UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

Methods Based on Random Finite Sets for Object Tracking in Computer Vision and Robotics

Eine vom Promotionsausschuss des Fachbereichs 4: Informatik der Universität Koblenz-Landau zur Verleihung des akademischen Grades Doktor der Naturwissenschaften (Dr. rer. nat.) genehmigte

Dissertation

von

Dipl.-Inform. Nicolai Wojke

Vorsitzende des Promotionsausschusses:	Prof. Dr. Maria A. Wimmer
Vorsitzende der Promotionskommission:	Prof. Dr. Petra Schubert
Erster Berichterstatter:	Prof. Dr. Dietrich Paulus
Zweiter Berichterstatter:	Prof. Dr. Marcin Grzegorzek
Dritter Berichterstatter:	Prof. Lennart Svensson, Ph.D.
Datum der Aussprache:	18. Juli 2018

Koblenz, 2018

Acknowledgements

It has been thirteen years since I moved to Koblenz to begin my studies at the unversity and this thesis marks the end of a long journey that has been full of excitement, learning, and experiences. I thank all the friends and colleagues that went a part of this way along with me. It has truly shaped me in many ways.

There are few people I want to thank in particular. I want to thank Dietrich Paulus for providing me with a comforting environment to grow. You gave me the freedom to explore research on my own and I have always felt well respected and heard. My gratitude goes beyond the typical academic supervision as there are many memories to remember. Second, I want to thank Frank Neuhaus for many fruitful discussions on any topic and the clear-minded, straight-forward feedback on my work. I hope I was no less of a valuable discussion partner to you. I thank Alex Bewley for his contributions and long discussions on metric learning and Raphael Memmesheimer for his contributions to the operator following method. Finally, Ana, thank you for going this way with me. We have gone long distances, and you have always coped with my working habits. On the way, your personal success has been an inspiration on its own.

Abstract

This thesis addresses the automated identification and localization of a time-varying number of objects in a stream of sensor data. The problem is challenging due to its combinatorial nature: If the number of objects is unknown, the number of possible object trajectories grows exponentially with the number of observations. Random finite sets are a relatively new theory that has been developed to derive at principled and efficient approximations. It is based around set-valued random variables that contain an unknown number of elements which appear in arbitrary order and are themselves random. While extensively studied in theory, random finite sets have not yet become a leading paradigm in practical computer vision and robotics applications.

This thesis explores random finite sets in visual tracking applications. The first method developed in this thesis combines set-valued recursive filtering with global optimization. The problem is approached in a min-cost flow network formulation, which has become a standard inference framework for multiple object tracking due to its efficiency and optimality. A main limitation of this formulation is a restriction to unary and pairwise cost terms. This circumstance makes integration of higher-order motion models challenging. The method developed in this thesis approaches this limitation by application of a Probability Hypothesis Density filter. The Probability Hypothesis Density filter was the first practically implemented state estimator based on random finite sets. It circumvents the combinatorial nature of data association itself by propagation of an object density measure that can be computed efficiently, without maintaining explicit trajectory hypotheses. In this work, the filter recursion is used to augment measurements with an additional hidden kinematic state to be used for construction of more informed flow network cost terms, e.g., based on linear motion models. The method is evaluated on public benchmarks where a considerate improvement is achieved compared to network flow formulations that are based on static features alone, such as distance between detections and appearance similarity.

A second part of this thesis focuses on the related task of detecting and tracking a single robot operator in crowded environments. Different from the conventional multiple object tracking scenario, the tracked individual can leave the scene and later reappear after a longer period of absence. Therefore, a re-identification component is required that picks up the track on re-entrance. Based on random finite sets, the Bernoulli filter is an optimal Bayes filter that provides a natural representation for this type of problem. In this work, it is shown how the Bernoulli filter can be combined with a Probability Hypothesis Density filter to track operator and non-operators simultaneously. The method is evaluated on a publicly available multiple object tracking dataset as well as on custom sequences that are specific to the targeted application. Experiments show reliable tracking in crowded scenes and robust re-identification after long-term occlusion.

Finally, a third part of this thesis focuses on appearance modeling as an essential aspect of any method that is applied to visual object tracking scenarios. Therefore, a feature representation that is robust to pose variations and changing lighting conditions is learned offline, before the actual tracking application. This thesis proposes a joint classification and metric learning objective where a deep convolutional neural network is trained to identify the individuals in the training set. At test time, the final classification layer can be stripped from the network and appearance similarity can be queried using cosine distance in representation space. This framework represents an alternative to direct metric learning objectives that have required sophisticated pair or triplet sampling strategies in the past. The method is evaluated on two large-scale person re-identification datasets where competitive results are achieved overall. In particular, the proposed method better generalizes to the test set compared to a network trained with the well-established triplet loss.

Kurzfassung

Die Problemstellung der Objektverfolgung beschäftigt sich mit dem Schätzen von Objekttrajektorien aus Sensordaten. Da in den meisten Anwendungsszenarien die Zuordnung von observierten Positionen zu in der Szene vorhandenen Objekten unbekannt ist, entsteht bei der Sensordatenverarbeitung eine exponentiell wachsende Anzahl möglicher Trajektorienhypothesen deren vollumfängliche Aufzählung und Auswertung nicht möglich ist. Random Finite Sets beschreiben einen speziell für dieses Problem entwickelten mengentheoretischen Ansatz, der mit der Zielsetzung entwickelt wurde, wohl fundierte und effiziente Approximationen herzuleiten. Dabei wird das Problem basierend auf einer mengenwertigen Problemformulierung in einer Bayes'schen Zustandsschätzung gelöst.

Die erste in dieser Arbeit entwickelte Methode kombiniert mengentheoretische Zustandsschätzung mit globaler Optimierierung im Kontext der Verfolgung einer unbekannten, zeitlich variierenden Anzahl von Objekten. Dabei wird die Trajektoriensuche als Netzwerkflussproblem formuliert, in dem Messungen zu Trajektorien verbunden werden. Netzwerkflussformulierungen sind globale Optimierungsroutinen, die aufgrund effizienter und exakter Lösungsmethoden heute zu den Standardverfahren in der Objektverfolgung gehören. Netzwerkflussprobleme sind in der Problemformulierung jedoch auf paarweise Kostenterme beschränkt. Aus diesem Grund werden Objekttransitionen zwischen Messungen traditionell durch statische Kostenterme, wie z.B. Erscheinungsähnlichkeit und Distanz zwischen Messungen modelliert. Der in dieser Arbeit entwickelte Ansatz nähert sich dieser Restriktion durch Anwendung eines mengentheoretischen Zustandsschätzers an, auf dessen Basis Messungen mit zusätzlichen geschätzen Zustandsvariablen angereichert werden. Unter Verwendung dieser Zustandsvariablen können z.B. lineare Bewegungsmodelle in die Netzwerkflussformulierung integriert werden. Eine Evaluation auf öffentlichen Datensätzen zeigt, dass sich durch die Kombination von mengentheoretischer Zustandsschätzung und globaler Optimierung eine erhöhte Genauigkeit erzielen lässt.

In einem zweiten Teil der Arbeit wird das verwandte Problem der Detektion und Verfolgung einer einzelnen Person in großen Personenmengen behandelt. Im Unterschied zum ersten Verfahren kann die in diesem Teil der Arbeit entwickelte Methode mit Situationen umgehen, in denen die Person die Szene verlässt und zu einem späteren Zeitpunkt wieder betritt. Dazu wird ein mengentheoretischer Ansatz verwendet, der eine für diese Problemart natürliche Repräsentation darstellt. Es wird außerdem gezeigt, wie durch Integration von Erscheinungsinformationen und parallele Ausführung mehrerer mengentheoretischer Filter die Leistung des Systems verbessert werden kann. Zur Evaluation wird sowohl ein öffentlich zugänglicher Datensatz als auch ein speziell auf die Problemcharakteristik abgestimmter, eigens aufgenommener Datensatz verwendet. In Experimenten wird gezeigt, dass das entwickelte System Personen erfolgreich verfolgt und nach Langzeitverdeckungen wiedererkennt.

In einem dritten Teil wird schließlich das Lernen einer Merkmalsrepräsenation für die Personenverfolgung behandelt, die robust gegenüber Artikulationen, variierenden Hintergründen und wechselnden Beleuchtungsbedingungen ist. Der in dieser Arbeit entwickelte Ansatz formuliert die Aufgabenstellung als Klassifikationsproblem. Dazu wird ein Klassifikator formuliert, der eine Kosinusmetrik auf dem darunterliegenden Merkmalsraum forciert. Während des Trainings werden die Merkmalsrepräsentation und der Klassifikator gemeinsam trainiert. Nach Abschluss des Trainings kann der Klassifikator entfernt werden. Ähnlichkeitsanfragen werden dann über die Kosinusdistanz im Merkmalsraum durchgeführt. Auf diese Weise generalisiert der Merkmalsraum über die im Trainingsdatensatz beinhalteten Individuen hinaus. In einer Evaluation auf zwei Datensätzen zur Personenwiedererkennung werden konkurenzfähige Ergebnisse erzielt.

Contents

1 Introduction **1**
- 1.1 Applications . . . 3
- 1.2 Contributions . . . 4
- 1.3 Outline . . . 5

2 Background **7**
- 2.1 Random Finite Sets . . . 8
  - 2.1.1 Multi-Object Probability Density Function . . . 8
  - 2.1.2 Probability Hypothesis Density . . . 10
  - 2.1.3 Relevant Classes . . . 11
- 2.2 Multi-Object State Estimation . . . 12
  - 2.2.1 System Representation . . . 13
  - 2.2.2 Motion Model . . . 14
  - 2.2.3 Measurement Model . . . 14
- 2.3 Probability Hypothesis Density Filter . . . 15
  - 2.3.1 Prediction . . . 16
  - 2.3.2 Update . . . 16
  - 2.3.3 Gaussian Mixture Implementation . . . 17
- 2.4 Min-Cost Flow Problem . . . 21
  - 2.4.1 Problem Formulation . . . 22
  - 2.4.2 Example . . . 23

3 Deep Cosine Metric Learning **27**
- 3.1 Introduction . . . 28
- 3.2 Related Work . . . 29
- 3.3 Joint Classification and Metric Learning . . . 31
  - 3.3.1 Problem Formulation . . . 31
  - 3.3.2 Standard Softmax Classifier . . . 32
  - 3.3.3 Cosine Softmax Classifier . . . 33
- 3.4 Network Architecture . . . 36
- 3.5 Evaluation . . . 38

3.5.1 Datasets and Evaluation Protocols . . . 38
3.5.2 Baseline Methods . . . 39
3.5.3 Results . . . 40

4 Min-Cost Flow PHD Tracker 49
4.1 Introduction . . . 49
4.2 Related Work . . . 51
4.3 Track-Oriented PHD Filter . . . 54
4.3.1 Prediction . . . 55
4.3.2 Update . . . 56
4.3.3 State Extraction . . . 57
4.4 Min-Cost Flow Trajectory Estimation . . . 59
4.4.1 Objective Function . . . 60
4.4.2 Min-Cost Flow Solution . . . 62
4.5 Comparison to Multiple Hypothesis Tracking . . . 65
4.6 Integration of Detector Confidence Scores . . . 69
4.7 Integration of Appearance Information . . . 71
4.8 Gaussian Mixture Implementation . . . 75
4.8.1 Track-Oriented GM-PHD Filter . . . 75
4.8.2 Trajectory Estimation Cost Terms . . . 78
4.9 Evaluation . . . 79
4.9.1 Metrics . . . 80
4.9.2 Simulation . . . 83
4.9.3 PETS 2009 . . . 86
4.9.4 MOT Challenge and RGB-D People . . . 93

5 Operator Following 99
5.1 Introduction . . . 99
5.2 Related Work . . . 101
5.3 Joint Detection and Tracking . . . 102
5.3.1 System Representation . . . 103
5.3.2 Interacting Bernoulli/PHD Filter Update . . . 105
5.3.3 Classification and Online Learning . . . 107
5.4 Experiments . . . 108
5.4.1 Parametrization . . . 108
5.4.2 PETS 2009 . . . 109
5.4.3 Custom Datasets . . . 110

6 Conclusion and Future Work 115

A Metric Learning Plots **119**

B Single Object Modeling **123**
B.1 Constant Velocity Motion Model 123
B.2 Linear Measurement Model 124
B.3 Linear Birth Model 125

C Comparison to Multiple Hypothesis Tracking **127**

D Gaussian Mixture Bernoulli Filter **131**

E Results on 2D MOT 2015 **133**

Abbreviations & Acronyms **139**

Symbols **141**

Bibliography **144**

Chapter 1

Introduction

Object tracking addresses the problem of estimating the location and assessing the identity of a single or multiple objects from a stream of sensor data, taken from a possibly moving platform. This problem naturally arises in many computer vision and robotics applications where it is often a basic requirement for higher-order reasoning. For example, intelligent vehicles that monitor outside traffic participants have the potential to warn about unforeseen events and to trigger emergency stops in situations where humans are unable to react. Likewise, consistent object labeling is necessary in video surveillance scenarios in order to detect unusual or suspicious behavior. Similar reasoning about human activities may be implemented on a service robot to perform complex tasks that include human-robot interaction.

The design of a concrete tracking system involves modeling aspects that depend on the application scenario and sensor setup at hand, but the typical tracking engineer is faced with the following situation: In a preprocessing stage, observation vectors are extracted from raw sensor data to be grouped into consistent object trajectories by the tracker. The provided observations are imperfect in the following ways. First, not all detections belong to a true object identity and false alarms must be correctly identified and disregarded. Second, some existing objects are not detected due to sensor failures and occlusions, but the application may require output even when no observation is available, e.g., for collision avoidance in automotive context. Finally, observations usually have no identifying label attached to them, such that the association between objects and unlabeled observations must be handled as part of the tracking system. Due to these imperfections, multiple object tracking is a challenging combinatorial problem: When the identity of observations is unknown, the number of possible combinations of observations that make up individual trajectories grows exponentially with the number of measurements in the observation sequence.

Handling of sensor noise and uncertainty in object motion models is a well-controlled subject. In the 1960s, the Kalman filter [Kal60] has been developed as a

closed-form recursive filter for systems where all involved distributions are Gaussian and the motion and measurement model are linear. For these problems, the Kalman filter represents a Bayes optimal state estimator. Approximations of the Kalman filter can be applied to nonlinear systems. The extended Kalman filter linearizes nonlinear motion and measurement models around the mean estimate by Taylor expansion [TBF05, Chapter 3]. The unscented Kalman filter [WVDM00, VDMW01] provides a more accurate approximation using moment matching. Instead of linearizing the model, a set of deterministically placed points is propagated through the nonlinear function. Then, moments of the posterior are reconstructed from the propagated points. Even highly nonlinear systems can be solved, though at higher computational cost, using sequential Monte-Carlo filtering [TBF05, Chapter 4].

Data association uncertainty is more difficult to deal with due to the combinatorial nature of the problem. Since their formulation in the 1980s, Joint Probabilistic Data Association (JPDA) [FBSS83] and Multiple Hypothesis Tracking (MHT) [Rei79] have largely influenced the tracking community. Both methods approach the problem by evaluation of multiple association hypotheses. In JPDA, a joint probabilistic score for the measurement-to-track association is computed under consideration of all assignments. In MHT, the space of object trajectories is explored in a breadth-first search from the first time step onwards. Due to this systematic exploration, MHT has long been seen as a gold standard. However, the method is computationally costly and it has taken until the mid 1990s until a practical implementation was proposed by Cox [CH96]. Since then, the overwhelming combinatorial nature of multi-object systems has lead to different conclusions on how to approach the problem systematically.

In information fusion, random finite sets and finite set statistics [Mah07b] have been developed for a rigorous treatment of multi-object phenomena under the Bayesian paradigm. The theory provides mathematical procedures intended for derivation at principled approximations with certain guarantees on their validity and applicability in real-time. In contrast to MHT, where optimality was never proven, the framework provides means to derive an optimal multi-object Bayes filter in which the state of all objects is a single set-valued random variable. It is only recently that a computationally tractable implementation has been developed [VVP14]. Earlier, computationally cheaper approximations have emerged out of this theory, such as the Probability Hypothesis Density filter [Mah03], the cardinalized Probability Hypothesis Density filter [Mah07a], and the Multi-Target Multi-Bernoulli filter [VVC09]. These filters impose additional assumptions on the multi-object state, such as independent motion and independent measurement generation, that yield efficient but principled approximations. The second intention behind the development of random finite sets and finite set statistics was to provide an engineering friendly theory for multi-object problems. It is arguable if random

finite sets are the "practitioner friendly version of point process theory" [Mah07b, p.8] as intended. Bringing its own mathematical notation and calculus, there certainly is an entry hurdle that must be overcome to design appropriate models for algorithms derived from finite set statistics.

There has been tremendous progress on object detection and feature learning within the last decade [FGMR10, RHGS15, SRASC14]. In vision, this progress has lead to a shift away from recursive filtering towards global optimization strategies. Notable advances have been made under the *tracking-by-detection* paradigm which typically refers to a tighter coupling of detection and tracking. Trajectories are often recovered directly in image space and often without estimation of a hidden kinematic state. Interestingly, the related field of simultaneous localization and mapping has undergone a similar shift away from the Bayesian paradigm towards efficient global optimization strategies [GKSB10] and is now, apart from conventional methods that still dominate the field, a prime research area for methods derived from finite set statistics [MVAV11]. A particular formulation that has found wide application in tracking is based on a min-cost flow transportation problem [ZLN08]. The popularity is due to availability of optimal, yet very efficient inference algorithms that exploit the specific structure of the tracking problem [BFTF11, LGU15]. However, a key limitation of the min-cost flow formulation is its restriction to only pairwise cost terms. This makes integration of motion models challenging because multiple observations are required to compute object motion. Consideration of higher-order potentials, however, makes the problem NP-hard and approximate inference must applied [Col12, BC13]. Then, efficiency and optimality are lost.

1.1 Applications

As a fundamental environment perception problem, object tracking has application in numerous domains. The methods in this thesis have been developed with the following applications in mind.

Automotive The automotive industry is a major end user of object tracking technologies. The demand stems from a need for reliable environment perception in intelligent driver assistance systems. All major vehicle manufacturers offer advanced systems that enhance safety and driving comfort, such as autonomous cruise control, collision avoidance, and blind spot monitoring [LKVN07]. The European Road Safety Observatory has summarized the casualty reduction effect of various safety technologies [Eur16b]. For example, autonomous emergency breaking systems effectively prevent up to 44% of front to rear collisions. In consequence, they demand that from 2018 on all new vehicles are equipped with such systems.

Autonomous driving is a long followed goal with first success stories in the 1980s [VPRH87]. The topic has received wide public attention due to a DARPA challenge that was held in 2007 [BIS09] where vehicles were required to drive 90 km through an urban environment. Since then, major manufacturers have developed prototype autonomous vehicles. Due to the ongoing automation in passenger transportation, the need for reliable object tracking can be expected to increase further in the future.

Robotics Research on service robots is dedicated to the development of autonomous systems that assist humans in daily life activities. Activity monitoring and behavior analysis play an essential role in this task. Object tracking is a low-level vision task that works towards this goal, enabling the robot to move through populated environments and interact with people. Robotics research has reached an exciting stage where first autonomous systems are deployed in industrial and service applications. Within the SPENCER project [TAA+16] for example, a robot was developed to guide passengers to their designated gate at a large European airport. The subject can be expected to gain further importance. To better cope with the challenges of an aging society, the European Commission funds research into robotics for aging well under the Horizon 2020 framework programme [Eur16a].

Surveillance Video surveillance is a safety and security task where detailed information about the density, movements, and actions involving people and other road users are obtained from one or multiple cameras in an automated fashion. Besides obvious security applications, such information are also used to assess bottlenecks in infrastructure or analyze traffic situations that lead towards dangerous situations [Eur17]. On a regular basis, the Performance Evaluation of Tracking and Surveillance (PETS) workshop is held in conjunction with the International Conference on Computer Vision and Pattern Recognition. One of the most popular crowd analysis datasets is due to this workshop [FS09].

1.2 Contributions

Methods based on random finite sets have yet found little adoption in vision and robotics. From a historical perspective this is surprising, since information fusion, vision, and robotics have long considered the same algorithms for application in their respective domains [Rei79, FBSS83, CH96]. Potentially, this is due to an entry hurdle that must be overcome to apply these methods in practice. This thesis aims at bridging the gap between theoretically capable methods that have emerged out random finite set theory and practical tracking applications in computer vision in robotics. This goal is reached by the following contributions:

- The main contribution of this thesis is the development of a multi-object tracker that combines recursive multi-object filtering with global data association, thus combining methods from computer vision with random finite set theory. The tracker works with general point targets and can be applied to a broad range of problems. From a computer vision perspective, the problem formulation enables integration of object motion models through recursive filtering without increasing the complexity of the data association problem. From a random finite set perspective, the approach offers a novel data association scheme to extract trajectories from the output of a Probability Hypothesis Density filter [Mah03]. This work was presented at a renowned international conference for robotics research [WP16] and is the first method to combine random finite set theory with global data association. A practical sequence Monte-Carlo implementation was presented in [WP17].

- The second contribution is the application of an optimal multi-object Bayes filter to the specific problem of tracking a single robot operator through crowded environments. It is shown that the random finite set formulation offers a natural representation for this type of problem and how the performance of such a system can be improved by integration of appearance features and running multiple filters concurrently. This work has been presented at a major conference on information fusion [WMP17].

- In a final contribution, person re-identification is explored as an environment to train an appearance descriptor for people tracking that is robust to articulation, changing lighting conditions, and viewpoint changes. For this purpose, a light-weight deep convolutional neural network architecture is presented that permits application in online tracking scenarios. In addition, a joint classification and metric learning framework is developed that overcomes practical issues related to well-established metric learning objectives. Evaluation shows that this framework improves the final model performance when compared to these objectives. The work has been developed as part of a multi-object tracker that has been presented at a major image processing conference [WBP17a]. The details of the learning framework have been published separately [WB18].

1.3 Outline

The following chapters are structured as follows. Chapter 2 introduces a theoretical background on random finite set theory and the min-cost flow transportation problem. The material presented in this chapter builds a foundation for the methods that have been developed in this thesis. Chapter 3 diverts from the tracking

application to a person re-identification task in order to learn an appearance descriptor for people tracking that is robust towards articulation, background, and changing lighting conditions. A multi-object tracker that combines multi-object recursive state estimation with global data association in a tracking-by-detection paradigm is presented in Chapter 4. In Chapter 5, a solution to a practical robotics tracking task is developed in which random finite sets offer a natural representation for the problem at hand. Finally, a conclusion and directions for future work are presented in Chapter 6.

Chapter 2

Background

This chapter provides a theoretical foundation on Finite Set Statistics (FISST), a specialized theory for set-valued random variables that has been developed to make results from point process theory more accessible to engineers [Mah07b]. The basic idea underlying FISST is to provide statistical descriptors and mathematical procedures by which multi-object, multi-sensor problems can be treated similar to the well-established single-object, single-sensor case. To this end, the theory generalizes many concepts from conventional vector-valued probability theory to set-valued phenomena. In the FISST methodology, the engineer defines appropriate models for the practical application at hand and then applies a specialized multi-object calculus to derive at a solution. The potential benefit of this approach lies in the separation of application engineering and mathematical background. The engineer is not required to be an expert on the mathematics behind the theory in order to apply it. Yet, the theory itself is mathematically grounded and, in some cases, offers guarantees such as Bayes optimality that are otherwise hard to prove (e.g., [RVVF13]).

The following description covers only a minimal set of theoretical background rather than diving deep into derivations or formalisms that may impair the comprehension of a potentially unfamiliar reader. For a broader, but relatively gentle introduction to the topic refer to tutorial texts [Mah04] and [Mah13]. An extensive treatment with formal definitions and derivations is given in Mahler's book [Mah07b]. A recent sequel [Mah14] contains a discussion of advances in the field. The remainder of this chapter begins with a discussion of set-valued random variables in Section 2.1 and continues with a presentation of the multi-object system representation in Section 2.2. The Probability Hypothesis Density filter, a set-valued state estimator that plays a central role in this thesis, is described in Section 2.3. The chapter concludes with an introduction of the min-cost flow problem in Section 2.4. This problem will later be used as an inference framework to recover object trajectories from the Probability Hypothesis Density filter.

2.1 Random Finite Sets

Random Finite Sets (RFSs) are set-valued random variables that contain an unknown number of elements which are themselves random. Within the scope of this thesis, these are the set of objects present in the scene or the set of measurements generated by these objects at a particular point in time. Given an underlying space $\mathcal{X}$, such as a single-object state or measurement space, a random finite set draws instantiations from the hyperspace of all finite subsets $\mathcal{F}(\mathcal{X})$ of $\mathcal{X}$. As, in general, a random finite set may contain any (finite) number of elements, possible instantiations of a random finite set $X \in \mathcal{F}(\mathcal{X})$ are

$$X = \emptyset, \quad X = \{\boldsymbol{x}_1\}, \quad X = \{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \quad \dots \quad X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}, \quad \dots \tag{2.1}$$

where $\boldsymbol{x}_i \in \mathcal{X}$ and $\forall i, j : \boldsymbol{x}_i \neq \boldsymbol{x}_j$. Random finite sets impose no ordering on their elements, making them particularly well suited to model phenomena where (1) the ordering has no physical interpretation or (2) elements are not directly observable and the association between elements and their measurements is unknown.

Two probability distributions are needed to characterize the uncertainty involved in a random finite set: A discrete probability distribution for the cardinality of the set and a joint probability distribution for the individual members, given the cardinality. Then, sampling from a random finite set can be illustrated as follows. First, draw the number of elements n from the cardinality distribution. Then, sample $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ from the joint probability distribution over its elements conditional on n.

2.1.1 Multi-Object Probability Density Function

Remark 2.1. Let $x \in \mathbb{R}$ be a continuous random variable on the set of real numbers. Then its probability density function is a real-valued, non-negative function $p(x)$ that, for all intervals $[a, b]$, integrates to the probability that x takes on a value in $a \leq x \leq b$:

$$\mathrm{P}(a \leq x \leq b) = \int_a^b p(x)\,\mathrm{d}x \tag{2.2}$$

with $\forall x \in \mathbb{R} : p(x) > 0$ and $\mathrm{P}(-\infty \leq x \leq \infty) = 1$. Alternatively, this can be written as

$$\mathrm{P}(x \in S) = \int_S p(x)\,\mathrm{d}x \tag{2.3}$$

where $S = \{x \mid \forall x \in \mathbb{R} : a \leq x \leq b\}$ is the interval in set-builder notation.

The multi-object probability density function of a random finite set is an analogous concept to the probability density function of a continuous random vector[1]. Let $X \in \mathcal{F}(\mathcal{X})$ be a random finite set on some underlying space $\mathcal{X}$. Then its multi-object probability density function is a real-valued, non-negative function $\pi(X)$ that, for all regions $S \subseteq \mathcal{X}$, integrates to the probability that X takes on a subset of S:

$$P(X \subseteq S) = \int_S \pi(X)\, \delta X \tag{2.4}$$

with $\forall X \in \mathcal{F}(\mathcal{X}) : \pi(X) \geq 0$ and $P(X \subseteq \mathcal{X}) = 1$. Since this definition contains an integral over a set-valued random variable with an unknown number of elements that appear in arbitrary order, it requires application of a specialized set-integral which is denoted by $\int \delta X$.

The first step towards defining this integral is to express the multi-object probability density function in a vector notation. Therefore, define for each $n \geq 2$ a function in n vector variables that distributes the probability density of the set equally among all $n!$ permutations of the given inputs:

$$f_n(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \begin{cases} \frac{1}{n!}\pi(\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}) & \text{if } \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \text{ are distinct,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.5}$$

Then, the set-integral in (2.4) can be expressed as an infinite sum over the cardinality of the set by integrating over all possible assignments [Mah07b, Section 11.3.3]:

$$\int \pi(X)\, \delta X = \pi(\emptyset) + \int \pi(\{\boldsymbol{x}\})\, \mathrm{d}\boldsymbol{x} + \sum_{n=2}^{\infty} \int_{S^n} f_n(\boldsymbol{x}_i, \ldots, \boldsymbol{x}_n)\, \mathrm{d}\boldsymbol{x}_1 \ldots\, \mathrm{d}\boldsymbol{x}_n, \tag{2.6}$$

where $S^n = S \times \ldots \times S$ is the Cartesian product over n sets. It is obvious that, due to the infinite sum over all cardinalities, evaluation of a set-integral is generally computationally intractable. However, the set-integral simplifies to tractable terms for certain classes of random finite sets. This is illustrated in the following Example 2.1.

Example 2.1. The purpose of this example is to show how a random vector can be modeled as a random finite set and that, in this case, the multi-object probability density function is equivalent to the conventional probability density function. Therefore, assume a single object $\boldsymbol{x} \in \mathbb{R}^2$ that moves on the

[1] In accordance with existing literature, the term multi-object probability density function is used to refer to the probability density of a set-valued random variable. The theory is not limited to modeling of multi-object tracking problems, but has been developed with this application in mind.

ground plane. The object is always present in the scene, but its position is subject to spatial uncertainty that is modeled by a bivariate normal distribution

$$p(\boldsymbol{x}) = \det(2\pi \boldsymbol{P})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m})^{\mathrm{T}} \boldsymbol{P}^{-1}(\boldsymbol{x}-\boldsymbol{m})\right) \tag{2.7}$$

with mean $\boldsymbol{m}$ and covariance $\boldsymbol{P}$. In this simple example, the random finite set $X = \{\boldsymbol{x}\}$ is a singleton set and its multi-object probability density function is

$$\pi(X) = \begin{cases} p(\boldsymbol{x}) & \text{if } X = \{\boldsymbol{x}\}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

Since $\pi(X)$ evaluates to 0 whenever $|X| \neq 1$, it is easily verified that this function fulfills the properties of a multi-object probability density function. Further, for this example the set-integral (2.6) simplifies to a conventional integral

$$\int \pi(X)\, \delta X = \pi(\emptyset) + \int \pi(\{\boldsymbol{x}\})\, \mathrm{d}\boldsymbol{x} + \sum_{n=2}^{\infty} \int_{S^n} f_n(\boldsymbol{x}_i, \ldots, \boldsymbol{x}_n)\, \mathrm{d}\boldsymbol{x}_1 \ldots\, \mathrm{d}\boldsymbol{x}_n \tag{2.9}$$

$$= 0 + \int p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} + 0 + \ldots \tag{2.10}$$

and dealing with the random finite set $X = \{\boldsymbol{x}\}$ is no harder than dealing with its random vector equivalent $\boldsymbol{x}$ directly.

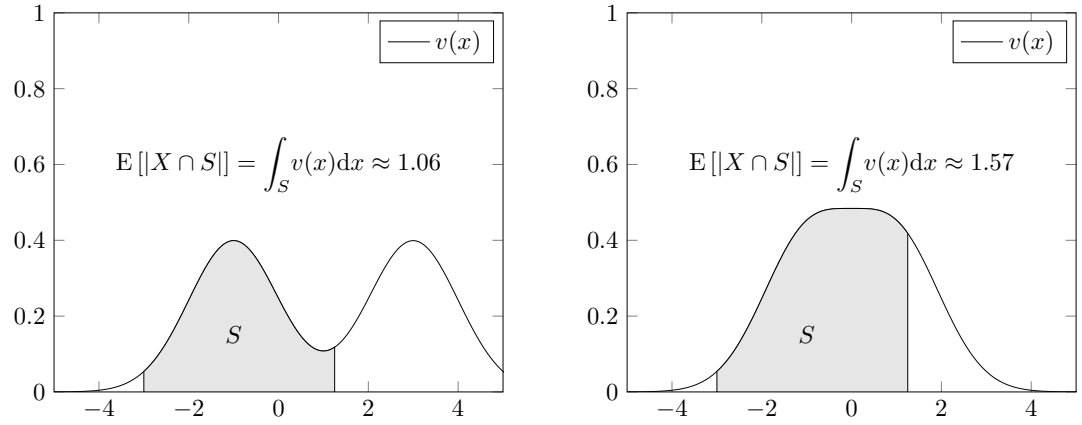
2.1.2 Probability Hypothesis Density

Let X be a random finite set that draws its instantiations from the hyperspace of all finite subsets $\mathcal{F}(\mathcal{X})$ of some space $\mathcal{X}$. The first-order moment of X is a non-negative function $v(\boldsymbol{x})$ defined on $\mathcal{X}$. Formally, it is defined as a set-integral [Mah07b, p. 580]:

$$v(\boldsymbol{x}) = \int \pi(\{\boldsymbol{x}\} \cup X)\, \delta X. \tag{2.11}$$

This function is called the Probability Hypothesis Density (PHD) or *intensity* of X. Intuitively, it can be interpreted as a fuzzy-membership function that represents the zero-probability event that an element $\boldsymbol{x}$ is contained in the set X, i.e., $v(\boldsymbol{x}) \mathrel{\widehat{=}} P(\boldsymbol{x} \in X)$. However, the PHD is not a proper probability density in the sense that the integral $\int_{\mathcal{X}} v(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$ is not equal to 1. Instead, for all closed subsets $S \subseteq \mathcal{X}$, the PHD integrates to the expected number of elements in X that are also present in S:

$$\mathrm{E}[|X \cap S|] = \int_S v(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}. \tag{2.12}$$



(a) $x_1 \sim \mathcal{N}(-1, 1^2)$, $x_2 \sim \mathcal{N}(3, 1^2)$ **(b)** $x_1 \sim \mathcal{N}(-1, 1^2)$, $x_2 \sim \mathcal{N}(1, 1^2)$

Figure 2.1: Illustration of the PHD of a random finite set $X = \{x_1, x_2\}$ that contains two normally distributed objects. In (a) both objects are well separated, and the integral over region S returns approximately 1.06 objects. As the objects move closer together in (b), the expected number of objects in S increases to approximately 1.57.

This property gives rise to a second interpretation: The PHD is an object density measure in the sense that elements are concentrated in regions of high intensity mass. Figure 2.1 illustrates this property for a random finite set that contains two statistically independent objects. In (a) both objects are well separated and the PHD has two well separated peaks from which the location of its elements can be inferred. The two modes join when both objects move closer together in (b). At this point, a clear distinction between the two objects is not possible. However, the expected number of objects contained in any region can be obtained by integrating over the intensity function.

2.1.3 Relevant Classes

Several common classes of random finite sets can be found in the literature. Relevant to this thesis are the following two classes.

Poisson RFS A Poisson RFS X contains an expected number of λ objects that are independent and identically distributed (i.i.d.) according to a spatial density $p(\boldsymbol{x})$. The cardinality distribution is Poisson with mean λ. The probability density of the Poisson RFS is

$$\pi(X) = e^{-\lambda} \prod_{\boldsymbol{x} \in X} \lambda\, p(\boldsymbol{x}) = e^{-\int v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}} \prod_{\boldsymbol{x} \in X} v(\boldsymbol{x}). \tag{2.13}$$

Therefore, Poisson RFSs are completely characterized by their PHD

$$v(\boldsymbol{x}) = \lambda \cdot p(\boldsymbol{x}). \tag{2.14}$$

The expected number of elements in the set can be computed from the PHD by $\lambda = \int v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$ and the spatial density of its elements by $p(\boldsymbol{x}) = v(\boldsymbol{x})/\lambda$.

Bernoulli RFS A Bernoulli RFS X contains at most one element. The cardinality distribution is a Bernoulli distribution with parameter q, such that the set is empty with probability $1 - q$ and contains an element with probability q. The probability density of the Bernoulli RFS is

$$\pi(X) = \begin{cases} 1 - q & \text{if } X = \emptyset, \\ q \cdot p(\boldsymbol{x}) & \text{if } X = \{\boldsymbol{x}\}, \\ 0 & \text{otherwise,} \end{cases} \tag{2.15}$$

where $p(\boldsymbol{x})$ is the spatial density. The PHD of the Bernoulli RFS is $v(\boldsymbol{x}) = q \cdot p(\boldsymbol{x})$. As for the Poisson RFS, conversion between the multi-object density and the PHD is possible without loss of information. The probability of existence can be recovered from the PHD by $q = \int v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$ and the spatial density by $p(\boldsymbol{x}) = v(\boldsymbol{x})/q$. Throughout this thesis, the probability density of a Bernoulli RFS is abbreviated by its parameters $\pi = (q, p(\boldsymbol{x}))$ when appropriate.

2.2 Multi-Object State Estimation

In the typical object tracking application, raw sensor data is preprocessed by a detector to obtain likely object locations. These are then passed on to the tracker to filter false alarms and extract object trajectories. This processing pipeline is subject to various sources of uncertainty that are left for the tracker to resolve. In particular, detector-based preprocessing steps usually cannot resolve data association ambiguity themselves. Instead, they provide locations of likely object occurrences for all objects of a particular class, for example, pedestrians, vehicles, or bicycles. Under these circumstances, the association of detections to individual object identities is one of the most fundamental challenges of the tracking component. Specifically designed to deal with set-valued phenomena, FISST provides the means to address this problem in a mathematically principled way.

2.2.1 System Representation

In a random finite set methodology, the set of all object states X_k and measurements Z_k at time k are reconceptualized as set-valued random variables

$$X_k = \{\boldsymbol{x}_{k,1}, \ldots, \boldsymbol{x}_{k,N_k}\} \in \mathcal{F}(\mathcal{X}), \tag{2.16}$$
$$Z_k = \{\boldsymbol{z}_{k,1}, \ldots, \boldsymbol{z}_{k,M_k}\} \in \mathcal{F}(\mathcal{Z}), \tag{2.17}$$

on state and measurement spaces $\mathcal{X}$ and $\mathcal{Z}$ without imposing a specific ordering on the respective collections. This representation is natural in that the exact number of objects as well as their association to individual measurements is unknown and their ordering has no physical interpretation[2]. Through application of multi-object calculus, a specialized Bayes recursion

$$\pi\left(X_k \mid Z_{1:k}\right) = \frac{\pi\left(Z_k \mid X_k\right) \pi\left(X_k \mid Z_{1:k-1}\right)}{\pi\left(Z_k \mid Z_{1:k-1}\right)} \tag{2.18}$$

can be derived to solve the estimation problem optimally, thus leading to a mathematically principled way to incorporate data association uncertainty into the Bayes recursion. The modeling decisions to be made in this framework are on par with the conventional vector-valued Bayes filter: In Equation 2.18, $\pi\left(X_k \mid X_{k-1}\right)$ is a set-valued Markov state transition density that describes the evolution of the multi-object state over time. This density not only characterizes object motion, but also appearances and disappearances between consecutive time steps. Likewise, $\pi\left(Z_k \mid X_k\right)$ is a set-valued measurement likelihood function that describes the measurement generation process. The aspects involved in designing appropriate models are similar to those in conventional Bayesian filtering. The random finite set system representation not so much supersedes the conventional formalization as it compromises it. At the core, conventional vector-valued probabilistic motion and measurement models describe the single-object state evolution. These models are complemented by the set-valued system representation to characterize multi-object phenomena such as object appearances and disappearances, sensor failures, and false alarms. The following two sections present the standard models commonly employed in practical implementations [Mah03, VSD05, VM06, VVC09].

[2]One may argue that object states are ordered, in order to express the identity associated with each object. This ordering, however, would be arbitrary. Any permutation of the order that maintains consistent identities over all time steps is equally valid. Therefore, in the random finite set methodology, maintaining object identities is handled by tracking on a state space that is augmented with an identifying label [VVP14].

2.2.2 Motion Model

In the standard formulation, the set of objects at the time k is a union of newly appearing objects and surviving objects from time $k-1$. Therefore, let B_k denote the random finite set of spontaneously appearing objects at time k. Further, assume that each object $\boldsymbol{x}' \in X_{k-1}$ either disappears from the scene with probability $1 - p_{\mathrm{S}}(\boldsymbol{x}')$ or transitions to a new state $\boldsymbol{x} \in \mathcal{X}$ according to $p_{k|k-1}(\boldsymbol{x} \mid \boldsymbol{x}')$. Then, this behavior can be modeled by a Bernoulli RFS $S_k(\boldsymbol{x}')$ with parameters $(p_{\mathrm{S}}(\boldsymbol{x}'), p_{k|k-1}(\boldsymbol{x} \mid \boldsymbol{x}'))$ and the set of objects at time k can be expressed by

$$X_k = B_k \cup \left[\bigcup_{\boldsymbol{x}' \in X_{k-1}} S_k(\boldsymbol{x}') \right]. \tag{2.19}$$

The individual random finite sets in the above union are considered statistically independent [Mah07b, Section 13.2]. Depending on the Bayes filter approximation, the birth set may take on different forms. For example, if the task is to track a single object through clutter, B_k is a Bernoulli RFS and the Bayes filter recursion can be implemented exactly [Mah07b, Section 14.7]. If multiple objects are present in the scene, then B_k may be modeled by a Poisson RFS [Mah03]. Based on this decision, the multi-object Markov transition density $\pi(X_k \mid X_{k-1})$ can be derived by application of multi-object calculus. However, since the multi-object Markov state transition density is not used directly in this thesis, the interested reader is referred to [Mah07b, Chapter 13] for a discussion on further modeling aspects and a mathematical derivation.

2.2.3 Measurement Model

The measurement model draws a generative relationship between the set-valued object state and the set of observed measurements. This relationship covers the spatial location of object-generated measurements, but also includes sensor characteristics such as the probability of detection, sensor field of view, clutter characteristics, etc. Therefore, let $p_{\mathrm{D}}(\boldsymbol{x})$ denote a probability of detection for given state $\boldsymbol{x} \in \mathcal{X}$ and let $p_k(\boldsymbol{z} \mid \boldsymbol{x})$ denote a conventional single-object measurement model that draws a relationship between $\boldsymbol{x}$ and measurement $\boldsymbol{z} \in \mathcal{Z}$. Then, object-generated measurements are modeled by a Bernoulli RFS $H_k(\boldsymbol{x})$ with parameters $(p_{\mathrm{D}}(\boldsymbol{x}), p_k(\boldsymbol{z} \mid \boldsymbol{x}))$. This RFS is either empty with probability $1 - p_{\mathrm{D}}(\boldsymbol{x})$ or contains exactly one element that is distributed according to $p_k(\boldsymbol{z} \mid \boldsymbol{x})$. If further K_k denotes the clutter RFS, then the set of measurements observed at time k is

$$Z_k = K_k \cup \left[\bigcup_{\boldsymbol{x}_k \in X_k} H_k(\boldsymbol{x}) \right]. \tag{2.20}$$

It is generally assumed that clutter is independent of object-generated measurements and that object-generated measurements are conditionally independent of object state [Mah07b, Chapter 12]. Throughout this thesis, clutter is modeled by a Poisson RFS. As for the Markov transition density, the multi-object measurement likelihood $\pi(Z_k \mid X_k)$ will not be used directly in this thesis. A detailed discussion with examples and derivations, including a relationship to the likelihood function that is used in MHT, can be found in [Mah07b, Chapter 12].

2.3 Probability Hypothesis Density Filter

The PHD filter was the first successfully applied multi-object state estimator based on FISST [Mah03]. It alleviates the computational intractability of the exact multi-object Bayes recursion by propagating a first-order moment statistic instead of a full multi-object probability density. This is conceptually closer to tracking entire groups of objects rather than individuals. To see why this is true, recall that in Section 2.1.2 it has been pointed out that for a random finite set $X \in \mathcal{F}(\mathcal{X})$ with PHD $v(\boldsymbol{x})$ the integral over any region $S \subset \mathcal{X}$ yields the expected number of elements in X that are also in S. Due to this property, locations of high object concentration can be extracted from peaks of the PHD. It does not, however, contain information about individual object identities in the set. Therefore, these must be extracted from the PHD in a post processing step. Despite this theoretical limitation, the PHD filter offers several potential advantages over other methods [Mah07b, p. 571]:

- The PHD filter does not suffer from an exponential growth of track hypotheses. At each time step, the computational complexity is $\mathcal{O}(mn)$ where m is the number of measurements and n is the number of objects in the scene.

- The PHD filter does not require explicit decisions on measurement-to-track associations which potentially introduce errors into state estimates.

- There exist efficient sequential Monte-Carlo [VSD05] and Gaussian mixture [VM06] implementations.

- The number of objects in the scene is estimated directly from data under consideration of sensor characteristics such as probability of detection, clutter statistics, and sensor field of view.

- The filter has shown good performance compared to conventional methods and there is evidence of successful applications, e.g., [PVS+04, JB09, KLW+10, MĆP15].

The remainder of this section outlines the PHD filter recursion without further discussion. A practical Gaussian mixture implementation will be presented in the following section.

2.3.1 Prediction

Given the posterior intensity $v_{k-1}(\boldsymbol{x})$ of the multi-object state X_{k-1} at time $k-1$, the predicted intensity at time k is

$$v_{k|k-1}(\boldsymbol{x}) = b_k(\boldsymbol{x}) + \int p_{\mathrm{S}}(\boldsymbol{x}')p_{k|k-1}(\boldsymbol{x} \mid \boldsymbol{x}')v_{k-1}(\boldsymbol{x}')\,\mathrm{d}\boldsymbol{x}_{k-1} \tag{2.21}$$

where $b_k(\boldsymbol{x}) = \lambda_{\mathrm{b},k} \cdot p_{\mathrm{b},k}(\boldsymbol{x})$ is the intensity of a Poisson birth RFS with mean cardinality $\lambda_{\mathrm{b},k}$ and spatial density $p_{\mathrm{b},k}(\boldsymbol{x})$. For an intuitive interpretation, note that the second term in Equation 2.21 reduces to down-scaled Bayes filter prediction for a constant probability of survival $p_{\mathrm{S}}(\boldsymbol{x}) = p_{\mathrm{S}}$. If X_k is approximately Poisson, then $\hat{N}_{k|k-1} = \int v_{k|k-1}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$ objects in X_k are distributed proportional to $v_{k|k-1}(\boldsymbol{x})$. Therefore, the PHD predictor propagates the spatial density of surviving object according to the single-object motion model and adapts the cardinality estimate to account for disappearing objects. Additionally, objects that enter the scene are accounted for by adding intensity mass at locations where objects are likely to appear.

2.3.2 Update

Let $v_{k|k-1}(\boldsymbol{x})$ denote the predicted intensity of the multi-object state X_k at time k. Given a set of newly arrived measurements Z_k, the posterior intensity at time k is

$$v_k(\boldsymbol{x}) = v_{\mathrm{L},k}(\boldsymbol{x}) + \sum_{\boldsymbol{z} \in Z_k} v_{\mathrm{U},k}(\boldsymbol{z}, \boldsymbol{x}), \tag{2.22}$$

where the first term accounts for missed detections

$$v_{\mathrm{L},k}(\boldsymbol{x}) = [1 - p_{\mathrm{D}}(\boldsymbol{x})]\, v_{k|k-1}(\boldsymbol{x}) \tag{2.23}$$

and the second term accounts for measurement-corrected updates

$$v_{\mathrm{U},k}(\boldsymbol{z}, \boldsymbol{x}) = \frac{p_{\mathrm{D}}(\boldsymbol{x})p_k(\boldsymbol{z} \mid \boldsymbol{x})v_{k|k-1}(\boldsymbol{x})}{c_k(\boldsymbol{z}_{k,j}) + \int p_{\mathrm{D}}(\boldsymbol{x})p_k(\boldsymbol{z} \mid \boldsymbol{x})v_{k|k-1}(\boldsymbol{x})} \tag{2.24}$$

with $c_k(\boldsymbol{z}) = \lambda_{\mathrm{c},k} \cdot p_{\mathrm{c},k}(\boldsymbol{z})$ denoting the intensity of the Poisson clutter RFS with mean cardinality $\lambda_{\mathrm{c},k}$ and spatial density $p_{\mathrm{c},k}(\boldsymbol{z})$.

In the missed-detection term, the predicted intensity is down-weighted by one minus the probability of detection to account for objects that remain present in

the scene, but are not detected at the current time. Each measurement-corrected partition resembles a Bayes-like update of the predicted intensity with one of the measurements. By taking into account clutter statistics $c_k(\boldsymbol{z})$, the intensity mass weights off the likelihood of the measurement being generated by an object against the clutter intensity.

2.3.3 Gaussian Mixture Implementation

The PHD filter can be implemented in closed form under linear Gaussian assumptions. The corresponding Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter [VM06] maintains a Gaussian mixture representation of the intensity

$$v_k(\boldsymbol{x}) = \sum_{i=1}^{L_k} w_k^{(i)} \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}_k^{(i)}, \boldsymbol{P}_k^{(i)}) \tag{2.25}$$

where the weights sum up to the expected number of objects $\int v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \sum_{i=1}^{L_k} w_k^{(i)}$. The implementation presented in this section loosely follows an extension of the GM-PHD filter where it is assumed that objects are detected on their first entrance into the scene [RCVV12]. This leads to a more efficient processing of birth components if no a priori knowledge about the location of appearing object is available.

Prediction

The pseudo code for the GM-PHD predictor is given in Algorithm 1. In line 3, component weights are multiplied by a constant probability of survival $p_\mathrm{S}(\boldsymbol{x}) = p_\mathrm{S}$. A Kalman filter prediction step is carried out in line 4. If object motion is described by a linear function $\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{\epsilon}_k$ with zero-mean additive noise $\boldsymbol{\epsilon}_k \propto \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$, then

$$\boldsymbol{m}_{k|k-1}^{(i)} = \boldsymbol{F}\boldsymbol{m}_{k-1}^{(i)}, \tag{2.26}$$

$$\boldsymbol{P}_{k|k-1}^{(i)} = \boldsymbol{F}\boldsymbol{P}_{k-1}^{(i)}\boldsymbol{F}^\mathrm{T} + \boldsymbol{Q} \tag{2.27}$$

is the corresponding Kalman filter prediction (see, e.g., [TBF05]). The constant velocity motion model is given as an example in Appendix B.1. This model assumes objects follow a linear motion pattern with constant velocity. Errors in this assumption are compensated by noise covariance $\boldsymbol{Q}$ that increases the uncertainty associated with the predicted object state in Equation 2.27. Note that there are no birth components in the presented implementation of the GM-PHD predictor. Following [RCVV12], birth components are created during the filter update step.

Algorithm 1 Gaussian mixture implementation of the PHD filter prediction step

1: **procedure** GMPHD_PREDICT($\{w_{k-1}^{(i)}, \boldsymbol{m}_{k-1}^{(i)}, \boldsymbol{P}_{k-1}^{(i)}\}_{i=1}^{L_{k-1}}$)
2: **for** $i = 1, \ldots, L_{k-1}$ **do**
3: $w_{k|k-1}^{(i)} = p_{\mathrm{S}}\, w_k^{(i)}$
4: $\boldsymbol{m}_{k|k-1}^{(i)}, \boldsymbol{P}_{k|k-1}^{(i)} = \text{KALMAN\_PREDICT}(\boldsymbol{m}_{k-1}^{(i)}, \boldsymbol{P}_{k-1}^{(i)})$
5: **end for**
6: **return** $\{w_{k|k-1}^{(i)}, \boldsymbol{m}_{k|k-1}^{(i)}, \boldsymbol{P}_{k|k-1}^{(i)}\}_{i=1}^{L_{k-1}}$
7: **end procedure**

Update

Pseudo-code for the GM-PHD corrector is given in Algorithm 2. The algorithm is presented in terms of a procedure that computes the missed-detection intensity partition of Equation 2.23 in lines 1–8 and a procedure that computes the measurement-corrected partitions of Equation 2.24 in lines 10–21. The missed-detection partition is generated from the predicted intensity by scaling component weights by a constant miss probability $1 - p_{\mathrm{D}}(\boldsymbol{x}) = 1 - p_{\mathrm{D}}$. Each measurement-corrected partition is computed from the predicted intensity by application of a Kalman filter update (line 12 and 13). If measurement $\boldsymbol{z}_k$ is generated from the single-object state $\boldsymbol{x}_k \sim \mathcal{N}(\boldsymbol{m}_{k|k-1}^{(i)}, \boldsymbol{P}_{k|k-1}^{(i)})$ by a linear function $\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{\delta}_k$ with zero-mean additive noise $\boldsymbol{\delta}_k = \mathcal{N}(\boldsymbol{0}, \boldsymbol{R})$, then

$$\boldsymbol{m}_k^{(i)} = \boldsymbol{m}_{k|k-1}^{(i)} + \boldsymbol{K}_k^{(i)} \left(\boldsymbol{z}_k - \boldsymbol{n}_k^{(i)}\right), \tag{2.28}$$

$$\boldsymbol{P}_k^{(i)} = \boldsymbol{P}_{k|k-1}^{(i)} - \boldsymbol{K}_k^{(i)} \boldsymbol{S}_k^{(i)} \left(\boldsymbol{K}_k^{(i)}\right)^{\mathrm{T}}, \tag{2.29}$$

$$l_k^{(i)} = \mathcal{N}(\boldsymbol{z}_k; \boldsymbol{n}_k^{(i)}, \boldsymbol{S}_k^{(i)}) \tag{2.30}$$

with

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}^{(i)} \boldsymbol{H}^{\mathrm{T}} \left(\boldsymbol{S}_k^{(i)}\right)^{-1}, \tag{2.31}$$

$$\boldsymbol{n}_k^{(i)} = \boldsymbol{H}\boldsymbol{m}_{k|k-1}^{(i)}, \tag{2.32}$$

$$\boldsymbol{S}_k^{(i)} = \boldsymbol{H}\boldsymbol{P}_{k|k-1}^{(i)} \boldsymbol{H}^{\mathrm{T}} + \boldsymbol{R} \tag{2.33}$$

is the corresponding Kalman filter update (see, e.g., [TBF05]). A linear measurement model for a partially observed state is given as an example in Appendix B.2.

The algorithm proceeds by generating a single birth component in line 16. This birth component is generated as follows. If at time k on average $\lambda_{\mathrm{b},k}$ objects enter the surveillance volume where they are always detected and distributed according

to $p_{\mathrm{b},k}(\boldsymbol{z})$, then a Gaussian component is generated for each measurement $\boldsymbol{z}$ such that

$$\begin{aligned} \tau_k^{(L_{k|k-1}+1)} &= \lambda_{\mathrm{b},k}\, p_{\mathrm{b},k}(\boldsymbol{z}) \\ &= \lambda_{\mathrm{b},k} \int p_k(\boldsymbol{z} \mid \boldsymbol{x}) \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}_k^{(L_{k|k-1}+1)}, \boldsymbol{P}_k^{(L_{k|k-1}+1)})\, \mathrm{d}\boldsymbol{x}. \end{aligned} \tag{2.34}$$

In practice, this is implemented by drawing the component mean $\boldsymbol{m}_k^{(L_{k|k-1}+1)}$ and covariance $\boldsymbol{P}_k^{(L_{k|k-1}+1)}$ from an inverse measurement model $p_k(\boldsymbol{x} \mid \boldsymbol{z})$. An example of such a model is given in Appendix B.3. The general idea followed here is to initialize the observed components of the state, e.g., the position, to the measured values and initialize the remaining components, e.g., velocities, to zero mean. The procedure is finalized by computing the updated mixture weights in lines 17–19.

The updated intensity is computed in lines 23 to 29 by taking the union of the missed detection and measurement-corrected components. Therefore, the updated intensity contains $L_k = L_{k|k-1} + L_{k|k-1} \cdot M_k$ components in total, where $L_{k|k-1}$ is the number of mixture components in the predicted intensity and M_k is the number of measurements. In practice, most of the updated components have negligible weights because the observation likelihood is peaked around the predicted object location. Then, many of the computed update components have small weights.

Algorithm 2 Gaussian mixture implementation of the PHD filter update step

procedure GMPHD_MISSED_DETECTIONS($\{w_{k|k-1}^{(i)}, \boldsymbol{m}_{k|k-1}^{(i)}, \boldsymbol{P}_{k|k-1}^{(i)}\}_{i=1}^{L_{k|k-1}}$)
  for $i = 1, \ldots, L_{k|k-1}^{(i)}$ **do**
    $w_k^{(i)} = (1 - p_{\mathrm{D}})\, w_{k|k-1}^{(i)}$
    $\boldsymbol{m}_k^{(i)} = \boldsymbol{m}_{k|k-1}^{(i)}$
    $\boldsymbol{P}_k^{(i)} = \boldsymbol{P}_{k|k-1}^{(i)}$
  end for
  return $\{w_k^{(i)}, \boldsymbol{m}_k^{(i)}, \boldsymbol{P}_k^{(i)}\}_{i=1}^{L_{k|k-1}}$
end procedure

procedure GMPHD_CORRECT($\{w_{k|k-1}^{(i)}, \boldsymbol{m}_{k|k-1}^{(i)}, \boldsymbol{P}_{k|k-1}^{(i)}\}_{i=1}^{L_{k|k-1}}$, $\boldsymbol{z}$)
  for $i = 1, \ldots, L_{k|k-1}$ **do**
    $\boldsymbol{m}_k^{(i)}, \boldsymbol{P}_k^{(i)} = \mathrm{KALMAN\_UPDATE}(\boldsymbol{m}_{k|k-1}^{(i)}, \boldsymbol{P}_{k|k-1}^{(i)}, \boldsymbol{z})$
    $l_k^{(i)} = \int p_k(\boldsymbol{z} \mid \boldsymbol{x}) \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}_{k|k-1}^{(i)}, \boldsymbol{P}_{k|k-1}^{(i)})\, \mathrm{d}\boldsymbol{x}$
    $\tau_k^{(i)} = p_{\mathrm{D}}\, l_k^{(i)} w_{k|k-1}^{(i)}$
  end for
  $\boldsymbol{m}_k^{(L_{k|k-1}+1)}, \boldsymbol{P}_k^{(L_{k|k-1}+1)}, \tau_k^{(L_{k|k-1}+1)} = \mathrm{BIRTH\_COMPONENT}(\boldsymbol{z})$
  for $i = 1, \ldots, L_{k|k-1+1}^{(i)}$ **do**
    $w_k^{(i)} = \dfrac{\tau_k^{(i)}}{c_k(\boldsymbol{z}) + \sum_{i=1}^{L_{k|k-1}+1} \tau_k^{(i)}}$
  end for
  return $\{w_k^{(i)}, \boldsymbol{m}_k^{(i)}, \boldsymbol{P}_k^{(i)}\}_{i=1}^{L_{k|k-1}+1}$
end procedure

procedure GMPHD_UPDATE($X_{k|k-1} = \{w_{k|k-1}^{(i)}, \boldsymbol{m}_{k|k-1}^{(i)}, \boldsymbol{P}_{k|k-1}^{(i)}\}_{i=1}^{L_{k|k-1}}$, Z_k)
  $X_{\mathrm{L},k} = \mathrm{GMPHD\_MISSED\_DETECTIONS}(X_{k|k-1})$
  for $j = 1, \ldots, M_k$ **do**
    $X_{\mathrm{U},k}^{(j)} = \mathrm{GMPHD\_CORRECT}(X_{k|k-1}, \boldsymbol{z}_{k,j})$
  end for
  return $X_{\mathrm{L},k} \cup X_{\mathrm{U},k}^{(1)} \cup \ldots \cup U_{\mathrm{U},k}^{(M_k)}$
end procedure

Algorithm 3 Gaussian mixture PHD filter component pruning [VM06]

procedure PRUNE_COMPONENTS($\{w_k^{(i)}, \boldsymbol{m}_k^{(i)}, \boldsymbol{P}_k^{(i)}\}_{i=1}^{L_k}$)
  $I = \{i \in \{1, \dots, L_k\} \mid w_k^{(i)} \geq w_{\min}\}$
  $l = 0$
  while $I \neq \emptyset$ **do**
    $l := l + 1$
    $j := \underset{i}{\operatorname{argmax}}\, w_k^{(i)}$
    $G := \{i \in I \mid \left(\boldsymbol{m}_k^{(i)} - \boldsymbol{m}_k^{(j)}\right) \left(\boldsymbol{P}_k^{(j)}\right)^{-1} \left(\boldsymbol{m}_k^{(i)} - \boldsymbol{m}_k^{(j)}\right)^{\mathrm{T}} < U\}$
    $\tilde{w}_k^{(l)} = \sum_{i \in L} w_k^{(i)}$
    $\tilde{\boldsymbol{m}}_k^{(l)} = \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in G} w_k^{(i)} \boldsymbol{m}_k^{(i)}$
    $\tilde{\boldsymbol{P}}_k^{(l)} = \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in G} w_k^{(i)} \left(\boldsymbol{P}_k^{(i)} + \left(\tilde{\boldsymbol{m}}_k^{(l)} - \boldsymbol{m}_k^{(i)}\right) + \left(\tilde{\boldsymbol{m}}_k^{(l)} - \boldsymbol{m}_k^{(i)}\right)^{\mathrm{T}}\right)$
    $I := I \setminus G$
  end while
  return $\{\tilde{w}_k^{(i)}, \tilde{\boldsymbol{m}}_k^{(i)}, \tilde{\boldsymbol{P}}_k^{(i)}\}_{i=1}^{l}$
end procedure

Pruning

During the GM-PHD filter update step, new components are added to the mixture for each observed measurement. This leads to an unbounded growth of mixture components as time progresses. Vo *et al.* [VM06] propose a simple pruning scheme to reduce the number of Gaussian components in the mixture. The idea is to remove those components that have negligible weights, i.e., a weight smaller than $w_{\min}$, and to merge components that are so close together that they can be accurately represented by a single Gaussian. The corresponding algorithm is given in Algorithm 3. Here, all components that have squared Mahalanobis distance smaller than U are merged into a single component. The algorithm is executed once at the end of each time step, subsequently to prediction and update.

2.4 Min-Cost Flow Problem

The min-cost flow problem describes a general optimization problem that has applications in many fields, including but not limited to distributed systems planning,

human resource management, transportation, and computer vision (for an overview of different applications, see [AMO93, Chapter 9]). The problem is well studied and specialized algorithms exploit the structure of the underlying task to find solutions efficiently. Therefore, the min-cost flow problem provides a powerful and efficient inference framework for many applications. In this section, the min-cost flow problem is presented in a general form. Later in Chapter 4, the min-cost flow problem serves as the inference framework for solving the multiple object tracking problem.

2.4.1 Problem Formulation

In a general setting, the min-cost flow problem describes the search for the least cost shipment of some commodity through a network in order to satisfy a demand at certain nodes from available supply at other nodes [AMO93, p. 4]. This network is a directed acyclic graph $G(V, E)$ where each edge $(u, v) \in E$ has an associated capacity $m(u, v) > 0$ which indicates the maximum amount of commodity that can be pushed along from u to v, an assigned flow $f(u, v) \geq 0$ that is the amount of commodity that is shipped along the edge, and a cost per unit of flow $c(u, v)$ that may take on negative values to indicate that traversing an edge produces a certain amount of profit instead of incurring a penalty. Associated with every node $v \in V$ is a value $b(v)$ which expresses the balance between supply and demand. The balance is positive if the node supplies commodity and negative if the node demands commodity. The min-cost flow problem can be stated by the following linear program:

$$\begin{aligned} \text{minimize} \quad & \sum_{(u,v)\in E} c(u,v) \cdot f(u,v) \\ \text{subject to} \quad & \sum_{w:(v,w)\in E} f(v,w) - \sum_{u:(u,v)\in E} f(u,v) = b(v) \quad \forall v \in V \\ & 0 \leq f(u,v) \leq m(u,v) \qquad\qquad \forall (u,v) \in E \end{aligned} \tag{2.35}$$

The first constraint ensures *flow conservation*. It is satisfied if all nodes $v \in V$ consume the amount of commodity that is expressed by balance $b(v)$ and pass on the remaining flow to their successors. The second constraint is called *capacity constraint*. It is satisfied if all edges $(u, v) \in E$ transport no more commodity than they can carry. A solution to the min-cost flow problem may not exist if the capacity of the network is insufficient to satisfy all demands given the supply. However, if it exists, the solution can be found in polynomial time [AMO93, Chapter 10].

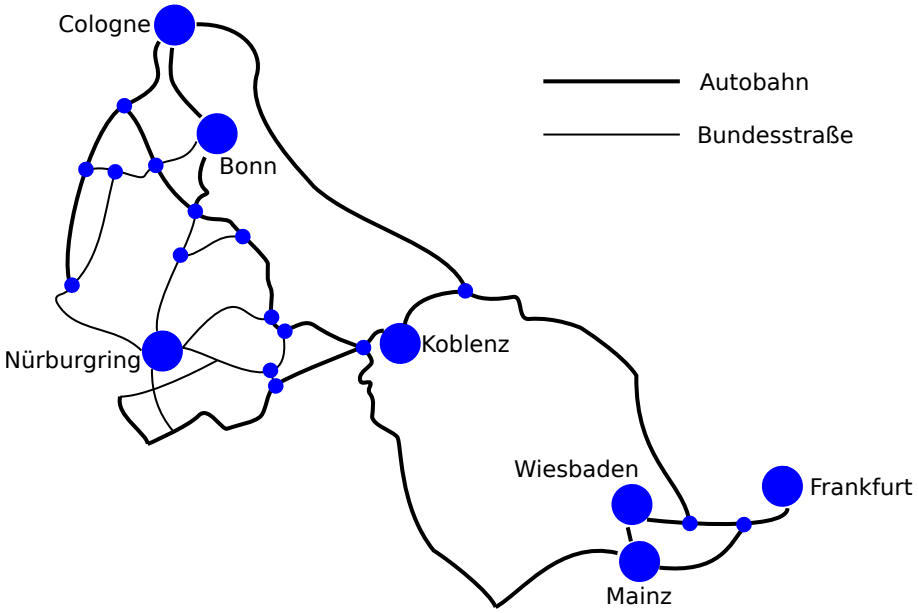
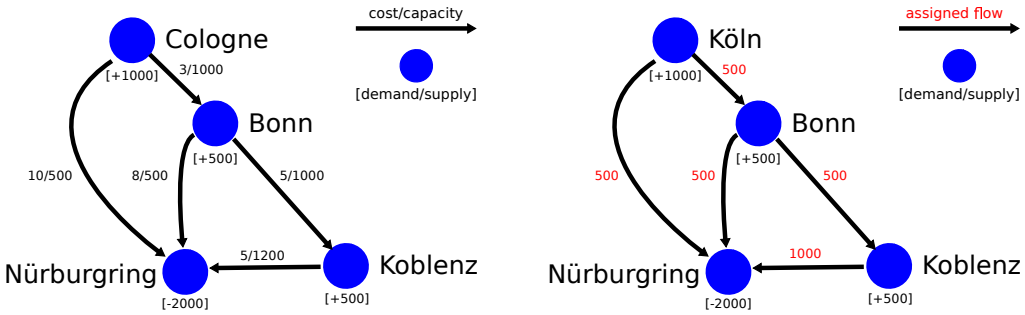


Figure 2.2: The Nürburgring is a well-known race track in Germany that is host of public events with over 100 000 visitors. A road network connects the Nürburgring with major cities in its surrounding.

2.4.2 Example

The following example illustrates the min-cost flow problem on a traffic routing task. Figure 2.2 shows the route network that connects the Nürburgring, a well-known race track in the surrounding of Koblenz, with its neighboring cities. This road network consists of large-capacity interstates (Autobahn) and highways of smaller capacities (Bundesstraße). During major public events, thousands of visitors travel to and from the Nürburgring via its neighboring cities, causing traffic jams and delays on this road network. With planning ahead of time, traffic jams and resulting delays can be avoided by creating a routing plan that directs vehicles towards their destination under consideration of road capacities.

A simplified version of the full road network is shown in Figure 2.3a. This simplified network shall serve as basis for the following example. Each node in the network is annotated with the number of vehicles that start (positive balance) or terminate (negative balance) at the corresponding location. In this example, 1000 vehicles start their journey in Cologne, 500 in Bonn, and another 500 in Koblenz. Since all traffic is routed towards the Nürburgring, the balance at the corresponding node has a value of -2000 vehicles. Edges in the graph are annotated with a cost that corresponds to the time of travel and a capacity that corresponds to the maximum number of vehicles that can travel on the road between the linked



(a) Flow network **(b)** Assigned flow

Figure 2.3: Illustration (a) shows a simplified flow network that connects the Nürburgring with major cities in is surrounding. Each node is annotated with a balance. Edges are annotated with their corresponding cost and capacity. Illustration (b) shows the min-cost flow solution of the corresponding traffic routing problem.

locations. Thus, the objective of the given min-cost flow problem is to minimize the total time of travel of all vehicles under consideration of road capacities. For the given flow network, the shortest path from any of the surrounding cities towards the Nürburgring follows the direct connection towards the destination. However, routing all vehicles on their direct path produces an infeasible solution due to limited capacity on the road between Cologne and Nürburgring. The solution obtained by solving the min-cost flow problem (2.35) for the given network is visualized in Figure 2.3b. The corresponding optimal routing plan can be stated as follows.

1. Route 500 of the 1000 vehicles along the direct path from Cologne to Nürburgring, leaving 0 capacity on this road.

2. Re-route the remaining 500 vehicles towards Bonn, leaving a remaining capacity of 500 vehicles on this road.

3. Route 500 of the 1000 vehicles from Bonn (500 vehicles originate in Bonn, 500 have been re-routed from Cologne) directly to Nürburgring, leaving 0 capacity on this road.

4. Re-route the remaining 500 vehicles via Koblenz, leaving a remaining capacity of 500 vehicles on this road.

5. Route all remaining 500 vehicles that originate from Koblenz and the 500 vehicles that have been re-routed from Cologne and Bonn towards Nürburgring, leaving a capacity of 200 vehicles on this road.

Note that in the given example all costs are positive. Therefore, each vehicle that travels between two locations incurs a penalty. The optimal solution is found

when all demand and supply is satisfied by a routing that incurs the lowest penalty/minimum cost. In certain applications, edge costs may become negative. In this case, traversing an edge generates a certain profit that improves the solution. Such an application can be found in [AMO93, Chapter 9]. In the *hopping flight* example, a small commuter airline operates a plane that stops at several locations on its way to the destination. In this case, the airline would like to determine the number of passengers that should be transported between the locations in order to maximize its profit. In this case, edge costs are negative such that each transported passenger improves the overall solution. Since the min-cost flow problem is not constrained to positive edge costs, this type of task can be solved in the min-cost flow framework just as well.

Chapter 3

Deep Cosine Metric Learning

The data association problem in multiple object tracking refers to the task of finding the correct association between new detections at time k and existing detections from previous times 1 to $k-1$. This problem shares similar characteristics to an image retrieval task where the correct association between a given query image and a gallery of unlabeled images must be established. Due to availability of large datasets, such image retrieval tasks have received considerate attention as a play ground for deep learning methods [CHL05, HRBLM07, SKP15, ZBS+16]. The problem is typically tackled in a metric learning framework where during training a suitable feature representation is learned to follow a predefined similarity metric. At test time, this metric is then used to rank the similarity of gallery images to the given query. This chapter focuses on person re-identification as an image retrieval task that shares strong similarity to people tracking. A representation space that has been trained in the person re-identification domain should be invariant to pose variations, changing background, and lighting conditions. Therefore, such a feature representation is also an ideal candidate to be employed in the tracking application for integration of appearance information.

The content of this chapter has been developed as part of a tracking system which has been published in [WBP17a]. A separate publication details the learning framework [WB18]. The remaining text is structured as follows. Section 3.1 provides an introduction to person re-identification and metric learning. A summary of related literature specific to this problem is provided in Section 3.2. A joint classification and metric learning framework is presented in Section 3.3. The concrete network architecture that will be used throughout experiments in this thesis is described in Section 3.4. A performance evaluation of the learning framework and network architecture is provided in Section 3.5.

3.1 Introduction

Person re-identification is a video surveillance problem. Given a query image, the task is to search in a gallery for images that show the same person. The search must be performed from a single camera shot because neither the person in the query nor the identities in the gallery are known to the re-identification system. As gallery images are usually taken from different cameras at different points in time, the system must deal with pose variations, different lighting conditions, and changing background. The problem is usually addressed in a metric learning framework. On a set of separate training identities, a feature representation is learned to reflect the task objective. At test time, the search for images of the same identity is performed using nearest neighbor queries. A feature representation that reflects the task objective is invariant to visual distractions such as pose changes, lighting, and background and at the same time follows a predefined metric that can be used for nearest neighbor queries.

Due to the annotation effort that is necessary to set up a person re-identification dataset, until recently only a limited amount of labeled images was available. This has changed with publication of the Market 1501 [ZST+15] and MARS [ZBS+16] datasets. MARS contains over one million images that have been annotated in a semi-supervised fashion. The data has been generated using a multi-target tracker that extracts short, reliable trajectory fragments that were subsequently annotated to consistent object trajectories. This annotation procedure not only leads to larger amount of data, but also puts the dataset closer to real-world applications where people are more likely extracted by application of a person detector rather than manual cropping.

Much like in other vision tasks, deep learning has become the predominant paradigm to person re-identification since the advent of larger datasets. Yet, the problem remains challenging and far from solved. In particular, there is an ongoing discourse over the performance of direct metric learning objectives compared to approaching the training procedure indirectly in a classification framework. The difference between the two methodologies is as follows. Metric learning objectives encode the similarity metric directly into the training objective. Classification-based methods train a classifier on the set of training identities and then use the underlying feature representation of the network to perform nearest neighbor queries. On the one hand, in the past direct metric learning objectives have suffered from undesirable properties that can hinder optimization, such as non-smoothness or missing contextual information about the neighborhood structure [RPDB16]. On the other hand, these problems have been approached with success in more recent publications [OSXJS16, HBL17]. Nevertheless, with similarity defined solely based on class membership, it remains arguable if direct metric learning has a clear advantage over training in a classification regime. In this setting, metric learning

is often reduced to minimizing the distance between samples of the same class and forcing a margin between samples of different classes [CHL05, HBL17]. A classifier that is set up with care might decrease intra-class variance and increase inter-class variance in a similar way to direct metric learning objectives.

The purpose of this chapter is to twofold: First, a light-weight Convolutional Neural Network (CNN) architecture is trained on the MARS dataset to be used in the people tracking application later. Second, a re-parametrization of the standard softmax classifier is presented that enforces a cosine similarity metric on the representation space. This parametrization is explored as an alternative to direct metric learning objectives that have suffered from inefficient training behavior in the past.

3.2 Related Work

Metric Learning CNNs have shown impressive performance on large scale computer vision problems and the representation space underlying these models can be successfully transferred to tasks that are different from the original training objective [DJV+14, SRASC14]. Therefore, in classification applications with few training examples a task-specific classifier is often trained on top of a general purpose feature representation that was learned beforehand on ImageNet [KSH12] or MS COCO [LMB+14]. There is no guarantee that the representation of a network which has been trained with a softmax classifier can directly be used in an image retrieval task such as person re-identification because the representation does not necessarily follow a certain (known) metric to be used for nearest-neighbor queries. Nevertheless, several successful applications in face verification and person re-identification exist [TYRW14, XLOW16, ZZS+17]. In this case, a softmax classifier is trained to discriminate the identities in the training set. When training is finished, the classifier is stripped of the network and distance queries are made using cosine similarity or Euclidean distance on the final layer of the network. If, however, the feature representation cannot be used directly, an alternative is to find a metric subspace in a post processing step [KHW+12, LHZL15].

Deep metric learning approaches encode notion of similarity directly into the training objective. The most prominent formulations are siamese networks with contrastive [CHL05] and triplet [WS09] loss. The contrastive loss minimizes the distance between samples of the same class and forces a margin between samples of different classes. Effectively, this loss pushes all samples of the same class towards a single point in representation space and penalizes overlap between different classes. The triplet loss relaxes the contrastive formulation to allow samples to move more freely as long as the margin is kept. Given an anchor point, a point of the same class, and a point of a different class, the triplet loss forces the distance to the

point of the same class to be smaller than the distance to the point of the different class plus a margin. Both formulations have been applied successfully to metric learning problems (e.g., [SKP15, VHW16, HBL17]), but the success has long been dependent on an intelligent pair/triplet sampling strategy. Many of the possible choices of pairs and triplets that one can generate from a given dataset contain little information about the relevant structures by which identities can be discriminated. If the wrong amount of hard to distinguish pairs/triplets are incorporated into each batch, the optimizer either fails to learn anything meaningful or does not converge at all. Development of a well-working sampling strategy can be a complex and time-consuming task, effectively limiting the practical applicability of siamese networks. A second issue related to the contrastive and triplet loss stems from the hard margin that is enforced between samples of different classes. The hard margin leads to a non-smooth objective function that is harder to optimize, because only few examples are presented to the optimizer at each iteration and there can be strong disagreement between different batches [RPDB16]. These problems have been addressed recently. For example, Song *et al.* [OSXJS16] formulate a smooth upper bound of the original triplet loss that can be implemented by drawing informative samples from each batch directly on GPU. A similar formulation of the triplet loss where the hard margin is replaced by a soft margin has shown to perform well on a person re-identification problem [HBL17].

Apart from siamese network formulations, the magnet loss [RPDB16] has been formulated as an alternative to overcoming many of the related issues. The loss is formulated as a negative log-likelihood ratio between the correct class and all other classes, but also forces a margin between samples of different classes. By operating on entire class distributions instead of individual pairs or triplets, the magnet loss potentially converges faster and leads to overall better solutions. The center loss [WZLQ16] has been developed in an attempt to combine classification and metric learning. The formulation utilizes a combination of a softmax classifier with an additional term that forces compact classes by penalizing the distance of samples to their class mean. A scalar hyperparameter balances the two losses. Experiments suggest that this joint formulation of classification and metric learning produces state-of-the-art results.

Person Re-Identification With availability of large datasets, person re-identification has become an application domain of deep metric learning. Several CNN architectures have been designed specifically for this task. Most of them focus on mid-level features and try to deal with pose variations and viewpoint changes explicitly by introducing special units into the architecture. For example, Li *et al.* [LZXW14] propose a CNN with a special patch matching layer that captures the displacement between mid-level features. Ahmed *et al.* [AJM15] capture feature

displacements similarly by application of special convolutions that compute the difference between neighborhoods in the feature map of two input images. The gating functions in the network of Varior *et al.* [VHW16] compare features along a horizontal stripe and output a gating mask to indicate how much emphasis should be paid to the local patterns. Finally, in [VSL+16] a recurrent siamese neural network architecture is proposed that processes images in rows. The idea behind the recurrent architecture is to increase contextual information through sequential processing.

More recent work on person re-identification suggests that baseline CNN architectures can compete with their specialized counter parts. In particular, the current best performing method on the MARS is a conventional residual network [HBL17]. Application of baseline CNN architectures can be beneficial if pre-trained models are available for fine-tuning to the person re-identification task. Influence of pre-training on overall performance is studied in [ZBS+16]. They report between 9.5% and 10.2% recognition rate is due to pre-training on ImageNet [KSH12].

3.3 Joint Classification and Metric Learning

This section presents a re-parameterization of the softmax classifier that enforces a cosine similarity metric on the representation space. This parameterization is explored as a way to train classifier and metric representation space jointly, motivated by the aforementioned issues related to training siamese networks. The section starts with a formal problem definition and a review of the standard softmax classifier. Then, a suitable re-parameterization for metric learning is presented.

3.3.1 Problem Formulation

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ of N training images $\boldsymbol{x}_i \in \mathbb{R}^D$ and associated class labels $y_i \in \{1, \dots, C\}$, metric learning refers to the problem of finding a parametric encoder function $\boldsymbol{r} = f_\Theta(\boldsymbol{x})$ with parameters Θ which projects input images $\boldsymbol{x} \in \mathbb{R}^D$ into a feature representation $\boldsymbol{r} \in \mathbb{R}^d$ that follows a predefined notion of similarity. If the encoder function is a deep neural network, then this problem is referred to as deep metric learning.

In person re-identification, similarity is expressed in terms of class membership. Therefore, according to *some* symmetric measure $\odot$, the similarity between features of the same class should be larger than similarity between features of different classes

$$y_i = y_j \wedge y_i \neq y_k \Leftrightarrow \boldsymbol{r}_i \odot \boldsymbol{r}_j < \boldsymbol{r}_i \odot \boldsymbol{r}_k, \quad \forall i, j, k \in \{1, \dots, N\}, \tag{3.1}$$

such that at test time queries can be made using neighbor search. The benefit of a metric learning formulation over a standard classification setting is that, if the

similarity measure $\odot$ is known, the feature representation generalizes to unseen classes. At best, the encoder can be transferred to new datasets without additional training. For example, a representation that is trained on a person re-identification dataset can potentially be employed for data association in an object tracking application that contains previously unseen object identities [LTCFS16, WBP17a].

This section focuses on a directional representation space where similarity is measured in terms of the angle between samples. More specifically, for any two unit-length features $\boldsymbol{r}_i, \boldsymbol{r}_j \in \mathbb{R}^d$ with $\|\boldsymbol{r}_i\|_2 = \|\boldsymbol{r}_j\|_2 = 1$, the cosine similarity

$$\boldsymbol{r}_i \odot \boldsymbol{r}_j = \boldsymbol{r}_i^{\mathrm{T}} \boldsymbol{r}_j. \tag{3.2}$$

is used to measure similarity. This metric is primarily chosen because it allows the metric learning problem to be embedded in a classification setting. Let $g_\Omega : \mathbb{R}^d \to \{1, \dots, C\}$ denote a classifier with parameter Ω that maps from representation space to one of the C classes. Then the feature encoder $f_\Theta(\boldsymbol{x})$ is trained jointly with the classifier by minimization of a classification loss

$$\underset{\Omega,\Theta}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}_{g_\Omega}(y_i, f_\Theta(\boldsymbol{x}_i)). \tag{3.3}$$

At test time, the classifier is stripped of the network and the feature representation is used on its own to perform nearest neighbor queries. Note that, because the classification loss does not optimize the metric learning objective (3.1) directly, the classifier must be set up in a way that good classification accuracy leads to compact classes in a representation space that follows the predefined metric. The following section reviews the standard softmax classifier in this regard. Then, a re-parametrization that leads to compact classes in a directional representation space is presented.

3.3.2 Standard Softmax Classifier

The standard approach to classification in the deep learning setting is to process input images by a CNN and place a softmax classifier on top of the network to obtain probability scores for each of the C classes. The softmax classifier $g_\Omega(\boldsymbol{r}) = \operatorname{argmax}_k p(y = k \mid \boldsymbol{r})$ chooses the class with maximum probability according to a parametric function

$$p(y = k \mid \boldsymbol{r}) = \frac{\exp\left({\boldsymbol{w}_k}^{\mathrm{T}} \boldsymbol{r} + b_k\right)}{\sum_{n=1}^{C} \exp\left({\boldsymbol{w}_n}^{\mathrm{T}} \boldsymbol{r} + b_n\right)} \tag{3.4}$$

with parameters $\Omega = \{\boldsymbol{w}_1, b_1, \dots, \boldsymbol{w}_C, b_C\}$. For the special case of $C = 2$ classes this formulation is equivalent to logistic regression. Further, the specific choice of

functional form can be motivated from a generative perspective on the classification problem. If the class-conditional densities are Gaussian

$$p(\boldsymbol{r} \mid y = k) = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{r} - \boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{r} - \boldsymbol{\mu}_k)\right) \tag{3.5}$$

with shared covariance $\boldsymbol{\Sigma}$, then the posterior class probability can be computed by Bayes' rule

$$p(y = k \mid \boldsymbol{r}) = \frac{p(\boldsymbol{r} \mid y = k)p(y = k)}{\sum_{n=1}^{C} p(\boldsymbol{r} \mid y = n)p(y = n)} = \frac{\exp\left(\boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{r} + b_k\right)}{\sum_{n=1}^{C} \exp\left(\boldsymbol{w}_n^{\mathrm{T}} \boldsymbol{r} + b_n\right)} \tag{3.6}$$

with $\boldsymbol{w}_k = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k$ and $b_k = -\frac{1}{2}\boldsymbol{\mu}_k^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log p(y_i = k)$ [Bis06]. However, the softmax classifier is trained in a discriminative regime. Instead of determining the parameters of the class-conditional densities and prior class probabilities, the parameters Ω of the conditional class probabilities are obtained directly by minimization of a classification loss. Let $\mathbb{1}_{y=k}$ denote the indicator function that evaluates to 1 if y is equal to k and 0 otherwise. Then, the corresponding loss

$$\mathcal{L}_{g_\Omega}(y, \boldsymbol{r}) = -\sum_{k=1}^{C} \mathbb{1}_{y=k} \cdot \log\, p(y = k \mid \boldsymbol{r}) \tag{3.7}$$

is called cross-entropy loss because it minimizes the cross-entropy between the *true* label distribution $p(y = k) = \mathbb{1}_{y=k}$ and estimated probabilities of the softmax classifier $p(y = k \mid \boldsymbol{r})$. By minimization of the cross-entropy loss, parameters are chosen such that the estimated probability is close to 1 for the correct class and close to 0 for all other classes.

Figure 3.1 illustrates a classification problem with three classes. In Figure 3.1a, three Gaussian densities $p(\boldsymbol{r} \mid y)$ are shown together with the corresponding decision boundary. The posterior class probabilities of this scenario are shown in Figure 3.1b together with a set of hypothesized training examples. Whereas the Gaussian densities peak around a class mean, the posterior class probability is a function of the distance to the decision boundary. When the feature encoder is trained with the classifier jointly by minimization of the cross-entropy loss, the parameters of the encoder network are adapted to push samples away from the decision boundary as far as possible, but not necessarily towards their class mean that has been taken to motivate the specific functional form. This behavior is problematic for metric learning because similarity in terms of class membership is encoded in the orientation of the decision boundary rather than in the feature representation itself.

3.3.3 Cosine Softmax Classifier

With few adaptations, the standard softmax classifier can be modified to produce compact clusters in representation space. First, ℓ_2 normalization must be

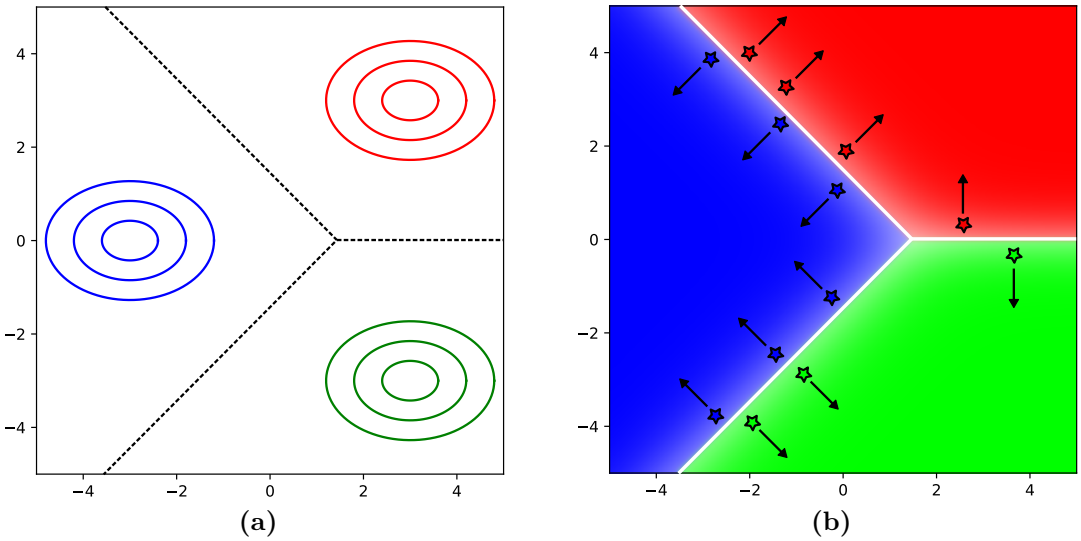


Figure 3.1: Plot (a) shows three Gaussian class-conditional densities (iso-contours) and the corresponding decision boundary (dashed lines). Plot (b) shows the conditional class probabilities (color coded) and a set of hypothesized training examples. The softmax classifier models the posterior class probabilities directly, without construction of Gaussian densities. By training with the cross-entropy loss, samples are pushed away from the decision boundary, but not necessarily towards a class mean.

applied to the final layer of the encoder network to ensure the representation is unit length $\|f_\Theta(\boldsymbol{x})\|_2 = 1, \forall \boldsymbol{x} \in \mathbb{R}^D$. Second, the weights must be normalized to unit-length as well, i.e., $\tilde{\boldsymbol{w}}_k = \boldsymbol{w}_k / \|\boldsymbol{w}_k\|_2, \forall k = 1, \ldots, C$. Then, the cosine softmax classifier can be stated by

$$p(y = k \mid \boldsymbol{r}) = \frac{\exp\left(\kappa \cdot \tilde{\boldsymbol{w}}_k^{\mathrm{T}} \boldsymbol{r}\right)}{\sum_{n=1}^{C} \exp\left(\kappa \cdot \tilde{\boldsymbol{w}}_n^{\mathrm{T}} \boldsymbol{r}\right)}, \tag{3.8}$$

where κ is a free scaling parameter. This parametrization has $C-1$ fewer parameters compared to the standard formulation (3.4) because the bias terms b_k have been removed, i.e., $\Omega = \{\kappa, \tilde{\boldsymbol{w}}_1, \ldots, \tilde{\boldsymbol{w}}_C\}$. Otherwise, the functional form resembles strong similarity to the standard parametrization and implementation is straight-forward. In particular, decoupling the length of the weight vector κ from its direction has been proposed before [SK16] as a way to accelerate convergence of stochastic gradient descent. Training itself can be carried out using the cross-entropy loss as usual since the cosine softmax classifier is merely a change of parametrization compared to the standard formulation.

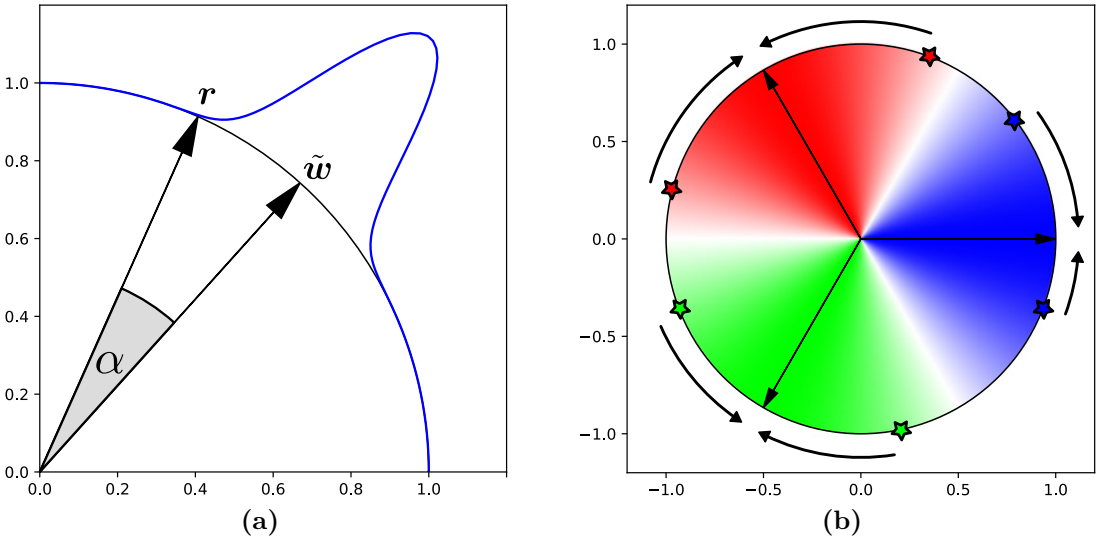


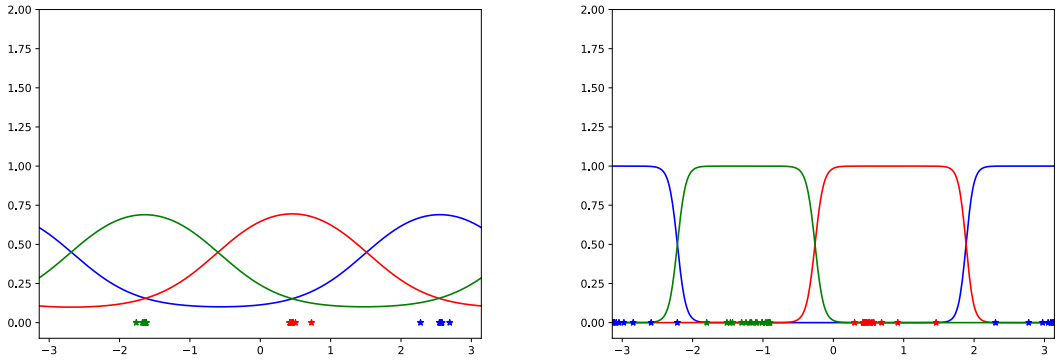
Figure 3.2: Plot (a) illustrates a von Mises-Fisher distribution. The probability density increases as the cosine of the angle α between sample $\boldsymbol{r}$ and mean direction $\tilde{\boldsymbol{w}}$ becomes smaller. Parameter κ controls the concentration of the distribution similar to how the standard deviation controls the spread of a Gaussian distribution. Plot (b) illustrates the posterior class probabilities (color coded) and decision boundary (white line) of the cosine softmax classifier for a problem with three classes. During training, all samples are pushed away from the decision boundary towards their parametrized class mean direction (indicated by an arrow).

The functional modeling of log-probabilities by $\kappa \cdot \tilde{\boldsymbol{w}}_k^{\mathrm{T}} \boldsymbol{r}$ can be motivated from a generative perspective as well. If the class-conditional likelihoods follow a von Mises-Fisher distribution

$$p(\boldsymbol{r} \mid y = k) = c_d(\kappa) \exp\left(\kappa \cdot \tilde{\boldsymbol{w}}_k^{\mathrm{T}} \boldsymbol{r}\right) \tag{3.9}$$

with shared concentration parameter κ and normalizer $c_d(\kappa)$, then Equation 3.8 is the posterior class probability under an equal prior assumption $p(y = k) = p(y = l)$, $\forall k, l \in \{1, \ldots, C\}$. The von Mises-Fisher distribution is an isotropic probability distribution on the $d-1$ dimensional sphere in $\mathbb{R}^d$ that peaks around mean direction $\tilde{\boldsymbol{w}}_k$ and decays as the cosine similarity decreases. This is illustrated in Figure 3.2a.

To understand why this parametrization enforces a cosine similarity on the representation space, observe that the log-probabilities are directly proportional to the cosine similarity between training examples and a parametrized class mean direction. By minimizing the cross-entropy loss, examples are pushed away from the decision boundary towards their parametrized class mean as illustrated in Figure 3.2b. In consequence, parameter vector $\tilde{\boldsymbol{w}}_k$ becomes a surrogate for all samples



(a) $\kappa = 1$ **(b)** $\kappa = 10$

Figure 3.3: Illustration of the free scaling parameter κ in a one dimensional problem with three classes. The conditional class probabilities are shown as colored functions. Optimized sample locations are visualized as stars at $y = 0$. A low κ value (a) leads to smoother functions with wider support such that samples are pushed into tight clusters. The shape becomes box-like for high values (b), allowing samples to move more freely within a region that is occupied by the class.

in class k. The scaling parameter κ controls the shape of the conditional class probabilities as illustrated in 3.3. A low value corresponds to smoother functions with wider support. A high κ value leads to conditional class probabilities that are box-like shaped around the decision boundary. This places a larger penalty on misclassified examples, but at the same time leaves more room for samples to move freely in the region of representation space that is occupied by its corresponding class. In this regard, the scale takes on a similar role to margin parameters in direct metric learning objectives. When the scale is left as a free parameter, the optimizer gradually increases its value as the overlap between classes reduces. A margin between samples of different classes can be enforced by regularizing the scale with weight decay.

3.4 Network Architecture

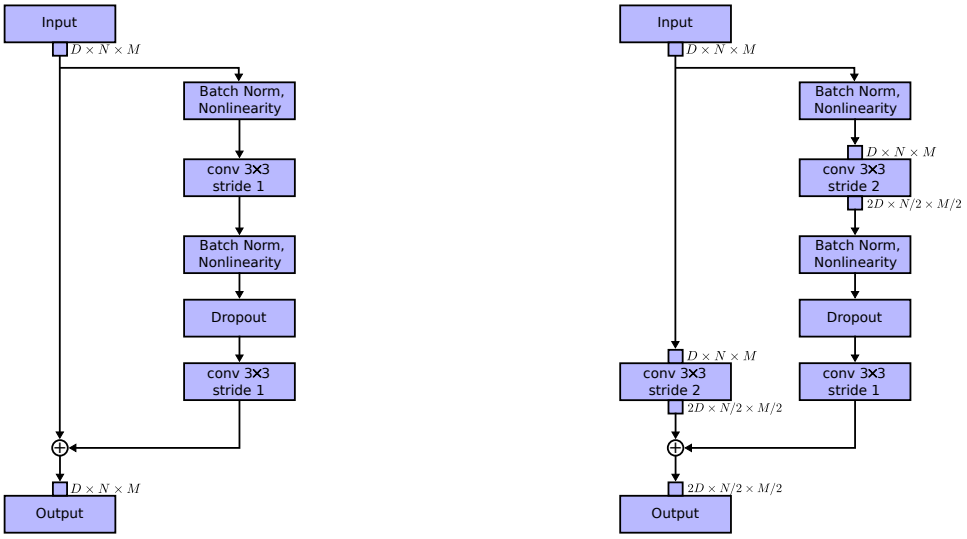
This section presents a light-weight CNN architecture for person re-identification and people tracking. The architecture is summarized in Table 3.1. Input images are rescaled to 128×64 and presented to the network in RGB color space. A series of convolutional layers reduces the size of the feature map to 16×8 before a global feature vector of length 128 is extracted by layer *Dense 10*. The final ℓ_2 normalization projects features onto the unit hypersphere for application of the cosine softmax classifier. The network contains several residual blocks that follow

Name	Patch Size/Stride	Output Size
Conv 1	$3 \times 3/1$	$32 \times 128 \times 64$
Conv 2	$3 \times 3/1$	$32 \times 128 \times 64$
Max Pool 3	$3 \times 3/2$	$32 \times 64 \times 32$
Residual 4	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 5	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 6	$3 \times 3/2$	$64 \times 32 \times 16$
Residual 7	$3 \times 3/1$	$64 \times 32 \times 16$
Residual 8	$3 \times 3/2$	$128 \times 16 \times 8$
Residual 9	$3 \times 3/1$	$128 \times 16 \times 8$
Dense 10		128
ℓ_2 normalization		128

Table 3.1: Overview of the CNN architecture. The final ℓ_2 normalization projects features onto the unit hypersphere.

the pre-activation layout proposed by He *et al.* [HZRS16b]. In general, a residual block maps the input $\boldsymbol{x}$ to an output $\boldsymbol{y}$ by $\boldsymbol{y} = h(i(\boldsymbol{x}) + g(\boldsymbol{x}))$ where $i(\boldsymbol{x})$ is an identity mapping, $g(\boldsymbol{x})$ is a residual function and $h(\boldsymbol{x})$ is a nonlinearity. In the pre-activation residual block, all non-linearities are moved into the residual function such that there is always a direct flow of information between the input and output, i.e., $\boldsymbol{y} = i(\boldsymbol{x}) + g(\boldsymbol{x})$. This has shown to ease learning and improve overall accuracy [HZRS16b]. The structure of the residual blocks is shown in Figure 3.4. The design follows the ideas of wide residual networks presented in [ZK16]: All convolutions are of size 3×3 and max pooling is replaced by convolutions of stride 2. When the spatial resolution of the feature map is reduced, then the number of channels is increased accordingly to avoid a bottleneck. Dropout and batch normalization are used as means of regularization.

Note that with in total 15 layers—including two convolutional layers in each residual block—the network is relatively shallow when compared to the current trend of ever deeper architectures [HZRS16b]. This decision has been made with the people tracking application in mind to permit application in online tracking scenarios and is backed up by the experiments of Zagoruyko and Komodakis [ZK16] that suggest shallow networks with increased width can be computationally more efficient and achieve similar accuracy than their deeper counterparts. Further, architectures that have been designed for person re-identification specifically [LZXW14, AJM15] put special emphasis on mid-level features. Therefore, the dense layer is added at a point where the feature map still provides *enough* spatial resolution. Exponential linear units [CUH15] are used as activation function in all layers.



(a) Dimensionality preserving

(b) Dimensionality reduction

Figure 3.4: Structure of pre-activation residual blocks. In (a) the output has the same dimensionality as the input. In (b) the spatial resolution is reduced by a factor of 2 and the number of channels is increased by a factor of 2. Input, output, and computational blocks that change the size of the feature map are annotated by input/output pins.

In total, the network has 2,800,864 parameters and one forward pass of 32 bounding boxes takes approximately 30 ms on an Nvidia GeForce GTX 1050 mobile GPU. Thus, this network is well suited for online tracking even on low-cost hardware.

3.5 Evaluation

This section presents an evaluation of the metric learning framework and network architecture on two publicly available person re-identification datasets.

3.5.1 Datasets and Evaluation Protocols

The two datasets have been selected for evaluation are large enough to train a CNN from scratch. A single-shot, cross-view evaluation protocol is adopted in all experiments, i.e., a single query image from one camera is matched against a gallery of images taken from different cameras. The gallery image ranking is established using cosine similarity or Euclidean distance, if appropriate. Training and test data splits are provided by the dataset authors. Additionally, 10% of the training data is split for hyperparameter tuning and early stopping. On both

datasets, cumulative matching characteristics (CMC) at rank 1 and 5 as well as mean average precision (mAP) are reported. The CMC rank k metric reports the frequency by which a matching image is contained in the top k images of the ranked gallery. If the precision at k is the number of correct items in the ranked gallery among all top k items, then average precision is the average precision over all k and mean average precision is the mean of average precision over all queries. For both metrics higher values indicate better performance. The scores are computed with evaluation software provided by the corresponding dataset authors.

Market 1501 The Market 1501 [ZST+15] dataset contains 1,501 identities and roughly 30 000 images taken from six cameras. The dataset is challenging due to frequent bounding box misalignment and additional false alarms in the test gallery. The standard evaluation protocol proposed for this dataset is followed here: Training and testing are carried out on provided data splits (751 identities for training, 750 for testing) using images that have been generated by a deformable part models detector. Additional distractor training images that are also provided by the authors are not used.

MARS The MARS [ZBS+16] dataset is an extension of Market 1501 that contains 1 261 identities and over 1 100 000 images. The data has been generated using a multi-target tracker that generates tracklets, i.e. short-term track fragments, which have then been manually annotated to consistent identities. Consequently, this dataset also contains significant bounding box misalignment and inaccurate labeling.

3.5.2 Baseline Methods

In order to assess the performance of the joint classification metric learning framework on overall performance, the network architecture is repeatedly trained with two baseline direct metric learning objectives.

Triplet loss The triplet loss [WS09] is defined over tuples of three examples $\boldsymbol{r}_a$, $\boldsymbol{r}_p$, and $\boldsymbol{r}_n$ that include a positive pair $y_a = y_p$ and a negative pair $y_a \neq y_n$. For each such triplet the loss demands that the difference of the distance between the negative and positive pair is larger than a pre-defined margin $m \in \mathbb{R}$:

$$\mathcal{L}_{\text{triplet}}(\boldsymbol{r}_a, \boldsymbol{r}_p, \boldsymbol{r}_n) = \left\{ \|\boldsymbol{r}_a - \boldsymbol{r}_p\|_2 - \|\boldsymbol{r}_a - \boldsymbol{r}_n\|_2 + m \right\}_+, \tag{3.10}$$

where $\{\}_+$ denotes the hinge function that evaluates to 0 for negative values and identity otherwise. In this experiment, a soft-margin version of the original triplet loss [HBL17] is used where the hinge is replaced by a soft plus function $\{x + m\}_+ =$

$\log(1+\exp(x))$ to avoid issues with non-smoothness [RPDB16]. Further, the triplets are generated directly on GPU as proposed by [HBL17] to avoid potential issues in the sampling strategy. Note that this particular triplet loss formulation has been used to train the current best performing model on MARS [HBL17].

Magnet loss The magnet loss has been proposed as an alternative to siamese loss formulations that works on entire class distribution rather than individual samples. The loss is a negative log-likelihood ratio measure that forces separation in terms of each sample's distance away from the means of other classes. In its original proposition [RPDB16] the loss takes on a multi-modal form. Here, a simpler, unimodal variation of this loss is employed because it better fits the single-shot person re-identification task:

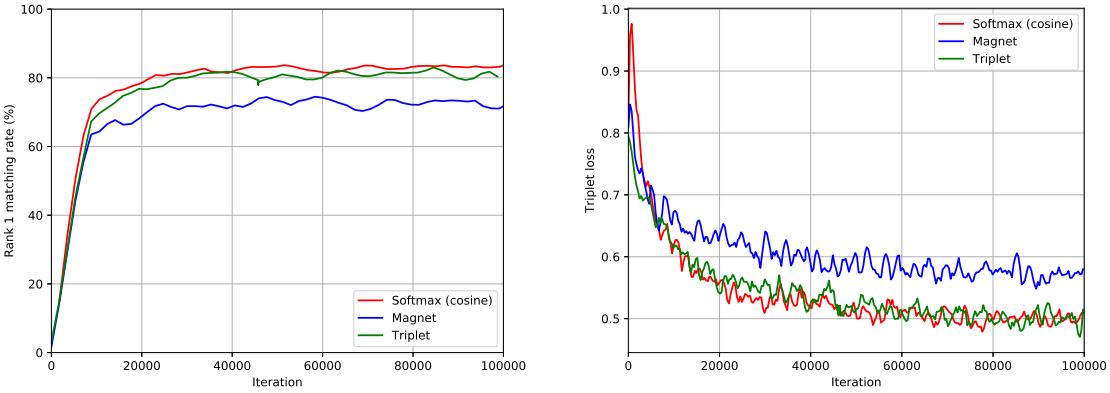
$$\mathcal{L}_{\text{magnet}}(y, \boldsymbol{r}) = \left\{ -\log \frac{\exp\left(-\frac{1}{2\hat{\sigma}^2}\|\boldsymbol{r} - \hat{\boldsymbol{\mu}}_y\|_2^2 - m\right)}{\sum_{k \in \bar{C}(y)} \exp\left(-\frac{1}{2\hat{\sigma}^2}\|\boldsymbol{r} - \hat{\boldsymbol{\mu}}_k\|_2^2\right)} \right\}_+, \tag{3.11}$$

where $\bar{C}(y) = \{1, \ldots, C\} \setminus \{y\}$, m is again a margin parameter, $\hat{\boldsymbol{\mu}}_y$ is the sample mean of class y, and $\hat{\sigma}^2$ is the variance of all samples away from their class mean. These parameters are computed on GPU for each batch individually.

3.5.3 Results

The results reported in this section have been established by training the network for a fixed number of 100 000 iterations using Adam [KB15]. The learning rate was set to 1×10^{-3}. As can be seen in Figure 3.5, all configurations have fully converged at this point. The network was regularized with a weight decay of 1×10^{-8} and dropout inside the residual units with probability 0.4. The margin of the magnet loss has been set to $m = 1$ and the cosine softmax scale κ was left as a free parameter for the optimizer to tune, but regularized with a weight decay of 1×10^{-1}. The batch size was fixed to 128 images. Gallery rankings have been established using Euclidean distance in case of magnet and triplet loss and cosine similarity in case of the softmax classifier. To increase variability in the training set, input images have been randomly flipped, but no random resizing or cropping has been performed.

Training Behavior Figure 3.5a shows the rank 1 matching rate on the validation set of MARS as a function of training iterations. The results obtained on Market 1501 are omitted here since the training behavior is similar. The network trained with the cosine softmax classifier achieves overall best performance, followed by the network trained with soft-margin triplet loss. The best validation performance of the softmax network is reached at iteration 49 760 with rank 1 matching rate 84.92%.



(a) Evolution of validation set accuracy

(b) Evolution of triplet loss on training set

Figure 3.5: The left plot shows the rank 1 matching accuracy on the validation set as a function of training iterations on MARS [ZBS+16]. The right plot shows how the triplet loss evolves on the training set. Note that the triplet loss is only used as training objective for the triplet network. For the other two methods the loss is only monitored to obtain insight into the training behavior.

The best performance of the triplet loss network is reached at iteration 86 329 with rank 1 matching rate 83.23%. The magnet loss network reaches its best performance at iteration 47 677 with rank 1 matching rate 77.34%. Overall, the convergence behavior of the three losses is similar, but the magnet loss falls behind on final model performance. In its original implementation [RPDB16] the authors sample batches such that similar classes appear in the same batch. For practical reasons such more informative sample mining has not been implemented. Instead, a fixed number of images per individual was randomly selected for each batch. Potentially, the magnet loss suffers from this less informative sampling strategy more than the other two losses.

During all runs, the triplet loss has been monitored as an additional information source on training behavior. Figure 3.5b plots the triplet loss as a function of training iterations. Note that the triplet loss has not been used as a training objective in runs softmax (cosine) and magnet. Nevertheless, both minimize the triplet loss indirectly. In particular the softmax classifier is quite efficient at minimizing the triplet loss. During iterations 20 000 to 40 000 the triplet loss drops even slightly faster when optimization is carried out with the softmax classifier rather than optimizing the triplet loss directly. Therefore, the cosine softmax classifier effectively enforces a similarity metric onto the representation space.

Re-Identification Performance All three networks have been evaluated on the provided test splits of the Market 1501 and MARS datasets. Figure 3.2 summarizes

Method	Market 1501			MARS		
	Rank 1	Rank 5	mAP	Rank 1	Rank 5	mAP
TriNet [HBL17][a,b]	84.92	94.21	69.14	79.80	91.36	67.70
LuNet [HBL17][b]	81.38	92.34	60.71	75.56	89.70	60.48
IDE + XQDA [ZBS+16][a,†]	73.60	-	49.05	65.30	82.00	47.60
MSCAN [LCZH17]	-	-	-	71.77	86.57	56.06
P-QAN [LYO17]	-	-	-	73.73	84.90	51.70
CaffeNet [ZHW+17]	-	-	-	70.60	90.00	50.70
DaF [YZBB17][a]	82.30	-	72.42	-	-	-
JLML [LZG17][a]	85.10	-	65.50	-	-	-
GoogLeNet [ZLWZ17][a]	81.00	-	63.40	-	-	-
SVDNet [SZDW17][a]	82.30	-	62.10	-	-	-
Gated CNN [VHW16][b]	65.88	-	39.55	-	-	-
Recurrent CNN [VSL+16][b]	61.60	-	35.30	-	-	-
Triplet[b]	74.88	88.72	53.04	71.31	85.55	54.30
Magnet	61.10	81.03	40.12	63.13	81.16	45.45
Cosine softmax	79.10	91.06	56.68	72.93	86.46	56.88

Table 3.2: Performance comparison on the Market 1501 [ZST+15] and MARS [ZBS+16] datasets. †: Numbers for Market 1501 taken from [HBL17]. [a]: Pre-trained on ImageNet. [b]: Siamese network.

the results and provides a comparison against the state of the art. The training behavior and rank 1 matching rates that have been observed on the validation set manifest in the final performance on the provided test splits. On both datasets, the cosine softmax network achieves the best results, followed by the siamese network. The gain in mAP due to the softmax loss is 3.64 on the Market 1501 dataset and 2.58 on MARS. This is a relative gain of 6.8% and 4.7% respectively. The state of the art contains several alternative siamese architectures that have been trained with a contrastive or triplet loss, marked by [b] in the table. The performance of these networks is not always directly comparable because the models have varying capacity. However, the LuNet of Hermans *et al.* [HBL17] is a residual network with roughly double the capacity of the proposed architecture. The reported numbers have been generated with test-time data augmentation that accounts for approximately 3 mAP points according to the corresponding authors. Thus, the proposed network comes in close range at much lower capacity. Further, the method of [ZBS+16] refers to a CaffeNet that has been trained with the conventional softmax classifier and the metric subspace has been obtained in a separate post processing step. The results suggest that the proposed joint classification and metric learning framework not only enforces a metric onto the representation space, but also that encoding the metric directly into the classifier works better than treating it in a subsequent post processing step.

The best performing method on Market 1501 has a 15.84 points higher mAP score than the cosine softmax network. On MARS, the best performing method achieves a 10.82 higher mAP. This is a large-margin improvement over the proposed network, which shows that considerate improvement is possible by application of larger capacity architectures with additional pre-training. For example, TriNet *et al.* [HBL17] is a ResNet-50 [HZRS16a] with 25.74 million parameters that has been pre-trained on ImageNet [KSH12]. Clearly, deeper networks have an advantage over the small architecture used in this work, but they are not applicable in online tracking scenarios due to the computational resources that they require.

The best performing network that has been trained from scratch, i.e., without pre-training on ImageNet, is the LuNet of Hermans *et al.* [HBL17]. With approximately 5 million parameters the network is still roughly double the size, but the final model performance in terms of mAP is only 4.03 and 3.6 points higher (including test-time augmentation). Therefore, the proposed architecture provides a good trade off between computational efficiency and re-identification performance considering the targeted tracking application.

Learned Embedding Figure 3.6 shows a series of exemplary queries computed from the Market 1501 test gallery. The queries shown in Figure 3.6a represent a selection of many identities that the network successfully identifies by nearest neighbor search. In many cases, the feature representation is robust to varying poses as well as changing background and image quality. Figure 3.6b shows some challenging queries and interesting failure cases. For example, the network focuses on the bright handbag in a low-resolution capture of a woman in the second row. The top five results returned by the network contain four women with colorful clothing. In the third row the network fails to correctly identify the gender of the queried identity. In the last example, the network successfully re-identifies a person that is first sitting on a scooter and later walks (rank 4 and 5), but also returns a wrong identity with similarly striped sweater (rank 3). A visualization of the learned embedding on the MARS test split is shown in Figure 3.7 for visual inspection.

Application to Multi-Object Tracking In a final experiment the cosine softmax network which has been trained on MARS is applied to a variety of multi-object tracking datasets to assess the performance in the targeted tracking application. On each dataset, the cosine similarity between detections at time k and time $k+n$ for $n = 1, \ldots, 50$ is computed. Then, positive and negative pairs are established by matching detections against ground truth annotations such that a positive pair corresponds to two detections of the same identity and a negative pair corresponds to two detections of different identities. Detections that cannot be matched against

the ground truth with bounding box overlap of at least 75% are counted as false alarms and are subsequently removed.

For each time gap n, histograms of the cosine similarity between positive and negative pairs are computed and histogram intersection is used to measure how well identities are separated in representation space (c.f. Figure 3.8). If identities fall into compact clusters, then the cosine similarity of positive pairs should generally be higher than the cosine similarity of negative pairs. In addition, there should be little overlap between the two histograms in order to identify correct matches by a threshold on the cosine similarity. In this regard, the histogram intersection area for a given time gap n indicates how suitable the feature representation can guide the tracker to recover from occlusions over n frames.

The datasets that have been selected for this experiment are: ETH [ELSVG08] sequences Bahnhof and Sunnyday, TUD [ARS08] sequences Campus and Stadtmitte, PETS 2009 [FS09] sequences S1L1-1, S1L1-2, S1L2-1, S1L1-2, S2L1, S2L2, S2L3, and KITTI [GLU12] training sequences 15–19. Figure 3.8 shows exemplary images taken from the ETH and PETS 2009 datasets as well as the cumulative histograms over the cosine similarity between positive and negative pairs for all $n = 1, \ldots, 50$. This corresponds to a time gap of up to 3.5 seconds on ETH and 7 seconds on PETS 2009. By comparison of the two histograms it can be observed that the performance differs considerably between both datasets. A possible explanation for the poor performance on PETS 2009 is threefold. First, the camera position is much higher than on MARS. Potentially, the network does not generalize well to the different perspective. Second, PETS 2009 contains many densely crowded scenes with frequent partial occlusions, causing multiple people to appear inside detection boxes. Such situations are likely harder to handle for the re-identification network than detections in front of static background. Third, ground truth identities that leave the scene are assigned a new identity when they re-enter at a later point in time. Therefore, some negative pairs actually represent the same identity.

Figure 3.9a plots the histogram intersection area as a function of time gap n on all four datasets. Given that the performance on ETH, TUD, and KITTI is relatively similar, it seems the network generalizes equally well to different domains. However, on all datasets the intersection area is low for small n and increases with time. Therefore, the network is well suited to establish associations on a frame-by-frame basis with short occlusions and misses, but becomes less discriminative after long-term occlusions. Detailed histograms for the four datasets can be found in Appendix A.

The experiment has been repeated using provided models of the TriNet of Hermans *et al.* [HBL17]. This is the current top performing network on MARS with a 10.82 points higher mAP than the cosine softmax network. The result of

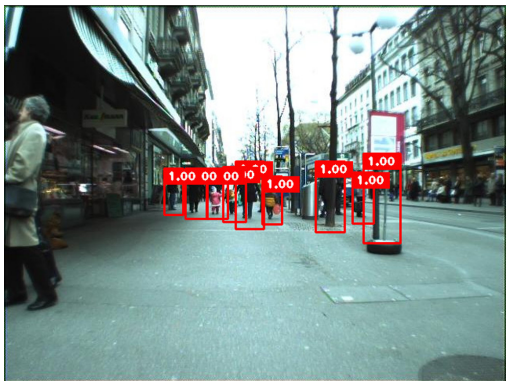
this repeated experiment is shown in Figure 3.9b. Surprisingly, the TriNet performs worse in the tracking application compared to the cosine softmax network. The results suggest that the increased network capacity of TriNet results in more dataset characteristic features that do not generalize to the tracking domain directly.

(a) Examples for successful queries without error in top 5 matches

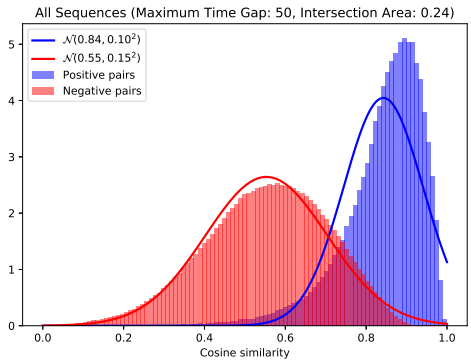
(b) Interesting failure cases

Figure 3.6: Example queries generated from the Market 1501 [ZST+15] test gallery. The first image in each row shows the query image. The second block shows the five most similar images in the gallery (errors marked by red border). The third block shows the five most dissimilar images in the gallery.

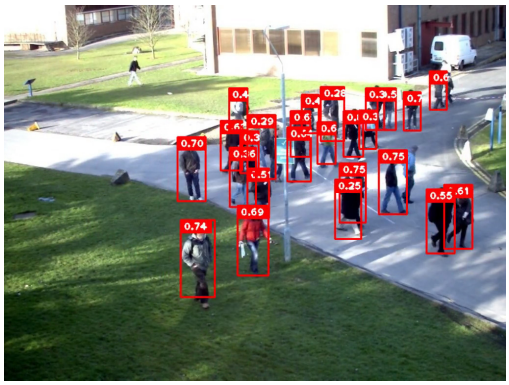
Figure 3.7: This plot shows an excerpt of the learned embedding on the MARS test split. The visualization has been generated with t-SNE [VDM14]. Images that are close in the image are also close in representation space. Identities shown here have not been used during training. The full embedding can be found in Appendix A.



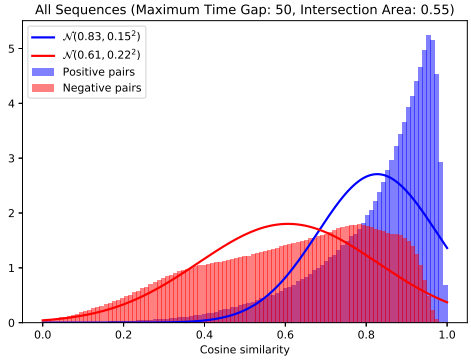
(a) ETH dataset



(b) ETH cosine similarity

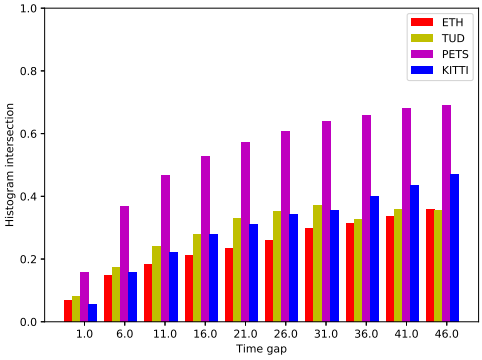


(c) PETS 2009 dataset

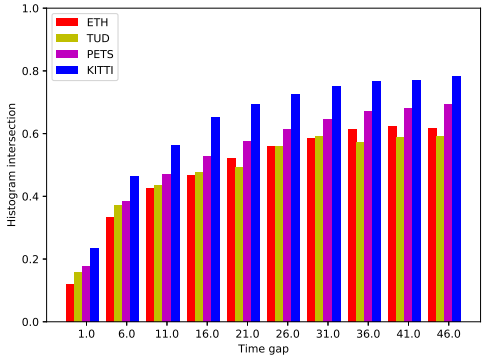


(d) PETS 2009 cosine similarity

Figure 3.8: This plot shows exemplary images from the ETH [ELSVG08] and PETS 2009 [FS09] dataset and corresponding histograms of the cosine similarity between positive and negative pairs. The PETS 2009 dataset is densely crowded with more frequent partial occlusions, leading to worse performance.



(a) Cosine softmax network



(b) TriNet [HBL17]

Figure 3.9: For each tracking dataset, histograms of the cosine similarity between positive (images of the same identity) and negative (images of different identities) pairs have been computed. This plot shows the histogram intersection area as a function of the time gap between detections within each pair. The smaller the intersection area, the better the representation space discriminates between identities.

Chapter 4

Min-Cost Flow Probability Hypothesis Density Tracker

This chapter presents a tracker based on the PHD filter that recovers trajectories in a min-cost flow problem formulation. Due to this property, the approach is termed as MCF-PHD tracker. Preliminary work on this method has been published [WP16] and [WP17]. This chapter extends these publications by a more in-depth treatment of the topic and broader evaluation on simulated data as well as public benchmarks. The text starts with an introduction in Section 4.1. A literature review on multiple object tracking is given in Section 4.2. Section 4.3 presents a re-formulation of the PHD filter in terms of single-object track hypotheses that are subsequently linked into consistent object trajectories in Section 4.4. A relationship between the MCF-PHD tracker and MHT is drawn in Section 4.5. The following sections discuss a practical application to visual tracking scenarios. Therefore, Section 4.6 shows how detector confidence scores can be integrated into the PHD recursion as an additional source of information to infer the number of objects in the scene. Section 4.7 discusses integration of appearance features, in particular the CNN feature representation that has been developed in the previous chapter. A practical Gaussian mixture implementation is given in Section 4.8. The MCF-PHD tracker is evaluated both under controlled conditions in a simulated environment and on public benchmarks. The results of this evaluation are presented in Section 4.9.

4.1 Introduction

The PHD filter provides an innovative solution to the multiple object tracking task that circumvents the combinatorial data association problem by propagating an object density measure which can be computed efficiently. In large, this efficiency stems from the fact that the propagated intensity can be computed without main-

taining object identities. Therefore, the PHD filter provides a set of likely object locations at each time step, but provides no information about the trajectories that individual objects have taken[1].

This chapter proposes a multi-object tracker that recovers trajectories from the PHD filter in a min-cost flow problem formulation. Therefore, the approach combines multi-object state estimation with discrete optimization in the search for globally consistent object trajectories. The resulting algorithm inherits the desirable properties of the min-cost flow formulation [ZLN08] which have lead to its widespread application, i.e., efficient and exact inference for a globally consistent solution, and at the same time approaches one of its limitations, the integration of higher-order motion models, without increasing the complexity of the data association problem.

The key observation that has led to the development of this algorithm is due to a *measurement-centric* view onto the tracking problem that the PHD filter inherits from RFS theory. More specifically, at any time the intensity can be written as a sum of measurement-oriented partitions that each describe the hidden state of an object that has generated the corresponding measurement. This measurement-centric view is exploited in order to augment observations with additional estimated state variables to be used in the cost terms of the min-cost flow problem. Therefore, a main contribution of this chapter is the integration of object motion into the min-cost flow problem that is traditionally solved using cost terms based on only static features, such as appearance similarity and distance between detections [ZLN08]. Object motion is particularly useful to maintain identities when there are large gaps between detections, e.g., due to occlusions, and broadens the application domain to scenarios where appearance is less discriminative, such as Light Detection and Ranging (LIDAR) and Radio Detection and Ranging (RADAR) tracking in automotive and robotics context.

Further, this chapter establishes a relationship between the objective function used in this work and the track hypothesis score used in MHT. The comparison gives rise to an interpretation of the MCF-PHD tracker as an approximation to the popular MHT that avoids its exponential growths of trajectory hypotheses at the cost of capturing only short-term dependencies outside of the recursive filtering framework. The proposed algorithm is general and can be applied to a broad range of tracking scenarios where observations can be represented as point measurements. Due to the efficiency of min-cost flow data association, the method can be applied both in online and offline tracking scenarios.

[1]Existing track management schemes that can be applied to the PHD filter will be discussed in the following section on related work.

4.2 Related Work

In many practical applications it is still common practice to adopt a single-hypothesis tracking approach where the data association problem is solved on a frame-by-frame basis [ELSVG08, LBLA16, WBP17a]. In this paradigm, a set of confirmed tracks is successively extended to incorporate new measurements. Therefore, newly arriving measurements are associated to existing tracks in a linear assignment problem that can be solved efficiently using the Hungarian method [Kuh55]. A recent study on multi-modal people tracking in crowded environments [LBLA16] suggests that this simple approach can outperform more advanced methods in practice due to easier parametrization. Yet, single-hypothesis tracking is error prone from a theoretical perspective because data association errors from the past cannot be corrected based on new evidence.

More advanced methods that have been employed traditionally are JPDA [FBSS83] and MHT [Rei79]. In JPDA, a joint probabilistic score is defined for the measurement-to-track association problem. In MHT, the space of object trajectories is explored in a breadth-first search starting from the first time step onwards. This approach explicitly considers multiple trajectory hypotheses in parallel such that errors from the past can be corrected. While both methods have received considerate attention during the 1990s, attention has decayed within the last decades due to combinatorial issues that come along with their application. However, recent revisitations [HRMZ+15, KLCR15] suggest that both methods still perform well compared to the state of the art.

Due to tremendous progress in object detection, the computer vision community has widely adopted a *tracking-by-detection* paradigm where multiple object tracking is typically solved in a global optimization problem over the measurement sequence. In consequence, many approaches following this paradigm do not perform recursive state estimation. Instead, they formulate a global optimization problem that takes into account the distance of detections and appearance similarity. Many optimization strategies of varying model capacity have been explored, including conditional random fields [YN12, YN14, MSR16], minimum cliques [ZDS12], network flows [ZLN08], message passing [SR13], and lifted multicuts [TAAS17]. The network flow formulation of Zhang *et al.* [ZLN08] has become a standard method within this paradigm due to its efficiency and optimality. The formulation extends the linear assignment problem to multiple frames by construction of a flow network in which the search for trajectories corresponds to a transportation problem. Dynamic programming solutions [PRF11] have been developed that find near-optimal solutions in time linear in the number of objects and linear in the length of the observation sequence. Further, due to the specific structure of the flow network, the optimal solution can obtained very efficiently by casting the problem as shortest path search [BFTF11]. More recently, clever caching strategies have been devel-

oped [LGU15] that permit application in online tracking scenarios by extending and re-using the solution of the previous time step. The resulting algorithm has bounded memory and computational complexity. Further, a way to learn the parameters of a flow network by backpropagation has been presented by Schulter *et al.* [SVCC17], leading to generally better performance and reduced parametrization effort.

The performance of a tracking-by-detection system largely depends on the cost terms that are used to model object transitions. Nowadays, most often CNNs are used to model appearance similarity [KLCR15, LTCFS16, TAAS17, SAS17, CAS+17]. Integration of motion models is more challenging. In particular, only static features can be used to model transitions in the network flow formulation because the cost terms consider only pairwise relationships. Integration of a constant velocity motion model has been studied by Butt *et al.* [BC13], but the resulting network structure can no longer be solved in a min-cost flow problem and approximate, iterative solutions must be applied to obtain a solution. This not only increases the computational cost but also convergence is not guaranteed. Nevertheless, integration of motion models is worthwhile as it can reduce the number of identity switches and provides useful information to maintain identities through occlusions [Col12]. An alternative strategy to incorporate motion into the network flow formulation can be established by using optical flow. For this purpose, Choi [Cho15] has developed an appearance similarity descriptor that delivers excellent performance on public benchmarks. Optical flow has been shown to perform particularly well when tracking is performed in image coordinates and video footage is subject to unknown camera motion.

Hierarchical association strategies, e.g., [YN12, YN14, WWCW17], represent a straight-forward way to incorporate higher-order dependencies without increasing the complexity of the optimization problem. Tracklet-based methods first solve for reliable, small trajectory fragments and then apply one or multiple additional optimization steps where these fragments are linked into longer trajectories. Both stages can be solved in a min-cost flow problem [WWCW17], but from the second stage onwards higher-order information can be integrated as each tracklet contains a sequence of detections from which a motion pattern can be computed. The drawback of this approach is that errors that have been made at lower association stages cannot be corrected later on anymore.

Methods based on RFS theory approach the tracking problem in a set-valued recursive filtering framework. The PHD filter does not provide object identities itself, but heuristic track management schemes exist both for the Gaussian mixture implementation [PCV09] and the sequential Monte-Carlo implementation [PVS05]. These methods assign a unique track label to each Gaussian mixture component, or particle respectively, that is created during object birth and then propagate

these labels to subsequent time steps during prediction and update. As an alternative, MHT can be applied on top of the output of a PHD filter to obtain trajectories [PVS07]. In this case, the PHD filter eliminates some clutter from the measurement set to reduce the computational complexity of MHT.

The track management scheme of Panta *et al.* [PVS07] can also be applied to the cardinalized PHD filter [Mah07a] and the Multi-Target Multi-Bernoulli (MeMBer) filter [VVC09]. The cardinalized PHD filter provides a more accurate cardinality estimate compared to its standard counterpart by propagating the cardinality distribution in addition to the PHD. The MeMBer filter has been proposed as an alternative that, instead of a moment approximation, propagates an approximate multi-object posterior density of a multi-Bernoulli RFS. It has approximately the same computational complexity, but suffers from a significant cardinality bias that has been addressed with a correction term in [VVC09]. Although object interactions can be modeled in RFS theory itself, the PHD filter, the cardinalized PHD filter, and the MeMBer filter all assume independent object motion and measurement generation.

More recent advances in the field have led to the development of a tractable approximation to the optimal multi-object Bayes filter, called the Generalized Labeled Multi-Bernoulli (GLMB) filter [VVP14]. This filter relaxes independence assumptions and is therefore capable to incorporate interactions in object motion and measurement models [BVV15, BRG$^+$16]. In addition, in contrast to the aforementioned filters, the GLMB filter outputs trajectories instead of object states. Therefore, it is not necessary to run a heuristic track management scheme on top of the filter. This is achieved by operating on a hybrid state space which contains a discrete class label for each identity. Similar to MHT, the filter maintains multiple trajectory hypotheses and computational tractability is maintained by pruning schemes that limit the number of hypotheses to follow. The method has only recently been applied to general [VVP14] and specialized applications such as tracking in high clutter scenarios [PVV$^+$15] and extended target tracking [BRG$^+$16].

An alternative formulation to obtain trajectories from filters based on RFS theory has been developed by Garcíi *et al.* [GFSM16]. Their approach models the multi-object state as trajectory set. Thus, filtering is performed directly in trajectory space. The theory is applied to the PHD filter in [GFS16] where they discuss practical issues related to the PHD approximation and propose a Gaussian mixture implementation that can be applied in practice.

Despite offering a diverse toolbox of algorithms that model multi-object tracking problems at different levels of accuracy and computational complexity, the RFS methodology has not yet become a standard approach in computer vision. Currently, only two out of forty published submissions to 2DMOT2015 [LTMR$^+$15], a standard multiple object tracking benchmark, apply a RFS methodology at some

stage of the tracking system [SMPC16, SJ16]. Instead, the domain is dominated by approaches that operate under the tracking-by-detection paradigm. However, a recent publication of Rezatofighi *et al.* [RMR+17] suggests that many vision tasks can potentially benefit from a RFS formulation. More specifically, they formulate the CNN object detection and multi-class classification task in a RFS framework and achieve state of the art or superior performance to existing methods. The importance of this work is twofold. First, the publication may be seen as foundational work that potentially leads to wider acceptance of RFS theory in the future. Second, their approach provides a way to learn a multi-object likelihood function directly from data. Thus, the approach may become a useful tool for motion and appearance modeling for trackers that operate under the RFS paradigm.

4.3 Track-Oriented PHD Filter

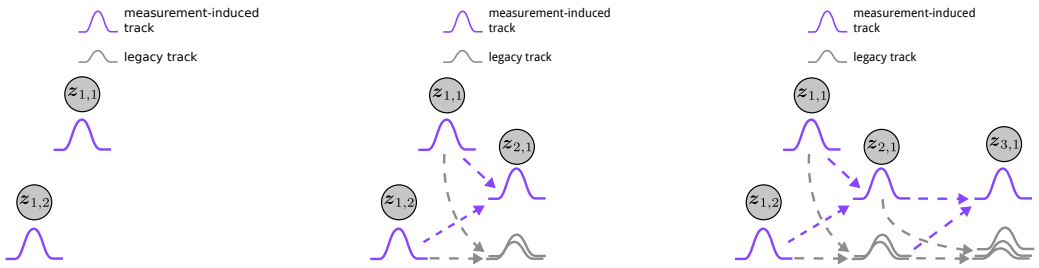
The following section presents a re-formulation of the PHD filter in terms of single-object track hypotheses. This re-formulation leads to a measurement-oriented partitioning of the PHD that is similar to the hypothesized tracks generated by the MeMBer filter [VVC09]. For each measurement in the observation sequence, there exists one track hypothesis that is characterized by a spatial density and a cardinality estimate. These track hypotheses are initialized during the PHD filter update step based on the predicted PHD and clutter characteristics. In subsequent time steps, track hypotheses are propagated in accordance with the PHD filter equations. This re-formulation is exact in the sense that it is merely a rewriting in terms of individual intensity partitions rather than a modified recursion. In particular, the intensity of the multi-object state can be constructed from its track partitions at every time step. Let $Z_{1:k}$ denote the set of all measurements obtained up to time k. Then, the posterior intensity at time k can be written as a linear combination of individual track hypotheses

$$v_k(\boldsymbol{x}) = u_k(\boldsymbol{x}) + \sum_{\boldsymbol{z}_{t,i} \in Z_{1:k}} r_{t,i}\, v_k^{(t,i)}(\boldsymbol{x}) \tag{4.1}$$

where $u_k(\boldsymbol{x})$ is the intensity of objects that remain undetected until time k, where $v_k^{(t,i)}(\boldsymbol{x})$ is the intensity of a single-object track hypothesis that originated at measurement $\boldsymbol{z}_{t,i}$, and where $r_{t,i} \in [0, 1]$ is a scaling parameter that characterizes the expected number of objects that have generated the corresponding measurement[2].

The following two sections present the rewritten PHD recursion that initiates and maintains these individual track hypotheses. An illustration of this algorithm is

[2] According to the standard multi-object measurement model, a measurement can be generated by at most one target. Consequently, the expected number of objects that have generated each measurement is at most one, i.e., $r_{t,i} \in [0, 1]$.



(a) Time 1 **(b)** Time 2 **(c)** Time 3

Figure 4.1: Illustration of the track-oriented PHD filter recursion. (a) During the filter update step a new measurement-induced track partition is created for each measurement. (b–c) In subsequent time steps these become legacy tracks that account for the missed-detection case of the PHD filter update. Measurement-induced tracks at time k are computed from all track hypotheses of previous time steps 1 to $k-1$.

given in Figure 4.1. During the update step, a *measurement-induced* track hypothesis is created for each measurement. These are subsequently propagated in time according to the PHD predictor and down-scaled to account for the missed-detection case of the PHD filter update step. As such, each track hypothesis represents the event that the object that has generated a particular measurement has since then not been detected.

4.3.1 Prediction

If at time $k-1$ the posterior intensity of the multi-object state partitions into measurement-oriented track hypotheses

$$v_{k-1}(\boldsymbol{x}) = u_{k-1}(\boldsymbol{x}) + \sum_{\boldsymbol{z}_{t,i} \in Z_{1:k-1}} r_{t,i}\, v_{k-1}^{(t,i)}(\boldsymbol{x}), \tag{4.2}$$

then the predicted multi-object intensity at time k partitions into measurement-oriented tracks as well, i.e.,

$$v_{k|k-1}(\boldsymbol{x}) = u_{k|k-1}(\boldsymbol{x}) + \sum_{\boldsymbol{z}_{t,i} \in Z_{1:k-1}} r_{t,i}\, v_{k|k-1}^{(t,i)}(\boldsymbol{x}) \tag{4.3}$$

with

$$u_{k|k-1}(\boldsymbol{x}) = b_k(\boldsymbol{x}) + \int p_{\mathrm{S}}(\boldsymbol{x}') p_{k|k-1}(\boldsymbol{x} \mid \boldsymbol{x}') u_{k-1}(\boldsymbol{x}')\, \mathrm{d}\boldsymbol{x}', \tag{4.4}$$

$$v_{k|k-1}^{(t,i)}(\boldsymbol{x}) = \int p_{\mathrm{S}}(\boldsymbol{x}') p_{k|k-1}(\boldsymbol{x} \mid \boldsymbol{x}') v_{k-1}^{(t,i)}(\boldsymbol{x}')\, \mathrm{d}\boldsymbol{x}'. \tag{4.5}$$

The result is established by application of the PHD predictor (2.21) to the individual partitions with the intensity of appearing objects $b_k(\boldsymbol{x})$ added to the partition of undetected targets. During the prediction step, the intensity of each track partition is scaled down by a state-dependent probability of survival $p_{\mathrm{S}}(\boldsymbol{x}')$ to account for objects leaving the scene. Surviving objects are propagated in time by application of the single-object motion model $p_k(\boldsymbol{x} \mid \boldsymbol{x}')$. The constant scaling factor $r_{t,i}$ is independent of time and remains unchanged.

4.3.2 Update

If at time k a given predicted intensity partitions into measurement-oriented track hypotheses

$$v_{k|k-1}(\boldsymbol{x}) = u_{k|k-1}(\boldsymbol{x}) + \sum_{\boldsymbol{z}_{t,i} \in Z_{1:k-1}} r_{t,i}\, v_{k|k-1}^{(t,i)}(\boldsymbol{x}), \tag{4.6}$$

then the posterior intensity partitions into a set of legacy track hypotheses that account for the missed-detection case of the PHD filter update (2.22) and a set of newly created measurement-induced track hypotheses that account for the measurement-corrected terms:

$$v_k(\boldsymbol{x}) = \underbrace{u_k(\boldsymbol{x}) + \sum_{\boldsymbol{z}_{t,i} \in Z_{1:k-1}} r_{t,i}\, v_k^{(t,i)}(\boldsymbol{x})}_{= v_{\mathrm{L},k}(\boldsymbol{x})} + \underbrace{\sum_{\boldsymbol{z}_{k,j} \in Z_k} r_{k,j}\, v_k^{(k,j)}(\boldsymbol{x})}_{= \sum_{\boldsymbol{z}_{k,j} \in Z_k} v_{\mathrm{U},k}(\boldsymbol{z}_{k,j}, \boldsymbol{x})} . \tag{4.7}$$

The update rule for legacy tracks is obtained from Equation 2.23 by scaling the intensity mass down by missed-detection probability $1 - p_{\mathrm{D}}(\boldsymbol{x})$:

$$u_k(\boldsymbol{x}) = [1 - p_{\mathrm{D}}(\boldsymbol{x})]\, u_{k|k-1}(\boldsymbol{x}), \tag{4.8}$$

$$v_k^{(t,i)}(\boldsymbol{x}) = [1 - p_{\mathrm{D}}(\boldsymbol{x})]\, v_{k|k-1}^{(t,i)}(\boldsymbol{x}). \tag{4.9}$$

The measurement-induced track hypotheses are computed from Equation 2.24, which is re-written into a scaling parameter and an intensity for the track hypothesis:

$$r_{k,j} = \frac{\tau_k(\boldsymbol{z}_{k,j})}{c_k(\boldsymbol{z}_{k,j}) + \tau_k(\boldsymbol{z}_{k,j})}, \tag{4.10}$$

$$v_k^{(k,j)}(\boldsymbol{x}) = \frac{p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}(\boldsymbol{x})}{\int p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}}, \tag{4.11}$$

with *PHD likelihood* $\tau_k(\boldsymbol{z}_{k,j}) = \int p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$[3]. The following Remark 4.1 shows that measurement-induced track hypotheses are indeed scaled

[3] An explanation of this term is given in Remark 4.3.

partitions of the multi-object intensity[4]. An intuitive interpretation for this update can be established as follows. First, observe that, for a constant probability of detection $p_{\mathrm{D}}(\boldsymbol{x}) = p_{\mathrm{D}}$, Equation 4.11 resembles a Bayes' rule-like update of the predicted multi-object intensity $v_{k|k-1}(\boldsymbol{x})$ with the newly arrived measurement $\boldsymbol{z}_{k,j}$ using the single-object measurement model $p(\boldsymbol{z}_{k,j} \mid \boldsymbol{x})$. Assuming the underlying Poisson assumption of the PHD filter holds, the predicted multi-object state at time k contains $\hat{N}_{k|k-1} = \int v_{k|k-1}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$ i.i.d. objects that are distributed according to $p_{k|k-1}(\boldsymbol{x}) = v_{k|k-1}(\boldsymbol{x})/\hat{N}_{k|k-1}$. Consequently, if measurement $\boldsymbol{z}_{k,j}$ has been generated by an object (not clutter), then the posterior density of this object can indeed be computed by application of a conventional Bayes update with the newly arrived measurement. Therefore, the initiated track partition $v_k^{(k,j)}(\boldsymbol{x})$ at time k corresponds to the spatial density of an object that has generated $\boldsymbol{z}_{k,j}$ under the assumption that its prior distribution is proportional to the predicted multi-object intensity. Next, observe that the track partition is created such that its intensity mass integrates to 1, i.e., the expected number of object in this partition is $\int v_k^{(k,j)}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = 1$. Therefore, scaling parameter $r_{t,i}$ is introduced to compensate for the difference in the cardinality estimate with respect to the full multi-object intensity. In consequence, Equation 2.24 can be written as $v_{\mathrm{U},k}(\boldsymbol{z}_{k,j}, \boldsymbol{x}) = r_{k,j}\, v_k^{(k,j)}(\boldsymbol{x})$.

Remark 4.1. To see why the track initialization scheme in Equation 4.10 and 4.11 holds, observe that

$$
\begin{aligned}
r_{k,j}\, v_k^{(k,j)}(\boldsymbol{x}) &= \frac{\int p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}}{c_k(\boldsymbol{z}_{k,j}) + \int p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}} \\
&\quad \cdot \frac{p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}(\boldsymbol{x})}{\int p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}} && (4.12) \\
&= \frac{p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}(\boldsymbol{x})}{c_k(\boldsymbol{z}_{k,j}) + \int p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}} && (4.13) \\
&= v_{\mathrm{U},k}(\boldsymbol{z}_{k,j}, \boldsymbol{x}). && (4.14)
\end{aligned}
$$

Therefore, each track hypothesis $v_k^{(k,j)}(\boldsymbol{x})$ corresponds to a scaled measurement-corrected partition of the multi-object intensity $v_k(\boldsymbol{x})$.

4.3.3 State Extraction

The PHD of the multi-object state characterizes the number of objects in the scene as well as their spatial density. The most commonly applied approach to extract

[4]The term multi-object intensity is used to refer to the intensity of the multi-object state $v_k(\boldsymbol{x})$ for a clear distinction from the intensity of individual track hypotheses $v_k^{(t,i)}(\boldsymbol{x})$.

states from a given intensity is to select peaks that have a high concentration of intensity mass. This task is straight-forward for the Gaussian mixture implementation of the PHD filter where—assuming components are well separated—local minima can be found at means of the Gaussian components [VM06], but more involved for the sequential Monte Carlo implementation that requires an additional particle clustering step [VSD05]. The measurement-oriented track partitioning offers a general, intuitive approach that provides similar results under reasonable assumptions on the filter parameters.

Assuming a partitioned intensity of form (4.1), the expected number of objects in the scene can be computed by

$$\begin{aligned}
\hat{N}_k &= \int v_k(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} && (4.15)\\
&= \int \left[u_k(\boldsymbol{x}) + \sum_{\boldsymbol{z}_{t,i} \in Z_{1:k}} r_{t,i} v_k^{(t,i)}(\boldsymbol{x}) \right] \mathrm{d}\boldsymbol{x} && (4.16)\\
&= \int u_k(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} + \sum_{\boldsymbol{z}_{t,i} \in Z_{1:k}} r_{t,i} \int v_k^{(t,i)}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} && (4.17)\\
&= \hat{N}_k^{(\mathrm{u})} + \sum_{\boldsymbol{z}_{t,i} \in Z_{1:k}} \hat{N}_k^{(t,i)} && (4.18)
\end{aligned}$$

with $\hat{N}_k^{(\mathrm{u})} = \int u_k(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}$ and $\hat{N}_k^{(t,i)} = r_{t,i} \int v_k^{(t,i)}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}$. If the expected number of objects that have never been detected $\hat{N}^{(\mathrm{u})}$ is small, then most of the intensity mass in Equation 4.18 is due to measurement-induced partitions. In many practical applications this is indeed a reasonable assumption because the probability of detection is typically large enough to assume objects are detected on their first appearance in the scene. In particular, the Gaussian mixture implementation of the PHD filter that was presented in Section 2.3.3 was based on the assumption that appearing objects are always detected. Then, the set of undetected objects is empty $u_k(\boldsymbol{x}) = 0$. Under these circumstances, a straight-forward way to extract likely object locations from the partitioned intensity is to confirm any track hypothesis with expected number of objects larger than a given threshold $\hat{N}_{\min}$ and to take its location from the mode of its corresponding partition:

$$\hat{X}_k = \{ \operatorname*{argmax}_{\boldsymbol{x}} v_k^{(t,i)}(\boldsymbol{x}) \mid \forall \boldsymbol{z}_{t,i} \in Z_{1:k} : \hat{N}_k^{(t,i)} > \hat{N}_{\min} \}. \tag{4.19}$$

Since the expected number of objects $\hat{N}_k^{(t,i)}$ in each measurement-induced partition is bound to $[0, 1]$, a reasonable choice for the threshold is $\hat{N}_{\min} = 0.5$.

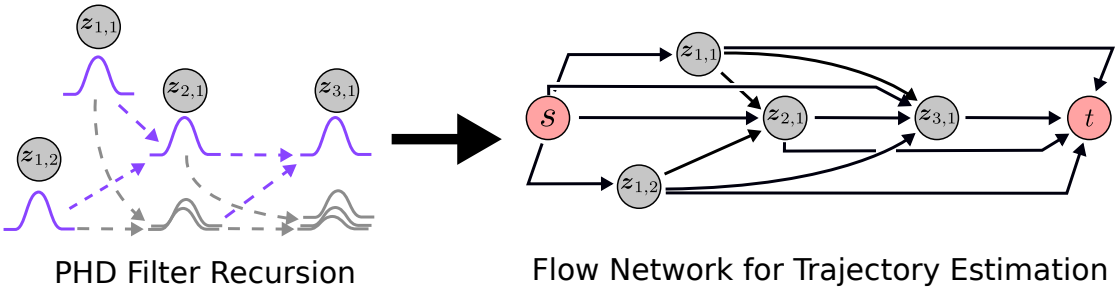


Figure 4.2: Illustration of the min-cost flow tracking framework. The left image shows a sequence of measurements and associated track hypotheses computed in the track-oriented PHD filter recursion. The right image shows the corresponding flow network that is built in parallel to running the PHD filter. From this network, object trajectories are found by solving a min-cost flow problem that optimizes for the minimum cost linking of measurement-oriented track hypotheses into globally consistent trajectories. The cost terms in the network are based on the track hypotheses obtained by the filter recursion.

4.4 Min-Cost Flow Trajectory Estimation

At each time step, the PHD filter provides an estimate of the number of objects in the scene as well as their location in state space. The filter does not, however, provide information about the trajectory that each individual object has taken. The measurement-oriented partitioning scheme also does not directly provide such information because a track hypothesis provides no information about the sequence of measurements generated by an individual object. Therefore, the goal of this section is to recover trajectories from the PHD recursion by linking the measurement-oriented tracks into consistent object trajectories. This is done in a min-cost flow problem formulation where the cost terms are defined based on the track hypotheses obtained from the filter recursion. Figure 4.2 illustrates this approach. Concurrently to running the PHD filter, a flow network is built incrementally where nodes represent measurements in the observation sequence. A virtual source s and sink t are added to denote the start and end of object trajectories such that every trajectory corresponds to an s-t path in the flow network. These are recovered by running a min-cost flow solver.

Remark 4.2. In the remainder of this section, track hypotheses $v_k^{(t,i)}(\boldsymbol{x})$ are treated as the intensity of a Poisson RFS $X_k^{(t,i)}$ that contains an expected number of $\int v_k^{(t,i)}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$ i.i.d. objects that are distributed proportional to $v_k^{(t,i)}(\boldsymbol{x})$. At this point it is worth noting that, according to the assumptions underlying the PHD filter, the posterior multi-object state X_k is itself a Poisson RFS with

intensity $v_k(\boldsymbol{x})$. Therefore, all objects contained in X_k are i.i.d. proportional to $v_k(\boldsymbol{x})$. This means, measurement partitions do not correspond to individual objects formally because this would impose a violation of the i.i.d. assumption.

In this light, the approach presented in this section is to be taken under the following premise: The object trajectories that are recovered from the following problem formulation are only loosely coupled to the PHD filter. Whenever a Poisson RFS is constructed from a measurement partition of the track-oriented PHD filter, this is based on the reasoning of the track initialization scheme in Equation 4.11. The experimental evaluation in Section 4.9.2 suggests that the recovered trajectories are nonetheless well aligned with the locations confirmed by the filter recursion.

4.4.1 Objective Function

Let $Z = Z_1 \cup \ldots \cup Z_M$ denote the set of measurements from time 1 to M and let $T_m = \{(k_{m_l}, j_{m_l})\}_{l=1}^{L_m}$ denote a single-object trajectory, such that $T_m(l)$ is an index to the j_{m_l}-th measurement at time k_{m_l}, i.e., $\boldsymbol{z}_{T_m(l)} \coloneqq \boldsymbol{z}_{k_{m_l},j_{m_l}}$. Further, define a multi-object trajectory as the set of single-object trajectories $T = \{T_m\}$. Then, trajectory estimation is defined as an optimization problem with linear cost function

$$T^* = \operatorname*{argmin}_T C(T) = \operatorname*{argmin}_T \sum_{T_m \in T} C_m(T_m) \tag{4.20}$$

where the space of valid solutions is constrained to non-overlapping trajectories, that is,

$$T_m \cap T_n = \emptyset, \quad \forall m \neq n. \tag{4.21}$$

The cost of each single-object trajectory is defined as a sum of an entry cost term and cost terms that link neighboring measurements on the trajectory:

$$C_m(T_m) = c_{\text{entry}}(\boldsymbol{z}_{T_m(1)}) + \sum_{l=2}^{L_m} c_{\text{link}}(\boldsymbol{z}_{T_m(l-1)}, \boldsymbol{z}_{T_m(l)}). \tag{4.22}$$

The concrete form of these cost terms is inspired by the track scoring function of the popular MHT [BP99, KLCR15], but it takes on a different form here because of the random finite set methodology that the method is based on. A comparison between both objectives is drawn in Section 4.5.

For the definition of the entry cost term, recall that in the predicted intensity (4.2), partition $u_{k|k-1}(\boldsymbol{x})$ at time k accounts for objects that are present in the scene but have not been detected in between times 1 to $k-1$. In particular, this partition contains the set of appearing objects at the current time $b_k(\boldsymbol{x})$. The cost for starting a new trajectory at measurement $\boldsymbol{z}_{k,j}$ is defined as

$$c_{\text{entry}}(\boldsymbol{z}_{k,j}) = -\log \frac{\tau_k^{(\text{u})}(\boldsymbol{z}_{k,j})}{c_k(\boldsymbol{z}_{k,j})}, \tag{4.23}$$

where

$$\tau_k^{(\mathrm{u})}(\boldsymbol{z}_{k,j}) = \int p_{\mathrm{D}}(\boldsymbol{x})\, p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) u_{k|k-1}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \tag{4.24}$$

is the PHD likelihood for a Poisson multi-object state with intensity $u_{k|k-1}(\boldsymbol{x})$ and where $c_k(\boldsymbol{z}_{k,j})$ is the intensity of the clutter RFS. Thus, the entry cost is defined as the negative log-intensity ratio between the birth and clutter process. It takes on negative values if the PHD likelihood of $u_{k|k-1}(\boldsymbol{x})$ becomes larger than the clutter intensity $c_k(\boldsymbol{z}_{k,j})$, zero if both intensities are equal, and positive otherwise.

Remark 4.3. Let $v(\boldsymbol{x})$ denote the intensity of a Poisson multi-object state X. Then

$$\tau(\boldsymbol{z}) = \int p_{\mathrm{D}}(\boldsymbol{x}) p(\boldsymbol{z} \mid \boldsymbol{x}) v(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \tag{4.25}$$

is the intensity of the projected state into measurement space for a given probability of detection $p_{\mathrm{D}}(\boldsymbol{x})$ and measurement model $p(\boldsymbol{z} \mid \boldsymbol{x})$. This result can be established by an integral transform of the Poisson multi-object density of X analogous to the PHD predictor, following its derivation in [Mah03, Section IVJ]. Here, the term PHD likelihood is used to refer to an evaluation of the projected state in measurement space. This is not to be confused with the *PHD pseudolikelihood* used by Mahler [Mah07b, Equation 16.109]. This term contains an additional component for the event that an existing object is not detected.

The cost term for linking neighboring measurements on a trajectory is defined in similar vein to the entry cost. Given two measurements, $\boldsymbol{z}_{t,i}$ from time t and $\boldsymbol{z}_{k,j}$ from time $k > t$, the cost of transitioning from measurement $\boldsymbol{z}_{t,i}$ to measurement $\boldsymbol{z}_{k,j}$ is defined as the negative log-intensity ratio

$$c_{\mathrm{link}}(\boldsymbol{z}_{t,i}, \boldsymbol{z}_{k,j}) = -\log \frac{\tau_k^{(t,i)}(\boldsymbol{z}_{k,j})}{c_k(\boldsymbol{z}_{k,j})}, \tag{4.26}$$

where

$$\tau_k^{(t,i)}(\boldsymbol{z}_{k,j}) = \int p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}^{(t,i)}\, \mathrm{d}\boldsymbol{x} \tag{4.27}$$

is the PHD likelihood of a Poisson RFS with intensity $v_{k|k-1}^{(t,i)}(\boldsymbol{x})$ and $c_k(\boldsymbol{z}_{k,j})$ is the intensity of the clutter RFS. Again, this term takes on negative values if the PHD likelihood of $v_{k|k-1}^{(t,i)}(\boldsymbol{x})$ becomes larger than the clutter intensity $c_k(\boldsymbol{z}_{k,j})$, zero if both intensities are equal, and positive otherwise.

Remark 4.4. In Equation 2.11 the PHD $v_X(\boldsymbol{x})$ of a random finite set X has been defined in terms of a set integral

$$v_X(\boldsymbol{x}) = \int \pi_X\left(\{\boldsymbol{x}\} \cup W\right) \delta W \tag{4.28}$$

that accumulates the probability density of all sets W that contain element $\boldsymbol{x}$. This gave rise to an interpretation of the PHD as a fuzzy-membership function that evaluates the likelihood that an element is contained in the set, informally written as $P(\boldsymbol{x} \in X)$. Then the intensity ratio

$$\frac{v_X(\boldsymbol{x})}{v_C(\boldsymbol{x})} = \frac{\int \pi_X\left(\{\boldsymbol{x}\} \cup W\right) \delta W}{\int \pi_C\left(\{\boldsymbol{x}\} \cup W\right) \delta W} \tag{4.29}$$

between two random finite sets X and C in the same way relates to the likelihood ratio $P(\boldsymbol{x} \in X)/P(\boldsymbol{x} \in C)$ that becomes larger than 1 if $\boldsymbol{x}$ is more likely an element contained in X than it is an element contained in C.

The motivation for this objective function is as follows. In the track-oriented PHD recursion, a track hypothesis $v_k^{(t,i)}(\boldsymbol{x})$ is created for each measurement. At time $k = t$, this hypothesis is initialized to the spatial density of an object that generated $\boldsymbol{z}_{t,i}$ and has prior density proportional to the predicted multi-object intensity (c.f. Equation 4.11). The intensity mass $\int v_t^{(t,i)}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$ is equal to 1. In subsequent times $k > t$, the track hypothesis is propagated according to the single-object motion model and the intensity mass is gradually scaled down to account for objects leaving the scene (4.5) and missed detections (4.9). As such, by application of the PHD filter recursion, $v_k^{(t,i)}(\boldsymbol{x})$ follows the spatial density and cardinality of an object that has generated measurement $\boldsymbol{z}_{t,i}$ at time t and has since then not been detected. The total trajectory cost (4.22) is a sum over negative log-intensity ratios that put the PHD likelihood of an object hypothesis in relation to the null hypotheses which assumes that the measurement sequence originates from clutter. In particular, a negative cost indicates that the generated trajectory is better explained by the linked tracks than by clutter.

4.4.2 Min-Cost Flow Solution

The min-cost flow problem has been introduced in Section 2.4 as a transportation problem where a commodity is routed through a network $G(V, E)$ in order to satisfy the demand and supply at certain nodes. Mapping the trajectory estimation objective into a min-cost flow network is illustrated in Figure 4.3. For each measurement $\boldsymbol{z}_{k,j} \in Z_{1:M}$ create two nodes $u_{k,j}, v_{k,j} \in V$ and add a special source $s \in V$ and

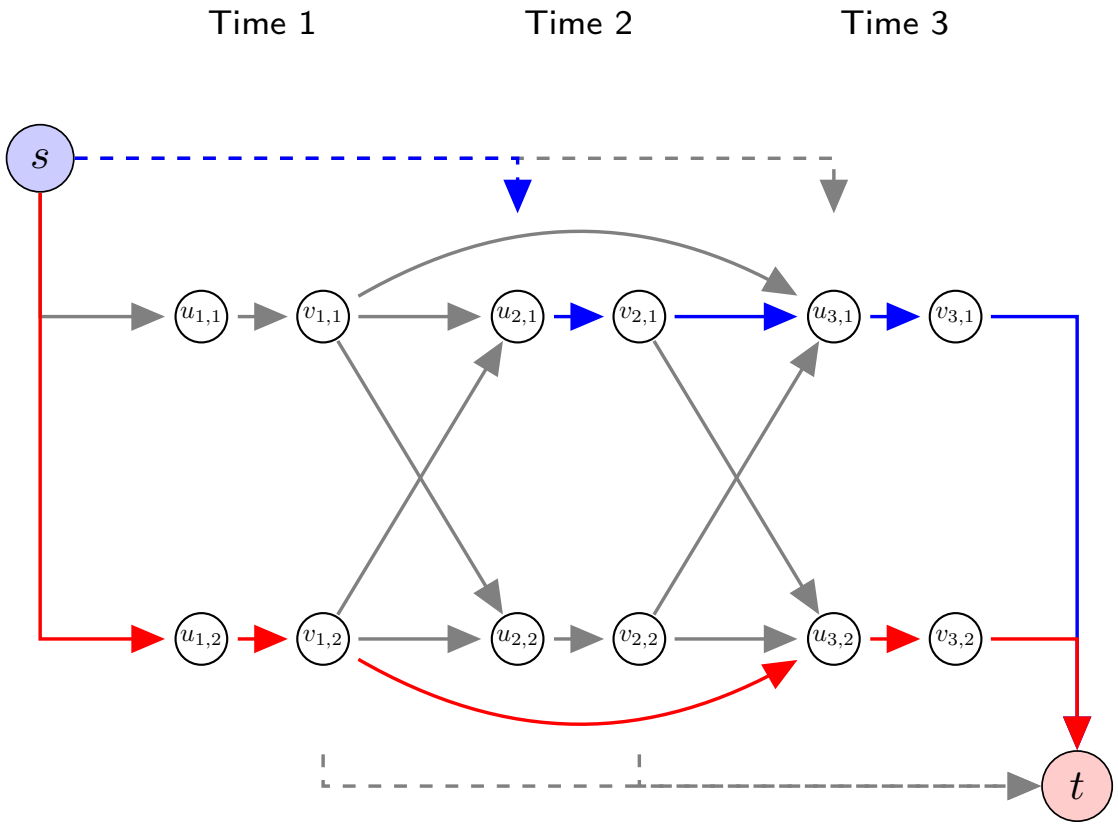


Figure 4.3: Example of a flow network over three time steps that each contain two measurements. The dashed lines indicate edges to all incoming and from all outgoing nodes at the corresponding time step. A red and a blue path show two non-overlapping trajectories. The red trajectory contains the measurement sequence $\{\boldsymbol{z}_{1,2}, \boldsymbol{z}_{3,2}\}$. The blue trajectory contains the measurement sequence $\{\boldsymbol{z}_{2,1}, \boldsymbol{z}_{3,1}\}$. For improved readability, edges between $v_{1,2}, u_{3,1}$ and $v_{1,1}, u_{3,2}$ are not shown.

terminal $t \in V$ node that mark the start and end of trajectories. Nodes $u_{k,j} \in V$ are connected to all candidate predecessors via an incoming edge $(v_{t,i}, u_{k,j}) \in E, t < k$ with cost and capacity

$$c(v_{t,i}, u_{k,j}) = c_{\text{link}}(\boldsymbol{z}_{t,i}, \boldsymbol{z}_{k,j}), \tag{4.30}$$
$$m(v_{t,i}, u_{k,j}) = 1. \tag{4.31}$$

Every node $v_{k,j}$ is also connected to all candidate successors $u_{m,n}$ with $m > k$ by this procedure. An edge $(u_{k,j}, v_{k,j}) \in E$ with zero cost and unit capacity

$$c(u_{k,j}, v_{k,j}) = 0, \tag{4.32}$$
$$m(u_{k,j}, v_{k,j}) = 1 \tag{4.33}$$

connects nodes $u_{k,j}$ and $v_{k,j}$. This edge encodes the non-overlapping trajectory constraint (4.21) into the network structure by limiting the amount of flow that can pass through each measurement: Because of flow conservation, the total amount of flow coming into $u_{k,j}$ as well as the total amount of flow going out of $v_{k,j}$ cannot take on values larger than 1 if $m(u_{k,j}, v_{k,j}) = 1$.

In general, trajectories can start and end at any time step. Therefore, all nodes $u_{k,j} \in V$ are connected to the source s by an edge $(s, u_{k,j}) \in E$ with cost and capacity

$$c(s, u_{k,j}) = c_{\text{entry}}(\boldsymbol{z}_{k,j}), \tag{4.34}$$
$$m(s, u_{k,j}) = 1 \tag{4.35}$$

and all nodes $v_{k,j} \in V$ are connected to the terminal t by an edge $(v_{k,j}, t) \in E$ with zero cost and unit capacity

$$c(v_{k,j}, t) = 0, \tag{4.36}$$
$$m(v_{k,j}, t) = 1. \tag{4.37}$$

Every s-t path in this network corresponds to a trajectory with sum of edge costs equal to the total trajectory cost (4.22). By solving the min-cost flow problem, each edge is assigned a flow in $\{0, 1\}$ that indicates if the edge is part of the minimum cost solution. The min-cost flow problem must be solved for a varying number of trajectories if the number of objects in the scene is unknown. The d trajectories with minimum cost are found by setting the balance at the source to the number of trajectories $b(s) = d$ and the balance at the terminal to $b(t) = -d$. Any non-virtual node is assigned balance $b(u_{k,j}) = b(v_{k,j}) = 0, \forall \boldsymbol{z}_{k,j} \in Z_{1:M}$ such that all flow that is pumped into the network at the source s must be transported to the terminal t. The optimal solution is found by iterating over the number of trajectories $d = 0, 1, \ldots$ until the retrieved cost is larger than the current solution. Since the trivial solution, i.e., no trajectories exist, has zero cost, all trajectories generated in this manner are better explained by an object hypothesis than clutter.

Compared to the general transportation problem, the flow network of the multiple object tracking problem has a very specific structure. First, all edges have unit capacity. Second, all edges point forward in time and there are no cycles. This specific structure can be exploited to obtain a much more efficient solution than solving the linear program of the min-cost flow problem directly. More specifically, whereas finding the optimal solution using a general min-cost flow algorithm has a computational complexity of $\mathcal{O}(n^2 m \log n)$ where n is the number of nodes and m is the number of edges in the graph [ZLN08], the current most efficient algorithm casts the problem in a successive shortest path search [BFTF11] that provides the optimal solution in $\mathcal{O}(k(m + n \log n))$ where k is the number of objects which is

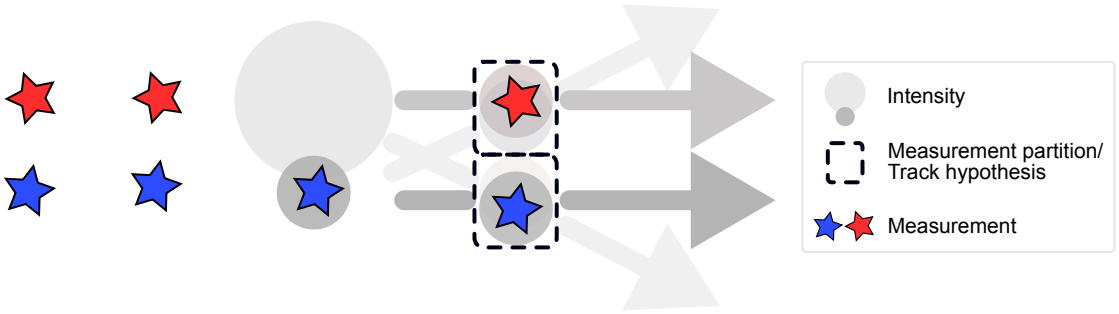
bounded by the number of nodes in the graph n. This algorithm can be outlined as follows. In a first step, the flow network is converted into an equivalent representation without negative edge costs. This can be done in linear time using dynamic programming because all edges point forward in time. Then, the optimal number of objects together with their trajectories are found by successive applications of Dijkstra's shortest path search and subsequent manipulation of the residual graph until the overall cost of the solution starts to increase[5]. This procedure is not only very fast, but also effectively eliminates the search for the optimal number of objects. Further, the successive shortest path algorithm can be extended to online tracking scenarios by caching and re-using the solution of the previous time step on arrival of new measurements [LGU15].

For a practical implementation, note that the flow network described in this section has a dense structure because every measurement $\boldsymbol{z}_{t,i}$ is a candidate predecessor of every measurement $\boldsymbol{z}_{k,j}$, $k > t$. Theoretically, this leads to a high worst-case computational complexity. In practice, however, the flow network can be sparsified by only adding edges between measurements in a given time window. Since the transition cost $c_{\text{link}}(\boldsymbol{z}_{t,i}, \boldsymbol{z}_{k,j})$ increases with time difference $k - t$, measurements with large time gap are unlikely direct neighbors on a trajectory.

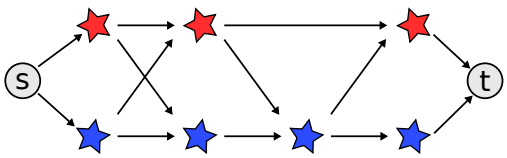
4.5 Comparison to Multiple Hypothesis Tracking

This section draws a relationship between the MCF-PHD tracker and MHT. In MHT the space of object trajectories is explored in a breadth-first search starting from the first time step onwards. This is done by maintaining and successively creating a number of likely trajectory hypotheses from newly arriving measurements. For this purpose, MHT maintains tree structures that contain all possible trajectories that originate from a single measurement. For an illustrative example, consider the scene depicted in Figure 4.4a where two objects move in parallel from left to right. Starting at the initial (leftmost) time step, either of the two observed locations may belong to either an object or clutter. This gives rise to two trajectory hypotheses with one measurement each. In the second time step, two observations can be associated to the two existing trajectories. The tree structure is created by adding a branch corresponding to each possible assignment. An example of such a trajectory tree is shown in Figure 4.4c for the leftmost red measurement. Each

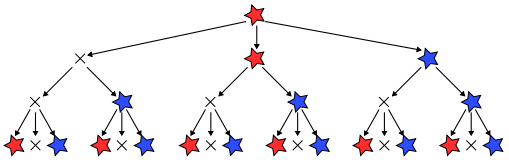
[5]The residual graph of a flow network is a data structure used for solving the k-shortest path search. Initially, the residual graph is equivalent to the flow network. After each iteration, edges along the shortest path are reversed in direction and the edge cost is multiplied by -1. Refer to the original publication [BFTF11] for more information.



(a) Illustration of two objects that move left to right in parallel. Arrows indicate multiple weighted motion hypotheses due to the two objects moving in close proximity. A dark arrow corresponds to a high weight.



(b) The corresponding flow network of the MCF-PHD tracker (simplified)



(c) MHT trajectory tree for the leftmost red measurement (× denotes a missed detection)

Figure 4.4: Comparison of MHT and MCF-PHD tracker on an illustrated problem that contains two objects moving in parallel. Illustration (a) shows how measurement partitions and corresponding track hypotheses are created from the multi-object intensity of the previous time step. In particular, each measurement partition contains multiple weighted motion hypotheses. Illustration (b) shows a simplified flow network for this problem where each trajectory is an s-t path. One trajectory tree that is constructed in MHT from the leftmost red measurement is shown in (c).

branch from the root of the tree to one of its leafs corresponds to one trajectory that originates at the red measurement in time step one and terminates at time step four. One such tree is created for each measurement in the observation sequence.

Following all trajectory hypotheses becomes infeasible quickly. For the relatively small scene depicted in Figure 4.4a, eight trajectory hypotheses exist at the second time step already: four hypotheses accounting for possible assignments of new measurements to the two existing trajectories, two hypotheses that account for occlusions, and two hypotheses that account of new objects. Therefore, unlikely trajectories must be pruned to keep the method computationally tractable. This is achieved by scoring each trajectory by a likelihood ratio of the observation sequence against a null hypothesis that assumes observations originate from clutter. Let p_{D} denote a constant probability of detection, let λ_{FA} denote the expected number of false alarms per unit volume of measurement space, and let λ_{new} denote the expected

number of newly appearing objects per unit volume of measurement space. Then, the track score is defined as [VMBS+15, Section V]:

$$S(T_m) = (p_{\mathrm{D}})^{L_m - 1} \cdot (1 - p_{\mathrm{D}})^{U_m} \cdot \frac{\lambda_{\mathrm{new}}}{\lambda_{\mathrm{FA}}} \cdot \prod_{l=2}^{L_m} \frac{p(\boldsymbol{z}_{T_m(l)} \mid \boldsymbol{z}_{T_m(1)}, \ldots, \boldsymbol{z}_{T_m(l-1)})}{\lambda_{\mathrm{FA}}} \tag{4.38}$$

where U_m denotes the number of missed detections and each factor

$$\begin{aligned} p(\boldsymbol{z}_{T_m(l)} \mid \boldsymbol{z}_{T_m(1)}, \ldots, \boldsymbol{z}_{T_m(l-1)}) = \\ \int p_k(\boldsymbol{z}_{T_m(l)} \mid \boldsymbol{x}_{k_{m_l}}) p(\boldsymbol{x}_{k_{m_l}} \mid \boldsymbol{z}_{T_m(1)}, \ldots, \boldsymbol{z}_{T_m(l-1)}) \, \mathrm{d}\boldsymbol{x} \end{aligned} \tag{4.39}$$

denotes the observation likelihood of a measurement on the trajectory that can be computed using a Bayes filter recursion, e.g., a Kalman filter for linear motion and measurement models. This scoring function is also used to compute the most likely set of compatible, i.e., non-overlapping, trajectories to be reported to the user.

The MCF-PHD tracker processes the observation sequence by application of the track-oriented PHD recursion where a partition of the multi-object intensity accounts for a Bayes rule-like update with each measurement. Scaled partitions form measurement-oriented tracks which are subsequently linked into globally consistent trajectories. The flow network corresponding to the scene depicted in Figure 4.4a is illustrated in Figure 4.4b. Compared to MHT, the approach leads to a considerate reduction of data association complexity. In particular, the MCF-PHD tracker does not suffer from the same exponential growth of trajectory hypotheses that MHT suffers from. The purpose of this section is to shed light on the approximation that leads to this reduced complexity and the limitations that come along, but it also provides justification for the trajectories generated by the MCF-PHD tracker in showing similarities between the two approaches.

It has been noted before that the trajectory cost (4.22) of the MCF-PHD tracker has been motivated by the track scoring function of MHT. In Appendix C it is shown that this cost is approximately equal to the negative logarithm of the MHT trajectory score

$$C_m(T_m) \approx -\log S(T_m) \tag{4.40}$$

under the following premises:

1. Let V denote the volume of the surveillance region. Then, clutter is a Poisson RFS with false alarm rate $\lambda_{\mathrm{FA}} \cdot V$ and spatial density $1/V$, i.e., $c_k(\boldsymbol{z}) = (\lambda_{\mathrm{FA}} \cdot V) \cdot \frac{1}{V} = \lambda_{\mathrm{FA}}$.

2. Newly appearing objects are always detected and the PHD likelihood of appearing object is equal to $\tau_k^{(\mathrm{u})}(\boldsymbol{z}) = (\lambda_{\mathrm{new}} \cdot V) \cdot \frac{1}{V} = \lambda_{\mathrm{new}}$.

3. The probability of detection is constant $p_{\mathrm{D}}(\boldsymbol{x}) = p_{\mathrm{D}}$ and the probability of survival $p_{\mathrm{S}}(\boldsymbol{x}) = 1$ evaluates to one.

4. In computation of observation likelihood (4.39), the predicted state is approximately distributed according to a probability density function that is proportional to the predecessor's track partition:

$$p(\boldsymbol{z}_{T_m(l)} \mid \boldsymbol{z}_{T_m(1)}, \ldots, \boldsymbol{z}_{T_m(l-1)}) \approx \int p_{k_{m_l}}(\boldsymbol{z}_{T_m(l)} \mid \boldsymbol{x}) p_{k_{m_l}|k_{m_l}-1}^{(k_{m_{l-1}}, j_{m_{l-1}})}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \tag{4.41}$$

where

$$p_{k_{m_l}|k_{m_l}-1}^{(k_{m_{l-1}}, j_{m_{l-1}})}(\boldsymbol{x}) \propto v_{k_{m_l}|k_{m_l}-1}^{(k_{m_{l-1}}, j_{m_{l-1}})}(\boldsymbol{x}). \tag{4.42}$$

Assumptions 1–3 are concrete modeling decisions that render the system representation underlying the PHD filter comparable to MHT. As such, they impose no general limitation on the performance of the MCF-PHD tracker. Therefore, the key approximation is in assumption 4. Whereas the MHT observation likelihood (4.39) is computed under consideration of the history of measurements on the trajectory, likelihood (4.41) is computed from the direct predecessor's track partition. This partition is created by a Bayes rule-like update of the predicted multi-object intensity with the preceding measurement on the trajectory. Since the intensity is a multi-object descriptor that characterizes motion of multiple objects, the measurement partition may also contain multiple (weighted) motion hypotheses. This circumstance is illustrated in Figure 4.4a. Track hypotheses in the track-oriented PHD filter summarize all trajectories that pass through the corresponding measurement. The practical consequence is the following. MHT captures long-term statistical dependencies by consideration of the full history of measurements on the trajectory. The MCF-PHD tracker, on the other hand, only evaluates the compatibility between the measurement and its direct predecessor under consideration of multiple weighted motion hypotheses that arise out of the PHD recursion. In contrast to MHT where the observation likelihood is computed with respect to exactly the one trajectory in question, the observation likelihood of the MCF-PHD tracker considers all trajectories that pass through the direct predecessor. It is due to this approximation that trajectories can be recovered in a min-cost flow problem and, therefore, it is not necessary to enumerate all possible trajectories as in MHT. In this regard, the MCF-PHD tracker trades off model capacity against computational complexity.

(a) (b) (c)

Figure 4.5: Three examples for low-confidence false alarms taken from the PETS 2009 dataset [FS09]. These typically originate (a) from misclassified static scene geometry, (b)–(c) poor detector localization accuracy. Each detection is annotated with a detector confidence score that is often much lower for false alarms than for real objects.

4.6 Integration of Detector Confidence Scores

FISST has been presented as a theory that provides comprehensive means of modeling multi-object phenomena and the PHD recursion itself deals with all notable sources of uncertainty involved in multi-object state estimation, including process and measurement noise as well as the uncertainty involved in data association. However, successful application of the PHD filter requires knowledge of clutter characteristics and the performance of the filter degrades substantially if these parameters are chosen incorrectly. In practical applications, precise knowledge about the clutter process is often unavailable. Then it is common practice to assume a uniform spatial distribution. This is usually a poor choice in visual tracking scenarios because detections are generated from a classifier that processes the image in a sliding window. Due to localization inaccuracies, this object detector more likely generates false alarms in the surrounding of the true object locations.

Figure 4.5 shows three examples for false alarms as they are often found in visual tracking applications. In Figure 4.5a, part of the static scene geometry is mistakenly detected as person. Figure 4.5b and 4.5c show poorly localized detections that are generated around the true object. All three depicted cases pose a challenge to the tracker because false alarms of this sort are typically reappearing from time to time—though at a lower frequency—and must be suppressed.

One of the key strengths of the random finite set formulation is the ability to handle data association implicitly, that is, no hard decision on measurement-

to-track assignments need to be made. Instead, methods following this paradigm aggregate evidence over multiple time steps and association hypotheses to infer the number of objects in the scene directly from data. Towards this end, the detector confidence provides a valuable source of information. As can be seen in Figure 4.5, the detector confidence score is often much lower for false alarms than for correct detections.

Integration of detector confidence scores into the PHD filter is straight-forward by adapting the clutter density and measurement likelihood function. If each measurement $\boldsymbol{z} = (\boldsymbol{y}, s)^{\mathrm{T}}$ contains a spatial component $\boldsymbol{y}$, i.e., bounding box position or any other observed part of the object state, and a detector confidence score s, then a likelihood function for the detector confidence can be learned from extraneous training data. For the most widely-adopted multiple object tracking benchmarks such data is provided, but even if training data for the particular tracking application is not available, there typically exists a dataset that has been used to train and validate the object detector. This dataset can be used if the two scenarios do not diverge *too much*. Denote by $p_{\mathrm{bg}}(s)$ the likelihood that a detection with confidence score s has been generated by clutter and denote by $p_{\mathrm{fg}}(s)$ the likelihood that the corresponding detection has been generated by an object. Assuming the detector confidence is independent of object location, the clutter density and measurement likelihood factorize into a product of independent components:

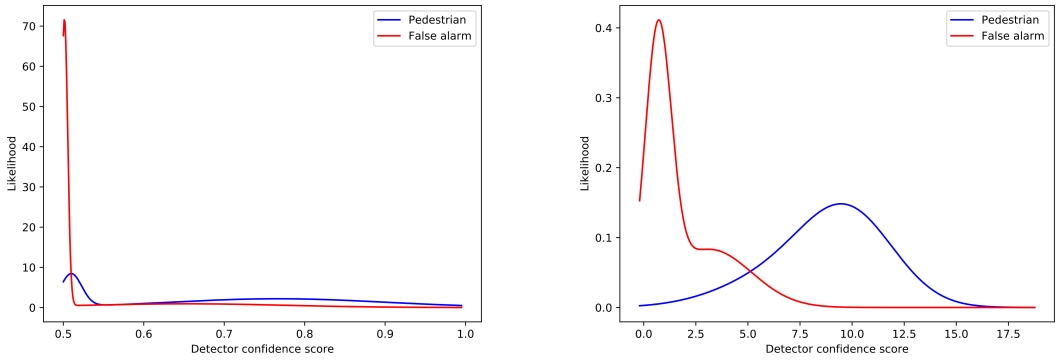
$$p_{\mathrm{c},k}(\boldsymbol{z}) = p_{\mathrm{bg}}(s) p_{\mathrm{c},k}(\boldsymbol{y}), \tag{4.43}$$

$$p_k(\boldsymbol{z} \mid \boldsymbol{x}) = p_{\mathrm{fg}}(s) p_k(\boldsymbol{y} \mid \boldsymbol{x}). \tag{4.44}$$

Such a factorization only affects the cardinality estimate of the PHD filter, not the location of individual objects, because the clutter intensity and PHD likelihood in the measurement-corrected partitions are re-weighted by the detector confidence likelihood globally, independent of state and observed location. Using likelihood functions (4.43) and (4.44), the measurement-corrected partitions of the PHD filter update (2.24) can be written as

$$v_{\mathrm{U},k}(\boldsymbol{z}_{k,j},\boldsymbol{x}) = \frac{p_{\mathrm{fg}}(s_{k,j}) \cdot p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{y}_{k,i} \mid \boldsymbol{x}) v_{k|k-1}(\boldsymbol{x})}{p_{\mathrm{bg}}(s_{k,j}) \cdot \lambda_{\mathrm{c},k}\, p_{\mathrm{c},k}(\boldsymbol{y}) + p_{\mathrm{fg}}(s_{k,j}) \int p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{y}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}(\boldsymbol{x})}. \tag{4.45}$$

The potential benefit of this approach is a faster confirmation at high-confidence locations and better suppression of low-confidence detections. How well the detector confidence guides the cardinality estimate highly depends on the detector and tracking scenario. Figure 4.6 shows two examples of a learned detector confidence likelihood. The first example has been trained on the PETS 2009 [FS09] dataset. This dataset contains many densely crowded scenes with partial occlusions. Consequently, the detector does not discriminate false alarms from objects



(a) **(b)**

Figure 4.6: Detector confidence likelihood that has been estimated from training data of the (a) PETS 2009 dataset [FS09] and (b) KITTI vision benchmark [GLU12].

well. In particular, the detector produces many low-confidence detections near to the cut-off threshold 0.5. The second example has been obtained from the KITTI vision benchmark [GLU12]. On this dataset, false alarms and true detections are separated much better. Through the detector confidence likelihood, these dataset characteristics are transparent to the PHD filter.

4.7 Integration of Appearance Information

Appearance provides a valuable source of information for the tracking task. In particular when occlusions cause motion to become less discriminative after several prediction steps which successively increase the uncertainty associated with each track hypothesis. In such situations, appearance potentially remains similar to the last observed detection. So far, the MCF-PHD tracker has been described as a general tracking method for point targets where the cost terms are aligned with the PHD filter recursion. The purpose of this section is to outline how appearance can be integrated into this formulation.

A conceptually straight-forward way to approach the integration of appearance into the tracking framework would be to augment the state space with appearance information and apply the PHD recursion on a hybrid motion-appearance space $\boldsymbol{x} = (\boldsymbol{x}_{\mathrm{mot}}, \boldsymbol{x}_{\mathrm{app}})^{\mathrm{T}}$. Then, appearance can be integrated into the measurement likelihood function as follows. If each measurement $\boldsymbol{z} = (\boldsymbol{y}, \boldsymbol{r})^{\mathrm{T}}$ contains a spatial component $\boldsymbol{y}$

and an appearance feature $\boldsymbol{r}$, and if appearance is taken independent of motion, then the measurement likelihood function can be written as

$$p_k(\boldsymbol{z} \mid \boldsymbol{x}) = p_k(\boldsymbol{y}, \boldsymbol{r} \mid \boldsymbol{x}_{\text{mot}}, \boldsymbol{x}_{\text{app}}) \tag{4.46}$$
$$= p_k(\boldsymbol{y} \mid \boldsymbol{x}_{\text{mot}}) p(\boldsymbol{r} \mid \boldsymbol{x}_{\text{app}}). \tag{4.47}$$

There are two problems with this approach. First, since there is usually no model available to describe how object appearance changes over time, filtering on this space is not straight-forward. Second, such a hybrid motion-appearance space would become high dimensional, making filtering generally inefficient.

The path followed here is conceptually similar, but simpler: The PHD filter recursion is only applied to the motion component of the object state, i.e., $\boldsymbol{x} = \boldsymbol{x}_{\text{mot}}$. Assuming appearance remains relatively stationary over time and is independent of motion, each track hypothesis is updated according to a measurement model that takes into account its associated feature:

$$p_k^{(t,i)}(\boldsymbol{z} \mid \boldsymbol{x}) = p_k^{(t,i)}(\boldsymbol{y}, \boldsymbol{r} \mid \boldsymbol{x}) \tag{4.48}$$
$$= p_k(\boldsymbol{y} \mid \boldsymbol{x}) p(\boldsymbol{r} \mid \boldsymbol{r}_{t,i}). \tag{4.49}$$

The measurement likelihood of unseen objects and the spatial density of the Poisson clutter RFS are also adapted to account for object appearance. Since it is usually not possible to describe the appearance of false alarms and unseen objects accurately, both are characterized by the same appearance model $p(\boldsymbol{r} \mid \emptyset)$ that takes a simple form, e.g., uniform over a part of the feature space. A concrete parametrization based on the feature representation that has been trained in the metric learning framework of Chapter 3 will be discussed together with a practical Gaussian mixture implementation in Section 4.8. In its general form, the corresponding likelihood functions of clutter and unseen objects are:

$$p_k^{(\text{u})}(\boldsymbol{z} \mid \boldsymbol{x}) = p(\boldsymbol{r} \mid \emptyset) p_k(\boldsymbol{y} \mid \boldsymbol{x}), \tag{4.50}$$
$$p_{\text{c},k}(\boldsymbol{z}) = p(\boldsymbol{r} \mid \emptyset) p_{\text{c},k}(\boldsymbol{y}). \tag{4.51}$$

The following two paragraphs outline the integration into the PHD filter recursion and trajectory estimation objective. Section 4.8 sheds further light on a concrete parametrization in a practical implementation.

PHD Filter In the track-oriented PHD filter, a new track hypothesis is created from the predicted intensity and clutter characteristics for each measurement according to Equation 4.10 and 4.11. The measurement model that is used therein $p_k(\boldsymbol{z} \mid \boldsymbol{x})$ does not account for a particular track hypothesis that the object $\boldsymbol{x}$ originates from, as proposed in Equation 4.49. It has been noted before that the MCF-PHD tracker

is only loosely coupled to the PHD filter because measurement-corrected partitions cannot formally be treated as individual objects within the PHD filtering framework. The remainder of this paragraph discusses how appearance can be integrated practically. Then, a remark shows how this approach is related to the conceptual idea outlined at the beginning of this section.

In Equation 4.10, new track hypotheses are constructed from the PHD likelihood of the multi-object state and the clutter intensity. The PHD likelihood can be written as a sum over individual measurement partitions:

$$\tau_k(\boldsymbol{z}) = \int p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{z} \mid \boldsymbol{x}) v_{k|k-1}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} \tag{4.52}$$

$$\begin{aligned} &= \int p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{z} \mid \boldsymbol{x}) u_{k|k-1}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} \\ &\quad + \sum_{\boldsymbol{z}_{t,i} \in Z_{1:k-1}} r_{t,i} \int p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{z} \mid \boldsymbol{x}) v_{k|k-1}^{(t,i)}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}. \end{aligned} \tag{4.53}$$

If appearance information are available, these are integrated into the PHD likelihood by replacing measurement likelihood $p_k(\boldsymbol{z} \mid \boldsymbol{x})$ by its counterpart (4.49) and (4.50):

$$\begin{aligned} \tau_k(\boldsymbol{z}) &= \int p_{\mathrm{D}}(\boldsymbol{x}) p_k^{(\mathrm{u})}(\boldsymbol{z} \mid \boldsymbol{x}) u_{k|k-1}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} \\ &\quad + \sum_{\boldsymbol{z}_{t,i} \in Z_{1:k-1}} r_{t,i} \int p_{\mathrm{D}}(\boldsymbol{x}) p_k^{(t,i)}(\boldsymbol{z} \mid \boldsymbol{x}) v_{k|k-1}^{(t,i)}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} \end{aligned} \tag{4.54}$$

$$\begin{aligned} &= p(\boldsymbol{r} \mid \emptyset) \int p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{y} \mid \boldsymbol{x}) u_{k|k-1}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} \\ &\quad + \sum_{\boldsymbol{z}_{t,i} \in Z_{1:k-1}} r_{t,i}\, p(\boldsymbol{r} \mid \boldsymbol{r}_{t,i}) \int p_{\mathrm{D}}(\boldsymbol{x}) p_k(\boldsymbol{y} \mid \boldsymbol{x}) v_{k|k-1}^{(t,i)}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}, \end{aligned} \tag{4.55}$$

Using this modified PHD likelihood, the track-oriented PHD recursion can be implemented as previously presented in Section 4.3.

Remark 4.5. A similar result to the practical implementation proposed in this paragraph can be established formally within the PHD framework when filtering is performed on the augmented motion-appearance space $\boldsymbol{x} = (\boldsymbol{x}_{\mathrm{mot}}, \boldsymbol{x}_{\mathrm{app}})^{\mathrm{T}}$ that was introduced at the beginning of this section. Then, appearance becomes part of the state and, consequently, available in the measurement model $p_k(\boldsymbol{z} \mid \boldsymbol{x}) = p_k(\boldsymbol{y}, \boldsymbol{r} \mid \boldsymbol{x}_{\mathrm{mot}}, \boldsymbol{x}_{\mathrm{app}})$. If the uncertainty in the appearance likelihood is very small, then $\boldsymbol{x}_{\mathrm{app}}$ is sharply peaked around the last feature that has been used to update the state. This is feature $\boldsymbol{r}_{t,i}$ that is associated with track hypothesis $v_k^{(t,i)}(\boldsymbol{x})$. During succeeding prediction steps, this uncertainty can be inflated again. Consequently, a measurement model $p_k(\boldsymbol{y}, \boldsymbol{r} \mid \boldsymbol{x}_{\mathrm{mot}}, \boldsymbol{x}_{\mathrm{app}})$

can be established that performs similar to the pairwise relationship $p(\boldsymbol{r} \mid \boldsymbol{r}_{t,i})$ that is used here.

Trajectory Estimation The entry cost (4.23) of the trajectory estimation objective remains unaffected by integration of appearance information because unseen objects and false alarms follow the same appearance likelihood:

$$c_{\text{entry}}(\boldsymbol{z}_{k,j}) = -\log \frac{\tau_k^{(\text{u})}(\boldsymbol{z}_{k,j})}{c_k(\boldsymbol{z}_{k,j})} \tag{4.56}$$

$$= -\log \frac{p(\boldsymbol{r}_{k,j} \mid \emptyset) \int p_{\text{D}}(\boldsymbol{x})\, p_k^{(\text{u})}(\boldsymbol{y}_{k,j} \mid \boldsymbol{x}) u_{k|k-1}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}}{\lambda_{\text{c},k} \cdot p_{\text{c},k}(\boldsymbol{y}_{k,j}) p(\boldsymbol{r}_{k,j} \mid \emptyset)} \tag{4.57}$$

$$= -\log \frac{\int p_{\text{D}}(\boldsymbol{x})\, p_k(\boldsymbol{y}_{k,j} \mid \boldsymbol{x}) u_{k|k-1}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}}{c_k(\boldsymbol{y}_{k,j})} \tag{4.58}$$

$$= c_{\text{entry}}(\boldsymbol{y}_{k,j}). \tag{4.59}$$

The transition cost (4.26) on the other hand factorizes into an appearance and a motion component that each score the PHD likelihood of the predecessor against clutter:

$$c_{\text{link}}(\boldsymbol{z}_{t,i}, \boldsymbol{z}_{k,j}) = -\log \frac{\int p_{\text{D}}(\boldsymbol{x}) p_k^{(t,i)}(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}^{(t,i)}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}}{c_k(\boldsymbol{z}_{k,j})} \tag{4.60}$$

$$= -\log \frac{p(\boldsymbol{r}_{k,j} \mid \boldsymbol{r}_{t,i}) \cdot \int p_{\text{D}}(\boldsymbol{x}) p_k(\boldsymbol{y}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}^{(t,i)}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}}{\lambda_{\text{c},k} \cdot p_{\text{c},k}(\boldsymbol{y}_{k,j}) p(\boldsymbol{r}_{k,j} \mid \emptyset)} \tag{4.61}$$

$$= -\log \left(\frac{p(\boldsymbol{r}_{k,j} \mid \boldsymbol{r}_{t,i})}{p(\boldsymbol{r}_{k,j} \mid \emptyset)} \cdot \frac{\tau_k^{(t,i)}(\boldsymbol{y}_{k,j})}{c_k(\boldsymbol{y}_{k,j})} \right). \tag{4.62}$$

Alternatively, this can be written as a sum of an appearance cost term and a motion cost term:

$$c_{\text{link}}(\boldsymbol{z}_{t,i}, \boldsymbol{z}_{k,j}) = c_{\text{link}}^{(\text{app})}(\boldsymbol{r}_{t,i}, \boldsymbol{r}_{k,j}) + c_{\text{link}}^{(\text{mot})}(\boldsymbol{y}_{t,i}, \boldsymbol{y}_{k,j}), \tag{4.63}$$

where

$$c_{\text{link}}^{(\text{app})}(\boldsymbol{r}_{t,i}, \boldsymbol{r}_{k,j}) = -\log \frac{p(\boldsymbol{r}_{k,j} \mid \boldsymbol{r}_{t,i})}{p(\boldsymbol{r}_{k,j} \mid \emptyset)}, \tag{4.64}$$

$$c_{\text{link}}^{(\text{mot})}(\boldsymbol{y}_{t,i}, \boldsymbol{y}_{k,j}) = -\log \frac{\tau_k^{(t,i)}(\boldsymbol{y}_{k,j})}{c_k(\boldsymbol{y}_{k,j})}. \tag{4.65}$$

A concrete implementation will be presented in the following section.

4.8 Gaussian Mixture Implementation

This section describes a Gaussian mixture implementation of the MCF-PHD tracker. The description starts with pseudo code for the track-oriented GM-PHD filter that is based on the procedures of the standard GM-PHD recursion of Section 2.3.3 and continues with a practical parametrization of the appearance likelihood for a feature representation that has been trained in the metric learning framework of Chapter 3.

4.8.1 Track-Oriented GM-PHD Filter

At each time k, the Gaussian mixture MCF-PHD tracker maintains a partitioned intensity (4.1) that contains $L_k^{(t,i)}$ components for each track hypothesis $v_k^{(t,i)}(\boldsymbol{x})$:

$$v_k(\boldsymbol{x}) = \sum_{\boldsymbol{z}_{t,i} \in Z_{1:k-1}} r_{t,i} \underbrace{\sum_{n=1}^{L_k^{(t,i)}} w_k^{(t,i,n)} \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}_k^{(t,i,n)}, \boldsymbol{P}_k^{(t,i,n)})}_{v_k^{(t,i)}(\boldsymbol{x})}. \tag{4.66}$$

Note that this mixture does not contain components that account for the set of undetected objects $u_k(\boldsymbol{x})$. Following the implementation of the GM-PHD filter in Section 2.3.3, for efficiency reasons it is assumed that objects are detected when they first enter the scene. Therefore, the set of undetected objects is empty and $u_k(\boldsymbol{x}) = 0$. Instead, for each measurement a single Gaussian birth component is created during initialization of the track hypothesis. Further, in practice it is undesirable to maintain track hypotheses indefinitely because this would lead to an unbounded growth of Gaussian mixture components. As will be seen shortly, components with low weight are removed from the mixture during a pruning stage in which track hypotheses with low intensity mass vanish.

Algorithm 4 presents pseudo code for a practical implementation of the track-oriented GM-PHD measurement correction step. The procedure largely follows the standard GM-PHD measurement correction of Algorithm 2. The most prominent changes are the integration of the appearance likelihood in line 7 and the integration of detector confidence scores in line 14.

Pseudo code for the track-oriented GM-PHD recursion are given in Algorithm 5 and 6. For notational brevity, parameters of the Gaussian mixture corresponding to track hypothesis $v_k^{(t,i)}(\boldsymbol{x})$ are denoted by

$$X_k^{(t,i)} = \{w_k^{(t,i,n)}, \boldsymbol{m}_k^{(t,i,n)}, \boldsymbol{P}_k^{(t,i,n)}\}_{n=1}^{L_k^{(t,i)}}. \tag{4.67}$$

Algorithm 5 describes the propagation of an existing track hypothesis from time $k-1$ to time k by application of the GM-PHD predictor (line 2), followed by a down-

Algorithm 4 Track-Oriented GM-PHD corrector

1: **procedure** MCFPHD_CORRECT($\boldsymbol{z}_{k,j}$)
2: **for** $\boldsymbol{z}_{t,i} \in Z_{1:k-1}$ **do**
3: m = 1
4: **for** $n = 1, \ldots, L_{k|k-1}^{(t,i)}$ **do**
5: $\boldsymbol{m}_k^{(k,j,m)}, \boldsymbol{P}_k^{(k,j,m)} = \text{KALMAN\_UPDATE}(\boldsymbol{m}_{k|k-1}^{(t,i,n)}, \boldsymbol{P}_{k|k-1}^{(t,i,n)}, \boldsymbol{y}_{k,j})$
6: $l_k^{(k,j,m)} = \int p_k^{(t,i)}(\boldsymbol{y}_{k,j} \mid \boldsymbol{x}) \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}_{k|k-1}^{(t,i,n)}, \boldsymbol{P}_{k|k-1}^{(t,i,n)}) \, \mathrm{d}\boldsymbol{x}$
7: $\tau_k^{(k,j,m)} = r_{t,i} \cdot p(\boldsymbol{r}_{k,j} \mid \boldsymbol{r}_{t,i}) p_{\mathrm{D}} \, l_k^{(k,j,m)} w_{k|k-1}^{(t,i,n)}$
8: $m := m + 1$
9: **end for**
10: **end for**
11: $\boldsymbol{m}_k^{(k,j,m)}, \boldsymbol{P}_k^{(k,j,m)}, \tau_k^{(k,j,m)} = \text{BIRTH\_COMPONENT}(\boldsymbol{z}_{k,j})$
12: $L_k^{(k,j)} = m$
13: **for** $m := 1, \ldots, L_k^{(k,j)}$ **do**
14: $$w_k^{(k,j,m)} = \frac{p_{\mathrm{fg}}(s_{k,j}) \tau_k^{(k,j,m)}}{p_{\mathrm{bg}}(s_{k,j}) p(\boldsymbol{r}_{k,j} \mid \emptyset) c_k(\boldsymbol{y}_{k,j}) + p_{\mathrm{fg}}(s_{k,j}) \sum_{i=1}^{L_k^{(k,j)}} \tau_k^{(k,j,i)}}$$
15: **end for**
16: **return** $\{w_k^{(k,j,m)}, \boldsymbol{m}_k^{(k,j,m)}, \boldsymbol{P}_k^{(k,j,m)}\}_{m=1}^{L_k^{(k,j,m)}}$
17: **end procedure**

Algorithm 5 Propagation of an existing track hypothesis $v_{k-1}^{(t,i)}(\boldsymbol{x})$

1: **procedure** PROPAGATE_TRACK($X_{k-1}^{(t,i)} = \{w_{k-1}^{(t,i)}, \boldsymbol{m}_{k-1}^{(t,i)}, \boldsymbol{P}_{k-1}^{(t,i)}\}_{i=1}^{L_{k|k-1}^{(t,i)}}$)
2: $X_{k|k-1}^{(t,i)} = \text{GMPHD\_PREDICT}(X_{k-1}^{(t,i)})$
3: $X_k^{(t,i)} = \text{GMPHD\_MISSED\_DETECTIONS}(X_{k|k-1}^{(t,i)})$
4: $X_k^{(t,i)} := \text{PRUNE\_COMPONENTS}(X_{k-1}^{(t,i)})$
5: **end procedure**

weighting of mixture weights to account for missed detections (line 3). The procedure is terminated by pruning components with negligible weights and merging similar components. As in the original GM-PHD recursion, this is necessary to avoid an exponential growth of mixture components. During the pruning step, a

Algorithm 6 Initialization of a new track hypothesis $v_k^{(k,j)}$ from measurement $\boldsymbol{z}_{k,j}$

1: **procedure** INIT_TRACK($\boldsymbol{z}_{k,j}$)
2: $\{w_k^{(k,j,n)}, \boldsymbol{m}_k^{(k,j,n)}, \boldsymbol{P}_k^{(k,j,n)}\}_{n=1}^{L_k^{(k,j)}} = \text{MCFPHD\_CORRECT}(\boldsymbol{z}_{k,j})$
3: $r_{k,j} = \sum_{n=1}^{L_k^{(t,i)}} w_k^{(k,j,n)}$
4: $w_k^{(k,j,n)} := \dfrac{w_k^{(k,j,n)}}{r_{k,j}}$, for $n = 1, \ldots, L_k^{(k,j)}$
5: $X_k^{(k,j)} = \text{PRUNE\_COMPONENTS}(\{w_k^{(k,j,n)}, \boldsymbol{m}_k^{(k,j,n)}, \boldsymbol{P}_k^{(k,j,n)}\}_{n=1}^{L_k^{(k,j)}})$
6: **end procedure**

track hypothesis may loose all of its mixture components when the intensity mass becomes negligible. Then, the track hypothesis also vanishes from the Gaussian mixture multi-object intensity (4.66).

Initialization of a new Gaussian mixture for track hypothesis $v_k^{(k,j)}(\boldsymbol{x})$ that originates from measurement $\boldsymbol{z}_{k,j}$ is given in Algorithm 6. In line 2, the intensity partition is created by application of the track-oriented GM-PHD corrector (Algorithm 4). This procedure functions as a plug-in replacement for the standard GM-PHD corrector of Algorithm 1. In line 3, scaling term $r_{k,j}$ is computed from the intensity mass in the measurement-corrected partition. Consequently, in line 4 weights are normalized to sum up to one, i.e., $\int v_k^{(k,j)}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = 1$. The algorithm is finalized by a pruning step in line 5 where low-weighted components are removed and similar components are merged.

It is worth noting that the presented algorithm is computationally more complex than the standard GM-PHD implementation because each Gaussian component is associated to exactly one track hypothesis, whereas the standard GM-PHD filter also merges similar components that originate from different measurements. This leads to an increased number mixture components. A computationally more efficient implementation of the track-oriented GM-PHD filter keeps a global pool of Gaussian mixture components that each have an associated weight vector for the individual track hypotheses. Then, the computational efficiency of the standard GM-PHD filter can be retained by application of the pruning and merging strategy to the global component pool. However, this more efficient implementation is also less accurate in dense tracking scenarios. If multiple track hypotheses are represented by the same Gaussian component, their transition cost terms are computed from the same state hypothesis. Then, it is left solely to the appearance likelihood to discriminate between individuals.

4.8.2 Trajectory Estimation Cost Terms

The flow network of the trajectory estimation problem can be constructed during application of the track-oriented PHD recursion by adding nodes for new measurements and linking them to candidate predecessors. In the following, the cost terms of this network are described in closer detail for a very general tracking scenario where no special knowledge about the environment is assumed. Therefore, clutter and birth are modeled by Poisson RFSs with uniform spatial density. More specifically, if V denotes the volume of the surveillance region and $\lambda_{\mathrm{c},k}$ is the expected number of clutter returns at time k, then $c_k(\boldsymbol{y}) = \lambda_{\mathrm{c},k} \cdot V^{-1}$ is the intensity of the Poisson clutter RFS. Likewise, if $\lambda_{\mathrm{b},k}$ is the expected number of appearing objects at time k that are uniformly distributed in the surveillance volume, then $\tau_k^{(\mathrm{u})}(\boldsymbol{y}) = \lambda_{\mathrm{b},k} \cdot V^{-1}$ is the PHD likelihood associated with birth components. Under these very general modeling assumptions, the trajectory entry cost (4.23) simplifies to

$$c_{\mathrm{entry}}(\boldsymbol{z}_{k,j}) = -\log \frac{\tau_k^{(\mathrm{u})}(\boldsymbol{z}_{k,j})}{c_k(\boldsymbol{z}_{k,j})} = -\log \frac{\lambda_{\mathrm{b},k}}{\lambda_{\mathrm{c},k}}. \tag{4.68}$$

It is straight-forward to adapt this model to scenarios where scene-specific information are available by exchanging the uniform distribution with more informed densities. For example, in some surveillance scenarios objects are more likely to appear at image borders. This can be accounted for by increasing the spatial density of the clutter and birth RFS at these locations.

In Equation 4.63, the transition cost has been formulated as a sum of a motion and an appearance term. For uniform clutter and birth densities, the motion term can be computed from the Gaussian mixture implementation as follows:

$$c_{\mathrm{link}}^{(\mathrm{mot})}(\boldsymbol{y}_{t,i}, \boldsymbol{y}_{k,j}) = -\log \frac{p_{\mathrm{D}} \int p_k(\boldsymbol{y}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}^{(t,i)}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}}{c_k(\boldsymbol{y}_{k,j})} \tag{4.69}$$

$$= -\log \frac{p_{\mathrm{D}} \sum_{n=1}^{L_{k|k-1}^{(t,i)}} l_k^{(t,i,n)} w_{k|k-1}^{(t,i,n)}}{\lambda_c V^{-1}}, \tag{4.70}$$

where

$$l_k^{(t,i,n)} = \int p_k(\boldsymbol{y}_{k,j} \mid \boldsymbol{x}) \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}_{k|k-1}^{(t,i,n)}, \boldsymbol{P}_{k|k-1}^{(t,i,n)}) \,\mathrm{d}\boldsymbol{x} \tag{4.71}$$

is the observation likelihood computed by the Kalman filter update of the n-th component. For computation of the appearance term, recall the metric learning framework of Chapter 3. By application of a neural network, a feature representation has been trained that assigns images of the same identity a high cosine similarity and images of different identities a small cosine similarity. Given any two unit-

length features $\boldsymbol{r}_{t,i}, \boldsymbol{r}_{k,j} \in \mathbb{R}^d$ with $\|\boldsymbol{r}_{t,i}\|_2 = 1$ and $\|\boldsymbol{r}_{k,j}\|_2 = 1$, the cosine similarity can be computed by:

$$s(\boldsymbol{r}_{t,i}, \boldsymbol{r}_{k,j}) = \boldsymbol{r}_{t,i}^{\mathrm{T}} \boldsymbol{r}_{k,j}. \tag{4.72}$$

This relationship can be expressed by a von Mises-Fisher distribution placed around $\boldsymbol{r}_{t,i}$ with concentration κ:

$$p(\boldsymbol{r}_{k,j} \mid \boldsymbol{r}_{i,j}) = c_d(\kappa) \exp\left(\kappa \cdot s(\boldsymbol{r}_{k,j}, \boldsymbol{r}_{t,i})\right), \tag{4.73}$$

where $c_d(\kappa)$ is a normalizing constant. This distribution peaks around $\boldsymbol{r}_{t,i}$ and decays when the cosine similarity decreases. The concentration parameter κ determines how sharp the distribution is peaked. Determining meaningful parameters for this distribution can be challenging in practice. A practical parametrization can be established as follows. By choosing

$$p(\boldsymbol{r}_{k,j} \mid \emptyset) = c_d(\kappa) \exp\left(\kappa \cdot s_{\min}\right) \tag{4.74}$$

for some minimum cosine similarity $s_{\min}$ between two neighboring detections on the same trajectory, the appearance component of the transition cost term becomes a log-linear model, parametrized by concentration κ and cosine similarity threshold $s_{\min}$:

$$c_{\text{link}}^{(\text{app})}(\boldsymbol{r}_{t,i}, \boldsymbol{r}_{k,j}) = -\log \frac{p(\boldsymbol{r}_{k,j} \mid \boldsymbol{r}_{t,i})}{p(\boldsymbol{r}_{k,j} \mid \emptyset)} = \kappa \cdot \left(s(\boldsymbol{r}_{k,j}, \boldsymbol{r}_{t,i}) - s_{\min}\right). \tag{4.75}$$

This term becomes negative if the cosine similarity is higher than the threshold $s_{\min}$ and positive or zero otherwise. In consequence, the transition cost term is a weighted combination of a motion and an appearance model with motion costs computed from the PHD filter recursion and appearance costs based on a pairwise similarity metric. Both models are weighted off by concentration parameter κ that scales the importance of the cosine similarity between neighboring measurements on the trajectory.

4.9 Evaluation

This section provides an evaluation of the MCF-PHD tracker in simulation and on standard benchmarks. The first part of the evaluation is dedicated to a comparison of the locations reported by the MCF-PHD tracker against the locations reported by the PHD filter. The second part of the evaluation draws a comparison against the state of the art, in particular to the well-established tracking-by-detection network flow formulation of Zhang *et al.* [ZLN08] and its derivatives. Evaluation is carried out on standard benchmarks to provide insight on the overall performance of the method.

Note that the most limiting factor of tracking performance is quality of detections [LBLA16, WBP17a]. Due to the success of deep learning, detector performance has increased rapidly within the last few years [RHGS15, HRS+16]. With more stable detections, fewer data association ambiguities arise and the gap between simpler and more complex tracking methods decreases. In consequence, it is easy to improve the presented results by application of a better object detector, but this can be expected to hold for any tracking framework. Standard detections are used in all benchmarks in order to eliminate the influence of object detection on overall reported performance.

4.9.1 Metrics

Evaluation of multiple object tracking systems is a complex task and a number of metrics have been proposed for this purpose. This section presents the most widely adopted metrics that will be used—depending on the specific task sometimes in part, in combination, or in full—to assess tracking performance throughout this chapter.

Precision, Recall, $\mathbf{F_1}$ score The first three metrics are used to evaluate the tracker's ability to filter false alarms from correct object detections. Therefore, these metrics do not take into account the identity of each object, but treat the problem as binary classification. Precision is the ratio of true positive tracker responses (TP) against the sum of true positives and false alarms (FA):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FA}}. \tag{4.76}$$

It measures the fraction of correct responses among all reported locations. Recall is the ratio of true positive tracker responses against the sum of true positives and false negatives (FN):

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{4.77}$$

It measures the fraction of correct responses among all ground truth positions. The F_1 score measures the overall accuracy by taking the harmonic mean of precision and recall:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{4.78}$$

All three measures take values in $[0, 1]$ and a higher value indicates better performance.

Mostly Tracked, Mostly Lost, ID switches, Fragments The following metrics express tracking performance in terms of individual ground truth object identities. These metrics have first been proposed by Li *et al.* [LHN09] and are now widely adopted:

- *Mostly Tracked* Number of ground truth tracks that are covered by tracker output for more than 80% of their life span (higher is better).
- *Mostly Lost* Number of ground truth tracks that are covered by tracker output for less than 20% of their life span (smaller is better).
- *ID switches* The total number of times that a tracked object changes its assigned ground truth identity (smaller is better).
- *Fragments* The total number of times that a ground truth trajectory is interrupted in the tracking output, e.g., due to an identity switch or a missed detection.

Note that the Most Tracked and Mostly Lost metrics do not penalize identity switches because they evaluate whether the ground truth track is covered by *any* track hypothesis.

CLEAR MOT The CLEAR MOT metrics [BS08] have been developed out of the need to establish standard evaluation criteria for multiple object tracking and they have become the most widely accepted metrics, today.

The MOTP metric evaluates tracking precision in terms of localization accuracy without consideration of assigned identities. The metric is defined as the total error in the estimated position normalized by the number of tracker-to-ground-truth associations:

$$\mathrm{MOTP} = 100 \cdot \frac{\sum_{t,i} d_t^{(i)}}{\sum_t c_t}, \tag{4.79}$$

where $d_t^{(i)}$ is, for the i-th association at time t, the distance between the estimated and the ground truth position and where c_t is the total number of associations made at time t. The distance is established either by bounding box overlap, if evaluation is performed in image coordinates, or by the Euclidean distance, if evaluation is carried out in world coordinates. A higher MOTP value indicates higher precision.

The MOTA metric summarizes tracking accuracy in terms of false negatives, false alarms, and identity switches:

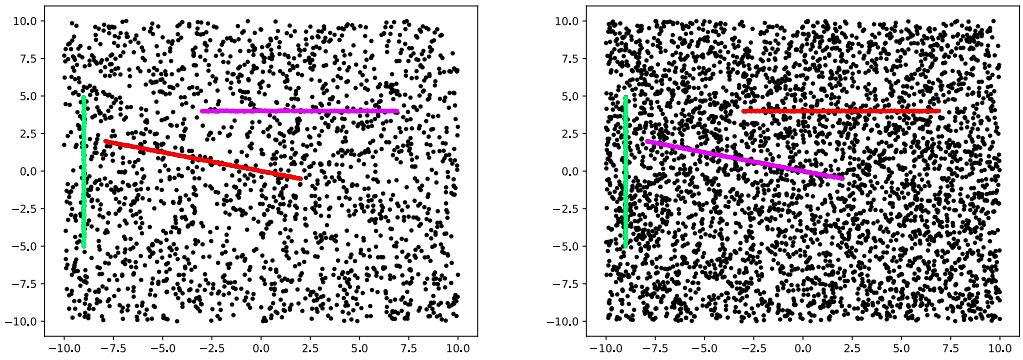
$$\mathrm{MOTA} = 100 \cdot \left(1 - \frac{\sum_t \mathrm{FN}_t + \mathrm{FA}_t + \mathrm{ID}_t}{\sum_t \mathrm{GT}_t}\right), \tag{4.80}$$

(a) Time $t-1$ **(b)** Time t

Figure 4.7: Illustration of the CLEAR MOT association strategy. At time $t-1$ an object hypothesis 1 has been associated to the red ground truth track. At time t, a new object hypothesis 2 is better aligned to the ground truth, but both associations represent a valid assignment. In this scenario, the CLEAR MOT association strategy gives priority to object hypothesis 1, because this is the more consistent solution over both frames.

where FN_t denotes the number of false negatives/missed detections, FA_t the number of false alarms, ID_t the number of identity switches, and GT_t the number of ground truth objects at time t. The metric is derived from the three error ratios $\sum_t \mathrm{FN}_t / \sum_t \mathrm{GT}_t$, $\sum_t \mathrm{FA}_t / \sum_t \mathrm{GT}_t$, $\sum_t \mathrm{ID}_t / \sum_t \mathrm{GT}_t$ that express the tracking error in terms of missed detections, false alarms, and identity switches over all frames. MOTA takes all three ratios into account equally to compute the tracking accuracy in terms of overall object configuration errors made by the tracker.

Computation of the CLEAR MOT metrics requires tracker-to-ground-truth associations. The proposed strategy to compute these associations follows a complex procedure based on a linear assignment problem that prioritizes existing associations from previous times. The procedure is illustrated in Figure 4.7 in a simplified example. If at time t an existing tracker-to-ground-truth association from time $t-1$ can be maintained at a higher overall association cost, then this solution should be preferred if the cost is below a predefined association threshold, because it provides a more globally consistent assignment over both frames. If the association between tracker and ground-truth were established purely based on the minimum cost linear assignment, small localization errors would lead frequent identity switches in densely crowded scenes with occlusions. This behavior is unwanted. However, implementation of the CLEAR MOT association strategy is not straight-forward and reported metrics can differ significantly between different implementations. If



(a) $\lambda_{c,k} = 20$ **(b)** $\lambda_{c,k} = 40$

Figure 4.8: Visualization of the simulated tracking scenario. Left: medium clutter profile with $\lambda_{c,k} = 20$ clutter returns per time step, right: high clutter profile with $\lambda_{c,k} = 40$ clutter returns per time step. Object trajectories are shown in green, red, and magenta. Black dots show the simulated sensor returns over 100 time steps.

available, the numbers report in this thesis have been computed with the software provided by benchmark authors.

4.9.2 Simulation

In a first experiment the MCF-PHD tracker is evaluated against the standard PHD filter to compare object locations reported by both methods. The experiment is carried out in a simulation to have full control over the parametrization. In the simulated environment, three objects follow a linear motion pattern with constant velocity. The scene is observed by a sensor with sensor field $20\,\mathrm{m} \times 20\,\mathrm{m}$ and zero-mean isotropic noise of standard deviation $0.01\,\mathrm{m}$. The scenario is evaluated once in a medium clutter profile with $\lambda_{c,k} = 20$ clutter returns per time step and once in a high clutter profile with $\lambda_{c,k} = 40$ clutter returns per time step. In both cases, clutter follows a uniform distribution over the sensor field. Evaluation is carried out for detection probabilities $p_D = 0.6$, $p_D = 0.7$, and $p_D = 0.8$. The simulation contains 100 time steps in total. Figure 4.8 shows the object trajectories and accumulated sensor returns.

Gaussian mixture implementations of the PHD filter (Section 2.3.3) and MCF-PHD tracker (Section 4.8) are compared. The state vector $\boldsymbol{x}_k = (x_k, y_k, \dot{x}_k, \dot{y}_k)^{\mathrm{T}}$ contains the object's position $(x_k, y_k)^{\mathrm{T}}$ and velocity $(\dot{x}_k, \dot{y}_k)^{\mathrm{T}}$. A constant velocity motion model is used to model state transitions. The motion model adds zero-mean noise with standard deviation $0.1\,\mathrm{m}$ for the position and $0.1\,\mathrm{m\,s^{-1}}$ for the velocity. The position is taken as direct observation of the object state and measurement uncertainty is set according to the simulated sensor setup, i.e., the standard deviation

	$p_D = 0.6$			$p_D = 0.7$			$p_D = 0.8$		
	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
PHD filter	0.88	0.45	0.59	0.90	0.53	0.66	0.93	0.72	0.81
MCF-PHD (online)	0.97	0.42	0.58	0.98	0.51	0.67	0.99	0.71	0.83
MCF-PHD (offline)	0.99	0.57	0.72	1.00	0.65	0.79	1.00	0.81	0.89

Table 4.1: Simulation results in a medium clutter profile ($\lambda_{c,k} = 20$). The best performing method is shown in red, the first runner up in blue.

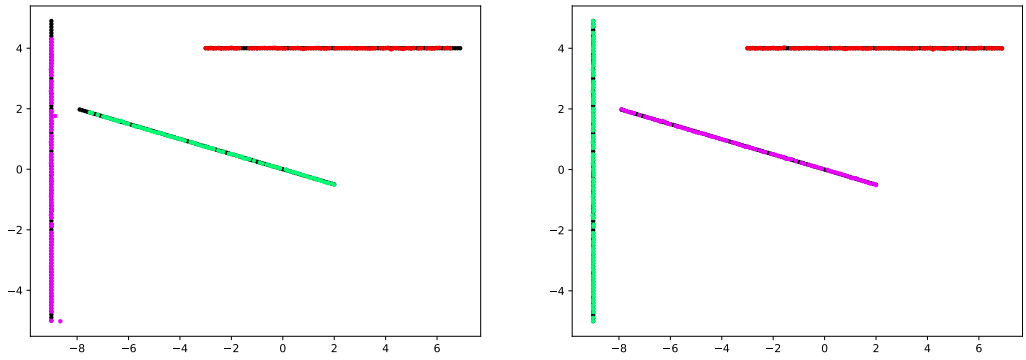
	$p_D = 0.6$			$p_D = 0.7$			$p_D = 0.8$		
	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
PHD filter	0.80	0.01	0.03	0.90	0.55	0.69	0.92	0.63	0.75
MCF-PHD (online)	1.00	0.01	0.02	0.97	0.55	0.70	0.99	0.63	0.77
MCF-PHD (offline)	1.00	0.03	0.06	1.00	0.67	0.80	1.00	0.74	0.85

Table 4.2: Simulation results in a high clutter profile ($\lambda_{c,k} = 40$). The best performing method is shown in red, the first runner up in blue. The ranking is only shown for $p_D \geq 0.7$ because all three methods fail to pick up track with $p_D = 0.6$.

is 0.01 m. For a given measurement $\boldsymbol{y} = (x_k, y_k)^T$, the birth component is initialized to mean $\boldsymbol{m}(\boldsymbol{y}) = (x_k, y_k, 0, 0)^T$ and covariance $\boldsymbol{P}(\boldsymbol{y}) = \mathrm{diag}(0.01^2, 0.01^2, 1.0, 1.0)$. The applied motion, measurement, and birth models are given in Appendix B.1–B.3 for completeness.

The remaining parameters of the PHD filter are: The probability of survival is set to $p_S = 0.95$, the clutter intensity is parametrized by $c_k(\boldsymbol{y}) = \lambda_{c,k} \cdot V^{-1}$ where $V = 400\,\mathrm{m}^2$ is the volume of the surveillance region, and the expected number of appearing objects is set to $\lambda_{b,k} = 4 \times 10^{-3}$. Gaussian mixture components with weight smaller than $w_{\min} = 1 \times 10^{-8}$ are pruned from the intensity and the component merging threshold is set to $U = 6$. Following the standard methodology, locations reported by the PHD filter are taken from mixture components with weight larger than 0.5. The threshold that is used to establish tracker-to-ground-truth associations is set to 0.1 m.

Table 4.1 and 4.2 summarize the results. Precision, recall, and F_1 score are reported for each simulated configuration. The MCF-PHD tracker is evaluated in two modes of operation. In offline mode, trajectories are found in a batch optimization over the entire observation sequence. In online mode, a fixed length history of 30 frames is optimized and matched against the previous solution at each time step. Configuration $\lambda_{c,k} = 40$ and $p_D = 0.6$ is excluded from the following evaluation because none of the evaluated methods was able to pick up objects in this high clutter, low detection profile. Therefore, $p_D = 0.7$ presents a lower bound on the signal-to-noise ratio in which the filter can be employed successfully.



(a) MCF-PHD (online) **(b)** MCF-PHD (offline)

Figure 4.9: Tracking output comparison of the MCF-PHD tracker in online and offline mode. Ground truth locations are shown by black dots, reported trajectories are shown in colors. In online mode the tracker solves the min-cost flow problem at every time step and reports the current solution. Therefore, the tracker requires a few time steps before the trajectory is confirmed. If offline mode, the full trajectory is recovered.

Throughout all simulations, the best result is obtained by using the offline MCF-PHD tracker. The F_1 score is 0.08 to 0.13 points higher than the score of the standard PHD filter. In offline mode, the MCF-PHD tracker optimizes the entire sequence in one batch. Therefore, information from the past and the future is available when a decision is made about object locations at the current time. This has a positive effect on all reported metrics. Note that, in this experiment, recall is limited by the detection probability because trajectories are defined as sequence of observations and the location at missed detections is not reported by the tracker. Therefore, the precision and recall reported by the offline MCF-PHD tracker are close to the theoretical maximum in all configurations. Since missed detections can be interpolated in a postprocessing step if necessary, this behavior imposes no limitation on the applicability of the MCF-PHD tracker.

In online mode, the min-cost flow problem is solved at each time step and current object locations are reported. While it would be possible to correct errors from the past based on the new current min-cost flow solution, such corrections are not made to render the results closer to real online tracking applications. Figure 4.9 shows the trajectory output obtained in both operational modes. Object trajectories are usually confirmed when a series of consecutive measurements can be explained by a linear motion pattern. In online mode, the object remains undetected until enough evidence is collected. In offline mode, the full trajectory is recovered under this circumstance.

The performance of the online MCF-PHD tracker is generally on par with the standard PHD filter. A general trend throughout the simulation is that the MCF-

(a) S2L1 **(b)** S2L2

Figure 4.10: Example images of two sequences from the PETS 2009 [FS09] dataset. Sequence S2L1 shows a sparse crowd, S2L2 a medium dense crowd. Detections are shown as red boxes with an annotated detector confidence score.

PHD tracker produces higher precision at lower recall. This effect is more eminent for low detection probabilities and vanishes at about $p_{\mathrm{D}} = 0.7$. Only for $p_{\mathrm{D}} = 0.6$ the tracker receives a lower F_1 score than the filter. By qualitative analysis, higher precision and low recall of the tracker are due to overall fewer reported object locations, both true positives and false alarms. In this regard, the MCF-PHD tracker confirms object locations more conservatively. This is not surprising as the PHD filter accumulates multiple trajectory hypotheses into its state estimate whereas the MCF-PHD tracker reports locations based on individual trajectory scores. Nevertheless, in terms of F_1 score the performance of both methods compares over all simulated configurations. Thus, the good filtering performance that the PHD filter exhibits under controlled simulation parameters is retained in the MCF-PHD tracker.

4.9.3 PETS 2009

The second experiment is conducted on the PETS 2009 dataset [FS09]. This dataset consists out of 7 medium and densely crowded sequences that are observed from multiple calibrated cameras. The scenario is characterized by substantial occlusions, missed detections, and false alarms as well as a largely varying number of pedestrians in the sensor field and changing lighting conditions. Therefore, the dataset is well suited to test the general applicability and sensitivity to parameter changes under changing conditions. Two representative images of this dataset are shown in Figure 4.10. Following an evaluation procedure that is comparable to other methods, tracking is carried out in view of camera 1 using standard detections and ground truth provided by Andriyenko *et al.* [ASR12]. The detections are projected

onto the ground plane using known camera calibration and evaluation is carried out in 3D. The tracker-to-ground-truth association threshold is set to 0.5 m. The dataset is split up into training and test sets as follows:

- Sequences S1L1-1 and S1L2-2 are used for parameter tuning. In particular, the parameters of the detector confidence likelihood and the appearance cost model are trained on these sequences.
- Sequences S1L1-2 (medium density crowd), S1L2-1 (high density crowd), S2L1 (sparse crowd), S2L2 (medium density crowd), S2L3 (dense crowd) are used for evaluation.

This procedure is in accordance with the numbers reported by Rezatofighi *et al.* [HRMZ+15] which will be used for a comparison against the state of the art.

Parametrization and Results The tracker is set up similarly to the simulation environment: The state vector $\boldsymbol{x}_k = (x_k, y_k, \dot{x}_k, \dot{y}_k)^{\mathrm{T}}$ contains the object's position $(x_k, y_k)^{\mathrm{T}}$ and velocity $(\dot{x}_k, \dot{y}_k)^{\mathrm{T}}$ on the ground plane. A constant velocity motion model is used to model state transitions. The standard deviation is $\Delta t \cdot 0.2\,\mathrm{m}$ for the position and $\Delta t \cdot 1.0\,\mathrm{m\,s^{-1}}$ for the velocity, where $\Delta = 0.142\,\mathrm{s}$ is the approximate time between two camera images. The position on the ground plane is taken as direct observation of the object state. The measurement model adds isotropic noise with standard deviation 0.2 m. For a given measurement $\boldsymbol{y} = (x_k, y_k)^{\mathrm{T}}$, the birth component is initialized to mean $\boldsymbol{m}(\boldsymbol{y}) = (x_k, y_k, 0, 0)^{\mathrm{T}}$ and covariance $\boldsymbol{P}(\boldsymbol{y}) = \mathrm{diag}(0.2^2, 0.2^2, 1.0, 1.0)$. The probability of survival is set to $p_{\mathrm{S}} = 0.9$ and the probability of detection to $p_{\mathrm{D}} = 0.7$. Components with weight smaller than $w_{\min} = 1 \times 10^{-10}$ are pruned from the intensity and the merge threshold is set to $U = 6$. The following parameters are found on training sequences by visual comparison of tracking output against the provided ground truth: The clutter intensity is set to $c_k(\boldsymbol{y}) \approx 1.27 \times 10^{-1}\,\mathrm{m^{-2}}$ and the PHD likelihood of birth components is set to $\tau_k^{(\mathrm{u})}(\boldsymbol{y}) = 7.83 \times 10^{-3}\,\mathrm{m^{-2}}$. A Gaussian mixture with two components describes the detector confidence likelihood $p_{\mathrm{fg}}(s)$ and $p_{\mathrm{bg}}(s)$. This likelihood is visualized in Figure 4.6a. If appearance information are used, then $\kappa = 10.0$ and $s_{\min} = 0.9$.

The results of the experiment are given in Table 4.3. The numbers for the MCF-PHD tracker have been generated in offline mode, that is, trajectories are recovered by processing the entire sequence in a single optimization. In a preprocessing stage, non-maximum suppression is applied before feeding detections to the tracker. In a post processing stage, the location at missed detections are recovered by interpolation, but no additional filter logic is applied. This is different from at least one method (Rezatofighi *et al.* [HRMZ+15]) where small trajectories are removed from the final solution to reduce false alarms. The following paragraphs provide an analysis of the experimental results.

	Method	MOTA	MOTP	GT	MT	ML	ID	Rec	Prec
S1L1-2	Pirsiavash *et al.* [PRF11][a]	45.4	66.8	36	9	14	38	47.1	99.5
	Berclaz *et al.* [BFTF11][a]	51.5	64.8	36	16	14	4	55.5	93.6
	Wen *et al.* [WLY+14]	57.1	54.8	36	18	8	4	58.6	97.8
	Milan *et al.* [MRS14]	57.9	59.7	36	19	11	21	64.5	91.8
	Rezatofighi *et al.* [HRMZ+15]	70.0	64.8	36	21	5	10	74.5	94.7
	MCF-PHD tracker[a]	**65.6**	**65.8**	**36**	**21**	**8**	**6**	**68.2**	**96.7**
S1L2-1	Berclaz *et al.* [BFTF11][a]	19.5	60.6	43	4	29	7	21.4	92.6
	Milan *et al.* [MRS14][a]	30.8	49.0	43	7	20	61	38.5	86.4
	Rezatofighi *et al.* [HRMZ+15]	32.8	59.8	43	9	20	52	40.3	86.8
	MCF-PHD tracker[a]	**29.1**	**42.1**	**43**	**9**	**24**	**18**	**32.1**	**92.8**
S2L2	Pirsiavash *et al.* [PRF11][a]	45.0	64.1	74	7	17	137	49.0	95.4
	Berclaz *et al.* [BFTF11][a]	24.2	60.9	74	7	40	22	26.8	92.1
	Wen *et al.* [WLY+14]	62.1	52.7	74	27	3	125	71.2	90.3
	Milan *et al.* [MRS14]	56.9	59.4	74	28	12	99	65.5	89.8
	Rezatofighi *et al.* [HRMZ+15]	58.3	59.3	74	22	6	103	70.5	86.6
	MCF-PHD tracker[a]	**56.2**	**61.2**	**74**	**20**	**14**	**61**	**60.2**	**94.9**
S2L3	Pirsiavash *et al.* [PRF11][a]	43.0	63.0	44	5	18	52	46.0	97.0
	Berclaz *et al.* [BFTF11][a]	28.8	61.8	44	5	31	7	30.4	95.7
	Wen *et al.* [WLY+14]	55.3	53.2	44	12	9	36	61.0	93.0
	Milan *et al.* [MRS14]	45.4	64.6	44	9	18	38	51.8	90.9
	Rezatofighi *et al.* [HRMZ+15]	53.9	61.6	44	15	17	20	59.5	92.3
	MCF-PHD tracker[a]	**45.0**	**66.9**	**44**	**12**	**22**	**10**	**47.0**	**96.5**
S2L1	Breitenstein *et al.*[BRL+11]	79.7	56.3	-	-	-	-	-	-
	Bae *et al.*[BY14]	83.0	69.6	23	23	0	4	-	-
	Milan *et al.* [MRS14]	91.6	80.2	23	21	1	11	92.4	98.4
	MCF-PHD tracker[a]	**88.9**	**79.3**	**23**	**21**	**0**	**7**	**94.7**	**94.4**

Table 4.3: Evaluation on PETS 2009 [FS09]: MT = Mostly Tracked, ML = Mostly Lost, ID = Number of ID switches, Rec = Recall, Prec = Precision. Numbers of previous methods on sequences S1L1-2–S2L3 are taken from [HRMZ+15]. The best performing method is shown in red, the first runner up in blue. Min-cost flow tracking formulations are marked by [a].

Rank	Mean MOTA (Method)		Rank	Sum ID (Method)	
1	60.73	Rezatofighi *et al.* [HRMZ+15]	1	33	Berclaz *et al.* [BFTF11]
2	58.17	Wen *et al.* [WLY+14]	**2**	77	**MCF-PHD tracker**
3	55.60	**MCF-PHD tracker**	3	133	Rezatofighi *et al.* [HRMZ+15]
4	53.40	Milan *et al.* [MRS14]	4	158	Milan *et al.* [MRS14]
5	40.80	Pirsiavash *et al.* [PRF11]	5	165	Wen *et al.* [WLY+14]
6	34.83	Berclaz *et al.* [BFTF11]	6	227	Pirsiavash *et al.* [PRF11]

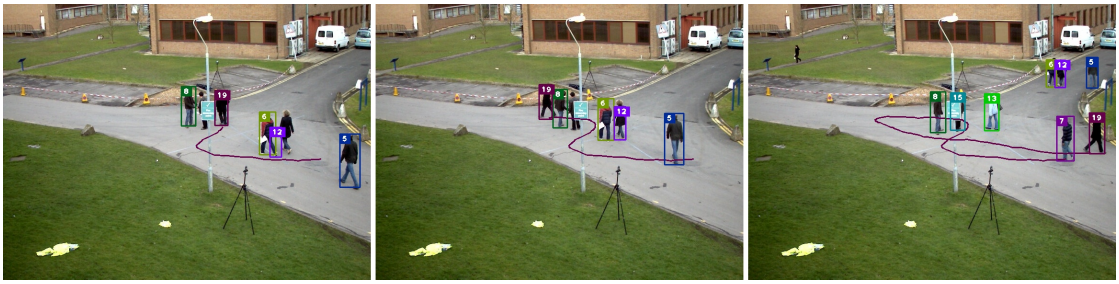
Table 4.4: Ranking in terms of mean MOTA and sum of ID switches over sequences S1L1-2, S2L2, S2L3 of PETS 2009 [FS09]. A method that successfully filters false alarms and keeps consistent object trajectories should be ranked high in both categories. S1L2-1 and S2L1 are not considered in this ranking because not all methods have reported numbers on these sequences.

(a) MCF-PHD tracker (motion)

(b) MCF-PHD tracker (appearance)

Figure 4.11: Tracking output on dense PETS 2009 [FS09] test sequences for (a) MCF-PHD tracker (motion) and (b) MCF-PHD tracker (appearance). Top to bottom: S1L1-2, S1L2-1, S2L2, S2L3.

(a)



(b)

Figure 4.12: This visualization shows how a track is successfully maintained through a complex tracking scenario in sequence S2L2 (a) and successful tracking through occlusion on S2L1 (b).

Comparison against State of the Art The state of the art in Table 4.3 contains two methods based on a min-cost flow problem formulation: Pirsiavash *et al.* [PRF11] refers to an efficient greedy algorithm with linear run-time and Berclaz *et al.* [BFTF11] refers to an exact k-shortest path search that is also used in the MCF-PHD tracker to recover trajectories. The difference between their formulation and the MCF-PHD tracker is due to the cost terms in the flow network. In particular, whereas the MCF-PHD tracker utilizes a constant velocity motion model, the other two methods are restricted to transition costs based on the distance between detections. In Wen *et al.* [WLY+14], short-term track fragments (tracklets) are linked into globally consistent trajectories using constant velocity motion and color-histogram appearance affinity metrics. The method of Milan *et al.* [MRS14] is a Markov random field formulation that encodes smooth trajectories as well as detection-level and trajectory-level exclusion terms to penalize overlapping trajectories. Finally, Rezatofighi *et al.* [HRMZ+15] is an efficient implementation of JPDA in combination with a heuristic track management scheme.

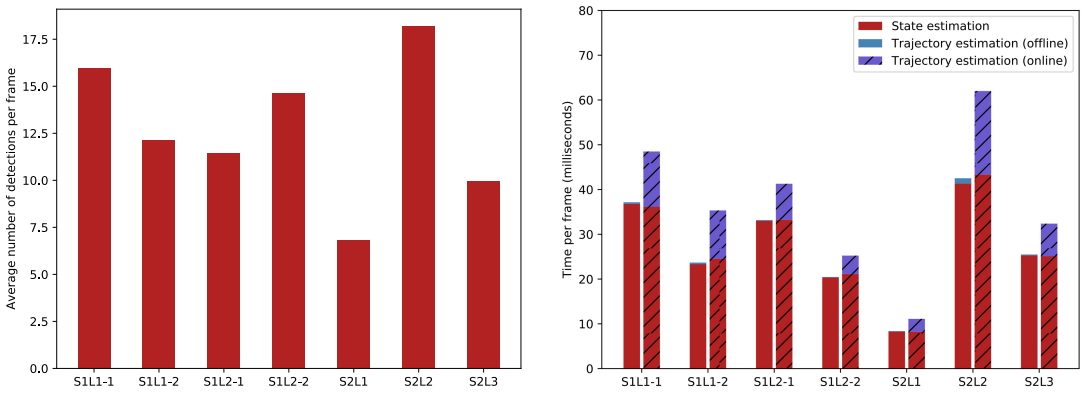
Overall, the MCF-PHD tracker performs well within the state of the art, achieving several first or first runner up positions on individual metrics. Note that these metrics are highly correlated. For example, the method of Berclaz *et al.* [BFTF11] produces very few ID switches on all sequences, but receives very low tracking

accuracy in terms of MOTA. At the other end, the method of Wen *et al.* [WLY+14] produces consistently high MOTA scores, but a high number of ID switches compared to other methods. From these examples it can be seen that trading off the trackers ability to produce high precision and recall often comes at the cost of an increased number of ID switches. This fact is not well captured by the MOTA metric that weights precision, recall, and ID switches equally in creating a summary descriptor. For a more detailed analysis, Table 4.4 ranks the tracking performance in terms of mean MOTA and sum of ID switches. A well performing method, both in terms of filtering false alarms and keeping consistent trajectories, should be ranked high in both categories. In this regard, the JPDA of Rezatofighi *et al.* [HRMZ+15] produces the best results (rank 1 on MOTA and rank 3 on ID switches). The performance of the MCF-PHD tracker is well summarized by a similar performance to Milan *et al.* [MRS14] in terms of MOTA at consistently lower number of ID switches. In particular, the method produces the lowest number of ID switches compared to all other methods that achieve MOTA above 50. The MCF-PHD tracker also performs significantly better than other min-cost flow tracking formulations. The gain in MOTA is between 2.0 and 14.1 points. This is a relative improvement of up to 27%. Therefore, object motion models greatly improve the performance on this dataset. Tracking output on dense sequences is shown in Figure 4.11 for visual inspection. Additionally, Figure 4.12 shows an example where the MCF-PHD tracker successfully maintains an object identity through complex tracking scenarios.

Influence of Batch Optimization and Appearance Features The experiments have been repeated with and without integration of appearance information and batch optimization in order to assess their influence on overall tracking performance. In offline mode, the entire sequence is processed in one batch. In online mode, a fixed-length history of 50 frames is optimized at each time step. Table 4.5 summarizes the results. The best results are obtained in offline mode. The gain in MOTA is 7% with appearance information and 9% without appearance information. The number of ID switches reduces by 16% and 17%, respectively. Whereas the number of ID switches and fragments can be consistently reduced by integration of appearance information, the MOTA decreases by 1.2 points (2%) when compared to using motion only. This can be attributed to two facts: First, the evaluation in Chapter 3 has revealed that due to frequent partial occlusions the feature representation does not work as well on PETS 2009 as on other datasets. Second, the detections on sequence S2L1 contain a number of false positives that continuously reoccur at static scene geometry. These false alarms are picked up more frequently with integrated appearance information because they receive a lower cost due to very similar appearance. If sequence S2L1 is taken out of the

Method	MOTA	MOTP	GT	MT	ML	ID	FM	Rec	Prec
MCF-PHD (appearance, offline)	56.9	65.8	220	83	68	102	153	60.6	95.0
MCF-PHD (motion, offline)	57.7	63.7	220	81	70	116	183	61.1	95.6
MCF-PHD (appearance, online)	53.0	59.2	220	73	68	122	174	56.6	94.9
MCF-PHD (motion, online)	53.1	55.8	220	74	76	140	188	57.1	94.5

Table 4.5: Summary of results on PETS 2009 [FS09] in different operational modes. In online mode, a fixed-length history of 50 frames is optimized at each time step. MT = Mostly Tracked, ML = Mostly Lost, ID = Number of ID switches, FM = Fragments, Rec = Recall, Prec = Precision. The best performing method is shown in red, the first runner up in blue.



(a) Object density **(b)** Run-time

Figure 4.13: This plot gives an overview of average algorithm run-time on individual sequences of PETS 2009 [FS09]. Plot (a) shows the average number of detections per sequence. Plot (b) shows the run-time of the GM-PHD filter and min-cost flow trajectory estimation. The reported numbers in offline mode are established by dividing the total run-time by the number of frames. The figures are exclusive of feature extraction. The CNN of Chapter 3 requires additional 30 milliseconds processing time for 32 bounding boxes.

evaluation, appearance and motion perform approximately equally well in terms of MOTA.

Figure 4.13 gives an overview of algorithm run-time on individual sequences. The reported numbers have been obtained on a consumer notebook with Intel i7-7700HQ CPU running at 2.8 GHz. The GM-PHD filter is implemented in Python, the min-cost flow solver in C++. On all sequences, most computation time is spent on state estimation. In particular, in online mode the cost of running the min-cost flow solver becomes negligible compared to state estimation. The tracker runs at

Method	MOTA	MOTP	MT	ML	ID	FM	FP	FN
DBN [KRH15]	51.1	62.2	28.7%	17.9%	380	418	2077	5746
GPDBN [KRH17]	49.8	61.0	25.7%	17.2%	311	386	1813	6300
MCF-PHD tracker	**39.9**	**53.6**	**25.7%**	**16.8%**	**363**	**529**	**3029**	**6700**
LPSFM [LTPMR11]	35.9	54.0	13.8%	21.6%	520	601	2031	8206
LP3D [LTMR+15]	35.9	55.8	20.9%	16.4%	580	659	3588	6593
SVT [WLC+17]	34.2	53.8	11.2%	25.4%	532	611	3057	7454
AMIR3D [SAS17]	25.0	55.6	3.0%	27.6%	1462	1647	2038	9084
KalmanSFM [PESVG09]	25.0	53.6	6.7%	14.6%	1838	1686	3161	7599

Table 4.6: Results on 3D MOT 2015 [LTMR+15] in offline mode (the entire sequence is optimized in a single batch). MT = Mostly Tracked, ML = Mostly Lost, ID = Number of ID switches, FM = Fragments, FP = False positives, FN = False negatives. The best performing method is shown in red, the first runner up in blue. Accessed on 11/18/2017.

over 60 frames per second on sparsely crowded scene S2L1. On the most crowded scene S2L2, the tracker runs at roughly 15 frames per second in offline mode and 22 frames per second in online mode.

4.9.4 MOT Challenge and RGB-D People

This section discusses results obtained on a number of different datasets that the MCF-PHD tracker has additionally been evaluated on. The focus of this evaluation is to provide a broad comparison against the state of the art on public benchmarks, and to study the performance when applied to different sensor modalities.

3D MOT 2015 The MCF-PHD tracker is further evaluated on 3D MOT 2015 [LTMR+15], a public benchmark that contains sequences PETS 2009 S2L2 and AVG-TownCentre in the test set. Despite the overlap with the previous experiment, submission to 3D MOT 2015 is interesting because evaluation is carried out against unpublished ground truth on a test server. Different results may be obtained on PETS 2009 S2L2 compared to the previous experiment due to utilization of different detections and possibly different ground truth. Sequence AVG-TownCentre has not been evaluated before. For the sake of generality, the MCF-PHD tracker is applied with the same parametrization as in the PETS 2009 experiment.

Results of the 3D MOT 2015 submission are summarized in Table 4.6. The MCF-PHD tracker comes in third of all published methods when ranked according to MOTA and ID switches. The top two methods utilize prior scene knowledge to suppress false alarms outside and reduce the number of fragments inside the region where people move. While such extraneous scene information could be integrated into the MCF-PHD tracker by adaptation of clutter and birth intensities, this path

Method	MOTA	MOTP	ID	FP	FN
Luber *et al.* [LSA11]	78.0	-	32	4.5%	16.8%
Munaro *et al.*[MBM12]	71.8	73.7	19	7.7%	20.0%
MCF-PHD tracker (online)	**74.1**	**74.1**	**18**	**2.5%**	**23.1%**
MCF-PHD tracker (offline)	**75.3**	**74.1**	**15**	**3.7%**	**20.6%**

Table 4.7: Results on RGB-D People dataset [LSA11] in different operational modes. In online mode a fixed-length history of 50 frames is optimized at each time step. ID = Number of ID switches, FP = False positives, FN = False negatives. The best performing method is shown in red, the first runner up in blue.

has not been followed here in order to draw a fair comparison to methods that work in a general setup. Out of the remaining, more general methods, the MCF-PHD tracker ranks highest according to most metrics. Compared to the baseline LP3D formulation, a min-cost flow linear programming solution on 3D positions in world coordinates, MOTA increases from 35.9 to 39.9 points and ID switches go down from 580 to 363. This is a relative improvement of 11% and 37%.

RGB-D People The second dataset contains a sequence of over 3000 RGB-D frames captured by three vertically mounted Microsoft Kinect sensors [LSA11]. The sensor configuration is placed in a busy university hall at approximately 1 m height. The provided ground truth contains several identities that are not annotated even though visible. To avoid a penalty during evaluation, the ground truth is interpolated and these locations are flagged to not be counted as false positive. Further, the corresponding dataset authors do not provide a set of standard detections. Therefore, point measurements on the ground plane are generated by a people detector that is available in current releases of the Point Cloud Library (PCL) [MBM12, RC11]. The parameters of the MCF-PHD tracker are adopted from the PETS 2009 experiment. The measurement noise uncertainty is reduced to standard deviation 0.1 m to account for higher detector localization accuracy. Remaining parameters are left unchanged.

The results are summarized in Table 4.7. The method of Luber *et al.* [LSA11] is an implementation of MHT that integrates color and depth information into the association likelihood by training online boosting classifiers for each track hypothesis. The lower number of false negatives compared to the other two methods can in part be explained by better detector output. Around 10% of people in the dataset appear on an elevated stairway where the PCL people detector provides no output. Consequently, these people are invisible to the tracker. The second method [MBM12] uses Kalman filtering and Global Nearest Neighbor (GNN) data association via Hungarian algorithm. Color features are integrated into the tracker

in similar fashion to Luber *et al.* [LSA11] by training online boosting classifiers on individual track hypotheses. Since this method also utilizes the PCL people detector, the numbers are directly comparable to the MCF-PHD tracker. In comparison, the MCF-PHD tracker has slightly better performance in terms of MOTA as well as ID switches.

2D MOT 2015 The 2D MOT 2015 [LTMR$^+$15] benchmark is a comprehensive collection of existing and new video sequences. The benchmark is particularly well suited to establish a performance evaluation over a broad range of tracking applications because it covers a large variety dataset characteristics. There exist sequences with low camera position (frontal view) as well as high camera position (surveillance setup), the frame rate ranges from 7 to 30 frames per second and image resolution from 640×480 to 1920×1020. The scene density varies from 5 to 22 objects per image. The camera is moving in some scenes and stationary in others. Weather conditions range from sunny to cloudy, yielding very different lighting conditions. In total, the benchmark contains 16 minutes and 30 seconds video footage (11 286 frames), 1221 ground truth objects and over 100 000 detections. Thus, it is hard to over-tune the tracker to a particular scenario. In addition, the ground truth for test sequences is unpublished and evaluation must be performed on a test server.

Note that this benchmark is particularly challenging for the MCF-PHD tracker because camera calibration is unavailable. The MCF-PHD tracker is built upon a recursive filtering framework which relies on informative motion models that are easier to describe in a world coordinate frame than in image coordinates. In particular, ego-motion can lead to substantial displacements in image coordinates that must be accommodated by high noise parameters, making these models less informative. This can be partially compensated by tuning the parameters to individual sequences or at least groups of sequences with similar characteristics, but this would provide little insight on the general performance of the tracker under varying situations. Therefore, a single parameter set has been chosen for all sequences within the benchmark. More specifically, the MCF-PHD tracker is set up to replicate the parametrization of a recent re-visitation of MHT [KLCR15] without performing additional parameter tuning. This also includes parametrization of the preprocessing and postprocessing stage where detections are filtered based on detector confidence and non-maximum suppression. While this evaluation procedure might lead to lower performance, it eliminates extraneous effects that can hinder a direct comparison.

The parameters of the MCF-PHD tracker are chosen as follows. The state vector $\boldsymbol{x} = (x, y, \dot{x}, \dot{y})^{\mathrm{T}}$ contains the bounding box center position $(x, y)^{\mathrm{T}}$ and velocity $(\dot{x}, \dot{y})^{\mathrm{T}}$ in pixel coordinates. The position is assumed to move accord-

Method	MOTA	MT	ML	ID	FM	FP	FN	Avg Rank	Hz
APRCNN_Pub [CAS+17]	38.5	8.7%	37.4%	586	1263	4005	33203	15.0	6.7
AMIR15 [SAS17]	37.6	15.8%	26.8%	1024	2024	7933	29397	19.8	1.9
NOMT [Cho15]	33.7	12.2%	44.0%	442	823	7762	32547	18.2	11.5
MHT-DAM [KLCR15]	32.4	16.0%	43.8%	435	826	9064	32060	20.8	0.7
MCF-PHD tracker[a]	**29.9**	**11.9%**	**44.0%**	**656**	**989**	**8892**	**33529**	**24.4**	**12.2**
SiameseCNN [LTCFS16][a]	29.0	8.5%	48.4%	639	1316	5160	37798	30.8	52.8
ELP [MDRM15][a]	25.0	7.5%	43.8%	1396	1804	7345	37344	36.7	5.7
JPDA_m [HRMZ+15]	23.8	5.0%	58.1%	365	869	6373	40084	31.3	32.6
EAMTTpub [SMPC16]	22.3	5.4%	52.7%	833	1485	7924	38982	38.3	12.2
LP2D [LTMR+15][a]	19.8	6.7%	41.2%	1649	1712	11580	36045	39.6	112.1
CEM [MRS14]	19.3	8.5%	46.5%	813	1023	14180	34591	38.5	1.1
GMPHD [SJ16]	18.5	3.9%	55.3%	459	1266	7864	41766	38.6	19.8
ALExTRAC [BORU16]	17.0	3.9%	52.4%	1859	1872	9233	39933	47.5	3.7
DP_NMS [PRF11][a]	14.5	6.0%	40.8%	4537	3090	13171	34814	40.7	444.8

Table 4.8: Results on 2D MOT 2015 [LTMR+15] in offline mode (the entire sequence is optimized in a single batch). Methods based on a min-cost flow formulation are marked by[a]. MT = Mostly Tracked, ML = Mostly Lost, ID = Number of ID switches, FM = Fragments, FP = False positives, FN = False negatives, Avg Rank = Average rank over all metrics, Hz = Reported tracker speed in frames per seconds. The full table can be found in Appendix E. Note that this is an ongoing benchmark; the ranking can change in future (accessed on 11/18/2017).

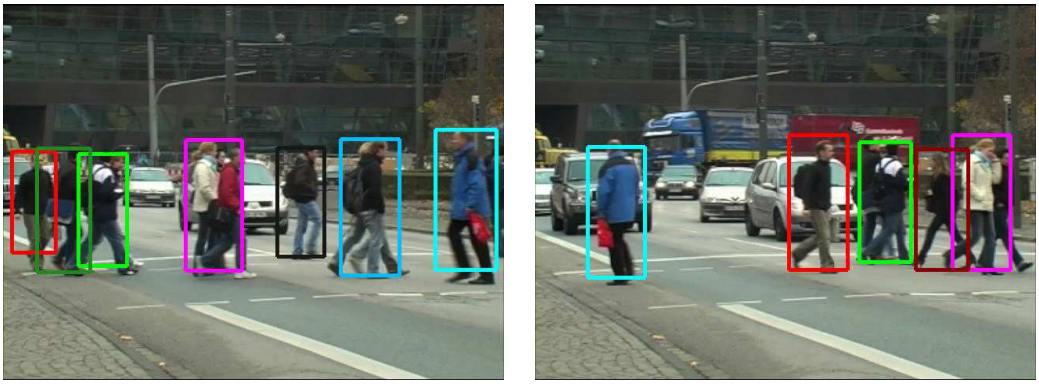
ing to a constant velocity motion model. The state transition covariance is set to $\mathrm{diag}(\sigma^2_{\mathrm{pos}}, \sigma^2_{\mathrm{pos}}, \sigma^2_{\mathrm{vel}}, \sigma^2_{\mathrm{vel}})$ where, for a predecessor detection of width w_{p} that has been used to perform the most recent measurement update, $\sigma^2_{\mathrm{pos}} = (\frac{1}{10} \cdot w_{\mathrm{p}})^2$ and $\sigma^2_{\mathrm{vel}} = \frac{1}{80} \cdot w_{\mathrm{p}}$. The bounding box center position is taken as direct observation of the object state and the measurement model adds isotropic noise with standard deviation $\frac{1}{10} \cdot w_{\mathrm{p}}$. To prevent associations between bounding boxes that have similar center position but different size, the measurement likelihood is set to 0 if the size differs by more than 40%. Survival and detection probabilities are set to $p_{\mathrm{S}} = 1$ and $p_{\mathrm{D}} = 0.9$. The clutter intensity is set to $c_k(\boldsymbol{y}) = 1 \cdot V^{-1}$ and the PHD likelihood of birth components to $\tau_k^{(\mathrm{u})}(\boldsymbol{y}) = 1 \times 10^{-3} \cdot V^{-1}$ where V is the number of pixels in the image. Detector confidence scores are not integrated into the GM-PHD recursion. Instead, detections are thresholded at a pre-defined minimum confidence value that is chosen on a per-sequence basis. The values are taken from the provided implementation of Kim *et al.* [KLCR15]. Components with weight smaller than $w_{\min} = 1 \times 10^{-10}$ are pruned from the intensity and the merge threshold is set to $U = 3$. The appearance model is parametrized by $\kappa = 10$ and $s_{\min} = 0.9$.

An excerpt of the benchmark results is shown in Table 4.8. This table contains only the top two performing methods together with a selection of submissions that are particularly interesting for comparison. The full table can be found in Appendix E. Of all published methods that have been submitted to 2D MOT 2015, the MCF-PHD tracker ranks 12 out of 40 with respect to MOTA, placing it at the top 30% and only two places below MHT-DAM [KLCR15], an implementation of MHT with online appearance modeling based on CNN features. The top performing methods put strong emphasize on visual aspects of the tracking problem. APRCNN_Pub [CAS+17] and AMIR15 [SAS17] focus on learning of rich features using CNNs. The main innovation behind NOMT [Cho15] is the development of a well-working optical flow descriptor that aids data association. At the lower end of the table are mostly methods that utilize motion alone. For example, JPDA_m [HRMZ+15] and CEM [MRS14] perform well on PETS 2009 where tracking is performed in 3D world coordinates, but the performance does not generalize to the image-space tracking scenario. By using appearance only, ALExTRAC [BORU16] is a peculiarity under all submissions.

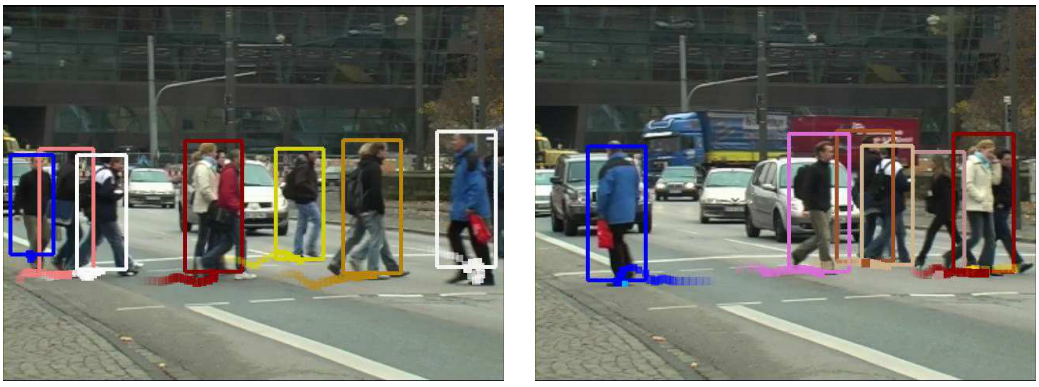
MHT-DAM [KLCR15] and the MCF-PHD tracker are both very general methods that have been evaluated using a mostly identical parameter set. As discussed in Section 4.5, the key strength of MHT over MCF-PHD is consideration of long-term dependencies. Whereas the MCF-PHD tracker utilizes a pairwise appearance model, MHT-DAM trains separate classifiers for each track hypothesis. This increases performance according to almost all metrics. In particular, MHT-DAM has an 8% higher MOTA and 34% fewer ID switches. A scenario where the advantage of capturing long-term dependencies becomes particularly eminent is shown in Figure 4.14. The number of false positives and false negatives on the other hand is relatively similar.

Several entries in the table refer to methods that are formulated in a min-cost flow framework. Out of these methods, the MCF-PHD tracker ranks highest according to most metrics. The baseline method LP2D [LTMR+15] is formulated on the distance between bounding box positions. DP_NMS [PRF11] is a fast dynamic programming approximation of the optimal solution that has also been evaluated on PETS 2009. Both methods fall behind in tracking performance, likely due to lack of motion and appearance information. ELP [MDRM15] is a min-cost flow network defined on tracklets. SiameseCNN [LTCFS16] utilizes a CNN that has been trained on appearance and optical flow features. The method produces similar MOTA and ID switches, but more fragments compared to the MCF-PHD tracker.

The table also contains two methods that use a GM-PHD filter. The method EAMTTpub [SMPC16] filters false alarms and fuse multiple detector outputs at a low level of a hierarchical association strategy using a GM-PHD filter. GM-



(a) MHT-DAM [KLCR15]



(b) MCF-PHD tracker

Figure 4.14: This figure shows tracking output of MHT-DAM [KLCR15] and MCF-PHD on TUD-Crossing. The video shows frequent occlusions from a low camera position. Bounding boxes typically have high overlap a the point where people cross their path. MHT-DAM deals with this situation very well (observe the person in a blue jacket walking right to left). For the MCF-PHD tracker this scenario is much more challenging.

PHD [SJ16] is a mostly straight-forward application of the GM-PHD filter with the widely applied track management scheme of Panta *et al.* [PCV09]. Both methods fall behind in tracking performance on this dataset.

Chapter 5

Operator Following

This chapter presents a practical application of a multi-object Bayes filter to the problem of jointly detecting and tracking a single person in crowded environments. The approach has been published in [WMP17]. Section 5.1 introduces the specific problem that motivated the development of such a system. Section 5.2 gives an overview of related work. The joint operator detection and tracking problem is formulated in Section 5.3. The chapter concludes with an experimental evaluation in Section 5.4.

5.1 Introduction

The motivation to jointly detect and track a single operator through crowded environments stems from a regular participation in RoboCup@Home challenges where fundamental as well as more advanced capabilities of service robots are tested in a benchmark environment. These benchmarks are typically set up in a way that a specific person is introduced to the robot as an operator. From that point onwards, this operator must be continuously identified and distinguished from remaining people, for example to take pick-and-deliver commands. Figure 5.1a shows service robot *Lisa* of the Active Vision Group at the University of Koblenz-Landau and an operator following scenario during the 2012 RoboCup@Home world cup in Mexico City. The operator following benchmark takes place in an open space that is crowded by spectators and press to increase the complexity of the problem. In addition, several intended distractions are incorporated to test re-detection capabilities after occlusions. For example, Figure 5.1b shows a person crossing onto the path in between robot and operator. The person will persist at this location for several seconds to test the robot's behavior when the field of view is blocked. In other benchmarks where operator identification is required, the operator may leave and

(a) **(b)**

Figure 5.1: (a) shows robot *Lisa* of the Active Vision Group at the University of Koblenz-Landau, (b) shows an operator following scenario during the 2012 RoboCup@Home world cup in Mexico City where a person blocks the path between robot and operator.

re-enter the scene at any time. Therefore, this is not just a regular tracking task, but rather a joint detection and tracking task.

The problem generalizes beyond the specific application in RoboCup@Home that is targeted here. In many tasks that involve human-robot interaction, knowledge about a specific person is fundamental to reliably carry out interaction. Consider, for example, an airport service robot that serves as help desk and takes customers to their designated gate. This robot must navigate through a dynamic, crowded environment and at the same time be aware of the customer that it is taking care of. While tracking a single individual is fundamentally easier than tracking *many* people at the same time, the problem scales with the complexity of the environment. The same challenges that make multiple object tracking difficult also affect the performance of a single-object tracker: data-association ambiguities, long-term occlusions, and social interactions. Therefore, while detecting and tracking an operator is relatively easy in controlled laboratory environments, deployment in real-world scenarios still poses a scientific challenge.

A key characteristic of the RoboCup@Home scenario is that the operator can leave and re-enter the scene at any time. Therefore, at each time the system must be capable to determine whether the operator is currently present in the scene and, if yes, at which location (c.f. Figure 5.2). In the end, a system to be successfully deployed under complex circumstances requires careful combination of a tracking component that resolves local ambiguities and a re-identification component that recovers from tracking failures and long-term occlusions. Due to these

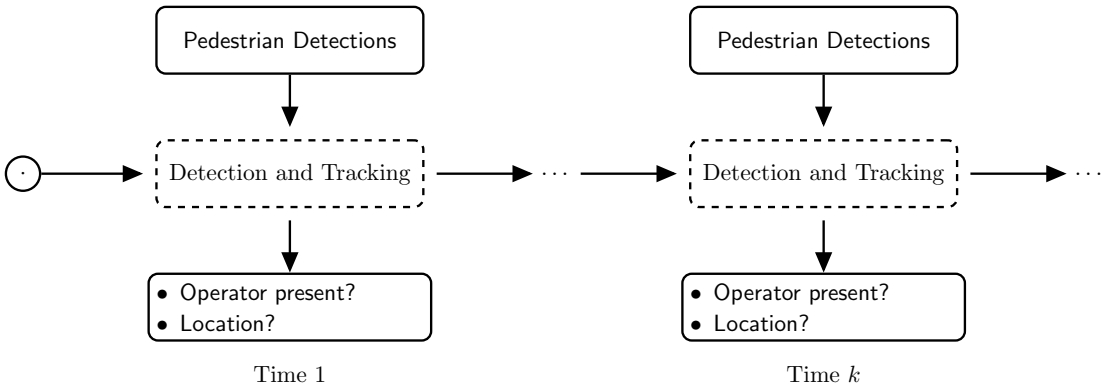


Figure 5.2: This diagram provides a schematic overview of the joint detection and tracking problem. Initially, the operator is introduced to the robot at a known starting position. From that point onwards, the system must determine whether the operator is currently present and, if yes, at which location.

specific requirements, most work on people tracking is not directly applicable to this application domain. In particular, most approaches to multiple object tracking lack a re-identification component, such that they can only deal with limited-time occlusions. On the other hand, RFS theory provides a natural framework to jointly treat the detection and tracking problem. The state variable at interest here, the operator, is a set that is either empty or contains exactly one element.

5.2 Related Work

Few integrated systems for operator following have been presented as a whole. Leigh *et al.* [LPOZ15] present a laser based people tracking approach. They detect people using extracted clusters of laser measurements at leg height. The clusters are then tracked using a combination of Kalman filter and GNN matching. Gockley *et al.* [GFS07] focus on the social aspect of people following in human-computer interaction. Their tracker is a simple particle filter on laser detections.

More frequently, operator following is solved as part of a multi-object tracking system. For example, Topp and Christensen [TC05] design a multiple object tracker for following and passing persons. Detections are laser-based and the tracker uses a sample-based JPDA filter. Munaro *et al.* [MBM12] design a multiple object tracking system with GNN data association based on Kinect RGB-D detections. They first find candidate clusters in the 3D point cloud. Then, they classify candidates with a support vector machine that is trained on histograms of oriented gradients. They apply the system to follow people in domestic environments.

In recent revisitations, classical online multi-object tracking frameworks have shown competitive performance when compared to more recent batch algorithms. Notably, Kim *et al.* [KLCR15] show that the classical MHT algorithm [Rei79] can achieve state-of-the-art results when combined with a visual classifier. Rezatofighi *et al.* [HRMZ+15] have investigated an efficient solution to the JPDA that, combined with a heuristic track handling scheme, achieves competitive results in dense tracking scenarios with substantial occlusions, false alarms, and missed detections. These methods are, however, only capable of short-term tracking. On the other side are tracking approaches based on FISST [Mah07b], a specialized mathematical theory for set-valued random variables. Within this theoretical framework, a number of recursive state estimators have been proposed. For example, the PHD filter [Mah03] is a moment approximation of the set-valued multi-object Bayes filter that is computationally efficient and has successfully been applied to various tracking tasks, e.g., [PVS+04, JB09, KLW+10, MĆP15]. In the case of a single target that is tracked through clutter, the Bernoulli filter [RVVF13] represents an exact solution to the Bayes recursion under consideration of data association uncertainty.

The problem formulation closest to the specific application scenario that is considered here can be found in a visual object tracking context. The problem is posed in an incremental learning framework that recovers from occlusions and tracking failures. For example, Hare *et al.* [HGS+16] apply a budgeted kernel support vector machine on a combination of Haar-like, histogram, and raw pixel features. They also propose to integrate this classifier into a structured prediction task to better cope with decreasing localization accuracy that is due to continuous model adaption. Nebehay *et al.* [NP14] explicitly address tracking articulated and deformable objects. They find matching key-points between successive frames based on descriptor similarity. Then, they group correspondences based on observed motion. While the work in this thesis is mostly in line with these methods, it tackles different aspects of the problem: Since RGB-D data is available, the tracking system does not suffer from drift and allows for more accurate integration of object motion models. Further, since the goal is to track pedestrians specifically, a specialized representation space can be employed that is optimized for this purpose.

5.3 Joint Detection and Tracking

A key characteristic of the detection and tracking problem is that the operator may be absent from the scene at a particular point in time. Therefore, the system must decide whether the operator is currently present and, if yes, at which location. In Section 2.1.3 the Bernoulli RFS has been introduced as a set-valued random variable that is either empty or contains exactly one element. This class of RFSs

provides a natural representation for the joint detection and tracking problem because the multi object density (c.f. Equation 2.15)

$$\pi(X) = \begin{cases} 1 - q & \text{if } X = \emptyset, \\ q \cdot p(\boldsymbol{x}) & \text{if } X = \{\boldsymbol{x}\}, \\ 0 & \text{otherwise,} \end{cases} \tag{5.1}$$

of a Bernoulli RFS X captures both the uncertainty about the probability of existence and the location of the operator. This set-valued problem formulation has the following benefits over a conventional vector-valued formulation: (1) The representation can deal with situations where the operator disappears and later re-enters the sensor field without any additional logic built on top of the tracking framework. (2) For this particular class of RFSs, a tractable implementation of the optimal multi-object Bayes filter exists. The Bernoulli filter [RVVF13] propagates the multi-object density of a Bernoulli state RFS under consideration of data association ambiguities that arise from additional clutter returns in the sensor field. By following a RFS methodology, hard decision on data association are avoided and left for the tracker to resolve. This is particularly important because a wrong decision on data association can be fatal for the operator following task if decisions from the past cannot be corrected in the future. Then, the tracker focuses on the wrong person from the point in time onwards where the error has been made. Derived from multi-object calculus, the Bernoulli filter considers multiple association hypotheses implicitly and, therefore, potentially recovers from errors that have been made in the past.

5.3.1 System Representation

The standard multi-object system representation has been presented in Section 2.2 as a formal model for the evolution of a multi-object state and the process by which measurements are generated. In this chapter, the representation must be adapted to account for the particular problem characteristics at hand. In the joint operator detection and tracking problem, the multi-object state at time k is the union of two disjoint sets

$$X_k = X_k^{(0)} \cup X_k^{(1)} \tag{5.2}$$

in which non-operators in $X_k^{(0)}$ are explicitly distinguished from the operator $X_k^{(1)}$. This modeling attributes to the fact that the operator moves in a populated environment. In particular, the sensor responds not only to detections originating from the operator, but to any person in the sensor field. Therefore, the set of

measurements at time k is a union of clutter returns, measurements that originate from non-operators, and up to one measurement that originates from the operator:

$$Z_k = K_k \cup \left[\bigcup_{\boldsymbol{x}_k \in X_k^{(0)}} H_k(\boldsymbol{x}) \right] \cup \left[\bigcup_{\boldsymbol{x}_k \in X_k^{(1)}} H_k(\boldsymbol{x}) \right]. \tag{5.3}$$

Following the modeling in Section 2.2, clutter K_k is Poisson with intensity $c_k(\boldsymbol{z})$ and $H_k(\boldsymbol{x})$ is a Bernoulli RFS with parameters $\{p_{\mathrm{D}}(\boldsymbol{x}), p_k(\boldsymbol{z} \mid \boldsymbol{x})\}$. In this application, non-operators are modeled as Poisson RFS with multi-object density (c.f. Equation 2.13)

$$\pi_k^{(0)}(X) = e^{-\int v_k^{(0)}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}} \prod_{\boldsymbol{x} \in X} v_k^{(0)}(\boldsymbol{x}) \tag{5.4}$$

such that the statistics of non-operators are completely characterized by their intensity $v_k^{(0)}(\boldsymbol{x})$. If there is at most one operator present, then $X_k^{(1)}$ is a Bernoulli RFS with multi-object probability density

$$\pi_k^{(1)}(X) = \begin{cases} 1 - q_k^{(1)} & \text{if } X = \emptyset, \\ q_k^{(1)} \cdot p_k^{(1)}(\boldsymbol{x}) & \text{if } X = \{\boldsymbol{x}\}, \\ 0 & \text{otherwise,} \end{cases} \tag{5.5}$$

where $q_k^{(1)}$ is the probability that the operator is currently present and $p_k^{(1)}(\boldsymbol{x})$ is the spatial density. In the following, the statistics of non-operators $X_k^{(0)}$ and operator $X_k^{(1)}$ are estimated by a PHD filter and a Bernoulli filter that run in parallel and interact in a combined update.

For the purpose of a unified system description, let $\boldsymbol{x}_k = (x, y, \dot{x}, \dot{y})^{\mathrm{T}}$ denote a single-object state that contains position and velocity and let $\beta \in \{0, 1\}$ denote a binary class label that takes on 0 for non-operators and 1 for the operator. Then, it is assumed that the operator does not change identity throughout the tracking application:

$$p_{k|k-1}(\boldsymbol{x}, \beta \mid \boldsymbol{x}', \beta') = \begin{cases} p_{k|k-1}(\boldsymbol{x} \mid \boldsymbol{x}') & \text{if } \beta = \beta', \\ 0 & \text{otherwise.} \end{cases} \tag{5.6}$$

In consequence, PHD and Bernoulli filter are non-interacting during the prediction step. The state transition density $p_{k|k-1}(\boldsymbol{x} \mid \boldsymbol{x}')$ follows a constant velocity motion model in this application, both in the PHD filter that is used to track non-operators as well as in the Bernoulli filter that is used to track the operator. A description of the constant velocity motion model is given in Appendix B.1.

Measurements are collected from a mobile robot by application of a standard pedestrian detector, e.g., from leg segmentation of laser scans [AMB07] or from 3D point clusters obtained by a Microsoft Kinect [MBM12]. The sensor is calibrated

against a color camera to perform image-based operator classification. Therefore, each measurement $\boldsymbol{z} = (\boldsymbol{y}, s)^{\mathrm{T}}$ contains a ground plane position $\boldsymbol{y} = (x, y)^{\mathrm{T}}$ and a classification confidence score s that is the distance to the decision surface. Section 5.3.3 outlines how this classifier is trained to discriminate the operator from remaining people. For integration into the detection and tracking module, the measurement likelihood is factorized into a spatial component and an appearance likelihood

$$p_k(\boldsymbol{y}, s \mid \boldsymbol{x}, \beta) = p_k(\boldsymbol{y} \mid \boldsymbol{x}) p(s \mid \beta) \tag{5.7}$$

with appearance taken independent of location. In the remainder of this chapter

$$p_k^{(\beta)}(\boldsymbol{z} \mid \boldsymbol{x}) = p_k(\boldsymbol{y} \mid \boldsymbol{x} \mid p)(s \mid \beta) \tag{5.8}$$

will be used to refer to the measurement likelihood of non-operators ($\beta = 0$) and operator ($\beta = 1$).

With exception of the combined filter update, the joint operator detection and tracking framework can be implemented based on this system description by a straight-forward application of the PHD and Bernoulli filter. Therefore, the remainder of this section will focus on the combined update step. Pseudo code for the PHD filter has been given in Section 2.3.3. An extensive tutorial on the Bernoulli filter, including a sequential Monte-Carlo and Gaussian mixture implementation, can be found in [RVVF13]. For completeness, the Gaussian mixture Bernoulli filter recursion is also provided in Appendix D.

5.3.2 Interacting Bernoulli/PHD Filter Update

The hybrid state space model with a common measurement space in Equation 5.2 and 5.3 has previously been studied by Mahler *et al.* [MVV11] in a PHD filter application where the clutter profile is estimated directly from data. Their derivation leads to a parallel execution of two PHD filters to jointly estimate true targets and clutter dynamics. The derivations in [MVV11] can be summarized as follows. If $v_{k|k-1}^{(\beta)}(\boldsymbol{x})$ denotes the predicted intensity of non-operators ($\beta = 0$) and operator ($\beta = 1$) respectively, then the posterior intensity can be computed by [MVV11]

$$\begin{aligned} v_k^{(\beta)}(\boldsymbol{x}) = {} & [1 - p_{\mathrm{D}}(\boldsymbol{x})]\, v_{k|k-1}^{(\beta)}(\boldsymbol{x}) + \\ & \sum_{\boldsymbol{z} \in Z_k} \frac{p_{\mathrm{D}}(\boldsymbol{x}) p_k^{(\beta)}(\boldsymbol{z} \mid \boldsymbol{x}) v_{k|k-1}^{(\beta)}(\boldsymbol{x})}{c_k(\boldsymbol{z}) + \sum_{i=0}^{1} \int p_{\mathrm{D}}(\boldsymbol{x}) p_k^{(i)}(\boldsymbol{z} \mid \boldsymbol{x}) v_{k|k-1}^{(i)}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}}, \end{aligned} \tag{5.9}$$

where a common probability of detection $p_{\mathrm{D}}(\boldsymbol{x})$ has been assumed. In essence, Equation 5.9 resembles a standard PHD filter update with an additional term in the denominator of measurement-corrected partitions that accounts for the PHD

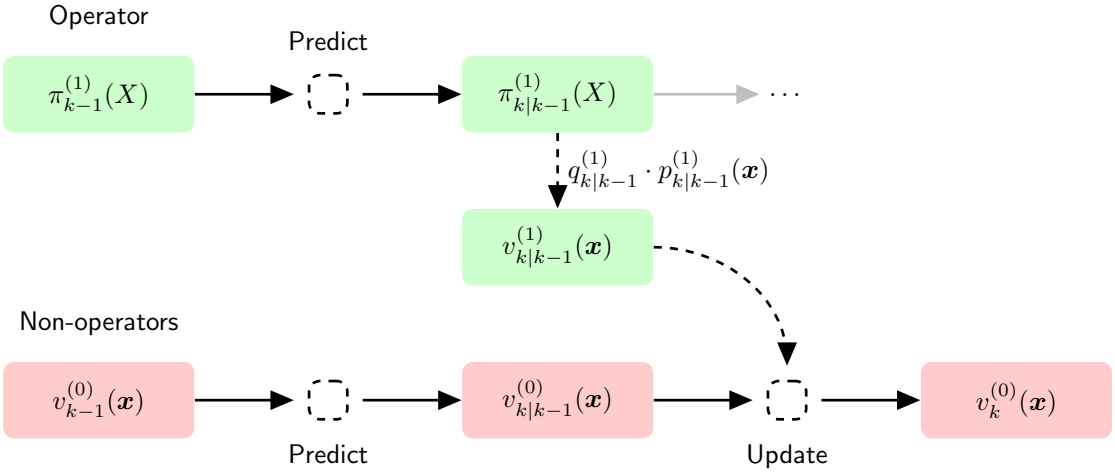


Figure 5.3: Interacting Bernoulli/PHD filter update: Schematic chart of the PHD filter update.

likelihood of $v^{(1-\beta)}_{k|k-1}(\boldsymbol{x})$ (c.f. Equation 2.22). Note that the same result could be established by application of the standard PHD update when each filter is updated according to a clutter set

$$K_k{}^{(\beta)} = K_k \cup \left[\bigcup_{\boldsymbol{x}_k \in X_k^{(1-\beta)}} H_k(\boldsymbol{x}) \right] \tag{5.10}$$

with intensity[1]

$$c_k{}^{(\beta)}(\boldsymbol{z}) = c_k(\boldsymbol{z}) + \int p_{\mathrm{D}}(\boldsymbol{x}) p_k^{(1-\beta)}(\boldsymbol{z} \mid \boldsymbol{x}) v_{k|k-1}{}^{(1-\beta)}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}. \tag{5.11}$$

Therefore, the operator represents itself to non-operators as an additional clutter source and vice versa.

Within the operator detection and tracking application, this framework is used to update the PHD filter and the Bernoulli filter jointly. Figure 5.3 illustrates the PHD filter update. First, the intensity of the operator RFS is constructed from the multi-object probability density by

$$v^{(1)}_{k|k-1}(\boldsymbol{x}) = q^{(1)}_{k|k-1} \cdot p^{(1)}_{k|k-1}(\boldsymbol{x}). \tag{5.12}$$

As noted in Section 2.1.3, this conversion is exact because the multi-object density of a Bernoulli RFS can be converted to and recovered from its intensity without loss

[1] Given two independent Poisson RFS X_1 and X_2 with intensity $v_1(\boldsymbol{x})$ and $v_2(\boldsymbol{x})$, the intensity of the union $X_1 \cup X_2$ is $v_1(\boldsymbol{x}) + v_2(\boldsymbol{x})$ [Mah07b]. In Equation 5.11, the first term is the clutter intensity and the second term is the intensity of measurements generated by objects in $X_k^{(1-\beta)}$.

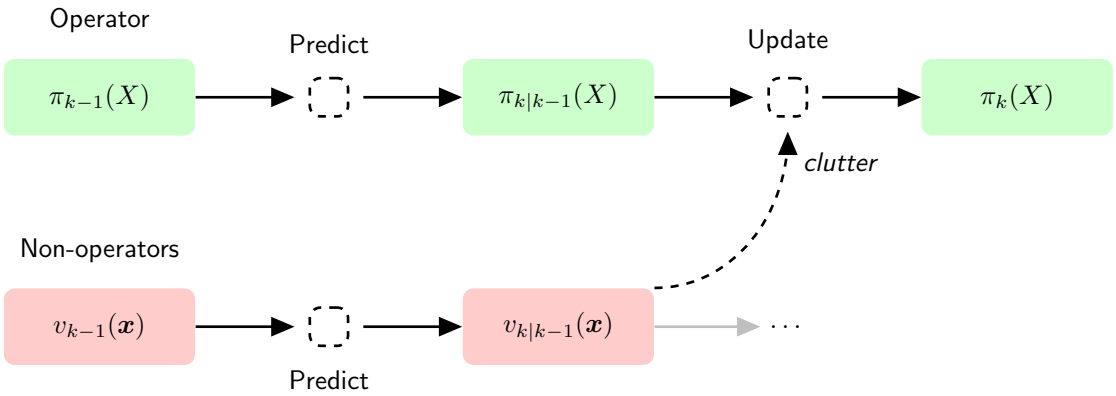


Figure 5.4: Interacting Bernoulli/PHD filter update: Schematic chart of the Bernoulli filter update.

of information. Then, Equation 5.9 is used to compute the posterior intensity of non-operators. In similar vein, non-operators are considered during the Bernoulli filter update by using a clutter intensity that accounts for $v^{(0)}_{k|k-1}(\boldsymbol{x})$ using Equation 5.11 with $\beta = 1$. This is illustrated in Figure 5.4.

5.3.3 Classification and Online Learning

The purpose of the classifier in the joint detection and tracking module is twofold: First, appearance aids the tracking component in case of occlusions when the uncertainty associated with the kinematic state increases. Second, and more importantly, if no extraneous information about where and when the operator enters the scene is available, appearance is necessary to discriminate the operator from any other person entering the scene. In Chapter 3, a CNN has been trained on a large-scale person re-identification dataset to be generally applicable to the task of discriminating people from one another. For the operator detection and tracking application, the final classification layer of this network is replaced by a linear support vector machine with squared hinge loss [Tan13]. Then, the network is fine-tuned to the operator classification task by keeping the parameters of the encoder network fixed, such that only the parameters of the classifier are adapted to the new scenario. The squared hinge loss is chosen over a softmax loss because in the online tracking scenario training data is scarce. By using a hinge loss function, the parameters of the classifier are only so much adapted as necessary to keep a margin between the positive and negative examples. The softmax classifier on the other hand would pull both classes indefinitely far apart. This could potentially lead to overfitting on few training examples that exist early on in tracking.

At each time step, training is carried out on newly arrived detections and a random selection of detections from the past by running a fixed number of iterations stochastic gradient descent. In order to avoid problems due to imbalanced classes—the number of positive training examples is much less than the number of negative examples when multiple non-operators are present in the scene—an equal number of positive and negative images is randomly sampled at each iteration. Since the Bernoulli filter does not perform explicit data association, training labels must be extracted by a heuristic scheme: At each time step, the detection corresponding to the most likely operator location is taken as positive example, remaining detections are used as negative examples. Further, the classifier is only trained when the probability of existence is above a predefined threshold to avoid learning when the operator is currently not visible or when the probability of existence is just building up on re-entrance into the scene.

5.4 Experiments

Joint operator detection and tracking is a very specific problem. Existing people tracking datasets have either been created for evaluation of multi-object tracking systems, or do not contain a combination of LIDAR and image data. There exists no publicly available dataset that targets the specific task of tracking a single operator and contains all required sensor data. In order to evaluate all the specific components of the system under these circumstances, experiments are carried out as follows. First, the system is evaluated on a widely accepted multi-object dataset [FS09] by tracking single individuals from a given starting position. Then, the Bernoulli filter is applied to a custom dataset with common pitfalls specific to single-operator tracking, in particular operator disappearance and re-entry. These situations are not commonly modeled in multi-object datasets. The method is compared against two well-established visual object tracking systems, namely CMT [NP14] and Struck [HGS+16], to establish a comparison against the current state of the art. Performance is measured in terms of a binary classification problem: (1) A precision score indicates the fraction of reported operator locations that coincide with the ground truth location, (2) a recall score indicates the fraction of ground truth locations that have been reported by the tracker. In addition, the number of ID switches is counted, that is the number of times that the tracker switches between an object identity.

5.4.1 Parametrization

The evaluated system is based on a Gaussian mixture implementation of the Bernoulli and PHD filter. A single set of parameters is adopted in all experiments.

The constant velocity motion model adds isotropic noise with standard deviation $\Delta t \cdot 1\,\mathrm{m}$ for the position and $\Delta t \cdot 1\,\mathrm{m\,s^{-1}}$ for the velocity, where Δt is the time gap between consecutive frames. The spatial component of the measurement model adds isotropic noise with standard deviation 0.1 m. In addition, the following set of parameters are used for the Bernoulli and PHD filter: The probability of survival is set to 0.95 and the probability of detection is set to 0.7. Birth components are created around each measurement as in the PHD filter of Section 2.3.3. The PHD likelihood of birth components is set to $3 \times 10^{-2}\,\mathrm{m}^{-1}$. The clutter intensity is set to $c_k(\boldsymbol{z}) = 3 \times 10^{-2}\,\mathrm{m}^{-1}$. At the beginning of the experiment, a single Gaussian birth component is centered around the known operator location to initiate tracking. The probability of existence of the operator state is clamped at 1×10^{-4} to prevent this value to become indefinitely small when the operator leaves the field of view. To measure the impact of the classification module on overall results, all experiments are run once with and once without appearance information. When classification is enabled, the class-conditional appearance likelihood is modeled by two normal distributions placed on each side of the decision boundary, i.e., $\mathcal{N}(s; -m, 1^2)$ and $\mathcal{N}(s; m, 1^2)$ where m is the margin of the squared hinge loss used for training. Without classification module, only the spatial component of the measurement model is taken into account.

5.4.2 PETS 2009

In the first experiment, the Bernoulli filter is applied to sequence S2L1 of the PETS 2009 [FS09] dataset. This sequence is only moderately crowded, but contains complex interactions and occlusions against static scene geometry. Bounding box detections and ground truth have been taken from the MOT challenge benchmark [LTMR+15]. Individuals that have been selected for the purpose of evaluation have been chosen to reflect a variety of appearances and difficulty levels. The results are summarized in Table 5.1. In general, the Bernoulli filter performs favorable compared to CMT and Struck. On all sequences, the Bernoulli filter obtains the highest precision and fewest ID switches. In particular, the Bernoulli filter with integrated classifier successfully discriminates the operator from remaining pedestrians, thus reaching a 100% precision score and 0 identity switches. However, the Bernoulli filter fails to pick up the correct object identity after long-term occlusion against static scene geometry (identity 3) without classifier (motion only). In this case, motion uncertainty raises to a level where the operator cannot be identified purely based on its position. Struck and CMT generally perform considerably worse. Struck successfully tracks the operator in two out of the four evaluation scenarios (identity 1 and 7). CMT generally looses the operator early on during tracking, as indicated by low recall rates. The lower performance of Struck and CMT may be attributed to the more general tracking application they have been

	Precision	Recall	ID
Identity 1			
CMT [NP14]	0.2526	0.2522	12
Struck [HGS+16]	0.9212	0.9212	5
Bernoulli filter (motion & appearance)	1.0000	0.8984	0
Bernoulli filter (motion only)	0.6056	0.4518	3
Identity 3			
Ours (motion & appearance)	1.0000	0.7444	0
Ours (motion only)	0.1473	0.1241	0
CMT [NP14]	0.5434	0.5414	1
Struck [HGS+16]	0.2105	0.2105	2
Identity 7			
CMT [NP14]	0.6713	0.6098	2
Struck [HGS+16]	1.0000	1.0000	0
Bernoulli filter (motion & appearance)	1.0000	0.9390	0
Bernoulli filter (motion only)	1.0000	0.8537	0
Identity 9			
CMT [NP14]	0.1699	0.1699	8
Struck [HGS+16]	0.6795	0.6795	8
Bernoulli filter (motion & appearance)	1.0000	0.8243	0
Bernoulli filter (motion only)	0.9891	0.6988	0

Table 5.1: Results on PETS 2009 S2L1 [FS09]. The best performing method is shown in red, the first runner up in blue. Refer to the tracking output in Figure 5.6 for a qualitative analysis.

.

designed for: Whereas the operator classifier in the Bernoulli filter is trained on positive and negative detections throughout the online tracking application, Struck and CMT collect negative training examples from the entire image, thus yielding a less informed model with respect to discrimination between persons in the scene. Further, Struck and CMT utilize a more general feature space that has not been designed for people tracking specifically. For a qualitative assessment of tracking performance, a selection of images is shown Figure 5.6.

5.4.3 Custom Datasets

In a second experiment, the performance of the Bernoulli filter is evaluated on a custom dataset that is specific to operator tracking. These sequences exhibit long-term occlusions and operator disappearances to stress the re-detection component

	Precision	Recall	ID
Sequence 1			
CMT [NP14]	0.6849	0.6849	–
Struck [HGS+16]	0.7112	0.7112	–
Bernoulli filter (motion & appearance)	1.0000	0.6083	0
Bernoulli filter (motion only)	0.7761	0.4551	5
Sequence 2			
CMT [NP14]	0.6518	0.6518	–
Struck [HGS+16]	0.6196	0.6196	–
Bernoulli filter (motion & appearance)	1.0000	0.7411	0
Bernoulli filter (motion only)	0.9805	0.6286	2
Sequence 3			
CMT [NP14]	0.3787	0.3787	–
Struck [HGS+16]	0.1837	0.1837	–
Bernoulli filter (motion & appearance)	0.9097	0.6168	3
Bernoulli filter (motion only)	0.6878	0.3447	6

Table 5.2: Results on custom datasets. The best performing method is shown in red, the first runner up in blue. No identity switches are reported for CMT and Struck because both trackers loose the object early on when the operator turns away from the camera. Refer to the tracking output in Figure 5.7 for a qualitative analysis.

of the tracking system. The dataset has been collected from a domestic service robot that is equipped with a Microsoft Kinect 2. Detections have been generated using a people detector [MBM12] that is available in current releases of PCL [RC11]. This detector generates point clusters on the ground plane which are subsequently filtered and classified. In total, three sequences of different complexity have been collected:

- Sequence 1: In this sequence, the operator walks away from the robot; people cross the path to block the view.
- Sequence 2: This sequence is similar to the first, but two identities wear identical shirts.
- Sequence 3: The most challenging sequence exhibits complex motion, interaction, and longer periods of operator disappearance. In addition, all identities are similarly dressed (c.f. Figure 5.5).

The ground truth operator position is obtained from a motion capturing system that is built of 12 OptiTrack Prime 13 cameras. These cameras record the 3D position of

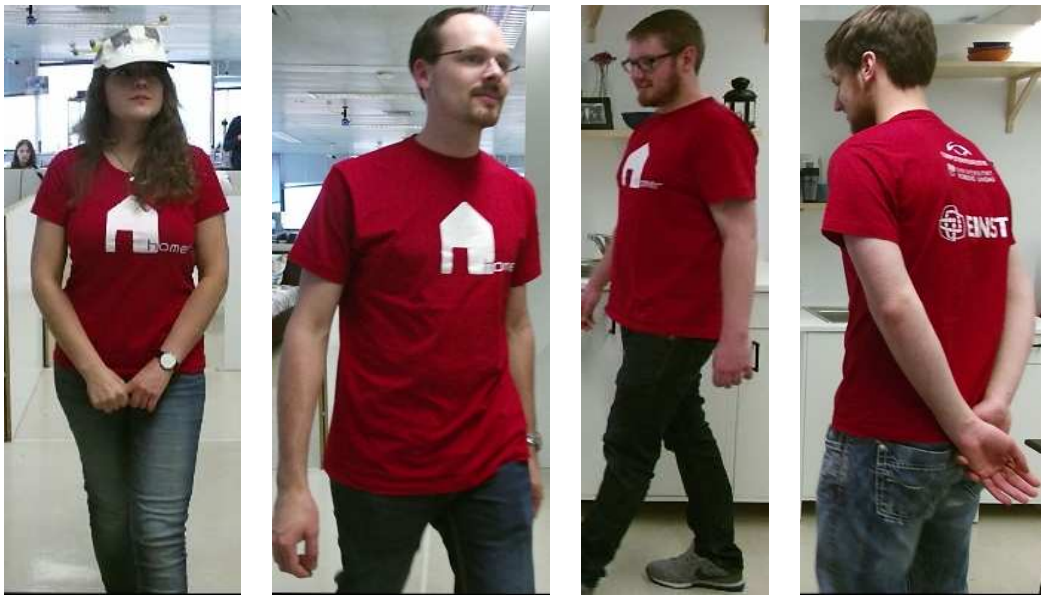
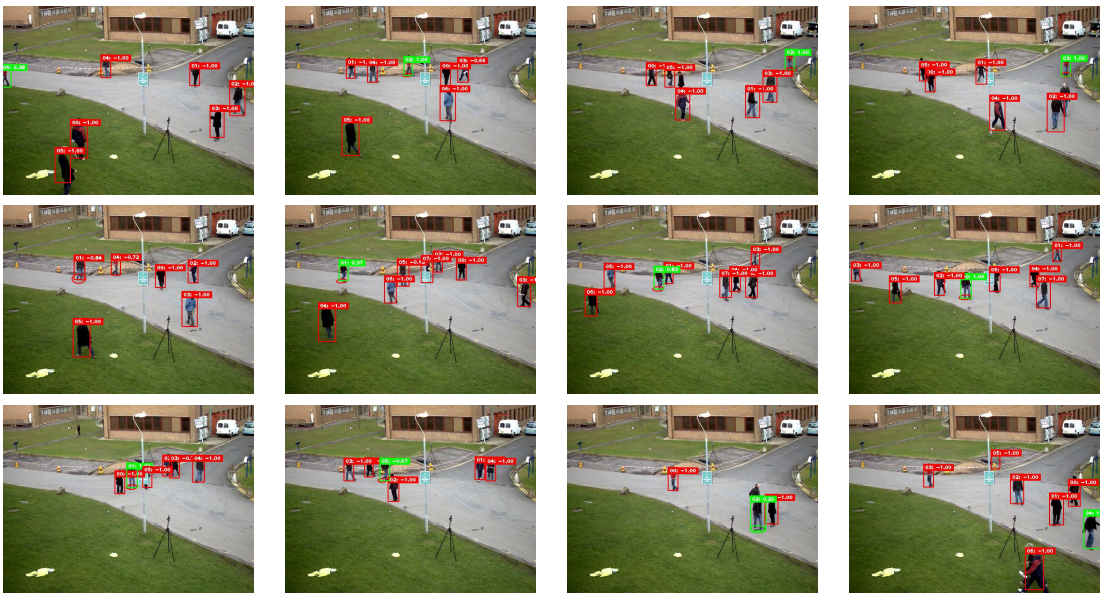


Figure 5.5: In sequence 3 of the custom dataset similar clothing stresses the re-identification component of the tracking framework.

the operator and the robot position at 120 Hz. For evaluation, detections and ground truth positions have been projected into the RGB-D camera frame. The results of the second experiment are summarized in Table 5.2. Again, Struck and CMT perform considerably worse than the Bernoulli filter. In particular, both approaches struggle with sudden appearance changes, e.g., due to the operator turning away from the camera in the beginning of the sequence. The Bernoulli filter on the other hand reliably tracks the operator throughout most of the sequences. The only tracking failure on the custom dataset occurred in the third, most challenging sequence. In this case, all identities are similarly dressed and discrimination based on object appearance alone is challenging (c.f. Figure 5.5). Consequently, during a period of longer absence from the scene the Bernoulli filter picks up a wrong identity twice before the operator is re-detected. A selection of images is shown in Figure 5.7 for a qualitative assessment of tracking performance.

(a)



(b)

Figure 5.6: Tracking output on PETS 2009 [FS09] sequence S2L1. (a) shows results for identity 1; top to bottom: Bernoulli filter (appearance & motion), Bernoulli filter (motion), CMT [NP14], Struck [HGS+16]. (b) shows results for Bernoulli filter (appearance & motion); top to bottom: Identity 3, 7, 9. Detections are visualized as red boxes. The reported operator location is highlighted in green. A confidence ellipse shows the estimated location on the ground plane in case of the Bernoulli filter.

(a)

(b)

Figure 5.7: Tracking output on custom dataset. (a) shows results for sequence 1; top to bottom: Bernoulli filter (appearance & motion), Bernoulli filter (motion), CMT [NP14], Struck [HGS+16]. (b) shows results for Bernoulli filter (appearance & motion) on sequence 2 (top) and sequence 3 (bottom). Detections are visualized as red boxes. The reported operator location is highlighted in green.

Chapter 6

Conclusion and Future Work

This thesis has studied random finite sets in computer vision and robotics applications. The work has focused on two application scenarios. The first scenario was a general multiple object tracking application which has been approached with a method that combines efficient multi-object recursive filtering with global optimization in the search for object trajectories. The second scenario was a single operator detection and tracking task that has been solved using a combination of specialized filters from random finite set theory. Finally, appearance modeling and feature learning has been addressed as a fundamental necessity for successful application of any method to visual tracking scenarios.

The results obtained during evaluation on public benchmarks have underlined the importance of good feature engineering on overall performance. The top performing methods put strong emphasis on this aspect of the tracking problem. Therefore, this is also the most promising direction to follow in future work. Within this thesis, feature learning has been addressed in a deep metric learning framework. A light-weight CNN architecture has been proposed for the people tracking application at hand. This architecture has shown to provide a very good trade off between computational complexity and model performance. In particular, the proposed network has surpassed a much higher capacity network when trained models were transferred from the person re-identification dataset to multi-object tracking datasets. The training of this network has been approached in a joint classification and metric learning framework. For this purpose, a re-parametrization of the conventional softmax classifier has been presented that enables metric learning to be posed in a classification objective. Trained in this regime, the final model performance of the network has been consistently improved compared to baseline direct metric learning methods. However, evaluation has also shown that overall model performance drops when the network is transfered from the person re-identification to the tracking domain. More specifically, experimental results suggest that the network can successfully guide data association on a frame-by-frame basis and that

it can handle short-term occlusions, but discriminative capabilities degrade when objects are occluded for multiple seconds. Since the network has shown robustness towards articulation and changing background on the re-identification test split, the problem is likely due to a distributional shift caused by changing from the re-identification to the tracking dataset. It is left for future work to study by which extent the performance can be improved using unsupervised domain adaption [WBP17b, KXFG15]. Similarly, future work could also investigate whether more advanced data augmentation techniques can improve the performance. Due to the tracklet-based annotation procedure by which the ground truth has been established on the person re-identification dataset, many samples contained in the training set have similar visual appearance. Possibly, variability in the training data can be improved by adding virtually generated samples from generative adversarial networks [GPAM$^+$14]. This methodology has led to increased performance in other applications [SPT$^+$17]. Further, though evaluation was only carried out in a person re-identification and tracking environment, the presented training regime is general, and not limited to these specific applications. In future work, the proposed method could be validated with higher-capacity networks, pre-training on ImageNet, and on more diverse datasets and applications, to study the applicability to general metric learning tasks such as face verification [HRBLM07] and object retrieval [PCI$^+$07].

The MCF-PHD tracker has been proposed as a general multi-object tracker that integrates higher-order motion models into a min-cost flow tracking formulation via multi-object recursive filtering. For this purpose, the PHD filter was re-written in terms of measurement-oriented track hypotheses that are proportional to partitions of the multi-object intensity. These tracks were subsequently linked into globally consistent trajectories. The objective function was formulated as a PHD likelihood ratio that takes on positive values if the observation sequence is more likely generated by an object than clutter. Due to this property, the number of objects could be inferred effectively along with the trajectories in a min-cost flow problem. Derivation of the objective function was largely led by intuition and an explanation in terms of conventional Bayesian filtering rather than strictly formal treatment in the random finite set paradigm. In particular, support for the approach has been established by a comparison to the objective function used in MHT. Future work could further investigate the formal relationship to existing methods. The MeMBer filter [VVC09] may provide the necessary framework to approach the problem formally in the random finite set paradigm. Compared to the PHD filter, the MeMBer filter propagates a full multi-Bernoulli posterior density that factorizes into measurement-oriented tracks similar to the track hypotheses that are formed in the track-oriented PHD filter. Further, recent research has led to a derivation of MHT using the multi-object calculus of finite set statistics [BC17]. This formulation is based on an alternative view that explicitly considers associa-

tion hypotheses that are only handled implicitly in the PHD filter. Since MHT and the PHD filter largely follow the same modeling assumptions, a formal relationship that has not been established in this thesis may be derived from their work.

In practice, the key strength of the MCF-PHD tracker is the integration of object motion models into the min-cost flow problem without increasing the cost of data association. Experimental results suggest that motion significantly contributes to the overall performance of the tracker. On all benchmarks, the MCF-PHD tracker has shown superior performance to min-cost flow formulations that utilize static features only. Compared to an implementation of MHT with CNN features [KLCR15], the MCF-PHD tracker has shown to produce more ID switches. Future work could investigate whether this gap can be bridged by integration of higher-order appearance information within the recursive filtering framework, similar to how object motion is handled. An alternative direction to consider is the following: multi-object recursive filtering provides an estimate for the number of objects in the scene. This information cannot be underestimated. If the number of objects is known, the combinatorial complexity of data association reduces significantly because it limits the number of possible object trajectories. Therefore, future work could explore new ways in which multi-object filtering in general and the cardinality estimate in particular can be help the search for high quality, globally consistent trajectories. For example, if the trajectory starting points are known, then identity-specific classifiers can be incorporated in a multi-commodity flow network from which high quality solutions can be obtained using branch-and-bound [JFL07] or Lagrangian relaxation [DTTS15]. Such a formulation would be much more robust in terms of identity switches because it relaxes the restriction to pairwise appearance costs through integration of global appearance models. Future work could investigate whether trajectory starting points can be obtained from multi-object filtering efficiently.

The final part of this thesis has been dedicated to a single operator detection and tracking task in which the tracked person can leave and re-enter the scene at any time. The Bernoulli filter has been shown to provide a natural representation for this type of problem. The filter has been combined with a PHD that was used to estimate the intensity of non-operators in the scene in order to suppress detection responses corresponding to remaining people in the environment. Both filters interacted in a combined update step. In order to increase the performance of the tracker and in order to enable re-detection on re-entry into the scene, a classifier has been integrated into the measurement likelihood function. Results obtained during the experimental evaluation suggest that this formulation is capable of robust operator detection and tracking, in principle. However, similarly clothing still poses a challenge to the system. This problem is best addressed in the feature representation that is learned offline, before the actual tracking application or in

the online learning component of the tracker. Therefore, any progress in metric learning directly contributes to the performance of the operator detection and tracking system.

Appendix A

Metric Learning Plots

Plots A.1–A.4 show histograms of the cosine similarity between image pairs of the same identity (positive pair) and image pairs of different identities (negative pair) for time gaps $1, 10, 20, 30, 40$ and cumulated histograms over all time gaps from 1 to 50. The time gap refers to the time difference (number of frames) at the observations within the pair. For example, the paired images have been observed in consecutive frames if the time gap is 1. The plots are annotated with the histogram intersection area and the sample mean and covariance. The results have been obtained with the cosine softmax network trained on MARS [ZBS+16]. A plot of the learned embedding on the MARS test split is shown in Figure A.5.

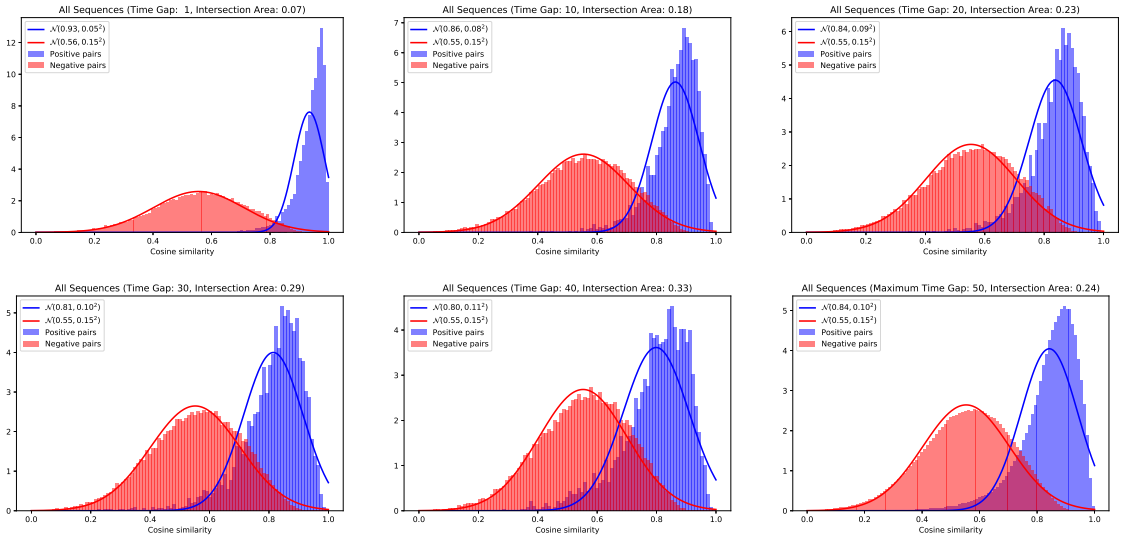


Figure A.1: Cosine distance of positive and negative image pairs computed on the ETH [ELSVG08] dataset, sequences Bahnhof and Sunnyday. The first five plots show the cosine distance at time gaps 1, 10, 20, 30, 40. The last plot shows cumulated histograms over all time gaps from 1 up to 50.

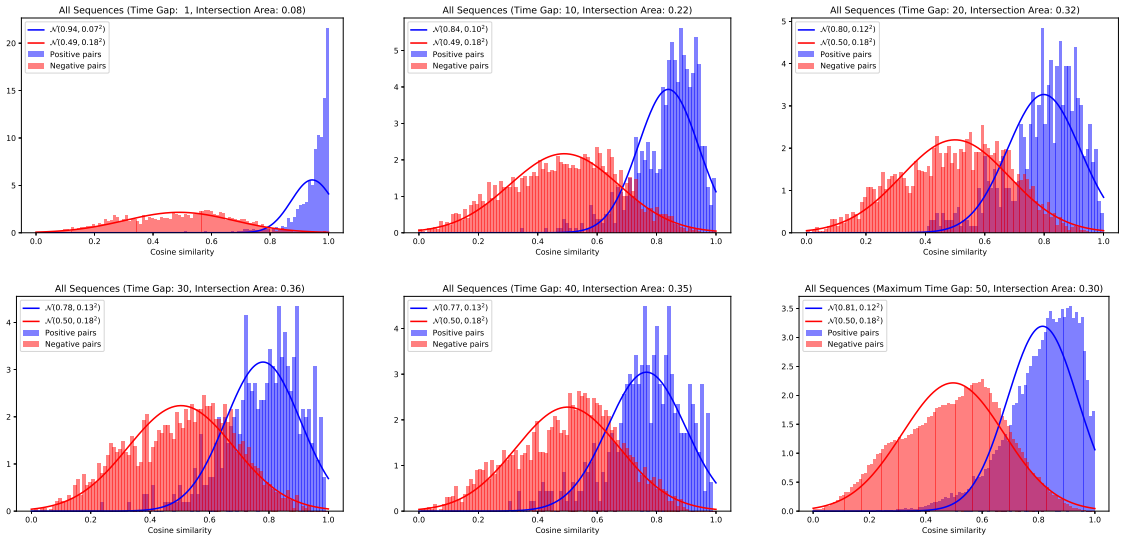


Figure A.2: Cosine distance of positive and negative image pairs computed on the TUD [ARS08] dataset, sequences Campus and Stadtmitte. The first five plots show the cosine distance at time gaps 1, 10, 20, 30, 40. The last plot shows cumulated histograms over all time gaps from 1 up to 50.

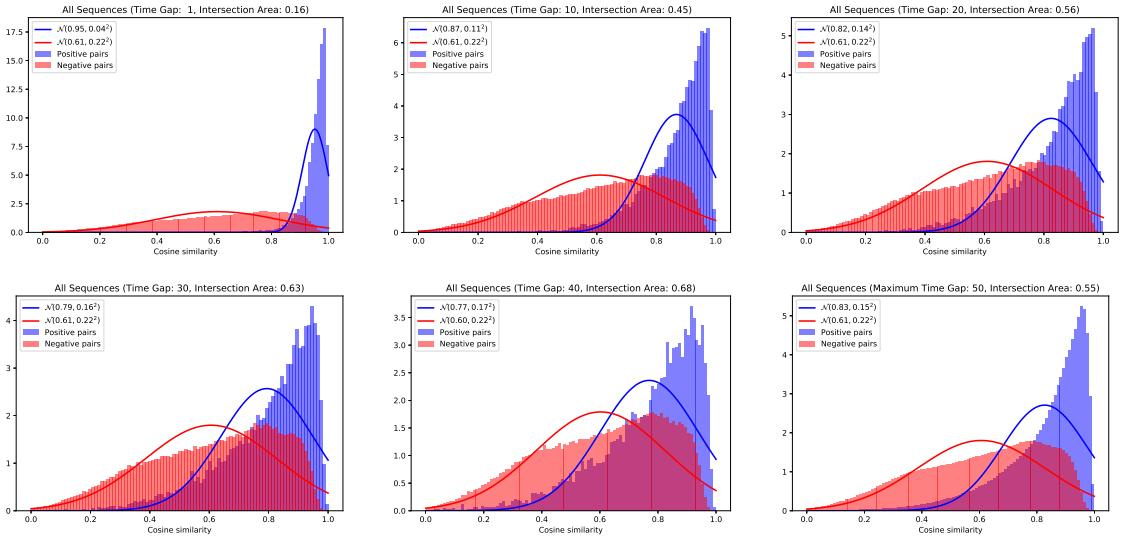


Figure A.3: Cosine distance of positive and negative image pairs computed on the PETS 2009 [FS09] dataset, sequences S1L1-1, S1L1-2, S1L2-1, S1L2-2, S2L1, S2L2, S2L3. The first five plots show the cosine distance at time gaps 1, 10, 20, 30, 40. The last plot shows cumulated histograms over all time gaps from 1 up to 50.

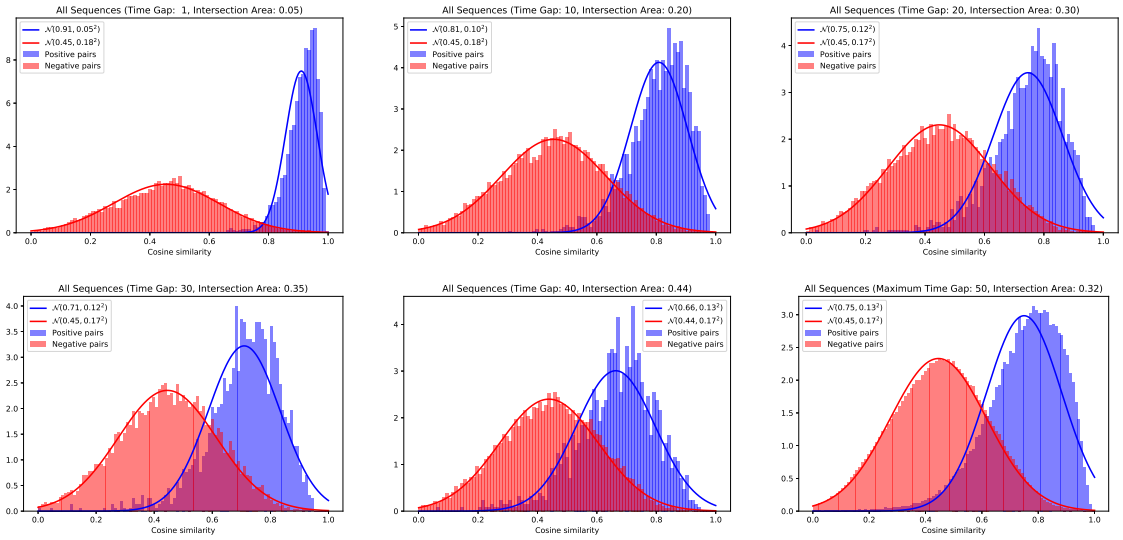


Figure A.4: Cosine distance of positive and negative image pairs computed on the KITTI [GLU12] dataset, training sequence 15–19. The first five plots show the cosine distance at time gaps 1, 10, 20, 30, 40. The last plot shows cumulated histograms over all time gaps from 1 up to 50.

Figure A.5: This plot shows a visualization of the learned embedding on the MARS test split obtained from t-SNE [VDM14]. Images that are close in the image are also close in representation space. The identities shown here have not been used during training.

Appendix B

Single Object Modeling

B.1 Constant Velocity Motion Model

The following motion model describes the state evolution of an object that moves in the two-dimensional plane following a linear motion pattern with constant velocity. The state of this object at time k is modeled by a vector

$$\boldsymbol{x}_k = \begin{pmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{pmatrix} \tag{B.1}$$

that contains the object's position $(x_k, y_k)^{\mathrm{T}}$ and velocity $(\dot{x}_k, \dot{y}_k)^{\mathrm{T}}$. Denote by Δt the elapsed time between time steps k and $k+1$. Then, the evolution of the state

$$\boldsymbol{x}_{k+1} = \boldsymbol{F}\boldsymbol{x}_k + \boldsymbol{\epsilon}_k \tag{B.2}$$

is described by a normally distributed random vector $\boldsymbol{\epsilon}_k \propto \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$ that characterizes the uncertainty about the object's motion and a state transition matrix

$$\boldsymbol{F} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{B.3}$$

If at time $k-1$ the object state is normally distributed with mean $\boldsymbol{m}_{k-1}$ and covariance $\boldsymbol{P}_{k-1}$:

$$p(\boldsymbol{x}_{k-1}) = \mathcal{N}(\boldsymbol{m}_{k-1}, \boldsymbol{P}_{k-1}), \tag{B.4}$$

then the predicted object state at time k is normally distributed as well

$$p(\boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{m}_{k|k-1}, \boldsymbol{P}_{k|k-1}) \tag{B.5}$$

with mean and covariance

$$\boldsymbol{m}_{k|k-1} = \boldsymbol{F}\boldsymbol{m}_{k-1}, \tag{B.6}$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\boldsymbol{P}_{k|k-1}\boldsymbol{F}^{\mathrm{T}} + \boldsymbol{Q}. \tag{B.7}$$

This is an example of the Kalman filter prediction step [TBF05].

B.2 Linear Measurement Model

The following observation model describes the measurement generation process for an object state

$$\boldsymbol{x}_k = \begin{pmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{pmatrix} \tag{B.8}$$

that contains the object's position $(x_k, y_k)^{\mathrm{T}}$ and velocity $(\dot{x}_k, \dot{y}_k)^{\mathrm{T}}$, to be used in combination with the constant velocity motion model when the sensor directly observes the object's position $\boldsymbol{z}_k = (x_k, y_k)$:

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{\delta}_k, \tag{B.9}$$

where $\boldsymbol{\delta}_k \propto \mathcal{N}(\boldsymbol{0}, \boldsymbol{R})$ and

$$\boldsymbol{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \tag{B.10}$$

If at time k the predicted state is normally distributed with mean $\boldsymbol{m}_{k|k-1}$ and covariance $\boldsymbol{P}_{k|k-1}$:

$$p(\boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{m}_{k|k-1}, \boldsymbol{P}_{k|k-1}) \tag{B.11}$$

then the posterior is normal as well

$$p(\boldsymbol{x}_k \mid \boldsymbol{z}_k) = \mathcal{N}(\boldsymbol{m}_k, \boldsymbol{P}_k) \tag{B.12}$$

with

$$\boldsymbol{m}_k = \boldsymbol{m}_{k|k-1} + \boldsymbol{K}_k \left(\boldsymbol{z}_k - \boldsymbol{n}_k\right), \tag{B.13}$$

$$\boldsymbol{P}_k = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k \boldsymbol{S}_k {\boldsymbol{K}_k}^{\mathrm{T}}, \tag{B.14}$$

where

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{S}^{-1}, \tag{B.15}$$

$$\boldsymbol{n}_k = \boldsymbol{H}\boldsymbol{m}_{k|k-1}, \tag{B.16}$$

$$\boldsymbol{S}_k = \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^{\mathrm{T}} + \boldsymbol{R}. \tag{B.17}$$

$$\tag{B.18}$$

This is an example of the Kalman filter update step [TBF05].

B.3 Linear Birth Model

This section describes how a partially observed object state can be initialized from a given measurement. The model described here can be used in combination with the linear measurement model of the previous section where the sensor measures the object position

$$\boldsymbol{z}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix} \tag{B.19}$$

and the state contains the object's position and velocity

$$\boldsymbol{x}_k = \begin{pmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{pmatrix}. \tag{B.20}$$

If $\boldsymbol{\delta}_k = \mathcal{N}(\mathbf{0}, \boldsymbol{R})$ denotes the measurement noise covariance, then

$$p(\boldsymbol{x}_k \mid \boldsymbol{z}_k) = \mathcal{N}(\boldsymbol{m}_k, \boldsymbol{P}_k) \tag{B.21}$$

with mean and covariance

$$\boldsymbol{m}_k = (x_k, y_k, 0, 0)^{\mathrm{T}}, \tag{B.22}$$

$$\boldsymbol{P}_k = \begin{pmatrix} \boldsymbol{R} & 0 \\ 0 & \sigma_v^2 \boldsymbol{I} \end{pmatrix} \tag{B.23}$$

where $\boldsymbol{I}$ is a 2×2 identity matrix and σ_v^2 is the variance of a spherical Gaussian.

Appendix C

Comparison to Multiple Hypothesis Tracking

The modeling assumptions underlying MHT differ from the multi-object system representation in a few aspects. In order to compare the PHD recursion to MHT, it is necessary to make the following assumptions:

1. The false alarm spatial density is uniform over a surveillance volume V and the expected number of clutter returns per volume of measurement space is λ_{FA}. Then, the false alarm rate is $\lambda = \lambda_{\mathrm{FA}} \cdot V$ and the clutter intensity becomes

$$c_k(\boldsymbol{z}) = \lambda \cdot \frac{1}{V} = (\lambda_{\mathrm{FA}} \cdot V) \cdot \frac{1}{V} = \lambda_{\mathrm{FA}}. \tag{C.1}$$

2. Newly appearing objects are always detected and the PHD likelihood of each appearing object is equal to

$$\tau_k^{(\mathrm{u})}(\boldsymbol{z}) = (\lambda_{\mathrm{new}} \cdot V) \cdot \frac{1}{V} = \lambda_{\mathrm{new}}. \tag{C.2}$$

3. The probability of detection and the probability of survival are constant, i.e., $p_{\mathrm{D}}(\boldsymbol{x}) = p_{\mathrm{D}}$ and $p_{\mathrm{S}}(\boldsymbol{x}) = p_{\mathrm{S}}$.

Entry Cost Under the given premises, the cost (4.23) of starting a trajectory at measurement $\boldsymbol{z}_{k,j}$ simplifies to

$$c_{\mathrm{entry}}(\boldsymbol{z}_{k,j}) = -\log \frac{\tau_k^{(\mathrm{u})}(\boldsymbol{z}_{k,j})}{c_k(\boldsymbol{z}_{k,j})} = -\log \frac{\lambda_{\mathrm{new}}}{\lambda_{\mathrm{FA}}}. \tag{C.3}$$

This is a direct consequence of assumptions 1 and 2.

Transition Cost Consider track hypothesis $v_t^{(t,i)}(\boldsymbol{x})$ that is created according to Equation 4.11 at time t where measurement $\boldsymbol{z}_{t,i}$ has been observed:

$$v_t^{(t,i)}(\boldsymbol{x}) = \frac{p_{\mathrm{D}}(\boldsymbol{x}) p_t(\boldsymbol{z}_{t,i} \mid \boldsymbol{x}) v_{t|t-1}(\boldsymbol{x})}{\int p_{\mathrm{D}}(\boldsymbol{x}) p_t(\boldsymbol{z}_{t,i} \mid \boldsymbol{x}) v_{t|t-1}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}}. \tag{C.4}$$

The predicted intensity $v_{t|t-1}(\boldsymbol{x})$ of multi-object state X_t characterizes a Poisson RFS that contains $\hat{N}_{t|t-1} = \int v_{t|t-1}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}$ i.i.d. objects that are distributed according to $p_{t|t-1}(\boldsymbol{x}) = v_{t|t-1}(\boldsymbol{x}) / \hat{N}_{t|t-1}$. For a constant probability of detection $p_{\mathrm{D}}(\boldsymbol{x}) = p_{\mathrm{D}}$ Equation 4.11 simplifies to

$$v_t^{(t,i)}(\boldsymbol{x}) = \frac{p_{\mathrm{D}}(\boldsymbol{x}) p_t(\boldsymbol{z}_{t,i} \mid \boldsymbol{x}) v_{t|t-1}(\boldsymbol{x})}{\int p_{\mathrm{D}}(\boldsymbol{x}) p_t(\boldsymbol{z}_{t,i} \mid \boldsymbol{x}) v_{t|t-1}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}} \tag{C.5}$$

$$= \frac{\hat{N}_{t|t-1} \, p_{\mathrm{D}} \, p_t(\boldsymbol{z}_{t,i} \mid \boldsymbol{x}) p_{t|t-1}(\boldsymbol{x})}{\hat{N}_{t|t-1} \, p_{\mathrm{D}} \, \int p_t(\boldsymbol{z}_{t,i} \mid \boldsymbol{x}) p_{t|t-1}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}} \tag{C.6}$$

$$= \frac{p_t(\boldsymbol{z}_{t,i} \mid \boldsymbol{x}) p_{t|t-1}(\boldsymbol{x})}{\int p_t(\boldsymbol{z}_{t,i} \mid \boldsymbol{x}) p_{t|t-1}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}}. \tag{C.7}$$

Therefore, track hypothesis $v_t^{(t,i)}(\boldsymbol{x})$ follows a conventional Bayes update of $p_{t|t-1}(\boldsymbol{x})$ with measurement $\boldsymbol{z}_{t,i}$ and the intensity mass $\int v_t^{(t,i)}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} = 1$ evaluates to one.

At times $l = t+1, \ldots, k$ this track hypothesis is propagated in time according to the track-oriented PHD recursion (4.5) and (4.9):

$$v_{l|l-1}^{(t,i)}(\boldsymbol{x}) = p_{\mathrm{S}} \int p_{l|l-1}(\boldsymbol{x} \mid \boldsymbol{x}') v_{l-1}^{(t,i)}(\boldsymbol{x}') \,\mathrm{d}\boldsymbol{x}', \tag{C.8}$$

$$v_l^{(t,i)}(\boldsymbol{x}) = (1 - p_{\mathrm{D}}) \, v_{l|l-1}^{(t,i)}(\boldsymbol{x}), \tag{C.9}$$

where again constant survival and detection probabilities have been used. In total, $k - t$ prediction steps (C.8) and $k - t - 1$ update steps (C.9) are applied to generate $v_{k|k-1}^{(t,i)}(\boldsymbol{x})$ from $v_t^{(t,i)}(\boldsymbol{x})$. Therefore, the intensity mass of the predicted track hypothesis at time k evaluates to

$$\eta_{k|k-1}^{(t,i)} = \int v_{k|k-1}^{(t,i)}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} = {p_{\mathrm{S}}}^{k-t} \cdot (1 - p_{\mathrm{D}})^{k-t-1}. \tag{C.10}$$

Denote by $p_{k|k-1}^{(t,i)}(\boldsymbol{x}) = \left(\eta_{k|k-1}^{(t,i)}\right)^{-1} \cdot v_{k|k-1}^{(t,i)}(\boldsymbol{x})$ a probability density function that is proportional to intensity $v_{k|k-1}^{(t,i)}(\boldsymbol{x})$. Then, this density arises as a consequence of a Bayes update of $p_{t|t-1}(\boldsymbol{x})$ with $\boldsymbol{z}_{t,i}$ (c.f. Equation C.7) followed by $k - t$ Bayes filter prediction steps (c.f. Equation C.8). The intensity can be written in terms of this probability density as follows:

$$v_{k|k-1}^{(t,i)}(\boldsymbol{x}) = \eta_{k|k-1}^{(t,i)} \cdot p_{k|k-1}^{(t,i)}(\boldsymbol{x}) \tag{C.11}$$

$$= {p_{\mathrm{S}}}^{k-t} \cdot (1 - p_{\mathrm{D}})^{k-t-1} \cdot p_{k|k-1}^{(t,i)}(\boldsymbol{x}). \tag{C.12}$$

Now, consider the cost (4.26) of linking two measurements $\boldsymbol{z}_{t,i}$ at time t and $\boldsymbol{z}_{k,j}$ at time $k > t$ on a trajectory. For a constant probability of detection and by using Equation C.12 this can be written as

$$c_{\text{link}}(\boldsymbol{z}_{t,i}, \boldsymbol{z}_{k,j}) = -\log \frac{\tau_k^{(t,i)}(\boldsymbol{z}_{k,j})}{c_k(\boldsymbol{z}_{k,j})} \tag{C.13}$$

$$= -\log \frac{\int p_{\text{D}}(\boldsymbol{x}) p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) v_{k|k-1}^{(t,i)}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}}{c_k(\boldsymbol{z}_{k,j})} \tag{C.14}$$

$$= -\log \left(p_{\text{D}} \, p_{\text{S}}^{k-t} (1 - p_{\text{D}})^{k-t-1} \frac{\int p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) p_{k|k-1}^{(t,i)}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}}{\lambda_{\text{FA}}} \right) \tag{C.15}$$

$$\begin{aligned} &= -\log p_{\text{D}} - \log (p_{\text{S}})^{k-t} - \log (1 - p_{\text{D}})^{k-t-1} \\ &\quad - \log \frac{\int p_k(\boldsymbol{z}_{k,j} \mid \boldsymbol{x}) p_{k|k-1}^{(t,i)}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}}{\lambda_{\text{FA}}}. \end{aligned} \tag{C.16}$$

Trajectory Cost The total cost of a trajectory $T_m = \{\boldsymbol{z}_{T_m(1)}, \ldots, \boldsymbol{z}_{T_m(L_m)}\}$ has been defined in Equation 4.22 as a sum over an entry cost term and $L_m - 1$ transition cost terms:

$$C_m(T_m) = c_{\text{entry}}(\boldsymbol{z}_{T_m(1)}) + \sum_{l=2}^{L_m} c_{\text{link}}(\boldsymbol{z}_{T_m(l-1)}, \boldsymbol{z}_{T_m(l)}). \tag{C.17}$$

By substitution of Equation C.3 and Equation C.16 into Equation C.17 the total trajectory cost becomes

$$\begin{aligned} C_m(T_m) = &-\log \frac{\lambda_{\text{new}}}{\lambda_{\text{FA}}} - \sum_{l=2}^{L_m} \log p_{\text{D}} - \sum_{l=2}^{L_m} \log (p_{\text{S}})^{k_{m_l} - k_{m_{l-1}}} \\ &- \sum_{l=2}^{L_m} \log (1 - p_{\text{D}})^{k_{m_l} - k_{m_{l-1}} - 1} \\ &- \sum_{l=2}^{L_m} \log \frac{\int p_{k_{m_l}}(\boldsymbol{z}_{T_m(l)} \mid \boldsymbol{x}) p_{k_{m_l}|k_{m_l}-1}^{(k_{m_{l-1}}, j_{m_{l-1}})}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}}{\lambda_{\text{FA}}}, \end{aligned} \tag{C.18}$$

where $T_m(l) = (k_{m_l}, j_{m_l})$, i.e., k_{m_l} denotes the time and j_{m_l} denotes the index of the l-th measurement on the trajectory. Using $\sum_{i=1}^{n} \log x^{y_i} = \log x^{(\sum_{i=1}^{n} y_i)}$ and $\sum_{l=2}^{L_m} (k_{m_l} - k_{m_{l-1}}) = k_{m_{L_m}} - k_{m_1}$, the trajectory cost can be written as

$$\begin{aligned} C_m(T_m) = &-\log \frac{\lambda_{\text{new}}}{\lambda_{\text{FA}}} - \log (p_{\text{D}})^{L_m - 1} - \log (p_{\text{S}})^{k_{m_{L_m}} - k_{m_1}} \\ &- \log (1 - p_{\text{D}})^{U_m} - \sum_{l=2}^{L_m} \log \frac{\int p_{k_{m_l}}(\boldsymbol{z}_{T_m(l)} \mid \boldsymbol{x}) p_{k_{m_l}|k_{m_l}-1}^{(k_{m_{l-1}}, j_{m_{l-1}})}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}}{\lambda_{\text{FA}}}, \end{aligned} \tag{C.19}$$

where $U_m = \sum_{l=2}^{L_m} \left(k_{m_l} - k_{m_{l-1}} - 1 \right)$ denotes the total number of missed detections on the trajecory.

The score of a trajectory in MHT has been introduced in Equation 4.38. It is re-stated here for a comparison:

$$S(T_m) = (p_{\mathrm{D}})^{L_m - 1} \cdot (1 - p_{\mathrm{D}})^{U_m} \cdot \frac{\lambda_{\mathrm{new}}}{\lambda_{\mathrm{FA}}} \cdot \prod_{l=2}^{L_m} \frac{p(\boldsymbol{z}_{T_m(l)} \mid \boldsymbol{z}_{T_m(1)}, \ldots, \boldsymbol{z}_{T_m(l-1)})}{\lambda_{\mathrm{FA}}}. \quad \text{(C.20)}$$

The negative logarithm of this function is

$$\begin{aligned} -\log S(T_m) = &-\log (p_{\mathrm{D}})^{L_m - 1} - \log (1 - p_{\mathrm{D}})^{U_m} - \log \frac{\lambda_{\mathrm{new}}}{\lambda_{\mathrm{FA}}} \\ &- \log \prod_{l=2}^{L_m} \frac{p(\boldsymbol{z}_{T_m(l)} \mid \boldsymbol{z}_{T_m(1)}, \ldots, \boldsymbol{z}_{T_m(l-1)})}{\lambda_{\mathrm{FA}}}. \end{aligned} \quad \text{(C.21)}$$

By comparison to Equation C.19, this function differs from the trajectory cost in the following two aspects:

1. The trajectory cost (C.19) contains a constant cost term $-\log (p_{\mathrm{S}})^{k_{m_{L_m}} - k_{m_1}}$ that depends on the length of the trajectory. It accounts for a probability of survival that does not exist in MHT. If the probability of survival $p_{\mathrm{S}} = 1$ is one, this term vanishes from the trajectory cost of the MCF-PHD tracker.

2. The observation likelihood in MHT is computed under consideration of all predecessors on the trajectory:

$$\begin{aligned} &p(\boldsymbol{z}_{T_m(l)} \mid \boldsymbol{z}_{T_m(1)}, \ldots, \boldsymbol{z}_{T_m(l-1)}) = \\ &\qquad \int p_{k_{m_l}}(\boldsymbol{z}_{T_m(l)} \mid \boldsymbol{x}) p(\boldsymbol{x} \mid \boldsymbol{z}_{T_m(1)}, \ldots, \boldsymbol{z}_{T_m(l-1)}) \, \mathrm{d}\boldsymbol{x}. \end{aligned} \quad \text{(C.22)}$$

   In the MCF-PHD tracker, this term is replaced by the observation likelihood of a state $\boldsymbol{x}$ that is distributed according to a probability density function that is proportional to the intensity of the predecessor's track hypothesis:

$$\int p_{k_{m_l}}(\boldsymbol{z}_{T_m(l)} \mid \boldsymbol{x}) p_{k_{m_l} | k_{m_l} - 1}^{(k_{m_{l-1}}, j_{m_{l-1}})}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}. \quad \text{(C.23)}$$

   This probability density function arises from a Bayes update of $p_{k_{m_{l-1}} | k_{m_{l-1}} - 1}(\boldsymbol{x})$ of with measurement $\boldsymbol{z}_{T_m(l-1)}$ at time $k_{m_{l-1}}$ and subsequent Bayes filter predictions up to time k_{m_l}.

Appendix D

Gaussian Mixture Bernoulli Filter

The following section presents the Gaussian mixture Bernoulli filter recursion as given by Ristic *et al.* [RVVF13]. At time $k-1$, let $\pi_{k-1}^{(1)} = \{q_{k-1}^{(1)}, p_{k-1}^{(1)}(\boldsymbol{x})\}$ denote the parameters of a Bernoulli state RFS where $q_{k-1}^{(1)}$ denotes the probability of existence and $p_{k-1}^{(1)}(\boldsymbol{x})$ denotes the spatial density that is represented by a Gaussian mixture

$$p_{k-1}^{(1)}(\boldsymbol{x}) = \sum_{i=1}^{L_{k-1}} w_{k-1}^{(i)} \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}_{k-1}^{(i)}, \boldsymbol{P}_{k-1}^{(i)}) \tag{D.1}$$

with $\sum_{i=1}^{L_{k-1}} w_{k-1}^{(i)} = 1$. Then, the parameters of the predicted multi-object density $\pi_{k|k-1}^{(1)} = \{q_{k|k-1}^{(1)}, p_{k|k-1}^{(1)}(\boldsymbol{x})\}$ can be established as follows. Denote by $p_{\mathrm{S}}(\boldsymbol{x}) = p_{\mathrm{S}}$ a constant probability of survival, by

$$p_{k|k-1}(\boldsymbol{x} \mid \boldsymbol{x}') = \mathcal{N}(\boldsymbol{x}; \boldsymbol{F}\boldsymbol{m}_{k|k-1}, \boldsymbol{F}\boldsymbol{P}_{k-1}\boldsymbol{F}^{\mathrm{T}} + \boldsymbol{Q}) \tag{D.2}$$

a linear Gaussian state transition model and by $\{q_{\mathrm{b},k}, p_{\mathrm{b},k}(\boldsymbol{x})\}$ the parameters of a Bernoulli birth RFS B_k that accounts for the object entering the scene at time k with

$$b_k(\boldsymbol{x}) = \sum_{i=1}^{L_{\mathrm{b},k}} w_{\mathrm{b},k}^{(i)} \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}_{\mathrm{b},k}^{(i)}, \boldsymbol{P}_{\mathrm{b},k}^{(i)}). \tag{D.3}$$

Then, the parameters of the predicted multi-object probability density are:

$$q_{k|k-1}^{(1)} = q_{\mathrm{b},k}\left(1 - q_{k-1}^{(1)}\right) + p_{\mathrm{S}} q_{k-1}^{(1)}, \tag{D.4}$$

$$\begin{aligned} p_{k|k-1}^{(1)}(\boldsymbol{x}) = {} & \frac{q_{\mathrm{b},k}\left(1 - q_{k-1}^{(1)}\right)}{q_{k-1}^{(1)}} b_k(\boldsymbol{x}) \\ & + \frac{p_{\mathrm{S}}\, q_{k-1}^{(1)}}{q_{k-1}^{(1)}} \sum_{i=1}^{L_{k-1}} w_{k-1}^{(i)} \mathcal{N}(\boldsymbol{x}; \boldsymbol{F}\boldsymbol{m}_{k-1}^{(i)}, \boldsymbol{F}\boldsymbol{P}_{k-1}^{(i)}\boldsymbol{F}^{\mathrm{T}} + \boldsymbol{Q}). \end{aligned} \tag{D.5}$$

Given a linear measurement model

$$p_k(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{H}\boldsymbol{x}, \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^{\mathrm{T}} + \boldsymbol{R}) \tag{D.6}$$

and a constant probability of detection $p_{\mathrm{D}}(\boldsymbol{x}) = p_{\mathrm{D}}$, the parameters $\pi_k = \{q_k^{(1)}, p_k^{(1)}(\boldsymbol{x})\}$ of the posterior multi-object density can be computed by

$$q_k^{(1)} = \frac{1 - \Delta_k}{1 - q_{k|k-1}^{(1)}\Delta_k} q_{k|k-1}^{(1)}, \tag{D.7}$$

$$\begin{aligned} p_k^{(1)}(\boldsymbol{x}) = {} & \frac{1 - p_{\mathrm{D}}}{1 - \Delta_k} p_{k|k-1}^{(1)}(\boldsymbol{x}) + \frac{p_{\mathrm{D}}}{1 - \Delta_k} \\ & \cdot \sum_{\boldsymbol{z} \in Z_k} \sum_{i=1}^{L_{k|k-1}} \frac{w_{k|k-1}^{(i)} l_k^{(i)}(\boldsymbol{z})}{c_k(\boldsymbol{z})} \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}_k^{(i)}(\boldsymbol{z}), \boldsymbol{P}_k^{(i)}), \end{aligned} \tag{D.8}$$

$$\Delta_k = p_{\mathrm{D}} \left[1 - \sum_{\boldsymbol{z} \in Z_k} \sum_{i=1}^{L_{k|k-1}} \frac{w_{k|k-1}^{(i)} l_k^{(i)}(\boldsymbol{z})}{c_k(\boldsymbol{z})} \right], \tag{D.9}$$

where $c_k(\boldsymbol{z})$ is the intensity of the Poisson clutter RFS and

$$\boldsymbol{m}_k^{(i)}(\boldsymbol{z}) = \boldsymbol{m}_{k|k-1}^{(i)} + \boldsymbol{K}_k^{(i)} \left(\boldsymbol{z} - \boldsymbol{n}_k^{(i)} \right), \tag{D.10}$$

$$\boldsymbol{P}_k^{(i)} = \boldsymbol{P}_{k|k-1}^{(i)} - \boldsymbol{K}_k^{(i)} \boldsymbol{S}_k^{(i)} {\boldsymbol{K}_k^{(i)}}^{\mathrm{T}}, \tag{D.11}$$

$$l_k^{(i)}(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{n}_k^{(i)}, \boldsymbol{S}_k^{(i)}), \tag{D.12}$$

$$\boldsymbol{n}_k^{(i)} = \boldsymbol{H}\boldsymbol{m}_{k|k-1}^{(i)}, \tag{D.13}$$

$$\boldsymbol{S}_k^{(i)} = \boldsymbol{H}\boldsymbol{P}_{k|k-1}^{(i)}\boldsymbol{H}^{\mathrm{T}} + \boldsymbol{R}, \tag{D.14}$$

$$\boldsymbol{K}_k^{(i)} = \boldsymbol{P}_{k|k-1}^{(i)}\boldsymbol{H}^{\mathrm{T}} {\boldsymbol{H}_k}^{-1}. \tag{D.15}$$

Refer to [RVVF13] for an extensive tutorial on the Bernoulli filter, including pseudo code for a sequential Monte-Carlo implementation and practical applications to unconventional problems.

Appendix E

Results on 2D MOT 2015

The following table contains the full list of published submissions to the 2DMOT2015 [LTMR$^+$15] benchmark.

Method	MOTA	MT	ML	ID	FM	FP	FN	Avg Rank	Hz
APRCNN_Pub [CAS+17]	38.5	8.7%	37.4%	586	1263	4005	33203	15.0	6.7
AMIR15 [SAS17]	37.6	15.8%	26.8%	1024	2024	7933	29397	19.8	1.9
HybridDAT [YWJ17]	35.0	11.4%	42.2%	358	1267	8455	31140	17.4	4.6
AM [COL+17]	34.3	11.4%	43.4	348	1463	5154	34848	18.4	0.5
TSMLCDEnew [WWCW17]	34.3	14.0%	39.4%	618	959	7869	31908	20.8	6.5
QuadMOT [SBCH17]	33.8	12.9%	36.9%	703	1430	7898	32061	23.9	3.7
NOMT [Cho15]	33.7	12.2%	44.0%	442	823	7762	32547	18.2	11.5
TDAM [YJ16]	33.0	13.3%	39.1%	464	1506	10064	30617	23.4	5.9
CDA_DDALpb [BY17]	32.8	9.7%	42.2%	614	1583	4983	35690	23.8	2.3
MHT-DAM [KLCR15]	32.4	16.0%	43.8%	435	826	9064	32060	20.8	0.7
MDP [XAS15]	30.3	13.0%	38.4%	680	1500	9717	32422	26.5	1.1
MCF-PHD tracker	**29.9**	**11.9%**	**44.0%**	**656**	**989**	**8892**	**33529**	**24.4**	**12.2**
CNNTCM [WWS+16]	29.6	11.2%	44.0%	712	943	7786	34733	25.4	1.7
SCEA [HYLYY16]	29.1	8.9%	47.3%	604	1182	6060	36912	29.8	6.8
SiameseCNN [LTCFS16]	29.0	8.5%	48.4%	639	1316	5160	37798	30.8	52.8
oICF [KBHA16]	27.1	6.4%	48.7%	454	1660	7594	36757	32.9	1.4
TO [MTDVG16]	25.7	4.3%	57.4%	383	600	4779	40511	35.7	5.0
LP_SSVM [WF17]	31.3	5.8%	53.0%	646	849	8369	36932	31.3	41.3
ELP [MDRM15]	25.0	7.5%	43.8%	1396	1804	7345	37344	36.7	5.7
LINF1 [FBADL16]	24.5	5.5%	64.6%	298	744	5864	40207	31.1	7.5
JPDA_m [HRMZ+15]	23.8	5.0%	58.1%	365	869	6373	40084	31.3	32.6
MotiCon [LTFK+14]	23.1	4.7%	52.0%	1018	1061	10404	35844	43.0	1.4
SegTrack [MLTSR15]	22.5	5.8%	63.9%	697	737	7890	39020	43.0	0.2
EAMTTpub [SMPC16]	22.3	5.4%	52.7%	833	1485	7924	38982	38.3	12.2
OMT_DFH [JKK+17]	21.2	7.1%	46.5%	563	1255	13218	34657	32.8	28.6
MTSTracker [AKNB17]	20.6	9.0%	36.9%	1387	2357	15161	32212	37.8	19.3
LP2D [LTMR+15]	19.8	6.7%	41.2%	1649	1712	11580	36045	39.6	112.1
DCO_X [MSR16]	19.6	5.1%	54.9%	521	819	10652	38232	39.9	0.3
CEM [MRS14]	19.3	8.5%	46.5%	813	1023	14180	34591	38.5	1.1
RNN_LSTM [MRD+17]	19.0	5.5%	45.6%	1490	2081	11578	36706	33.5	165.2
RMOT [YYLY15]	18.6	5.3%	53.3%	684	1282	12473	36835	43.4	7.9
TSDA_OAL [JKK+16]	18.6	9.4%	42.3%	806	1544	16350	32853	43.4	7.9
GMPHD [SJ16]	18.5	3.9%	55.3%	459	1266	7864	41766	38.6	19.8
SMOT [DCS13]	18.2	2.8%	54.8%	1148	2132	8780	40310	54.0	2.7
ALExTRAC [BORU16]	17.0	3.9%	52.4%	1859	1872	9233	39933	47.5	3.7
TBD [GLW+14]	15.9	6.4%	47.9%	1939	1963	14943	34777	51.3	0.7
GSCR [FBADL15]	15.8	1.8%	61.0%	514	1010	7597	43633	38.6	28.1
TC_ODAL [BY14]	15.1	3.2%	55.8%	637	1716	12970	38538	53.1	1.7
DP_NMS [LTMR+15]	14.5	6.0%	40.8%	4537	3090	13171	34814	40.7	444.8
LDCT [SCC15]	4.7	11.4%	32.5%	12348	2918	14066	32156	38.8	20.7

Table E.1: Full table of results on 2D MOT 2015 [LTMR+15] in offline mode (the entire sequence is optimized in a single batch). MT = Mostly Tracked, ML = Mostly Lost, ID = Number of ID switches, FM = Fragments, FP = False positives, FN = False negatives, Avg Rank = Average rank over all metrics, Hz = Reported tracker speed in frames per seconds. Note that this is an ongoing benchmark; the ranking can change in future (accessed on 11/18/2017).

List of Tables

3.1 Overview of the person re-identification CNN architecture 37
3.2 Person re-identification results on Market 1501 and MARS datasets 42

4.1 Simulation results in medium clutter profile 84
4.2 Simulation results in high clutter profile 84
4.3 Evaluation results on PETS 2009 88
4.4 Ranking on PETS 2009 in terms of MOTA and ID switches 88
4.5 Evaluation results on PETS 2009 in different operational modes . . 92
4.6 Evaluation results on 3D MOT 2015 93
4.7 Evaluation results on RGB-D People dataset 94
4.8 Evaluation results on 2D MOT 2015 96

5.1 Operator tracking results on PETS 2009 S2L1 110
5.2 Operator tracking results on custom dataset 111

E.1 Full table of results on 2D MOT 2015 134

List of Figures

2.1 Illustration of a Probability Hypothesis Density on the real line . . 11
2.2 Example flow network of a traffic routing problem 23
2.3 Min-cost flow solution to the traffic routing problem 24

3.1 Class-conditional densities and decision surface of a softmax classifier 34
3.2 Illustration of a von Mises-Fisher distribution and decision boundary of the cosine softmax classifier. 35
3.3 Effect of the free scaling parameter on the shape of conditional class probabilities in the cosine softmax classifier 36
3.4 Structure of pre-activation residual blocks 38
3.5 Evolution of validation accuracy and triplet loss on MARS 41
3.6 Example queries from Market 1501 test set 46
3.7 Visualization of the learned embedding on MARS 47
3.8 Cosine similarity between positive and negative pairs on datasets ETH and PETS 2009 48
3.9 Cosine similarity as function of the time gap between detections . . 48

4.1 Illustration of the track-oriented PHD filter recursion 55
4.2 Illustration of the min-cost flow tracking framework 59
4.3 Example of a flow network over three time steps 63
4.4 Illustrative comparison between MHT and MCF-PHD tracker . . . 66
4.5 Three examples for low-confidence false alarms on PETS 2009 . . . 69
4.6 Two examples of a learned detector confidence likelihood 71
4.7 Illustration of the CLEAR MOT association strategy 82
4.8 Visualization of the simulated tracking scenario 83
4.9 Tracking output comparison in the simulated scenario 85
4.10 Example images of two sequences from PETS 2009 86
4.11 Tracking output on PETS 2009 test sequences 89
4.12 Examples for successful tracking on PETS 2009 90
4.13 Run-time on PETS 2009 92
4.14 MHT-DAM and MCF-PHD tracker on TUD-Crossing 98

5.1 Robot *Lisa* in an operator following scenario 100
5.2 Schematic diagram of the joint detection and tracking problem . . . 101
5.3 Interacting Bernoulli/PHD filter update: Schematic chart of the PHD filter update 106
5.4 Interacting Bernoulli/PHD filter update: Schematic chart of the Bernoulli filter update 107
5.5 Similarly dressed people in custom operator tracking dataset 112
5.6 Operator tracking output on PETS 2009 113
5.7 Operator tracking output on custom dataset 114

A.1 Cosine distance of positive and negative image pairs on ETH 120
A.2 Cosine distance of positive and negative image pairs on TUD 120
A.3 Cosine distance of positive and negative image pairs on PETS 2009 121
A.4 Cosine distance of positive and negative image pairs on KITTI . . . 121
A.5 Visualization of the learned embedding on MARS test split 122

Abbreviations & Acronyms

Abbreviations

i.i.d.	independent and identically distributed

Acronyms

CNN	Convolutional Neural Network
FISST	Finite Set Statistics
GLMB	Generalized Labeled Multi-Bernoulli
GM-PHD	Gaussian Mixture Probability Hypothesis Density
GNN	Global Nearest Neighbor
JPDA	Joint Probabilistic Data Association
LIDAR	Light Detection and Ranging
MCF-PHD	Min-Cost Flow Probability Hypothesis Density (tracker)
MeMBer	Multi-Target Multi-Bernoulli
MHT	Multiple Hypothesis Tracking
PCL	Point Cloud Library
PHD	Probability Hypothesis Density
RADAR	Radio Detection and Ranging
RFS	Random Finite Set

Symbols

General Notation

$\lvert \cdot \rvert$	Cardinality of a set
$\emptyset$	Empty set
$\lVert \cdot \rVert_2$	ℓ_2 norm of a vector
$\det(\cdot)$	Matrix determinant
$\mathrm{diag}(\cdot)$	Constructs a diagonal matrix from the given argument
$\mathbb{1}_{y=k}$	Indicator function that evaluates to 1 if $y = k$ and 0 otherwise
$\mathcal{N}(\cdot; \boldsymbol{m}, \boldsymbol{P})$	Gaussian density with mean $\boldsymbol{m}$ and covariance $\boldsymbol{P}$
$p(\cdot)$	Conventional probability density function
$P(\cdot)$	Probability measure
$\mathcal{F}(\cdot)$	Set of all finite subsets
$\int \delta X$	Set integral
$\pi(\cdot)$	Multi-object probability density function

Multi-Object State Estimation

$\mathcal{X}$	State space
$\mathcal{Z}$	Measurement space
X_k	Multi-object state RFS at time k
Z_k	Multi-object measurement RFS at time k
B_k	Birth RFS that contains the set of appearing objects at time k
K_k	Clutter RFS that contains the set of false alarms at time k
N_k	Number of states $\lvert X_k \rvert$ at time k
M_k	Number of measurements $\lvert Z_k \rvert$ received at time k

$Z_{n:m}$	Short-hand notation for $Z_n, \ldots, Z_m$
$\boldsymbol{x}_{k,j}$	The j-th vector-valued state at time k
$\boldsymbol{z}_{k,j}$	The j-th vector-valued measurement observed at time k
$p_{k\|k-1}(\cdot \mid \cdot)$	Markov state transition density from time $k-1$ to time k
$p_k(\cdot \mid \cdot)$	Measurement likelihood function at time k
$p_{\mathrm{S}}(\boldsymbol{x})$	State-dependent probability of survival
$p_{\mathrm{D}}(\boldsymbol{x})$	State-dependent probability of detection
$b_k(\cdot)$	PHD of the Poisson birth RFS at time k
$\lambda_{\mathrm{b},k}$	Expected number of appearing objects at time k
$p_{\mathrm{b},k}(\cdot)$	Spatial density of the Poisson birth RFS B_k at time k
$c_k(\cdot)$	PHD of the Poisson clutter RFS at time k
$\lambda_{\mathrm{c},k}$	Expected number of clutter returns at time k
$p_{\mathrm{c},k}(\cdot)$	Spatial density of the Poisson clutter RFS K_k at time k
$v_k(\cdot)$	PHD of a multi-object state at time k

Gaussian Mixture Probability Hypothesis Density Filter

L_k	Number of GM-PHD components at time k
$w_k^{(i)}$	Weight of the i-th GM-PHD component at time k
$\boldsymbol{m}_k^{(i)}$	Mean of the i-th GM-PHD component at time k
$\boldsymbol{P}_k^{(i)}$	Covariance of the i-th GM-PHD component at time k
$\boldsymbol{F}$	Kalman filter state transition matrix
$\boldsymbol{Q}$	Kalman filter state transition noise covariance
$\boldsymbol{H}$	Kalman filter measurement matrix
$\boldsymbol{R}$	Kalman filter measurement noise covariance
$w_{\min}$	Pruning threshold: minimum GM-PHD component weight
U	Pruning threshold: GM-PHD component merge threshold

Metric Learning

$\mathcal{D}$	Datset of training images and associated class labels
C	Number of classes in the dataset
N	Number of examples in the dataset
$\boldsymbol{x}_i$	The i-th image in the dataset, $\boldsymbol{x}_i \in \mathbb{R}^D$
y_i	The i-th label in the dataset, $y_i \in \{1, \ldots, C\}$
$\boldsymbol{r}_i$	Feature representation $\boldsymbol{r}_i \in \mathbb{R}^d$ corresponding to the i-th image $\boldsymbol{x}_i$

$f_\Theta(\cdot)$	Encoder network with parameters Θ, such that $\boldsymbol{r}_i = f_\Theta(\boldsymbol{x}_i)$
$g_\Omega(\cdot)$	Classifier with parameters Ω that maps from feature to class label
$\boldsymbol{w}_k$	Standard softmax class parameter vector of class $k \in \{1, \ldots, C\}$
b_k	Standard softmax class bias parameter of class $k \in \{1, \ldots, C\}$
$\tilde{\boldsymbol{w}}_k$	Cosine softmax class parameter vector of class $k \in \{1, \ldots, C\}$
κ	Free scaling parameter of the cosine softmax classifier

Min-Cost Flow Probability Hypothesis Density Tracker

M	Length of the observation sequence (number of time steps)
$u_k(\cdot)$	PHD of objects that remain undetected until time k
$v_k^{(t,i)}(\cdot)$	PHD at time k of a track hypothesis that originates from measurement $\boldsymbol{z}_{t,i}$, with $k \geq t$
$p_k^{(t,i)}(\cdot)$	A probability density that is proportional to $v_k^{(t,i)}(\boldsymbol{x})$
$r_{t,i}$	Scaling parameter associated with $v_k^{(t,i)}(\boldsymbol{x})$
$\hat{N}_{\min}$	A confirmation threshold that defines the minimum intensity mass of confirmed tracks in the track-oriented PHD filter
$\tau_k(\cdot)$	PHD likelihood of the multi-object state at time k
$\tau_k^{(\mathrm{u})}(\cdot)$	PHD likelihood of undetected objects $u_k(\boldsymbol{x})$
$\tau_k^{(t,i)}(\cdot)$	PHD likelihood of track hypothesis $v_k^{(t,i)}(\boldsymbol{x})$
L_m	Length of the m-th trajectory
T_m	The m-th single-object trajectory $\{(k_{m_l}, j_{m_l})\}_{l=1}^{L_m}$
T	Set of non-overlapping trajectories $\{T_m\}$
T^*	Minimum cost non-overlapping trajectories
$\boldsymbol{z}_{T_m(l)}$	The l-th measurement on the m-th trajectory
$\boldsymbol{z}_{k_{m_l}, j_{m_l}}$	The l-th measurement on the m-th trajectory
$C_m(\cdot)$	Cost of a given trajectory
$c_{\mathrm{entry}}(\cdot)$	Cost of starting a trajectory at the given measurement
$c_{\mathrm{link}}(\cdot, \cdot)$	Cost of linking two measurements on a trajectory (transition cost)
$c_{\mathrm{link}}^{(\mathrm{mot})}(\cdot, \cdot)$	Motion component of trajectory transition cost
$c_{\mathrm{link}}^{(\mathrm{app})}(\cdot, \cdot)$	Appearance component of trajectory transition cost
s	Flow network source node
t	Flow network terminal node
$u_{k,j}$	Measurement entry node (connected to candidate predecessors)
$v_{k,j}$	Measurement exit node (connected to candidate successors)
$p_{\mathrm{fg}}(s)$	Foreground/true object likelihood over detector confidence scores

$p_{\text{bg}}(s)$	Background/false alarm likelihood over detector confidence scores
V	Volume of the surveillance region
$\boldsymbol{y}$	Spatial component of a measurement $\boldsymbol{z}$, e.g., ground plane or bounding box position
$\boldsymbol{r}$	Appearance feature representation of a measurement $\boldsymbol{z}$
s	Detector confidence score

Operator Following

$X_k^{(0)}$	Set of operators at time k, contains at most one object
$X_k^{(1)}$	Set of non-operators at time k
$q_k^{(1)}$	Operator probability of existence at time k
$p_k^{(1)}(\boldsymbol{x})$	Operator spatial density at time k
$v_k^{(0)}(\boldsymbol{x})$	PHD of the Poisson non-operator RFS at time k
$v_k^{(1)}(\boldsymbol{x})$	PHD of the Bernoulli operator RFS at time k
$p_k^{(0)}(\cdot \mid \cdot)$	Measurement likelihood function of non-operators at time k
$p_k^{(1)}(\cdot \mid \cdot)$	Measurement likelihood function of the operator at time k

Bibliography

[AJM15] Ejaz Ahmed, Michael Jones, and Tim K Marks. An improved deep learning architecture for person re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3908–3916, 2015.

[AKNB17] Nguyen Thi Lan Anh, Furqan M Khan, Farhood Negin, and Francois Bremond. Multi-object tracking using multi-channel part appearance representation. In *IEEE International Workshop on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, 2017.

[AMB07] Kai O Arras, Oscar Martinez Mozos, and Wolfram Burgard. Using boosted features for the detection of people in 2D range data. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3402–3407, 2007.

[AMO93] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ, 1993.

[ARS08] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. People-tracking-by-detection and people-detection-by-tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. Advance online publication. doi: 10.1109/CVPR.2008.4587583.

[ASR12] Anton Andriyenko, Konrad Schindler, and Stefan Roth. Discrete-continuous optimization for multi-target tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1926–1933, 2012.

[BC13] Asad A Butt and Robert T Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1846–1853, 2013.

[BC17] Edmund F Brekke and Mandar Chitre. The multiple hypothesis tracker derived from finite set statistics. In *International Conference on Information Fusion (FUSION)*, pages 630–637, 2017.

[BFTF11] Jerome Berclaz, Francois Fleuret, Engin Türetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011.

[Bis06] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, 2006.

[BIS09] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, volume 56. Springer, Berlin/Heidelberg, 2009.

[BORU16] Alex Bewley, Lionel Ott, Fabio Ramos, and Ben Upcroft. Alextrac: Affinity learning by exploring temporal reinforcement within association chains. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2212–2218, 2016.

[BP99] Samuel Blackman and Robert Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House, Norwood, MA, 1999.

[BRG+16] Michael Beard, Stephan Reuter, Karl Granström, Ba-Tuong Vo, Ba-Ngu Vo, and Alexander Scheel. Multiple extended target tracking with labeled random finite sets. *IEEE Transactions on Signal Processing*, 64(7):1638–1653, 2016.

[BRL+11] Michael D Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1820–1833, 2011.

[BS08] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):246–309, 2008.

[BVV15] Michael Beard, Ba-Tuong Vo, and Ba-Ngu Vo. Bayesian multi-target tracking with merged measurements using labelled random finite sets. *IEEE Transactions on Signal Processing*, 63(6):1433–1447, 2015.

[BY14] Seung-Hwan Bae and Kuk-Jin Yoon. Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1218–1225, 2014.

[BY17] Seung-Hwan Bae and Kuk-Jin Yoon. Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. Advance online publication. doi: 10.1109/TPAMI.2017.2691769.

[CAS+17] Long Chen, Haizhou Ai, Chong Shang, Zijie Zhuang, and Bo Bai. Online multi-target tracking with convolutional neural networks. In *IEEE International Conference on Image Processing (ICIP)*, pages 645–649, 2017.

[CH96] Ingemar J Cox and Sunita L Hingorani. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, 1996.

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 539–546, 2005.

[Cho15] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3029–3037, 2015.

[Col12] Robert T Collins. Multitarget data association with higher-order motion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1744–1751, 2012.

[COL+17] Qi Chu, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, Bin Liu, and Nenghai Yu. Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4836–4845, 2017.

[CUH15] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *International Conference on Learning Representations (ICLR)*, pages 1–14, 2015.

[DCS13] Caglayan Dicle, Octavia I Camps, and Mario Sznaier. The way they move: Tracking multiple targets with similar appearance. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2304–2311, 2013.

[DJV+14] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning (ICML)*, pages 647–655, 2014.

[DTTS15] Afshin Dehghan, Yicong Tian, Philip HS Torr, and Mubarak Shah. Target identity-aware network flow for online multiple target tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1146–1154, 2015.

[ELSVG08] Andreas Ess, Bastian Leibe, Konrad Schindler, and Luc Van Gool. A mobile vision system for robust multi-person tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. Advance online publication. doi: 10.1109/CVPR.2008.4587581.

[Eur16a] European Commission. EU-funded projects in robotics for ageing well, 2016. Online resource. url: `http://ec.europa.eu/newsroom/dae/document.cfm?doc_id=12942`.

[Eur16b] European Road Safety Observatory. Advanced driver assistance systems 2016, 2016. Online resource. url: `https://ec.europa.eu/transport/road_safety/sites/roadsafety/files/ersosynthesis2016-adas15_en.pdf`.

[Eur17] European Commission. XCYCLE: Advanced measures to reduce cyclists' fatalities and increase comfort in the interaction with motorised vehicles, 2017. Online resource. project id: 635975, url: `http://cordis.europa.eu/project/rcn/193364_en.html`.

[FBADL15] Low Fagot-Bouquet, Romaric Audigier, Yoann Dhome, and Frédéric Lerasle. Online multi-person tracking based on global sparse collaborative representations. In *IEEE International Conference on Image Processing (ICIP)*, pages 2414–2418, 2015.

[FBADL16] Loïc Fagot-Bouquet, Romaric Audigier, Yoann Dhome, and Frédéric Lerasle. Improving multi-frame data association with sparse representations for robust near-online multi-object tracking. In *European Conference on Computer Vision (ECCV)*, pages 774–790, 2016.

[FBSS83] T.E. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8(3):173–184, 1983.

[FGMR10] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[FS09] James Ferryman and Ali Shahrokni. Pets2009: Dataset and challenge. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS-Winter)*, pages 1–6, 2009.

[GFS07] Rachel Gockley, Jodi Forlizzi, and Reid Simmons. Natural person-following behavior for social robots. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 17–24, 2007.

[GFS16] Ángel F García-Fernández and Lennart Svensson. Trajectory probability hypothesis density filter. *arXiv preprint arXiv:1605.07264*, 2016.

[GFSM16] Ángel F García-Fernández, Lennart Svensson, and Mark R Morelande. Multiple target tracking based on sets of trajectories. *arXiv preprint arXiv:1605.08163*, 2016.

[GKSB10] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[GLW+14] Andreas Geiger, Martin Lauer, Christian Wojek, Christoph Stiller, and Raquel Urtasun. 3D traffic scene understanding from movable platforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(5):1012–1025, 2014.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

[HBL17] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[HGS+16] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr. Struck: Structured output tracking with kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2096–2109, 2016.

[HRBLM07] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[HRMZ+15] Seyed Hamid Rezatofighi, Anton Milan, Zhen Zhang, Qinfeng Shi, Anthony Dick, and Ian Reid. Joint probabilistic data association revisited. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3047–3055, 2015.

[HRS+16] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7310–7319, 2016.

[HYLYY16] Ju Hong Yoon, Chang-Ryeol Lee, Ming-Hsuan Yang, and Kuk-Jin Yoon. Online multi-object tracking via structural constraint event aggregation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1392–1400, 2016.

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–645, 2016.

[JB09] Radford Juang and Philippe Burlina. Comparative performance evaluation of GM-PHD filter in clutter. In *International Conference on Information Fusion (FUSION)*, pages 1195–1202, 2009.

[JFL07] Hao Jiang, Sidney Fels, and James J Little. A linear programming approach for multiple object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. Advance online publication. doi: 10.1109/CVPR.2007.383180.

[JKK+16] Jaeyong Ju, Daehun Kim, Bonhwa Ku, David K Han, and Hanseok Ko. Online multi-person tracking with two-stage data association and online appearance model mearning. *IET Computer Vision*, 11(1):87–95, 2016.

[JKK+17] Jaeyong Ju, Daehun Kim, Bonhwa Ku, David K Han, and Hanseok Ko. Online multi-object tracking with efficient track drift and fragmentation handling. *Journal of the Optical Society of America*, 34(2):280–293, 2017.

[Kal60] Rudolph E. Kalman. A new approach to linear filtering and prediction problems. *Transaction of the ASME - Journal of Basic Engineering*, 82(1):35–45, 1960.

[KB15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, pages 1–15, 2015.

[KBHA16] Hilke Kieritz, Stefan Becker, Wolfgang Hübner, and Michael Arens. Online multi-person tracking using integral channel features. In *IEEE International Workshop on Advanced Video and Signal Based Surveillance (AVSS)*, pages 122–130, 2016.

[KHW+12] Martin Koestinger, Martin Hirzer, Paul Wohlhart, Peter M Roth, and Horst Bischof. Large scale metric learning from equivalence constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2288–2295, 2012.

[KLCR15] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4696–4704, 2015.

[KLW+10] Bharath Kalyan, KW Lee, S Wijesoma, D Moratuwage, and Nicholas M Patrikalakis. A random finite set based detection and tracking using 3D LIDAR in dynamic environments. In *IEEE International Conference on Systems Man and Cybernetics (SMC)*, pages 2288–2292, 2010.

[KRH15] Tobias Klinger, Franz Rottensteiner, and Christian Heipke. Probabilistic multi-person tracking using dynamic bayesian networks. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, pages 435–442, 2015.

[KRH17] Tobias Klinger, Frank Rottensteiner, and Christian Heipke. Probabilistic multi-person localisation and tracking in image sequences. *ISPRS Journal of Photogrammetry and Remote Sensing*, 127:73–88, 2017.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[Kuh55] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.

[KXFG15] Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. Unsupervised domain adaptation for zero-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2452–2460, 2015.

[LBLA16] Timm Linder, Stefan Breuers, Bastian Leibe, and Kai O Arras. On multi-modal people tracking from mobile platforms in very crowded and dynamic environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5512–5519, 2016.

[LCZH17] Dangwei Li, Xiaotang Chen, Zhang Zhang, and Kaiqi Huang. Learning deep context-aware features over body and latent parts for person re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 384–393, 2017.

[LGU15] Philip Lenz, Andreas Geiger, and Raquel Urtasun. Followme: Efficient online min-cost flow tracking with bounded memory and computation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4364–4372, 2015.

[LHN09] Yuan Li, Chang Huang, and Ram Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2953–2960, 2009.

[LHZL15] Shengcai Liao, Yang Hu, Xiangyu Zhu, and Stan Z Li. Person re-identification by local maximal occurrence representation and metric learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2197–2206, 2015.

[LKVN07] Astrid Linder, Albert Kircher, Anna Vadeby, and Sara Nygårdhs. *Intelligent transport systems (ITS) in passenger cars and methods for assessment of traffic safety impact: a literature review.* Statens väg-och transportforskningsinstitut, Linköping, 2007. Online resource. url: `http://vti.diva-portal.org/smash/get/diva2:670550/FULLTEXT01.pdf`.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[LPOZ15] Angus Leigh, Joelle Pineau, Nicolas Olmedo, and Hong Zhang. Person tracking and following with 2D laser scanners. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 726–733, 2015.

[LSA11] Matthias Luber, Luciano Spinello, and Kai O Arras. People tracking in RGB-D data with on-line boosted target models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3844–3849, 2011.

[LTCFS16] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking: Siamese CNN for robust target association. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 33–40, 2016.

[LTFK+14] Laura Leal-Taixé, Michele Fenzi, Alina Kuznetsova, Bodo Rosenhahn, and Silvio Savarese. Learning an image-based motion context for multiple people tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3542–3549, 2014.

[LTMR+15] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*, 2015.

[LTPMR11] Laura Leal-Taixé, Gerard Pons-Moll, and Bodo Rosenhahn. Everybody needs somebody: Modeling social and grouping behavior on a

linear programming multiple people tracker. In *IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 120–127, 2011.

[LYO17] Yu Liu, Junjie Yan, and Wanli Ouyang. Quality aware network for set to set recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5790–5799, 2017.

[LZG17] Wei Li, Xiatian Zhu, and Shaogang Gong. Person re-identification by deep joint learning of multi-loss classification. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2194–2200, 2017.

[LZXW14] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 152–159, 2014.

[Mah03] Ronald P S Mahler. Multitarget bayes filtering via first-order multitarget moments. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4):1152–1178, 2003.

[Mah04] Ronald P S Mahler. "Statistics 101" for multisensor, multitarget data fusion. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):53–64, 2004.

[Mah07a] Ronald P S Mahler. PHD filters of higher order in target number. *IEEE Transactions on Aerospace and Electronic Systems*, 43(4):1523–1543, 2007.

[Mah07b] Ronald P S Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House, Norwood, MA, 2007.

[Mah13] Ronald P S Mahler. "statistics 102" for multisource-multitarget detection and tracking. *IEEE Journal of Selected Topics in Signal Processing*, 7(3):376–389, 2013.

[Mah14] Ronald P S Mahler. *Advances in Statistical Multisource-Multitarget Information Fusion*. Artech House, Norwood, MA, 2014.

[MBM12] Matteo Munaro, Filippo Basso, and Emanuele Menegatti. Tracking people within groups with RGB-D data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2101–2107, 2012.

[MĆP15] Ivan Marković, Josip Ćesić, and Ivan Petrović. Von mises mixture PHD filter. *IEEE Signal Processing Letters*, 22(12):2229–2233, 2015.

[MDRM15] Niall McLaughlin, Jesus Martinez Del Rincon, and Paul Miller. Enhancing linear programming with motion modeling for multi-target tracking. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 71–77. IEEE, 2015.

[MLTSR15] Anton Milan, Laura Leal-Taixé, Konrad Schindler, and Ian Reid. Joint tracking and segmentation of multiple targets. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5397–5406, 2015.

[MRD+17] Anton Milan, Seyed Hamid Rezatofighi, Anthony R Dick, Ian D Reid, and Konrad Schindler. Online multi-target tracking using recurrent neural networks. In *AAAI Conference on Artificial Intelligence*, pages 4225–4232, 2017.

[MRS14] Anton Milan, Stefan Roth, and Konrad Schindler. Continuous energy minimization for multitarget tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):58–72, 2014.

[MSR16] Anton Milan, Konrad Schindler, and Stefan Roth. Multi-target tracking by discrete-continuous energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2054–2068, 2016.

[MTDVG16] Santiago Manen, Radu Timofte, Dengxin Dai, and Luc Van Gool. Leveraging single for multi-target tracking using a novel trajectory overlap affinity measure. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016. Advance online publication. doi: 10.1109/WACV.2016.7477566.

[MVAV11] John Stephen Mullane, Ba-Ngu Vo, Martin David Adams, and Ba-Tuong Vo. *Random Finite Sets for Robot Mapping & SLAM: New Concepts in Autonomous Robotic Map Representations*, volume 72. Springer Science & Business Media, Berlin/Heidelberg, 2011.

[MVV11] Ronald P S Mahler, Ba-Tuong Vo, and Ba-Ngu Vo. CPHD filtering with unknown clutter rate and detection profile. *IEEE Transactions on Signal Processing*, 59(8):3497–3513, 2011.

[NP14] Georg Nebehay and Roman Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 862–869, 2014.

[OSXJS16] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4004–4012, 2016.

[PCI+07] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[PCV09] Kusha Panta, Daniel E Clark, and Ba-Ngu Vo. Data association and track management for the Gaussian mixture Probability Hypothesis Density filter. *IEEE Transactions on Aerospace and Electronic Systems*, 45(3):1003–1016, 2009.

[PESVG09] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *IEEE International Conference on Computer Vision (ICCV)*, pages 261–268, 2009.

[PRF11] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1201–1208, 2011.

[PVS+04] Kusha Panta, Ba-Ngu Vo, Sumeetpal Singh, Arnaud Doucet, and Iven Kader. Probability Hypothesis Density filter versus multiple hypothesis tracking. In *SPIE Signal Processing, Sensor Fusion, and Target Recognition*, volume 5429, pages 284–295, 2004.

[PVS05] K Panta, B Vo, and S Singh. Improved Probability Hypothesis Density (PHD) filter for multitarget tracking. In *IEEE International Conference on Intelligent Sensing and Information Processing (ICISIP)*, pages 213–218, 2005.

[PVS07] Kusha Panta, Ba-Ngu Vo, and Sumeetpal Singh. Novel data association schemes for the probability hypothesis density filter. *IEEE Transactions on Aerospace and Electronic Systems*, 43(2):1–11, 2007.

[PVV+15] Francesco Papi, Ba-Ngu Vo, Ba-Tuong Vo, Claudio Fantacci, and Michael Beard. Generalized labeled multi-bernoulli approximation of multi-object densities. *IEEE Transactions on Signal Processing*, 63(20):5487–5497, 2015.

[RC11] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point cloud library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011. Advance online publication. doi: 10.1109/ICRA.2011.5980567.

[RCVV12] B Ristic, D Clark, Ba-Ngu Vo, and Ba-Tuong Vo. Adaptive target birth intensity for PHD and CPHD filters. *IEEE Transactions on Aerospace and Electronic Systems*, 48(2):1656–1668, 2012.

[Rei79] D. B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[RMR+17] Hamid Rezatofighi, Anton Milan, Ian Reid, Ehsan Abbasnejad, and Anthony Dick. Deepsetnet: Predicting sets with deep neural networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 5247–5256, 2017.

[RPDB16] Oren Rippel, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev. Metric learning with adaptive density discrimination. *International Conference on Learning Representations (ICLR)*, pages 1–15, 2016.

[RVVF13] Branko Ristic, Ba-Tuong Vo, Ba-Ngu Vo, and Alfonso Farina. A tutorial on bernoulli filters: Theory, implementation and applications. *IEEE Transactions on Signal Processing*, 61(13):3406–3430, 2013.

[SAS17] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *IEEE International Conference on Computer Vision (ICCV)*, pages 300–311, 2017.

[SBCH17] Jeany Son, Mooyeol Baek, Minsu Cho, and Bohyung Han. Multi-object tracking with quadruplet convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5620–5629, 2017.

[SCC15] Francesco Solera, Simone Calderara, and Rita Cucchiara. Learning to divide and conquer for online multi-target tracking. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4373–4381, 2015.

[SJ16] Young-min Song and Moongu Jeon. Online multiple object tracking with the hierarchically adopted GM-PHD filter using motion and appearance. In *IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–4, 2016.

[SK16] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 901–909, 2016.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[SMPC16] Ricardo Sanchez-Matilla, Fabio Poiesi, and Andrea Cavallaro. Online multi-target tracking with strong and weak detections. In *European Conference on Computer Vision (ECCV) Workshops*, pages 84–99, 2016.

[SPT+17] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2107–2116, 2017.

[SR13] Aleksandr V Segal and Ian Reid. Latent data association: Bayesian model selection for multi-target tracking. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2904–2911, 2013.

[SRASC14] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 806–813, 2014.

[SVCC17] Samuel Schulter, Paul Vernaza, Wongun Choi, and Manmohan Chandraker. Deep network flow for multi-object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6951–6960, 2017.

[SZDW17] Yifan Sun, Liang Zheng, Weijian Deng, and Shengjin Wang. Svdnet for pedestrian retrieval. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3800–3808, 2017.

[TAA+16] Rudolph Triebel, Kai Arras, Rachid Alami, Lucas Beyer, Stefan Breuers, Raja Chatila, Mohamed Chetouani, Daniel Cremers, Vanessa Evers, Michelangelo Fiore, et al. SPENCER: A socially aware service robot for passenger guidance and help in busy airports. In *Field and Service Robotics*, pages 607–622. Springer, 2016.

[TAAS17] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3539–3548, 2017.

[Tan13] Yichuan Tang. Deep learning using linear support vector machines. In *International Conference on Machine Learning (ICML) Challenges in Representation Learning Workshop*, pages 1–6, 2013.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, MA, 2005.

[TC05] Elin A Topp and Henrik I Christensen. Tracking for following and passing persons. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2321–2327, 2005.

[TYRW14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1701–1708, 2014.

[VDM14] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[VDMW01] Rudolph Van Der Merwe and Eric A Wan. The square-root unscented Kalman filter for state and parameter-estimation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 6, pages 3461–3464, 2001.

[VHW16] Rahul Rama Varior, Mrinal Haloi, and Gang Wang. Gated siamese convolutional neural network architecture for human re-identification. In *European Conference on Computer Vision (ECCV)*, pages 791–808, 2016.

[VM06] Ba-Ngu Vo and W-K Ma. The gaussian mixture Probability Hypothesis Density filter. *IEEE Transactions on Signal Processing*, 54(11):4091–4104, 2006.

[VMBS+15] Ba-Ngu Vo, Mahendra Mallick, Yaakov Bar-Shalom, Stefano Coraluppi, Richard Osborne, Ronald P S Mahler, and Ba-Tuong Vo. Multitarget tracking. In John G Webster, editor, *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 1–15. John Wiley & Sons, Inc., 2015.

[VPRH87] Christian Voy, Ferdinand Panik, Dietrich Reister, and Ludwig Hamm. PROMETHEUS, ein europäisches Forschungsprojekt zur Gestaltung des Straßenverkehrs der Zukunft. *Automobil-Industrie*, 32(2), 1987.

[VSD05] Ba-Ngu Vo, Sumeetpal Singh, and Arnaud Doucet. Sequential monte carlo methods for multitarget filtering with random finite sets. *IEEE Transactions on Aerospace and Electronic Systems*, 41(4):1224–1245, 2005.

[VSL+16] Rahul Rama Varior, Bing Shuai, Jiwen Lu, Dong Xu, and Gang Wang. A siamese long short-term memory architecture for human re-identification. In *European Conference on Computer Vision (ECCV)*, pages 135–153, 2016.

[VVC09] Ba-Tuong Vo, Ba-Ngu Vo, and Antonio Cantoni. The cardinality balanced multi-target multi-bernoulli filter and its implementations. *IEEE Transactions on Signal Processing*, 57(2):409–423, 2009.

[VVP14] Ba-Ngu Vo, Ba-Tuong Vo, and Dinh Phung. Labeled random finite sets and the bayes multi-target tracking filter. *IEEE Transactions on Signal Processing*, 62(24):6554–6567, 2014.

[WB18] Nicolai Wojke and Alex Bewley. Deep cosine metric learning for person re-identification. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 748–756, 2018.

[WBP17a] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, 2017.

[WBP17b] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Addressing appearance change in outdoor robotics with adversarial domain adaptation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1551–1558, 2017.

[WF17] Shaofei Wang and Charless C Fowlkes. Learning optimal parameters for multi-target tracking with contextual interactions. *International Journal of Computer Vision*, 122(3):484–501, 2017.

[WLC+17] Longyin Wen, Zhen Lei, Ming-Ching Chang, Honggang Qi, and Siwei Lyu. Multi-camera multi-target tracking with space-time-view hypergraph. *International Journal of Computer Vision*, 122(2):313–333, 2017.

[WLY+14] Longyin Wen, Wenbo Li, Junjie Yan, Zhen Lei, Dong Yi, and Stan Z Li. Multiple target tracking based on undirected hierarchical relation hypergraph. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1282–1289, 2014.

[WMP17] Nicolai Wojke, Raphael Memmesheimer, and Dietrich Paulus. Joint operator detection and tracking for person following from mobile platforms. In *International Conference on Information Fusion (FUSION)*, pages 905–912, 2017.

[WP16] Nicolai Wojke and Dietrich Paulus. Global data association for the Probability Hypothesis Density filter using network flows. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 567–572, 2016.

[WP17] Nicolai Wojke and Dietrich Paulus. Confidence-aware Probability Hypothesis Density filter for visual multi-object tracking. In *International Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, pages 132–139, 2017.

[WS09] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[WVDM00] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC)*, pages 153–158, 2000.

[WWCW17] Bing Wang, Gang Wang, Kap Luk Chan, and Li Wang. Tracklet association by online target-specific metric learning and coherent dynamics estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(3):589–602, 2017.

[WWS+16] Bing Wang, Li Wang, Bing Shuai, Zhen Zuo, Ting Liu, Kap Luk Chan, and Gang Wang. Joint learning of convolutional neural networks and temporally constrained metrics for tracklet association. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 386–393, 2016.

[WZLQ16] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision (ECCV)*, pages 499–515, 2016.

[XAS15] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4705–4713, 2015.

[XLOW16] Tong Xiao, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang. Learning deep feature representations with domain guided dropout for pperson re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1249–1258, 2016.

[YJ16] Min Yang and Yunde Jia. Temporal dynamic appearance modeling for online multi-person tracking. *Computer Vision and Image Understanding*, 153:16–28, 2016.

[YN12] Bo Yang and Ram Nevatia. An online learned CRF model for multi-target tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2034–2041, 2012.

[YN14] Bo Yang and Ramakant Nevatia. Multi-target tracking by online learning a CRF model of appearance and motion patterns. *International Journal of Computer Vision*, 107(2):203–217, 2014.

[YWJ17] Min Yang, Yuwei Wu, and Yunde Jia. A hybrid data association framework for robust online multi-object tracking. *IEEE Transactions on Image Processing*, 26:5667–5679, 2017.

[YYLY15] Ju Hong Yoon, Ming-Hsuan Yang, Jongwoo Lim, and Kuk-Jin Yoon. Bayesian multi-object tracking using motion context from multiple objects. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 33–40, 2015.

[YZBB17] Rui Yu, Zhichao Zhou, Song Bai, and Xiang Bai. Divide and fuse: A re-ranking approach for person re-identification. In *British Machine Vision Conference*, pages 1–13, 2017.

[ZBS+16] Liang Zheng, Zhi Bie, Yifan Sun, Jingdong Wang, Chi Su, Shengjin Wang, and Qi Tian. MARS: A video benchmark for large-scale person re-identification. In *European Conference on Computer Vision (ECCV)*, pages 868–884, 2016.

[ZDS12] Amir Roshan Zamir, Afshin Dehghan, and Mubarak Shah. GMCP-tracker: Global multi-object tracking using generalized minimum clique graphs. In *European Conference on Computer Vision (ECCV)*, pages 343–356, 2012.

[ZHW+17] Zhen Zhou, Yan Huang, Wei Wang, Liang Wang, and Tieniu Tan. See the forest for the trees: Joint spatial and temporal recurrent neural networks for video-based person re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4747–4756, 2017.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*, pages 1–12, 2016.

[ZLN08] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. Advance online publication. doi: 10.1109/CVPR.2008.4587584.

[ZLWZ17] Liming Zhao, Xi Li, Jingdong Wang, and Yueting Zhuang. Deeply-learned part-aligned representations for person re-identification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3219–3228, 2017.

[ZST+15] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1116–1124, 2015.

[ZZS+17] Liang Zheng, Hengheng Zhang, Shaoyan Sun, Manmohan Chandraker, and Qi Tian. Person re-identification in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1367–1376, 2017.

Curriculum Vitae

Nicolai Wojke was born on May 6, 1986 in Berlin. He received his university entrance qualification (Abitur) in 2005. In the same year, Nicolai Wojke moved to Koblenz for his studies in Computational Visualistics at the University of Koblenz and Landau. There he graduated with honors and obtained a diploma degree in computer science in 2011. He worked as a student assistant in the image recognition laboratory of Prof. Dr. Lutz Priese from Februrary to July 2008 and June to December 2009. In between, he was awarded a scholarship from the German Academic Exchange Service for studying abroad at the Institute for Artificial Intelligence, University of Georgia from August 2008 to May 2009.

Nicolai Wojke has been a research associate in the Active Vision Group of Prof. Dr.-Ing. Dietrich Paulus from August to September 2011 and May 2012 to July 2017. During the period of October 2011 and April 2012, he was a research associate in the working group for real-time systems of Prof. Dr. Dieter Zöbel. Additionally, from September 2010 to September 2017 he was a part-time software developer at the spin-off company Vision & Robotics GmbH where he developed solutions to practical computer vision problems in industrial applications. In October 2017, he joined the German Aerospace Center where he works as a research associate at the Institute for Transportation Systems stationed in Berlin.

Publications

Nicolai Wojke and Alex Bewley. Deep cosine metric learning for person re-identification. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 748–756, 2018

Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, 2017

Nicolai Wojke, Raphael Memmesheimer, and Dietrich Paulus. Joint operator detection and tracking for person following from mobile platforms. In *International Conference on Information Fusion (FUSION)*, pages 905–912, 2017

Nicolai Wojke and Dietrich Paulus. Confidence-aware Probability Hypothesis Density filter for visual multi-object tracking. In *International Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, pages 132–139, 2017

Nicolai Wojke and Dietrich Paulus. Global data association for the Probability Hypothesis Density filter using network flows. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 567–572, 2016

Nicolai Wojke, Jens Hedrich, Detlev Droege, and Dietrich Paulus. Gaze-estimation for consumer-grade cameras using a Gaussian process latent variable model. *Pattern Recognition and Image Analysis*, 26(1):248–255, 1 2016

Frank Neuhaus, Nicolai Wojke, Christian Winkens, Bastian Krayer, Dietrich Paulus, and Marcel Häselich. Autonomous 3D terrain mapping and object localization for the Spacebot Camp 2015. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, pages 1–8, 2016

Nicolai Wojke, Frank Neuhaus, and Dietrich Paulus. Localization and pose estimation of textureless objects for autonomous exploration missions. In *IEEE International Conference on Image Processing (ICIP)*, pages 1304–1308, 2016

Nicolai Wojke, Jens Hedrich, and Detlev Droege. Latent-space Gaussian process gaze-tracking. In *9th German-Russian Workshop on Pattern Recognition and Image Understanding*, pages 144–149, 2015

Arend Buchacher and Nicolai Wojke. Graphenbasierte Segmentierung unter Verwendung von Structured Support Vector Machines im Kontext der Objekt-Lokalisierung. In *21. Workshop Farbbildverarbeitung*, pages 27–38, 2015

Viktor Seib, Nicolai Wojke, Malte Knauf, and Dietrich Paulus. Detecting fine-grained affordances with an anthropomorphic agent model. In *European Conference on Computer Vision (ECCV) Workshops*, pages 413–419, 2015

Marcel Häselich, Benedikt Jöbgen, Nicolai Wojke, Jens Hedrich, and Dietrich Paulus. Confidence-based pedestrian tracking in unstructured environments using 3D laser distance measurements. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4118–4123, 2014

Marcel Häselich, Marc Arends, Nicolai Wojke, Frank Neuhaus, and Dietrich Paulus. Probabilistic terrain classification in unstructured environments. *Robotics and Autonomous Systems*, 61(10):1051–1059, 10 2013

Jens Hedrich and Nicolai Wojke. Adaptivität-Sicherheit-Paradoxon in der Robotik. In *INFORMATIK 2013 – Informatik angepasst an Mensch, Organisation und Umwelt*, pages 1085–1095, 2013

Nicolai Wojke and Marcel Häselich. Moving vehicle detection and tracking in unstructured environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3082–3087, 2012