks. In *European Conference on Computer Vision*, 2018.
- [84] M. Kalayeh, S. Mussmann, A. Petrakova, N. Lobo, and M. Shah. Understanding Trajectory Behavior: A Motion Pattern Approach. *arXiv preprint arXiv:1501.00614*, 2015.
- [85] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, M. Boonstra, V. Kozhova, 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*, 31(2):319–336, 2009.
- [86] M. Keuper, S. Tang, B. Andres, T. Brox, and B. Schiele. Motion segmentation & multiple object tracking by correlation co-clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
- [87] M. Keuper, S. Tang, Y. Zhongjie, B. Andres, T. Brox, and B. Schiele. A Multi-Cut Formulation for Joint Segmentation and Tracking of Multiple Objects. *arXiv preprint arXiv:1607.06317*, 2016.
- [88] Z. Khan, T. Balch, and F. Dellaert. MCMC-Based Particle Filtering for Tracking a Variable Number of Interacting Targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1805–1918, 2005.
- [89] C. Kim, F. Li, A. Ciptadi, and J. Rehg. Multiple Hypothesis Tracking Revisited. In *International Conference on Computer Vision*, 2015.
- [90] C. Kim, F. Li, and J. M. Rehg. Multi-object tracking with neural gating using bilinear lstm. In *European Conference on Computer Vision*, 2018.
- [91] J. Kim and K. Grauman. Observe Locally, Infer Globally: A Space-Time MRF for Detecting Abnormal Activities with Incremental Updates. In *Conference on Computer Vision and Pattern Recognition*, 2009.
- [92] 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.
- [93] Y. J. Koh and C.-S. Kim. CDTS: Collaborative Detection, Tracking, and Segmentation for Online Multiple Object Segmentation in Videos. In *International Conference on Computer Vision*, 2017.

Bibliography

- [94] L. Kratz and K. Nishino. Going with the Flow: Pedestrian Efficiency in Crowded Scenes. In *European Conference on Computer Vision*, 2012.
- [95] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955.
- [96] C.-H. Kuo, C. Huang, and R. Nevatia. Multi-Target Tracking by On-Line Learned Discriminative Appearance Models. In *Conference on Computer Vision and Pattern Recognition*, 2010.
- [97] C.-H Kuo and R. Nevatia. How Does Person Identity Recognition Help Multi-Person Tracking? In *Conference on Computer Vision and Pattern Recognition*, 2011.
- [98] 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.
- [99] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler. Learning by tracking: Siamese CNN for robust target association. In *Conference on Computer Vision and Pattern Recognition Workshops*, 2016.
- [100] L. Leal-taixé, M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese. Learning an Image-Based Motion Context for Multiple People Tracking. In *CVPR*, 2014.
- [101] L. Leal-taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. Motchallenge 2015: Towards a Benchmark for Multi-Target Tracking. In *ARXIV*, 2015.
- [102] L. Leal-Taixé, A. Milan, K. Schindler, D. Cremers, I. Reid, and S. Roth. Tracking the trackers: an analysis of the state of the art in multiple object tracking. *arXiv preprint arXiv:1704.02781*, 2017.
- [103] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. HS Torr, and M. Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Conference on Computer Vision and Pattern Recognition*, 2017.
- [104] B. Leibe, K. Schindler, and L. Van Gool. Coupled Detection and Trajectory Estimation for Multi-Object Tracking. In *International Conference on Computer Vision*, October 2007.
- [105] P. Lenz, A. Geiger, and R. Urtasun. Followme: Efficient Online Min-Cost Flow Tracking with Bounded Memory and Computation. In *International Conference on Computer Vision*, pages 4364–4372, December 2015.
- [106] M. Leo, N. Mosca, P. Spagnolo, P.L. 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.
- [107] 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*, 12(5):546–566, 2008.

-
- [108] W. Li, V. Mahadevan, and N. Vasconcelos. Anomaly Detection and Localization in Crowded Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.
- [109] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A.V.D. Hengel. A Survey of Appearance Models in Visual Object Tracking. *ACM Transactions on Intelligent Systems and Technology*, 2013.
- [110] Y. Li, C. Huang, and R. Nevatia. Learning to Associate: Hybridboosted Multi-Target Tracker for Crowded Scene. In *Conference on Computer Vision and Pattern Recognition*, June 2009.
- [111] Z. Lin, H. Zheng, B. Ke, and L. Chen. Online multi-object tracking based on hierarchical association and sparse representation. In *International Conference on Image Processing*, 2017.
- [112] 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*, pages 1830–1837, 2013.
- [113] C. Long, A. Haizhou, Z. Zijie, and S. Chong. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In *International Conference on Multimedia and Expo*, 2018.
- [114] 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.
- [115] W. Luo, J. Xing, X. Zhang, X. Zhao, and T.-K. Kim. Multiple Object Tracking: A Literature Review. 2015.
- [116] W. Luo, Y. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Conference on Computer Vision and Pattern Recognition*, 2018.
- [117] C. Ma, C. Yang, F. Yang, Y. Zhuang, Z. Zhang, H. Jia, and X. Xie. Trajectory Factory: Tracklet Cleaving and Re-connection by Deep Siamese Bi-GRU for Multiple Object Tracking. In *International Conference on Multimedia and Expo*, 2018.
- [118] W.-C. Ma, D.-A. Huang, N. Lee, and K. M. Kitani. Forecasting interactive dynamics of pedestrians with fictitious play. In *Conference on Computer Vision and Pattern Recognition*, 2017.
- [119] D. R. Magee. Tracking Multiple Vehicles Using Foreground, Background and Motion Models. *Image and Vision Computing*, 22(2):143–155, February 2004.
- [120] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos. Anomaly Detection in Crowded Scenes. In *Conference on Computer Vision and Pattern Recognition*, 2010.

Bibliography

- [121] A. Maksai, X. Wang, F. Fleuret, and P. Fua. Globally Consistent Multi-People Tracking Using Motion Patterns. In *International Conference on Computer Vision*, 2017.
- [122] A. Maksai, X. Wang, and P. Fua. What Players Do with the Ball: A Physically Constrained Interaction Modeling. In *Conference on Computer Vision and Pattern Recognition*, 2016.
- [123] S. Manen, M. Gygli, D. Dai, and L. Van Gool. PathTrack: Fast Trajectory Annotation with Path Supervision. In *International Conference on Computer Vision*, 2017.
- [124] T. Mauthner, M. Donoser, and H. Bischof. Robust Tracking of Spatial Related Components. In *International Conference on Pattern Recognition*, 2008.
- [125] A. Milan, L. Leal-taixe, I. Reid, S. Roth, and K. Schindler. Mot16: A Benchmark for Multi-Object Tracking. *arXiv preprint arXiv:1603.00831*, 2016.
- [126] A. Milan, L. Leal-taixe, K. Schindler, and I. Reid. Joint Tracking and Segmentation of Multiple Targets. In *CVPR*, 2015.
- [127] A. Milan, S.H. Rezatofighi, A. Dick, I. Reid, and K. Schindler. Online Multi-Target Tracking Using Recurrent Neural Networks. In *American Association for Artificial Intelligence Conference*, 2017.
- [128] A. Milan, S. Roth, and K. Schindler. Continuous Energy Minimization for Multitarget Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:58–72, 2014.
- [129] A. Mittal and L. Davis. M2Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene. *International Journal of Computer Vision*, 51(3):189–203, 2003.
- [130] D. Mitzel and B. Leibe. Real-time multi-person tracking with detector assisted structure propagation. In *International Conference on Computer Vision Workshops*, 2011.
- [131] S. Murray. Real-Time Multiple Object Tracking-A Study on the Importance of Speed. *arXiv preprint arXiv:1709.03572*, 2017.
- [132] S. Oh, S. Russell, and S. Sastry. Markov Chain Monte Carlo Data Association for Multi-Target Tracking. *IEEE Transactions on Automatic Control*, 2009.
- [133] 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.
- [134] K. Okuma, A. Taleghani, N. de Freitas, J.J. Little, and D.G. Lowe. A Boosted Particle Filter: Multitarget Detection and Tracking. In *European Conference on Computer Vision*, May 2004.
- [135] P. Ondruska, J. Dequaire, D.Z. Wang, and I. Posner. End-To-End Tracking and Semantic Segmentation using Recurrent Neural Networks. In *Workshop on limits and potentials of Deep Learning in Robotics*, 2016.

- [136] P. Ondruska and I. Posner. Deep tracking: Seeing beyond seeing using recurrent neural networks. In *American Association for Artificial Intelligence Conference*, 2016.
- [137] Gurobi Optimization. Inc., “gurobi optimizer reference manual,” 2015. URL: <http://www.gurobi.com>, 2014.
- [138] P. Parisot and C. De Vleeschouwer. Graph-Based Filtering of Ballistic Trajectory. In *International Conference on Multimedia and Expo*, 2011.
- [139] P. Parisot and C. De Vleeschouwer. Consensus-Based Trajectory Estimation for Ball Detection in a Calibrated Cameras System. *EURASIP Journal on Image and Video Processing*, 2015.
- [140] 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.
- [141] S. Pellegrini, A. Ess, and L. Van Gool. Improving Data Association by Joint Modeling of Pedestrian Trajectories and Groupings. In *European Conference on Computer Vision*, 2010.
- [142] C. Piciarelli, G. Foresti, and L. Snidaro. Trajectory Clustering and Its Applications for Video Surveillance. In *Advanced Video and Signal Based Surveillance. IEEE Conference on*, 2005.
- [143] 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*, pages 1201–1208, June 2011.
- [144] Z. Qin and C. Shelton. Improving Multi-Target Tracking via Social Grouping. In *Conference on Computer Vision and Pattern Recognition*, pages 1972–1978, June 2012.
- [145] M. A. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, 2015.
- [146] M. Ravanbakhsh, M. Nabi, H. Mousavi, E. Sangineto, and N. Sebe. Plug-And-Play CNN for Crowd Motion Analysis: An Application in Abnormal Event Detection. *arXiv preprint arXiv:1610.00307*, 2016.
- [147] 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*, 18(3):350–362, 2008.
- [148] S. Hamid Rezaatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick, and I. Reid. Joint Probabilistic Data Association Revisited. In *International Conference on Computer Vision*, 2015.

Bibliography

- [149] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking. 2016.
- [150] E. Ristani and C. Tomasi. Features for Multi-Target Multi-Camera Tracking and Re-Identification. In *Conference on Computer Vision and Pattern Recognition*, 2018.
- [151] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European Conference on Computer Vision*, pages 549–565. Springer, 2016.
- [152] M. Rodriguez, I. Laptev, J. Sivic, and J. Audibert. Density-Aware Person Detection and Tracking in Crowds. In *International Conference on Computer Vision*, pages 2423–2430, 2011.
- [153] M. Rodriguez, J. Sivic, I. Laptev, and J.-Y. Audibert. Data-driven crowd analysis in videos. In *International Conference on Computer Vision*, 2011.
- [154] P. Rosello and M. J. Kochenderfer. Multi-agent reinforcement learning for multi-object tracking. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018.
- [155] D.A. Ross, S. Osindero, and R.S. Zemel. Combining Discriminative Features to Infer Complex Trajectories. In *International Conference on Machine Learning*, 2006.
- [156] A. Sadeghian, A. Alahi, and S. Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *International Conference on Computer Vision*, 2017.
- [157] R. Sanchez-matilla, F. Poiesi, and A. Cavallaro. Online Multi-Target Tracking with Strong and Weak Detections. In *European Conference on Computer Vision*, 2016.
- [158] V. K. Singh, B. Wu, and R. Nevatia. Pedestrian Tracking by Associating Tracklets Using Detection Residuals. *IEEE Workshop on Motion and Video Computing*, pages 1–8, 2008.
- [159] 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.
- [160] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Subgraph Decomposition for Multi-Target Tracking. In *Conference on Computer Vision and Pattern Recognition*, pages 5033–5041, 2015.
- [161] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Multi-Person Tracking by Multicut and Deep Matching. In *European Conference on Computer Vision*, 2016.
- [162] Y. T. Tesfaye, E. Zemene, A. Prati, M. Pelillo, and M. Shah. Multi-target tracking in multiple non-overlapping cameras using constrained dominant sets. *arXiv preprint arXiv:1706.06196*, 2017.

-
- [163] J. Varadarajan, R. Emonet, and J. Odobez. A Sequential Topic Model for Mining Recurrent Activities from Long Term Video Logs. *International Journal of Computer Vision*, 2013.
- [164] D. Varshneya and G. Srinivasaraghavan. Human trajectory prediction using spatially aware deep attention models. *arXiv preprint arXiv:1705.09436*, 2017.
- [165] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Conference on Computer Vision and Pattern Recognition*, 2015.
- [166] C. De Vleeschouwer, F. Chen, D. Delannay, C. Parisot, C. Chaudy, E. Martrou, A. Cavalario, et al. Distributed Video Acquisition and Annotation for Sport-Event Summarization. *New European Media*, 8, 2008.
- [167] B. Wang, G. Wang, K. Luk Chan, and L. Wang. Tracklet Association with Online Target-Specific Metric Learning. In *Conference on Computer Vision and Pattern Recognition*, 2014.
- [168] B. Wang, G. Wang, K. Luk Chan, and L. Wang. Tracklet Association by Online Target-Specific Metric Learning and Coherent Dynamics Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.
- [169] B. Wang, L. Wang, B. Shuai, Z. Zuo, T. Liu, K. Luk Chan, and G. Wang. Joint Learning of Convolutional Neural Networks and Temporally Constrained Metrics for Tracklet Association. In *Conference on Computer Vision and Pattern Recognition*, 2016.
- [170] X. Wang. Intelligent Multi-Camera Video Surveillance: A Review. *Pattern Recognition*, 2013.
- [171] X. Wang, V.H. 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*, 119:102–115, 2014.
- [172] X. Wang, E. Turetken, F. Fleuret, and P. Fua. Tracking Interacting Objects Optimally Using Integer Programming. In *European Conference on Computer Vision*, September 2014.
- [173] X. Wang, E. Turetken, F. Fleuret, and P. Fua. Tracking Interacting Objects Using Intertwined Flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2312–2326, 2016.
- [174] Xiaogang Wang et al. *Learning motion patterns using hierarchical bayesian models*. PhD thesis, 2009.
- [175] S. E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional Pose Machines. In *Conference on Computer Vision and Pattern Recognition*, 2016.
- [176] L.R. Williams and D.W. Jacobs. Stochastic Completion Fields: A Neural Model of Illusory Contour Shape and Saliency. *Neural Computation*, 9(4):837–858, 1997.

Bibliography

- [177] B. Wu and R. Nevatia. Detection of Multiple, Partially Occluded Humans in a Single Image by Bayesian Combination of Edgelet Part Detectors. In *International Conference on Computer Vision*, 2005.
- [178] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.
- [179] J. Xiang, G. Zhang, J. Hou, N. Sang, and R. Huang. Multiple Target Tracking by Learning Feature Representation and Distance Metric Jointly. *arXiv preprint arXiv:1802.03252*, 2018.
- [180] Y. Xiang, A. Alahi, and S. Savarese. Learning to Track: Online Multi-Object Tracking by Decision Making. In *Conference on Computer Vision and Pattern Recognition*, 2015.
- [181] M. Xu, J. Orwell, and G. Jones. Tracking Football Players with Multiple Cameras. In *International Conference on Image Processing*, pages 2909–2912, October 2004.
- [182] Y. Xu, Z. Piao, and S. Gao. Encoding crowd interaction with deep neural network for pedestrian trajectory prediction. In *Conference on Computer Vision and Pattern Recognition*, 2018.
- [183] B. Yang, C. Huang, and R. Nevatia. Learning Affinities and Dependencies for Multi-Target Tracking Using a CRF Model. In *Conference on Computer Vision and Pattern Recognition*, pages 1233–1240, 2011.
- [184] C. Yang, R. Duraiswami, and L. Davis. Fast Multiple Object Tracking via a Hierarchical Particle Filter. In *International Conference on Computer Vision*, 2005.
- [185] M. Yang and Y. Jia. Temporal Dynamic Appearance Modeling for Online Multi-Person Tracking. *Computer Vision and Image Understanding*, 2016.
- [186] M. Yang, F. Lv, W. Xu, and Y. Gong. Detection driven adaptive multi-cue integration for multiple human tracking. In *International Conference on Computer Vision*, 2009.
- [187] S. Yi, H. Li, and X. Wang. Pedestrian Behavior Understanding and Prediction with Deep Neural Networks. In *European Conference on Computer Vision*, pages 263–279, 2016.
- [188] S. Yoo, K. Yun, J. Y. Choi, K. Yun, and J. Choi. Action-decision networks for visual tracking with deep reinforcement learning. In *Conference on Computer Vision and Pattern Recognition*, 2017.
- [189] J. Hong Yoon, C.-R. Lee, M.-H. Yang, and K.-J. Yoon. Online Multi-Object Tracking via Structural Constraint Event Aggregation. In *Conference on Computer Vision and Pattern Recognition*, 2016.
- [190] K. Yoon, Y. m. Song, and M. Jeon. Multiple hypothesis tracking algorithm for multi-target multi-camera tracking with disjoint views. *IET Image Processing*, 2018.

- [191] S. Yu, D. Meng, W. Zuo, and A. Hauptmann. The Solution Path Algorithm for Identity-Aware Multi-Object Tracking. In *Conference on Computer Vision and Pattern Recognition*, 2016.
- [192] E. Zelniker, S. Gong, and T. Xiang. Global Abnormal Behaviour Detection Using a Network of CCTV Cameras. In *International Workshop on Visual Surveillance*, 2008.
- [193] M. Zhai, M. J. Roshtkhari, and G. Mori. Deep learning of appearance models for online object tracking. *arXiv preprint arXiv:1607.02568*, 2016.
- [194] D. Zhang, H. Maei, X. Wang, and Y.-F. Wang. Deep reinforcement learning for visual object tracking in videos. *arXiv preprint arXiv:1701.08936*, 2017.
- [195] K. Zhang, L. Zhang, and M.H. Yang. Real-Time Compressive Tracking. In *European Conference on Computer Vision*, 2012.
- [196] 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.
- [197] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang, and J. Sun. Alignedreid: Surpassing human-level performance in person re-identification. *arXiv preprint arXiv:1711.08184*, 2017.
- [198] Y. Zhang, H. Lu, and C. Xu. Collaborate Ball and Player Trajectory Extraction in Broadcast Soccer Video. In *International Conference on Pattern Recognition*, 2008.
- [199] Z. Zhang, J. Wu, lx. Zhang, and C. Zhang. Multi-Target, Multi-Camera Tracking by Hierarchical Clustering: Recent Progress on DukeMTMC Project. *arXiv preprint arXiv:1712.09531*, 2017.
- [200] Y. Zheng. Trajectory Data Mining: An Overview. *ACM Transactions on Intelligent Systems and Technology*, 2015.
- [201] B. Zhou, X. Wang, and X. Tang. Understanding Collective Crowd Behaviors: Learning a Mixture Model of Dynamic Pedestrian-Agents. In *Conference on Computer Vision and Pattern Recognition*, 2012.
- [202] G. Zhu, Q. Huang, C. Xu, Y. Rui, S. Jiang, W. Gao, and H. Yao. Trajectory Based Event Tactics Analysis in Broadcast Sports Video. In *ACM Multimedia*, pages 58–67, 2007.

ANDRII MAKSAI

Lausanne, Switzerland | amaksay@gmail.com

EDUCATION

Ph.D. in Computer Science *in progress* **Sept 2014 – Nov 2018**

Swiss Federal Institute of Technology (EPFL), Switzerland

Working on tracking multiple objects and understanding their behaviors in video sequences under the supervision of Prof. [Pascal Fua](#) in Computer Vision laboratory.

M.S. in Applied Mathematics with honors **Sept 2012 – July 2014**

Taras Shevchenko National University of Kyiv, Ukraine

GPA: 4.9 out of 5

Thesis: Improving tumor classification using peeling with convex structures.

Description: Developed an algorithm for cancer detection via classification based on the images of patient cells. Made analysis of approaches for intersecting clusters, introduced the convex peeling solution, and improved the accuracy by 8% compared to traditional approach.

B.S. in Applied Mathematics with honors **Sept 2008 – July 2012**

Taras Shevchenko National University of Kyiv, Ukraine

GPA: 4.8 out of 5

PUBLICATIONS

The WILDTRACK Multi-Camera Person Dataset **CVPR'18**

Tatjana Chavdarova, Pierre Baqué, Stéphane Bouquet, Andrii Maksai, Cijo Jose, Louis Lettry, Pascal Fua, Luc Van Gool, François Fleuret

Non-Markovian Globally Consistent Multi-Object Tracking **ICCV'17**

Andrii Maksai, Xinchao Wang, François Fleuret, Pascal Fua

What Players do with the Ball: A Physically Constrained Interaction Modeling **CVPR'16**

Andrii Maksai, Xinchao Wang, Pascal Fua

Predicting Online Performance of News Recommender Systems

Through Richer Evaluation Metrics **RecSys'15**

Andrii Maksai, Florent Garcin, Boi Faltings

Hierarchical Incident Ticket Classification with Minimal Supervision **ICDM'14**

Andrii Maksai, Jasmina Bogojeska, Dorothea Wiesmann

Reviewer for several journals and conferences: PAMI, ICCV workshops, CVPR

WORK EXPERIENCE

Software Engineering Intern, Google Research, Switzerland **Jun – Oct 2017**

Working in the handwriting recognition team of the Machine Perception group, created a pipeline for generating realistic handwriting based on the recent successes of neural networks in this area. Extended currently existing approaches by the ability to explicitly control the style of the generated text. Developed a pipeline for generating handwriting on a production scale and showed that augmenting training data for real recognizers with generated samples shows better results than those without training data augmentation for several scripts.

Research Intern, IBM Research, Switzerland **Feb – Aug 2014**

Developed an algorithm for automatic error messages (tickets) classification with minimal supervision. Achieved state-of-the-art accuracy and effort. Implemented the algorithm for production and introduced it into the on-line server incident prediction system.

Software Engineering Intern, Microsoft, USA **July – Sept 2013**

Created Twitter social news pipeline that produces real-time trending summaries for entities. Developed aggregation module for real-time processing large amounts of updates from Twitter.com. Constructed a novel feature set, examined the performance of different ML approaches for machine-learned ranking and evaluated the quality of the system.

Software Engineering Intern, Google, Switzerland **Jan – Apr 2012**

Created a prototype for an audio-Twitter system for emerging markets users to contact each other, subscribe, create threads, leave comments and feedback using cell phones w/o Internet.

Software Engineering Intern, Facebook, USA **June – Sept 2011**

Improved user database by separating it into a distributed ring of novel database systems with consistent hashing and arbitrarily large record additions. Used: C++, PHP, Hadoop, Hive.

PROGRAMMING CONTESTS, AWARDS & ACHIEVEMENTS

ACM ICPC World Finals 2013 – Silver Medal, 7th place in the world* **July 2013**

ACM ICPC South Eastern European Regional Contest – 2nd, 1st place **2013-2012**

- More details: <http://icpc.baylor.edu/download/worldfinals/pdf/Factsheet.pdf>

IEEE IEEExtreme Programming Competition – 1st, 1st, 2nd place in the world **2017-2015**

LauzHack 2017 Hackathon – Runner Up (2nd place), Logitech Challenge Winner **Oct 2017**

- Real-time gesture recognition for AR control via webcam, <https://devpost.com/software/iron-man-manipulator>

TEACHING EXPERIENCE

Teaching Assistant, EPFL, Switzerland **Fall 2015 – Fall 2017**

- Courses: Algorithms (x2), Computer Vision, Machine Learning

Programming Contests Weekend Section Coach, Kyiv, Ukraine **Sept 2008 – May 2012**

Among the students I taught there are 12 UOI medalists and one IOI gold medalist.

LANGUAGES

English – fluent: TOEFL iBT – 111; **Ukrainian** – Native; **Russian** – Native.

SKILLS, QUALIFICATIONS & INTERESTS

Skills & qualifications: practitioner: C++, Python, Matlab, beginner: R, Java

Interests: Object tracking; Computer vision; Artificial intelligence; Recommender systems;

