FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

People Tracking using Particle Filters and an advanced Human Motion Model on Mobile Robots

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Informatik

eingereicht von

Florian Beck, BSc
Matrikelnummer 1126203

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Johann Blieberger
Mitwirkung: Univ.Ass. Dipl.-Ing. Dr.techn. Markus Bader

Wien, 5. März 2018

Florian Beck	Johann Blieberger

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

People Tracking using Particle Filters and an advanced Human Motion Model on Mobile Robots

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Computer Engineering

by

Florian Beck, BSc
Registration Number 1126203

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Johann Blieberger
Assistance: Univ.Ass. Dipl.-Ing. Dr.techn. Markus Bader

Vienna, 5th March, 2018

Florian Beck | Johann Blieberger

Erklärung zur Verfassung der Arbeit

Florian Beck, BSc
Neustiftgasse 75,
7122 Gols

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 5. März 2018

Florian Beck

Kurzfassung

Für mobile Roboter, die sich in Umgebungen mit Menschen bewegen, ist es wichtig, die Position von Personen in ihrer Umgebung wahrzunehmen, sodass sich ein Roboter sicher bewegen kann. Um die Position einer Person stabil schätzen zu können, ist es erforderlich, sie nicht nur mit Sensoren wie Kameras oder Laserscannern zu erfassen, sondern auch Messungen verschiedener Sensoren über die Zeit zu kombinieren. Diese Diplomarbeit beschäftigt sich mit solchen Schätzungen für mehrere Personen.
Um die Position einer Person unter der Einbeziehung von Detektionen verschiedener Sensoren zu schätzen wird ein Partikel-Filter verwendet. Im Gegensatz zu Kalman Filter haben Partikel-Filter den Vorteil, dass diese auch beliebige, multimodale Wahrscheinlichkeitsverteilungen repräsentieren können. Das Filter benötigt ein menschliches Bewegungsmodell, um die Bewegung einer Person zu schätzen. In dieser Arbeit stellen wir einen neuen Ansatz basierend auf statischen Karten und historischen Daten vor, um die Qualität der Vorhersage im Gegensatz zu typischen, bisher verwendeten State-of-the-Art Ansätzen wie Constant-Velocity und Coordinated-Turn zu verbessern.
Um den Ansatz auf mehrere Personen zu erweitern, wird die Nearest-Neighbor-Data-Association Strategie verwendet, welche Detektionen zu existierenden Tracks zuordnet. Für das Initialisieren neuen Tracks auf Basis neu detektierter Personen beziehungsweise für das Löschen von veralteten Tracks werden gängige Lösungen aus der Literatur verwendet.
Um den Ansatz auszuwerten, wurde zunächst die Vorhersage-Qualität des vorgestellten Bewegungsmodells mit dem Constant-Velocity und dem Coordinated-Turn Bewegungsmodell in bestimmten Szenarien verglichen. Die Ergebnisse legen nahe, dass das vorgestellte Bewegungsmodell wesentlich bessere Vorhersagen liefert. Darüber hinaus wurden zwei reale People-Tracking-Szenarien getestet, um die allgemeine Tracking-Qualität zu beurteilen. Zu diesem Zweck wurden wieder die drei oben genannten Bewegungsmodelle verglichen. Die Ergebnisse zeigen, dass bei ausreichend hoher Frequenz and Detektionen das Bewegungsmodell einen relativ geringen Einfluss auf die Tracking Qualität hat, jedoch kann das vorgestellte Bewegungsmodell durch die genauere Vorhersage fehlende Detektionen besser ausgleichen.

Abstract

For mobile robots moving in environments alongside humans, it is crucial to perceive the location of people in their surroundings, such that a robot can move safely around them. In order to keep a stable estimate of a person's position it is required to not only detect them with sensors such as cameras or laser scanners, but also to combine measurements of different sensors over time. This thesis is concerned with such estimates for multiple people.

To estimate the position of a single person over time, incorporating detections acquired from different sensors, we apply particle filtering for state estimation. Particle filtering holds the advantage to be able to represent arbitrary multi-modal probability distributions in comparison with Kalman filtering. The filter requires a human motion model in order to forward predict the movement of a person. In this work, a novel approach based on static maps and historical data is introduced, in order to improve the prediction quality in contrast to typical state of the art approaches applying constant velocity or coordinated turn motion models.

To extend to multiple persons, well known nearest neighbor data association is applied which assigns received detections to existing tracks. In order to initialize tracks for newly observed people, as well as to delete obsolete tracks we apply common techniques proposed in literature.

To evaluate the approach we first compared forward prediction behavior of the proposed motion model with constant velocity and coordinated turn models in specific map scenarios, suggesting that the proposed motion model provides significantly better forward prediction. Furthermore, we tested two real world people tracking scenarios to assess overall tracking performance. We again compared the three motion models mentioned above, which show that if detections occur frequently enough, the motion model has a small influence on tracking quality, however using a more accurate model enables the filter to compensate for missing detections.

Contents

Kurzfassung **vii**

Abstract **ix**

Contents **xi**

1 Introduction **1**
1.1 Problem Statement 1
1.2 Aim of the Work 2
1.3 Methodological Approach 3
1.4 Structure of the Work 4

2 State of the Art **7**
2.1 People Detection 7
2.2 People Tracking 11
2.3 Summary 12

3 Background **15**
3.1 Probability Theory 15
3.2 Bayesian Filtering 20
3.3 Motion Models for People Tracking 24
3.4 The Assignment Problem for Data Association 29
3.5 CLEAR MOT Metrics 34
3.6 Summary 36

4 Approach **37**
4.1 Probabilistic Model for People Tracking 37
4.2 People Tracking Algorithm 40

5 Implementation **63**
5.1 People Tracking Package 63
5.2 Heat Map Package 64
5.3 Robot Setup 65

6 Results **69**
6.1 Experimental Setup 69
6.2 Forward Prediction Comparison Results 71
6.3 Tracking Performance Comparison Results 77
6.4 Summary 82

7 Conclusion **85**
7.1 Summary 85
7.2 Future Work 86

List of Figures **87**

List of Tables **89**

List of Algorithms **91**

Bibliography **93**

CHAPTER 1

Introduction

Mobile robotics is currently a very active field of research and engineering with a broad range of applications. Besides autonomous cars [Thr10], smaller mobile robots for tasks like transportation of goods[1], guidance or in home care assistance [FEP+16] increase in popularity. A significant factor for the success of such vehicles is the ability to complete their task in populated areas, alongside humans or in collaboration. To achieve this, a robot has to keep track of people in its surroundings. This work is concerned with an approach for tracking people near the robots location incorporating detections from different sensors.

1.1 Problem Statement

An autonomous vehicle in human environments requires the vehicle to be aware of people and adjust its behavior in order to move safely around humans or interact with them [KPAK13]. This awareness can be split into two parts, namely detection and tracking. The detection itself uses algorithms to identify people in sensor data e.g. from laser range finders [AMB07] or cameras [DT05, ML12]. After the detection process, the robot is only aware that there is a person at a certain position during the time of the sensor readings. However, the robot does not know whether the detected person is the same as in previous detections independent of whether he/she moved or not, furthermore every detector gives an estimate by itself ignoring possible other detections of the same person. This is where the tracking algorithm comes into play. It merges single detections of the same person, of several detectors, over time into a track. A track is therefore the current state (e.g. position and velocity) of a person based on its previous state estimates and the current detections. Therefore, as illustrated in Figure 1.1 the tracking algorithm receives several detector outputs as an input and provides a combined estimate of a all persons in the

[1] https://www2.ffg.at/verkehr/projekte.php?id=1423

robots vicinity based on the detections and information from previous time steps. In this thesis we are mainly concerned with the tracking part assuming that the detections are provided as an input. The tracking procedure can be split into four parts:

- **Forward prediction:**
  Predict next track state using a human motion model.
- **Data association:**
  Match received detections with existing tracks.
- **Detection integration:**
  Correct the predicted state incorporating the assigned detections.
- **Track management:**
  Create new tracks, if no existing tracks match with detections. Delete tracks for which no matching detections are received.

The track itself evolves over time using a model representing the human motion. Since the robot does not know the persons intentions, it is reasonable for the motion model to be of probabilistic nature to predict his/her next steps. Typically, a probabilistic filter is used to incorporate *forward prediction* and *detection integration*. In addition, applying sensor fusion, detections of multiple sensors can be integrated enabling the tracker to compensate for faulty detections from single detectors providing a more reliable result. Since fields of view differ for each sensor, sensor fusion can also enable the robot to observe a larger portion of the environment. For *data association*, one has to define a metric e.g. nearest neighbor data association as applied in [BH10], matching detections and tracks. On that basis, it can then be decided which forward predicted tracks should be corrected by which detection and furthermore, whether new tracks should be created or existing ones should be deleted by the *track management*.

1.2 Aim of the Work

This work proposes a novel approach to people tracking on a mobile robot, incorporating an advanced human motion model. In contrast to previously proposed people tracking implementations using a simple constant velocity model, or similar, for human motion (see Section 3.3), an advanced map based human motion model incorporating statistical data is developed and used for forward prediction of tracks. The model uses a grid map to accumulate previously observed human detections in a histogram. From this, we get probabilities of where the person will likely be in the next step. Since this type of model does not necessarily lead to Gaussian distributions, nor linear motion, Kalman filtering is only applicable through linearization, which leads to the choice of particle filtering in this work. We intend to show that this type of model improves forward prediction accuracy and hence also improves the overall tracking performance. To compare our approach to existing ones, the tracker is implemented C++. Additionally, it is wrapped into a node

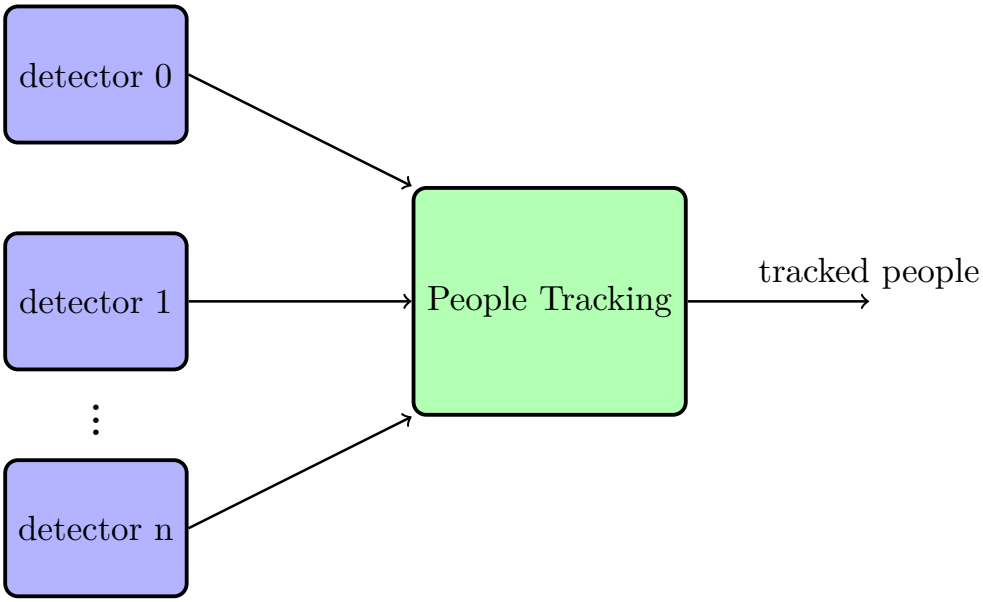


Figure 1.1: The people tracker integrates several detection algorithms to give a good estimate of a person's current position and velocity in the environment.

for the Robot Operating System (ROS)[2] such that it can easily use detections provided by other ROS nodes.

1.3 Methodological Approach

In a first step a literature review is deducted, discussing state of the art approaches to people tracking and corresponding people detection approaches. Furthermore, an approach to people tracking is developed and implemented. Finally, a comparison between different motion models is conducted. We also assess tracking performance compared to the existing SPENCER people tracking framework [LBLA16] using the CLEAR MOT metrics [BS08].

- **Review of used algorithms and state of the art**
  First, we review state of the art approaches to people tracking including current detection techniques and their limitations. Furthermore, we discuss the theoretical foundations of the data association and filtering methods adopted in the thesis.

- **Creation of a human motion model based on statistical maps**
  Using recorded data of people, we create grid maps containing values of how many people have been observed at a certain location, in the following referred to as heat

[2] `http://www.ros.org`

maps. From those maps we predict a person's movement based on the surroundings in the local neighborhood of his / her current position.

- **Implementation of the people tracker using particle filters**
  The tracker implementation is split into three parts:
  - **Filtering approach for track representation:** The tracks themselves are implemented as particle filters using integrating detections assigned to them by the data association and the human motion model to forward predict particle movement representing a possible human position and velocity.
  - **Data association:** In literature, several data association approaches are proposed. We implement a rather simple nearest neighbor strategy to test our overall implementation. In [LGA15] the authors show that nearest neighbor data association can compete with more sophisticated approaches while being less resource hungry.
  - **Track Management:** Furthermore, we implement a strategy for creating new tracks based on received detections that do not correspond to any of the current tracks and deleting obsolete tracks for which no corresponding detection is received.
- **Comparison of forward prediction for different human motion models**
  To compare the performance of the proposed human motion model, to other typically used models we choose specific map scenarios and record the particle distributions of the forward prediction without incorporating any measurements.
- **Comparison of tracking performance**
  We test the overall tracking performance of our approach utilizing the CLEAR MOT metrics [BS08]. For comparison we use a prerecorded dataset and apply our tracking approach with different motion models in use. The CLEAR MOT evaluation enables comparison of the overall tracking performance with existing work as proposed in papers [LBLA16] and [ML12].

1.4 Structure of the Work

The thesis is structured as follows. In Chapter 2 we discuss state of the art approaches to people tracking, including an overview of commonly used detection algorithms. Afterwards, Chapter 3 discusses basic concepts later used in the introduced people tracking approach. The concepts discussed include basics in probability theory, Bayesian filtering, especially particle filtering, and commonly used motion models in people tracking. Furthermore, we discuss the assignment problem solved by the Kuhn Munkres algorithm [Kuh55] [Mun57] which is later used for data association. In addition, we introduce the CLEAR MOT metrics [BS08] which are used to evaluate tracking performance. In Chapter 4 we propose our approach to people tracking, including the construction of the heat maps for the proposed motion model. Chapter 5 gives an overview of the mobile

robot used to conduct our experiments and furthermore provides an overview of the ROS packages implemented. The results for the motion model comparison and tracking performance comparison conducted are presented and discussed in Chapter 6. Finally, Chapter 7 concludes the thesis with a summary and an outlook on future work.

CHAPTER 2

State of the Art

Several forms of people tracking can be distinguished. For example one might be interested in tracking a single person or multiple persons. In addition, one can track specific movement patterns in 3D for gesture recognition or simply track 2D locations of people on the ground plane. Furthermore, it can be necessary to explicitly identify a person in scenarios in which the robot has to follow a specific person or interact with him / her. Here, we consider people tracking where it is not necessary to identify a specific person or movement pattern, but rather utilizing detections which provide the location of people we keep track of their trajectories in 2D on the ground plane. Additionally the state of the art review is restricted to tracking approaches based on Bayesian filtering.

Even though detection and tracking are not completely independent, we focus on state of the art people tracking approaches which do not directly depend on the detection methods used. Nevertheless, we give a brief review of detection approaches commonly used in people tracking.

2.1 People Detection

People detection can be categorized based on the underlying sensor used. Commonly used are 2D laser range finders, since they are already used for self localization, RGB cameras or cameras with additional depth information including RGB-D cameras, e.g. the Microsoft Kinect, or stereo cameras. In the following we will review state of the art approaches to people detection based on those sensors. Furthermore, we discuss positive and negative attributes of the particular approaches.

2.1.1 Laser Range Finder

Laser range finders obtain distance measurements using triangulation or time of flight principles. Typically the output of a laser range finder is a 2D scan of the environment

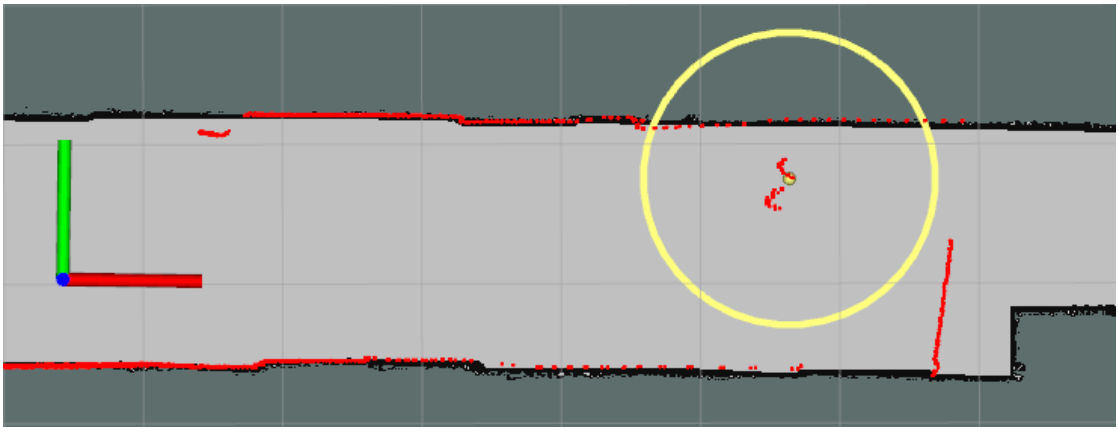


Figure 2.1: This Figure shows a detected person using a leg detector based on [AMB07]. In red one can see the laser scan. The yellow dot marks a detected person with the 99% covariance matrix visualized by the circle.

parallel to the ground plane, obtaining distance measurements within a field of view of about 270°. Their accuracy and range are strong arguments for laser range finders. Also lasers are more robust towards ambient conditions than cameras. Due to their 2D scan, typically obtained at a small distance from the ground plane, lasers are commonly used for detecting legs. An example visualization of legs detected in a scan is given in Figure 2.1. Several publications [FHM02], [KLF+02], [SMC04], [TC05] detect legs by identifying blobs of adjacent measurement beams. Those blobs are then classified as legs depending on their shape and size. Arras et al. [AMB07] use a similar approach, but apply machine learning techniques to train a classifier based on the features extracted from the laser scan. To detect legs they first segment the scan through jump distances, i.e. a set of beams is split if two adjacent beams are further away than a threshold. To decide whether a set of beams corresponds to a leg, a boosted classifier of 14 single feature classifiers is used. Examples of the features used are the radius of the cycle fitted into the segment or the standard deviation of the points in the segment. Laser based leg detection has several advantages. Lasers provide a wide field of view and both near and far measurements. Typically laser based leg detectors provide a high detection rate with the drawback of many false positives in environments with cylindric objects, such as tables or chairs which can incorrectly be classified as legs.

2.1.2 RGB Cameras

To detect people in RGB images a histogram of oriented gradients (HOG) detector proposed by Dalal and Triggs [DT05] is widely used. The basic idea is to divide the image into small regions and accumulate a histogram of intensity gradient orientations. Using a normalized version of the histograms in a larger spatial block serves as the descriptor. Then a window size is defined and slid over the image while computing descriptors and concatenating them into one feature vector. A SVM (Support Vector

Figure 2.2: This Figure shows a person detected in an image by a deep learning based pedestrian detector.

Machine) classifier is used to decide whether the feature vector corresponds to a person or not. The HOG detector reached a miss rate of 0.1 in their own INRIA dataset at 10^{-4} false positives per window. Since the HOG detector computes features for the whole human body, its detection rate decreases with occlusion scenarios like close range detection. Due to the sliding window approach the HOG approach is computationally expensive. Hence, practical implementations typically utilize a GPU. Prisacariu and Reid present a GPU implementation, using the NVIDIA CUDA framework, called fastHOG in [PR09] with the goal to be as close as possible to the sequential version in [DT05], however increasing speed through parallelization. Hence, they assume that miss rate and false positives are similar to the original algorithm in [DT05]. In [SL11] another GPU based implementation is proposed, performing similarly on the INRIA dataset from [DT05]. Additionally the authors propose geometric constraints to the image space based on ground plane estimation. This way regions of interest can be identified leading to significant speed up of the HOG detector.

Another approach to people detection in RGB images is the use of neural networks. An

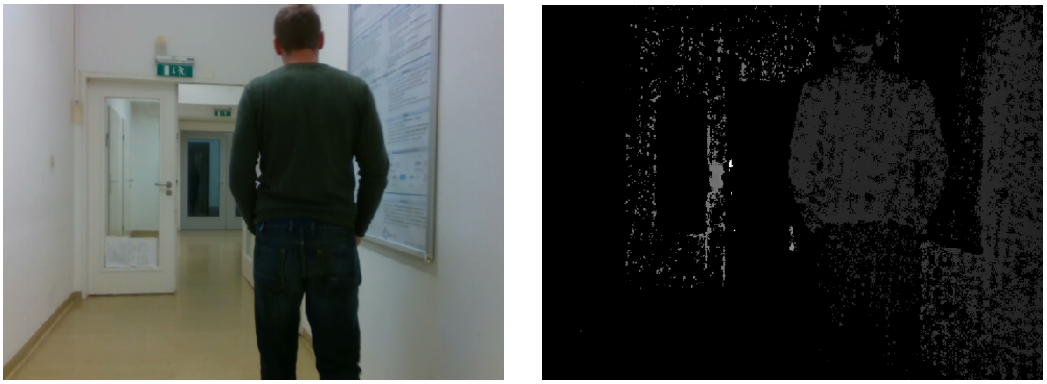


Figure 2.3: This Figure shows shows a color image on the left with its corresponding depth map on the right obtained using an Intel RealSense ZR300 camera. In the depth image the closer an object is to the camera, the brighter it appears.

example detection is shown in Figure 2.2 visualizing the detected bounding box. Recent developments in object detection neural networks enable real-time object detections [RDGF15], [RF16] turning them into a considerable option for people detection on mobile robots. Of course neural networks solutions also depend on the availability of a modern GPU like HOG detectors.

2.1.3 Depth Cameras

Figure 2.3 shows an example for RGB-D data providing an RGB color image as well as the corresponding depth data, which is typically visualized as a gray scale image. In the image closer objects appear brighter. RGB-D cameras are quite limited in range, as one can also spot in the image since a lot of data points appear black. However, the depth data obtained is rather accurate.

Spinello et al. [SA11] propose an approach utilizing depth orientation from RGB-D data for a so called histogram of oriented depths (HOD), similar to the HOG detector which works with intensity gradient orientation. The computation pipeline is equivalent to the HOG detector, except for using oriented depth gradients. Additionally, they propose a hybrid detector fusing RGB based HOG detections and their depth based HOD detector. This approach is more robust towards illumination due to the depth data. In their experiments HOD and the combined detector both outperform the traditional HOG detector. Munaro et al. propose another RGB-D detection scheme in [MBM12]. They remove the ground plane from the point cloud data received from the RGB-D camera and cluster the remaining points. Those clusters are then filtered based on the result of a HOG detector applied to the RGB image. The remaining clusters then correspond to detected persons.

In [ML12] and [JML14] a depth based upper-body detector is proposed intended for close

range detection. They utilize a learned depth template based on annotated upper body examples. Using a sliding window approach, a similarity measure between the window and the template is established. On that basis it can be decided whether their is a human present in the current region. To increase performance they do not consider the whole image but identify regions of interest by projecting the depth points onto the ground plane and identifying connected components. There evaluation gives a recall of 0.8 with 0.4 false positives per image at a range of up to 7 meters. Lower range results in higher recall, whereas it decreases towards higher ranges.

2.2 People Tracking

Many approaches to people tracking based on Bayesian filtering for movement prediction and measurement integration use concepts originating in radar target tracking. Basic techniques can be found in [BSLK04]. Also the survey series by Li and Jilkov provide valuable information on target tracking, for example describing common motion models [LJ03] and observation models [LJ01] used in target tracking which also occur in work on people tracking.

In [BH10], Bellotto and Hu give a comparison of filtering techniques for people tracking with a mobile robot. The tracker gets detection measurements from a laser based leg detector and a face recognition algorithm for RGB images. For the prediction a constant velocity model with zero-mean, Gaussian noise is used. To keep computational complexity low, they employ nearest-neighbor data association depending on the expected measurement of a track. New tracks are generated for three consecutive measurements on a possible human path which do not match with any existing track. If a track is not updated by measurements for a certain time, or its covariance reaches a certain limit it is deleted. Their results show that given their detection models and the constant velocity motion model an unscented Kalman filter (UKF) performs similarly well as the particle filter. In [DBJH15], a similar approach is implemented with the extension of nearest-neighbor joint probability data association (NNJPDA). The NNJPDA [BSDH09] is an approach to compute an estimate of the probability that a certain measurement corresponds to a certain object under the condition that the remaining measurements are assigned to specific other objects, assuming that the number of objects is known. The NNJPDA gives more reliable results under the cost of computational efficiency. The authors of [SBFC03] propose to use particle filters instead of Kalman filters in a sample based approach, arguing that particle filters provide better density estimates, since they are not restricted to Gaussian distributions. As a motion model they assume that a person adjusts his/her velocity and direction based on Gaussian distributions. To determine the number of tracks, the maximum likelihood given the number of measurements is estimated based on a learned distribution during test runs. Arras et al. [AGLB08] apply multi-hypothesis tracking [Rei78], [CH96] (MHT). Analogously to other implementations, they employ Kalman filters and a constant velocity human motion model. The difference lies in the multi-hypothesis data association strategy which also takes care of track

creation and deletion. As implied by the name, each track is represented by several possible hypotheses matching measurements to tracks, obviously leading to higher complexity depending on the number of hypotheses kept per track. In the European research project SPENCER, Linder et al. introduced a people tracking framework [LBLA16] for ROS [Fou17] implementing and comparing several tracking approaches. They compared two nearest-neighbor tracking approaches [DBJH15] [LGA15] and a multi-hypothesis tracker [AGLB08]. Their findings suggest that using a more simple data association technique in nearest-neighbor data association with advanced logic for track creation and deletion is not only computationally inexpensive but also performs best in most scenarios. The authors of [MBM12] also rely on nearest neighbor data association based on the implementations of Munkres algorithm in [KUS03]. For the cost of assigning a detection to a track they do not simply rely on Euclidean or Mahalanobis distance, but additionally utilize an online color classifier for every track and the likelihood obtained from a detector. The color classifier helps to distinguish people standing next to each other and to separate a person from the background clutter. To perform forward prediction and measurement integration an unscented Kalman filter is applied with a constant velocity motion model.

The previous people tracking approaches have in common that they all rely on a constant velocity motion model for forward prediction. The assumption of constant velocity is often violated by humans which for example cross into a room or quickly adjust their direction to avoid bumping into one another. In [OTWD04] Oh et al. suggest to use map based priors in Bayesian filtering to utilize known locations in a map where humans are more likely to move towards. Their results show that such priors improve tracking performance if sensor readings are unreliable at times. Similarly graphs generated from maps modeling the environment are used in [LFH+03] and [AM15]. Another approach is to learn human motion as Hidden Markov Models (HMM) from observed trajectories using expectation maximization (EM) as done by Bennewitz et al. in [BBCT05] or recurrent neural networks [AGR+16]. Influence of the environment can also be modeled by forces pushing humans away from each other or from static objects. In [AS08] the authors propose to use different kinds of force fields based on static obstacles, more frequent paths and crowd behavior around an individual. Luber et al. [LSTA10] also consider forces influencing the motion of people. Their work is based on a social force model [HFMV02] which incorporates both, humans and static objects in the vicinity of a person.

2.3 Summary

In this Chapter we discussed state of the art approaches to both people detection and people tracking. A lot of work has been done in order to detect people using different sensors and algorithms. Considering the drawbacks each sensor inherits to the detection algorithms, it makes sense to combine one approach from each category such that they compensate each other.

In people tracking we saw that most approaches [AGLB08], [MBM12], [LGA15], [DBJH15] focus on Kalman filter, respectively nonlinear extensions, based state estimation for people tracking utilizing a constant velocity motion model. However, results of [SBFC03] suggest that particle filters provide more accurate posterior density estimation. Furthermore, works on more advanced motion models such as [OTWD04] or [LSTA10] encourage the use of improved motion priors to improve tracking performance. Hence, the approach in this thesis focuses on particle filtering and proposes a novel motion model based on maps and historical data, while incorporating several detectors to compensate for their individual weaknesses.

In the following Chapter, theoretical background knowledge needed for the introduced people tracking approach, as well as the notation used throughout the thesis will be established.

CHAPTER 3

Background

In this chapter we discuss general concepts underlying the introduced approach to people tracking. We consider essential material in probability theory in Section 3.1 to provide common terminology. In order to later introduce a probabilistic model for people tracking we cover Bayesian filtering and specifically particle filtering in Section 3.2. Furthermore, typically used models for human motion, without knowing the persons intentions are provided in Section 3.3 which we will use for comparison with the model later introduced in this thesis. In order to apply nearest neighbor data association in the proposed people tracking algorithm and receive the best possible assignment one needs to solve the assignment problem which is described in Section 3.4 featuring the Kuhn Munkres algorithm as a solution. Finally, the CLEAR multi object tracking (MOT) metrics [BS08] are introduced in Section 3.5 which are used to evaluate the overall tracking performance of the proposed approach.

3.1 Probability Theory

In this section basic concepts in probability theory needed for the notion of Bayesian filtering are introduced. Since notation differs throughout literature, it is convenient to define common terminology and equations used in the rest of the thesis. The following is based on [HM14] and [Gri17].

First we define random variables. A random variable is defined as a function assigning real values to outcomes of a random experiment, i.e. $\mathsf{x} : \Omega \mapsto \mathcal{R}$ where Ω is the set off all possible outcomes. The set Ω is also called the certain event, since the outcome of the experiment is always an element of Ω. The probability of an event $\mathcal{A} \subseteq \Omega$ is then defined by the Kolmogorov axioms:

1. The probability of an event is always non-negative $\mathrm{P}\{\mathcal{A}\} \geq 0$

2. The probability of the certain event is one: $\mathrm{P}\{\Omega\} = 1$
3. If $\mathcal{A}_i$ for $i \in \mathbb{N}$ are mutually exclusive events, i.e. disjoint sets $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset,\ i \neq j$ then $\mathrm{P}\{\bigcup_{i\in\mathbb{N}} \mathcal{A}_i\} = \sum_{i\in\mathbb{N}} \mathrm{P}\{\mathcal{A}_i\}$

Subsequently, we will denote random vectors in bold font containing several scalar random variables $\mathbf{x} = [\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_N]$. Furthermore, deterministic vectors and variables are denoted as $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]^\intercal$. In the following we will focus on the more general case of random vectors, since we will work with vectors and matrices. Now we can define the cumulative distribution function (cdf) for a vector of random variables:

$$F_{\mathbf{x}}(\boldsymbol{x}) = \mathrm{P}\{\bigcap_{i=1}^{N} (\mathsf{x}_i \leq x_i)\} \tag{3.1}$$

Cumulative distribution functions are always monotonically increasing in any argument and bounded by the range $[0, 1]$. The probability density function (pdf) of a random vector for differentiable cdfs is then defined as:

$$f_{\mathbf{x}}(\boldsymbol{x}) = \frac{d}{d\boldsymbol{x}} F_{\mathbf{x}}(\boldsymbol{x}) = \frac{\partial}{\partial x_1}\frac{\partial}{\partial x_2}\ldots\frac{\partial}{\partial x_N} F_{\mathbf{x}}(\boldsymbol{x}) \tag{3.2}$$

The pdf of a random vector is always nonnegative. Furthermore, its integral over the whole N-dimensional space $\mathbb{R}^N$ is one.

$$\int_{\mathbb{R}^N} f_{\mathbf{x}}(\boldsymbol{x}) d\boldsymbol{x} = 1 \tag{3.3}$$

Using the pdf one can define the expectation operator for one random variable, which gives the mean or expected value of the variable, and consequently a vector of random variables as follows:

$$\mu_{\mathsf{x}} = \mathrm{E}\{\mathsf{x}\} = \int_{-\infty}^{\infty} x f_{\mathsf{x}}(x) dx \tag{3.4}$$

$$\boldsymbol{\mu}_{\mathbf{x}} = \mathrm{E}\{\mathbf{x}\} = [\mathrm{E}\{\mathsf{x}_1\}, \mathrm{E}\{\mathsf{x}_2\}, \ldots, \mathrm{E}\{\mathsf{x}_N\}]^\intercal \tag{3.5}$$

$$\boldsymbol{\mu}_{\mathbf{x}} = \mathrm{E}\{\mathbf{x}\} = \int_{-\infty}^{\infty} \boldsymbol{x} f_{\mathbf{x}}(\boldsymbol{x}) d\boldsymbol{x} \tag{3.6}$$

The expectation operator is a linear operator, i.e. $\mathrm{E}\{a\mathbf{x} + b\mathbf{y}\} = a\mathrm{E}\{\mathbf{x}\} + b\mathrm{E}\{\mathbf{y}\}$, which immediately follows from the linearity of integration. Besides the mean we additionally

define the $N \times N$ covariance matrix $\mathbf{C}_{\mathsf{x}}$ as an important measure of random vectors. The entries of the covariance matrix are given by the pairwise covariance of the random variables in the vector $C_{\mathsf{x_i},\mathsf{x_j}}$. In the special case where $i = j$ the entries correspondent to the variance $\sigma^2_{\mathsf{x_i}}$. Here $f_{\mathsf{x_i}\mathsf{x_j}}(x_i, x_j)$ is the marginal pdf obtained by integrating over the subset containing all random variables except $\mathsf{x_i}$ and $\mathsf{x_j}$.

$$\begin{aligned} C_{\mathsf{x_i},\mathsf{x_j}} &= \mathrm{E}\{(\mathsf{x_i} - \mu_{\mathsf{x_i}})(\mathsf{x_j} - \mu_{\mathsf{x_j}})\} && (3.7)\\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x_i - \mu_{\mathsf{x_i}})(x_j - \mu_{\mathsf{x_j}}) f_{\mathsf{x_i}\mathsf{x_j}}(x_i x_j)dx_i dx_j \\ \mathbf{C}_{\mathsf{x}} &= \mathrm{E}\{(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^{\mathsf{T}}\} && (3.8)\\ &= \int_{\mathbb{R}^N}(\boldsymbol{x} - \boldsymbol{\mu}_{\mathbf{x}})(\boldsymbol{x} - \boldsymbol{\mu}_{\mathbf{x}})^{\mathsf{T}} f_{\mathbf{x}}(\boldsymbol{x})d\boldsymbol{x} && (3.9) \end{aligned}$$

The variance of a single random variable describes how far a set of randomly drawn samples of the distribution is spread around the mean. The covariance between two distinct random variables in the vector is a measure of whether they show similar behavior. A positive covariance value means if one variable has a large value the other one does so too. Contrarily if the covariance is negative, large values of one variable imply small values of the other one.
Similarly, one can define the correlation matrix:

$$\begin{aligned} \mathbf{R}_{\mathsf{x}} &= \mathrm{E}\{\mathbf{x}\mathbf{x}^{\mathsf{T}}\} && (3.10)\\ &= \int_{\mathbb{R}^N}\boldsymbol{x}\boldsymbol{x}^{\mathsf{T}} f_{\mathbf{x}}(\boldsymbol{x})d\boldsymbol{x} && (3.11) \end{aligned}$$

The correlation matrix is related to the covariance matrix through the mean:

$$\mathbf{R}_{\mathbf{x}} = \mathbf{C}_{\mathbf{x}} + \boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^{\mathsf{T}} \tag{3.12}$$

To talk about statistical dependence lets consider two disjoint random vectors $\mathbf{x_1}$ of length N_1 and $\mathbf{x_2}$ of length N_2. If $\mathbf{x_1}$ is statistically dependent on $\mathbf{x_2}$, it is interesting to consider the probability of an event given that we already know the value of $\mathbf{x_2} = \boldsymbol{x_2}$. This leads to the notion of conditional probability. Let $\mathcal{R}_1$ be an arbitrary region in $\mathbb{R}^{N_1}$. Then the conditional probability of $\mathbf{x_1}$ being in the region $\mathcal{R}_1$ given $\mathbf{x_2} = \boldsymbol{x_2}$, under the condition that $f_{\mathbf{x_2}}(\boldsymbol{x_2}) > 0$, is:

$$\mathrm{P}\{\mathbf{x_1} \in \mathcal{R}_1 | \mathbf{x_2} = \boldsymbol{x_2}\} = \frac{\int_{\mathcal{R}_1} f_{\mathbf{x_1},\mathbf{x_2}}(\boldsymbol{x_1}, \boldsymbol{x_2})d\boldsymbol{x_1}}{f_{\mathbf{x_2}}(\boldsymbol{x_2})} \tag{3.13}$$

This result can be obtained using a limit argument where $\boldsymbol{x_2} \leq \mathbf{x_2} \leq (\boldsymbol{x_2} + \boldsymbol{\Delta x_2})$ and $\boldsymbol{\Delta x_2} \to 0$. Here $f_{\mathbf{x_1},\mathbf{x_2}}(\boldsymbol{x_1}, \boldsymbol{x_2})$ is the joint pdf of $\mathbf{x_1}$ and $\mathbf{x_2}$, i.e. $f_{\mathbf{x}}(\boldsymbol{x})$ where $\mathbf{x}$ is a random vector containing both $\mathbf{x_1}$ and $\mathbf{x_2}$. From (3.13) we get the conditional pdf of $\mathbf{x_1}$ given $\mathbf{x_2}$.

$$f_{\mathbf{x_1}|\mathbf{x_2}}(\boldsymbol{x_1}|\boldsymbol{x_2}) = \frac{f_{\mathbf{x_1},\mathbf{x_2}}(\boldsymbol{x_1}, \boldsymbol{x_2})}{f_{\mathbf{x_2}}(\boldsymbol{x_2})} \tag{3.14}$$

The conditional pdf for $\mathbf{x_2}$ is analogously obtained as:

$$f_{\mathbf{x_2}|\mathbf{x_1}}(\boldsymbol{x_2}|\boldsymbol{x_1}) = \frac{f_{\mathbf{x_1},\mathbf{x_2}}(\boldsymbol{x_1},\boldsymbol{x_2})}{f_{\mathbf{x_1}}(\boldsymbol{x_1})} \tag{3.15}$$

From equations (3.14) and (3.15) it follows that:

$$f_{\mathbf{x_1},\mathbf{x_2}}(\boldsymbol{x_1},\boldsymbol{x_2}) = f_{\mathbf{x_1}|\mathbf{x_2}}(\boldsymbol{x_1}|\boldsymbol{x_2})f_{\mathbf{x_2}}(\boldsymbol{x_2}) = f_{\mathbf{x_2}|\mathbf{x_1}}(\boldsymbol{x_2}|\boldsymbol{x_1})f_{\mathbf{x_1}}(\boldsymbol{x_1}) \tag{3.16}$$

With the notion of conditional pdfs we are able to define two important theorems, namely Bayes theorem and the total probability theorem. Bayes theorem is obtained by rearranging Equation 3.16:

$$f_{\mathbf{x_1}|\mathbf{x_2}}(\boldsymbol{x_1}|\boldsymbol{x_2}) = f_{\mathbf{x_2}|\mathbf{x_1}}(\boldsymbol{x_2}|\boldsymbol{x_1})\frac{f_{\mathbf{x_1}}(\boldsymbol{x_1})}{f_{\mathbf{x_2}}(\boldsymbol{x_2})} \tag{3.17}$$

This relation is the key concept used in Bayes filtering. It allows to interchange conditioning, which is used in probabilistic inference. We discuss its application in Section 3.2. As a remark please note that Bayes rule can be conditioned on arbitrary additional random variables:

$$f_{\mathbf{x_1}|\mathbf{x_2},\mathbf{y}}(\boldsymbol{x_1}|\boldsymbol{x_2},\boldsymbol{y}) = f_{\mathbf{x_2}|\mathbf{x_1},\mathbf{y}}(\boldsymbol{x_2}|\boldsymbol{x_1},\boldsymbol{y})\frac{f_{\mathbf{x_1}|\mathbf{y}}(\boldsymbol{x_1}|\boldsymbol{y})}{f_{\mathbf{x_2}|\mathbf{y}}(\boldsymbol{x_2}|\boldsymbol{y})} \tag{3.18}$$

This can easily be shown by using the definitions of conditional pdfs. Note that since we use random vectors of arbitrary length, one can easily split a vector into two parts or add another one. Therefore, given the conditional pdfs as:

$$f_{\mathbf{x_1}|\mathbf{x_2},\mathbf{y}}(\boldsymbol{x_1}|\boldsymbol{x_2},\boldsymbol{y}) = \frac{f_{\mathbf{x_1},\mathbf{x_2},\mathbf{y}}(\boldsymbol{x_1},\boldsymbol{x_2},\boldsymbol{y})}{f_{\mathbf{x_2},\mathbf{y}}(\boldsymbol{x_2},\boldsymbol{y})} \tag{3.19}$$

$$f_{\mathbf{x_2}|\mathbf{x_1},\mathbf{y}}(\boldsymbol{x_2}|\boldsymbol{x_1},\boldsymbol{y}) = \frac{f_{\mathbf{x_1},\mathbf{x_2},\mathbf{y}}(\boldsymbol{x_1},\boldsymbol{x_2},\boldsymbol{y})}{f_{\mathbf{x_1},\mathbf{y}}(\boldsymbol{x_1},\boldsymbol{y})} \tag{3.20}$$

$$f_{\mathbf{x}_1|\mathbf{y}}(\boldsymbol{x}_1|\boldsymbol{y}) = \frac{f_{\mathbf{x}_1,\mathbf{y}}(\boldsymbol{x}_1,\boldsymbol{y})}{f_{\mathbf{y}}(\boldsymbol{y})} \tag{3.21}$$

$$f_{\mathbf{x}_2|\mathbf{y}}(\boldsymbol{x}_2|\boldsymbol{y}) = \frac{f_{\mathbf{x}_2,\mathbf{y}}(\boldsymbol{x}_2,\boldsymbol{y})}{f_{\mathbf{y}}(\boldsymbol{y})} \tag{3.22}$$

We get a similar relation to Equation (3.16):

$$f_{\mathbf{x_1},\mathbf{x_2},\mathbf{y}}(\boldsymbol{x_1},\boldsymbol{x_2},\boldsymbol{y}) = f_{\mathbf{x_1}|\mathbf{x_2},\mathbf{y}}(\boldsymbol{x_1}|\boldsymbol{x_2},\boldsymbol{y})f_{\mathbf{x_2}|\mathbf{y}}(\boldsymbol{x_2}|\boldsymbol{y}) = f_{\mathbf{x_2}|\mathbf{x_1},\mathbf{y}}(\boldsymbol{x_2}|\boldsymbol{x_1},\boldsymbol{y})f_{\mathbf{x_1}|\mathbf{y}}(\boldsymbol{x_1}|\boldsymbol{y}) \tag{3.23}$$

Leading to the more general Bayes theorem:

$$f_{\mathbf{x_1}|\mathbf{x_2},\mathbf{y}}(\boldsymbol{x_1}|\boldsymbol{x_2},\boldsymbol{y}) = f_{\mathbf{x_2}|\mathbf{x_1},\mathbf{y}}(\boldsymbol{x_2}|\boldsymbol{x_1},\boldsymbol{y})\frac{f_{\mathbf{x}_1|\mathbf{y}}(\boldsymbol{x}_1|\boldsymbol{y})}{f_{\mathbf{x}_2|\mathbf{y}}(\boldsymbol{x}_2|\boldsymbol{y})} \tag{3.24}$$

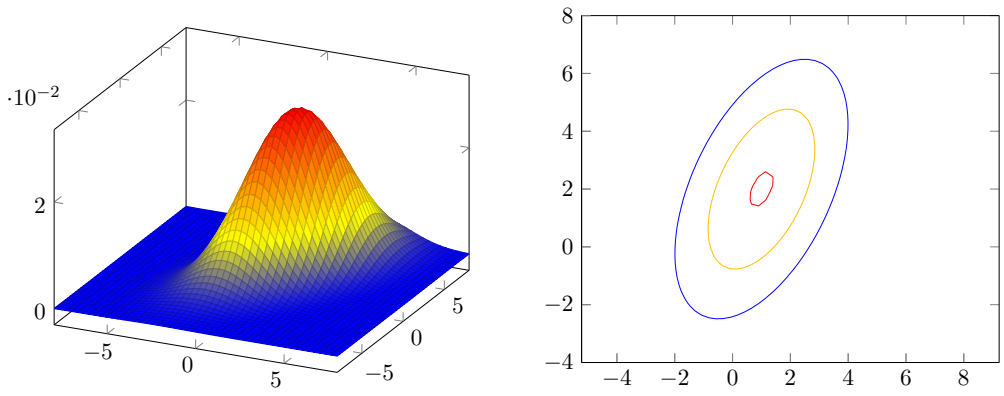


Figure 3.1: Example of 2D Gaussian pdf

The total probability theorem also directly follows from (3.16) and relates the marginal pdf to the conditional pdf stating that if you consider all possible values of the conditional random variable you get the unconditional marginal pdf.

$$f_{\mathbf{x_1}}(\boldsymbol{x_1}) = \int_{\mathcal{R}^{N_2}} f_{\mathbf{x_1}|\mathbf{x_2}}(\boldsymbol{x_1}|\boldsymbol{x_2}) f_{\mathbf{x_2}}(\boldsymbol{x_2}) d\boldsymbol{x_2} \tag{3.25}$$

In case $\mathbf{x_1}$ and $\mathbf{x_2}$ are statistically independent, the conditional pdf is equivalent to the marginal pdf:

$$f_{\mathbf{x_1}|\mathbf{x_2}}(\boldsymbol{x_1}|\boldsymbol{x_2}) = f_{\mathbf{x_1}}(\boldsymbol{x_1}) \tag{3.26}$$

$$f_{\mathbf{x_2}|\mathbf{x_1}}(\boldsymbol{x_2}|\boldsymbol{x_1}) = f_{\mathbf{x_2}}(\boldsymbol{x_2}) \tag{3.27}$$

One of the conditions in Equation (3.26) is sufficient since they imply each other. In addition, it follows from (3.16) that the joint pdf can be obtained by multiplying the marginals:

$$f_{\mathbf{x_1},\mathbf{x_2}}(\boldsymbol{x_1}, \boldsymbol{x_2}) = f_{\mathbf{x_1}}(\boldsymbol{x_1}) f_{\mathbf{x_2}}(\boldsymbol{x_2}) \tag{3.28}$$

As an example distribution of random vectors we consider the multivariate Gaussian (or normal) distribution commonly used to model noise in Bayesian filtering applications. It is defined given a mean $\boldsymbol{\mu_x}$ and a covariance matrix $\mathbf{C_x}$ and denoted as $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu_x}, \mathbf{C_x})$. The pdf is given as:

$$f_{\mathbf{x}}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^N \det(\mathbf{C_x})}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu_x})^\intercal \mathbf{C_x}^{-1}(\boldsymbol{x} - \boldsymbol{\mu_x})\right) \tag{3.29}$$

Figure 3.1 shows an example of a multivariate Gaussian pdf with $N = 2$, $\mu_{\mathrm{x}_1} = 1$, $\mu_{\mathrm{x}_2} = 2$, $\sigma^2_{\mathrm{x}_1} = 4$, $\sigma^2_{\mathrm{x}_2} = 9$ and a correlation coefficient $\rho_{\mathrm{x}_1,\mathrm{x}_2} = \frac{C_{\mathrm{x}_1,\mathrm{x}_2}}{\sigma_{\mathrm{x}_1},\sigma_{\mathrm{x}_2}} = 0.5$. One can see that the probability, defined by the volume under the curve, of the random variable $\mathbf{x}$ being around the mean is high while it gets lower towards the borders of the plot. The right

plot in Figure 3.1 shows cuts of the 3D curve which are often utilized to visualize the confidence, in the sense that one defines a fixed probability, i.e. 99% meaning $\mathbf{x}$ is inside this ellipse with a probability of 0.99, which is more spread if the confidence is low. Note that the major and minor axis of the ellipse are defined by the eigenvalues and eigenvectors of the covariance matrix.

In this Section we focused on continuous random variables, nevertheless most of the concepts introduced are also applicable to discrete random variables. The pdf of a discrete random variable can be defined using the Dirac delta function $\delta(x)$:

$$f_{\mathsf{x}}(x) = \sum_{x^{(i)} \in \mathcal{X}} p_{\mathsf{x}}(x^{(i)}) \delta(x - x^{(i)}) \tag{3.30}$$

Where $p_{\mathsf{x}}(x)$ is the probability mass function (pmf) of x for a finite or countably infinite set $\mathcal{X}$ defined as:

$$p_{\mathsf{x}}(x) = \begin{cases} \mathrm{P}\{\mathsf{x} = x^{(i)}\}, & x = x^{(i)} \\ \text{undefined}, & x \notin \mathcal{X} \end{cases} \tag{3.31}$$

In addition, we will not explicitly distinguish between discrete and continuous random variables in the following sections since conceptually it does not make a difference.

3.2 Bayesian Filtering

In this section we introduce Bayesian filtering using the probabilistic concepts from the previous Section 3.1. In general a Bayes filter estimates the conditional probability distribution $f_{\mathbf{x}_k|\mathbf{z}_{1:k}}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ of a state vector $\mathbf{x}_k$ at time k under the condition of noisy measurements from time 1 up to k $\mathbf{z}_{1:k}$, i.e. $\mathbf{z}_{1:k} = \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_k$. This state is often called belief state [TBF05]. Please note that we intentionally drop the input vector $\mathbf{u}_k$ typically involed in state estimation [TBF05], [RN16], due to unknown system inputs in object tracking. In the context of people tracking one would like to know the current position and velocity of a person at time k described by its state vector $\mathbf{x}_k$. The measurements are received in form of detections using sensor data. Since detections are not always accurate the state of a person can not be directly observed. The state evolution is modeled by a dynamic Bayesian network (DBN). A Bayesian network is a directed acyclic graph which models several random vectors, represented by the vertices, and their conditional dependencies given by the edges. If there is an edge from $\mathbf{x}$ to $\mathbf{z}$, $\mathbf{x}$ is a parent of $\mathbf{z}$. The probability distribution of a random vector is then conditioned on its parents. Considering the previous example one would get $f_{\mathbf{z}|\mathbf{x}}(\boldsymbol{z}|\boldsymbol{x})$. In the dynamic case these relationships can change over time. The DBN modeling state estimation from uncertain measurements is given in Figure 3.2. Estimation starts in the initial state $\mathbf{x}_0$ where no measurement is present, and then evolves over time. Using this model it is assumed that the random process of the state fulfills a first order Markov property. This means the current state $\mathbf{x}_k$ only depends on its immediate predecessor $\mathbf{x}_{k-1}$. The measurement $\mathbf{z}_k$

only depends on the current state $\mathbf{x}_k$. As stated before we would like to estimate the probability distribution of the current state observing the measurement:

$$f_{\mathbf{x}_k|\mathbf{z}_{1:k}}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) \tag{3.32}$$

In the following we derive a recursive way of updating our belief state (3.32) as given in [TBF05]. First Bayes theorem (3.24) is applied to calculate the belief state using the measurement pdf assuming the state is known.

$$f_{\mathbf{x}_k|\mathbf{z}_{1:k}}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) = \eta\ f_{\mathbf{z}_k|\mathbf{x}_k,\mathbf{z}_{1:k-1}}(\boldsymbol{z}_k|\boldsymbol{x}_k,\boldsymbol{z}_{1:k-1}) f_{\mathbf{x}_k|\mathbf{z}_{1:k-1}}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}) \tag{3.33}$$

$$\eta = \frac{1}{f_{\mathbf{z}_k|\mathbf{z}_{1:k-1}}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})} \tag{3.34}$$

Using that a node in a Bayesian network is conditionally independent of all other predecessors given its parents [RN16] $\mathbf{z}_k$ does not depend on any previous measurement given the current state $\mathbf{x}_k$. Therefore we further simplify equation (3.33) to:

$$f_{\mathbf{x}_k|\mathbf{z}_{1:k}}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) = \eta\ f_{\mathbf{z}_k|\mathbf{x}_k}(\boldsymbol{z}_k|\boldsymbol{x}_k) f_{\mathbf{x}_k|\mathbf{z}_{1:k-1}}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}) \tag{3.35}$$

Then we use the total probability theorem (3.25) to introduce the recursion. Furthermore, we again use conditional independence to simplify.

$$f_{\mathbf{x}_k|\mathbf{z}_{1:k}}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) = \eta\ f_{\mathbf{z}_k|\mathbf{x}_k}(\boldsymbol{z}_k|\boldsymbol{x}_k) \int_{\mathbb{R}^N} f_{\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{z}_{1:k-1}}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1},\boldsymbol{z}_{1:k-1}) \tag{3.36}$$
$$f_{\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}}(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{k-1}) d\boldsymbol{x}_{k-1}$$

$$= \eta\ f_{\mathbf{z}_k|\mathbf{x}_k}(\boldsymbol{z}_k|\boldsymbol{x}_k) \int_{\mathbb{R}^N} f_{\mathbf{x}_k|\mathbf{x}_{k-1}}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \tag{3.37}$$
$$f_{\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}}(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{k-1}) d\boldsymbol{x}_{k-1}$$

Equation (3.37) now defines the recursive update incorporating the previous belief state $f_{\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}}(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{k-1})$ and the measurement probability $f_{\mathbf{z}_k|\mathbf{x}_k}(\boldsymbol{z}_k|\boldsymbol{x}_k)$. This recursion is commonly split into two steps, namely a prediction step, forward predicting the state and a correction step including the new measurement. Hence, the two steps are given as:

$$\overline{bel}(\mathbf{x}_k) = \int_{\mathbb{R}^N} f_{\mathbf{x}_k|\mathbf{x}_{k-1}}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) bel(\mathbf{x}_{k-1}) d\boldsymbol{x}_{k-1} \tag{3.38}$$

$$bel(\mathbf{x}_k) = \eta\ f_{\mathbf{z}_k|\mathbf{x}_k}(\boldsymbol{z}_k|\boldsymbol{x}_k) \overline{bel}(\mathbf{x}_k) \tag{3.39}$$

3.2.1 Particle Filter

In this section we discuss a sampling based approximation for Bayesian filtering called particle filtering or sequential Monte Carlo [DdFG01]. Due to the sample based representation, particle filters are able to estimate nonlinear and non Gaussian posterior

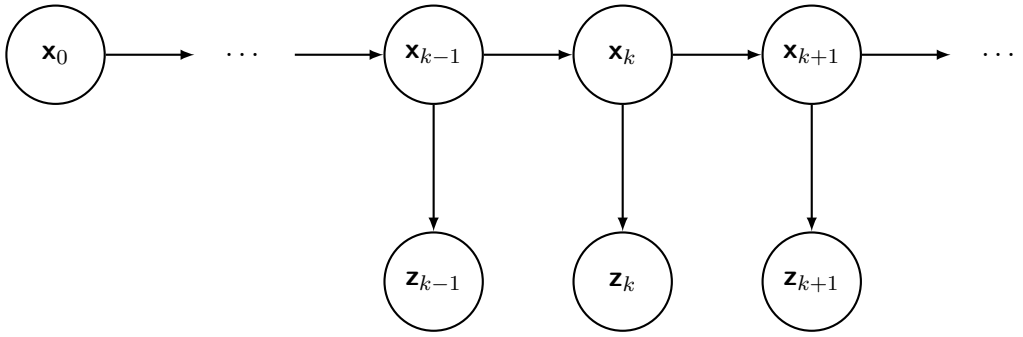


Figure 3.2: Dynamic Bayesian Network for Filtering

distributions in contrast to Kalman filters. However, depending on the complexity of the posterior a large number of samples can be required, which results in high computational complexity. Not relying on Gaussian distributions enables the key property to represent multi-modal distributions, i.e. probability distributions with several maxima. In the context of people tracking this means that the particle filter can represent belief states giving locations of several people at the same time. Similarly, if a previously tracked person is occluded or leaves the field of view of the robot, the multi-modal distribution can represent several likely locations for that particular person.

The posterior is approximated by a set of weighted samples called particles, where each one represents a possible state configuration. For an increasing number of particles, the approximation gets closer to the actual distribution. By the nature of this approximation we work with discrete time probability distributions in form of their probability mass function (pmf). For simplicity we still denote the pmf of x by $f_{\mathsf{x}}(x)$ as in the continuous case for the pdf. A formal argument for why the particle filter actually approximates the belief state $bel(\mathbf{x}_k)$ is given in [RN16] and its mathematical derivation in [Thr02].

In the following we describe the particle filter algorithm as depicted in [Thr02], referred to as Bootstrap filter or recursive sampling importance resampling (SIR) in literature [DdFG01] [Ber99] [AMGC02]. First we define the set of N particles at time step k as $\mathcal{X}_k = \{\boldsymbol{x}_k^0, \boldsymbol{x}_k^1, \dots \boldsymbol{x}_k^{N-1}\}$ where the superscript denotes the index of the particles ranging from 0 to $N-1$. Algorithm 3.1 shows the pseudo code as described in [Thr02]. Please note that we do not consider any system input here, since it is typically not available in tracking applications. Before actually applying Algorithm 3.1 one has to first draw a set of particles from the initial distribution. The initial distribution can be chosen as uniform over the whole state space if there is no knowledge beforehand. Since in people tracking, filters typically get initialized from a detection, one can choose a distribution centered around that detection e.g. a Gaussian distribution. Starting from the initial set of particles, Algorithm 3.1 can be applied at each time step. First, the algorithm iterates through the set of all particles from the previous time step and samples new ones from the state transition probability distribution (line 3). This corresponds to the

Algorithm 3.1: General Particle Filter Algorithm as in [Thr02]

Input: $\mathcal{X}_{k-1}$, $\boldsymbol{z}_k$
Output: $\mathcal{X}_k$
1 $\bar{\mathcal{X}}_k = \mathcal{X}_k = \emptyset$;
2 **for** $n = 0$ *to* $N-1$ **do**
3 | sample $\boldsymbol{x}_k^n \sim f_{\mathbf{x}_k|\mathbf{x}_{k-1}}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^n)$;
4 | $w_k^n = f_{\mathbf{z}_k|\mathbf{x}_k}(\boldsymbol{z}_k|\boldsymbol{x}_k^n)$;
5 | $\bar{\mathcal{X}}_k = \bar{\mathcal{X}}_k \cup (\boldsymbol{x}_k^n, w_k^n)$;
6 **end**
7 **for** $n = 0$ *to* $N-1$ **do**
8 | draw $\boldsymbol{x}_k^i$ with probability $\propto w_k^i$ from $\bar{\mathcal{X}}_k$;
9 | $\mathcal{X}_k = \mathcal{X}_k \cup \boldsymbol{x}_k^i$;
10 **end**
11 **return** $\mathcal{X}_k$

prediction step in the Bayes filter leading to $\overline{bel}(\mathbf{x}_k)$. Furthermore, the loop calculates weights for each particle drawn based on the conditional pmf of the measurement given the predicted particle $\boldsymbol{x}_k^n$ (line 4). The weighting essential for the particle filter to work, applying the principle of importance sampling. Importance sampling enables one to sample from a distribution, from which it is easy to draw samples, called the proposal distribution and still approximating a different target distribution by accounting for the difference through the weights. In the case of the particle filter one samples from the state transition probability distribution typically modeled by a motion model and accounts for the prediction error through the likelihood of a measurement given the predicted state. Now each pair of particle and weight is added to the temporary set $\bar{\mathcal{X}}_k$ (line 5). In the second for-loop the calculation of $bel(\mathbf{x}_k)$ takes place, actually incorporating the observed measurements. The loop draws N particles from the set $\bar{\mathcal{X}}_k$ with a probability according to their weight (line 8) and adds them to the final set of particles $\mathcal{X}_k$ (line 9). This step is called resampling, which makes sure that state configurations with higher probability are represented by more particles. It avoids degeneracy of the particle set, i.e. containing a single particle with high weight and a lot of particles with low weight not contributing much to the posterior approximation.

There are several issues with the Bootstrap particle filter algorithm mostly caused by the low number of particles used in a feasible implementation, that have to be addressed. First, due to sampling variance the distribution represented by the particles can largely differ from the true distribution. This difference obviously lessens as the number of samples increases. Here, the previously mentioned trade-off between an accurate representation of the belief state and computational power comes into play to get a better estimate. Another issue arises from the resampling step. Resampling reduces diversity in particles by design to avoid degeneracy of the particle set. However, this loss in diversity

introduces the problem of sample impoverishment [AMGC02]. Sample impoverishment is most present in the case of low noise in forward prediction. In that case the particle set degenerates into a set of equivalent particles after a few iterations which means the whole distribution is represented by a single particle. Two common counter measures as stated in [Thr02] are first to only resample if necessary, i.e. if the variance of the particle set is low, resampling should not be applied and second to apply low variance resampling. The low variance sampling algorithm as described in [Thr02] is stated in

Algorithm 3.2: Low Variance Sampling Algorithm as in [Thr02]

Input: $\mathcal{X}_k$, $\mathcal{W}_k$
Output: $\bar{\mathcal{X}}_k$

1 $\bar{\mathcal{X}}_k = \emptyset$;
2 $r = \text{rand}(0; N^{-1})$;
3 $c = w_k^0$;
4 $i = 0$;
5 **for** $n = 0$ *to* $N - 1$ **do**
6 $\quad U = r + n * N^{-1}$;
7 $\quad$ **while** $U > c$ **do**
8 $\quad\quad i = i + 1$;
9 $\quad\quad c = c + w_k^i$;
10 $\quad$ **end**
11 $\quad \bar{\mathcal{X}}_k = \bar{\mathcal{X}}_k \cup x_k^i$;
12 **end**
13 **return** $\bar{\mathcal{X}}_k$

Algorithm 3.2. The idea is to not randomly draw each particle proportionally to its weight by itself, but rather draw all particles at once based on a single random number r drawn uniform from the interval $[0, N^{-1}]$. To do this, the algorithm repeatedly adds N^{-1} to the initial random number (line 6) calculating U. Through the while loop (lines 7 to 10) the algorithm calculates the index of the next particle drawn by comparing whether U exceeds the current weight sum in c. This makes sure that particles are drawn proportional to their weight. The key advantage of the low variance sampling is that if all weights are equal, the returned set $\bar{\mathcal{X}}_k$ is equal to the initial set of particles $\mathcal{X}_k$. Furthermore, the low variance sampling algorithm has an upper bound complexity of $O(N)$ instead of $O(N \log N)$ of individually drawing each particle on its own, which requires drawing N random numbers and a search for the corresponding particle.

3.3 Motion Models for People Tracking

Applying Bayesian filtering to people tracking requires a motion model to predict future steps of the tracked target. In case of particle filtering we utilize the motion models as prior distribution to sample from. In this section we introduce commonly used motion

models applied in people tracking. In contrast to state estimation in robot localization, the models used naturally do not have a control input. The following sections are based on the description in [LJ03] which is a survey of motion models used in target tracking and the book of Bar-Shalom et al. [BSLK04].

Since the tracking approach taken in this thesis assumes that people move on the floor, i.e. $z = 0$, we define the state of a person representing their 2D position and velocity as $\boldsymbol{x} = [x, y, \dot{x}, \dot{y}]^\intercal$. Please also note that we neglect the shape of a person and only track his/her center point. Even though our tracking approach using particle filters is not restricted to linear models, the commonly used continuous models presented in this section are in fact linear. Hence, we define the general continuous linear, time-invariant (LTI) state model of a person as:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{w}(t), \quad \boldsymbol{x}(t_0) = \boldsymbol{x_0} \tag{3.40}$$

With the state vector at time t $\boldsymbol{x}(t)$, the possibly time variant state transition matrix $\boldsymbol{A}(t)$, the additive noise sequence $\boldsymbol{w}(t) = [w_x, w_y]^\intercal$ and the corresponding gain matrix and $\boldsymbol{B}(t)$. The noise $\boldsymbol{w}(t)$ is typically modeled as zero mean Gaussian with covariance $\boldsymbol{Q}(t)$. Such a linear continuous systems has the following solution:

$$\boldsymbol{x}(t) = \boldsymbol{F}(t - t_0)\boldsymbol{x}_0 + \int_{t_0}^{t} \boldsymbol{F}(t - \tau) + \boldsymbol{B}\boldsymbol{w}(\tau)\mathrm{d}\tau, \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{3.41}$$

$$\boldsymbol{F}(t) = \left(\sum_{i=0}^{\infty} \boldsymbol{A}^i \frac{t^i}{i!} \right) \boldsymbol{x}_0 = \exp(\boldsymbol{A}t) \tag{3.42}$$

This result can be easily proven using Picard's method of successive approximation for the homogeneous part of the linear ordinary differential equation and variation of parameters for the inhomogeneous part.

To use the model in an implementation it has to be discretized to the following general form:

$$\boldsymbol{x}_{k+1} = \boldsymbol{F}_k\boldsymbol{x}_k + \boldsymbol{w}_k \tag{3.43}$$

Where $\boldsymbol{x}_{k+1}$ and $\boldsymbol{x}_k$ represent the state at discrete time $k + 1$ and k respectively. The matrix $\boldsymbol{F}_k$ is a possibly time variant state transition matrix and $\boldsymbol{w}_k$ represents additive noise process with covariance $\boldsymbol{Q}_k$.

3.3.1 Nearly Constant Velocity Model

The nearly constant velocity (NCV) motion model assumes, that the tracked person moves at a certain velocity influenced by small acceleration modeled by a white noise process i.e. $\dot{\boldsymbol{x}}(t) = \boldsymbol{w}(t) \approx 0$. For simplicity the NCV model is often also referred to as constant velocity (CV) model in literature. In the following we will use NCV and CV interchangeably throughout the thesis and also occasionally omit the the "nearly" prefix

in subsequent models introduced in the following sections. According to the equations of motion the matrices for the model in Equation (3.40) are given by:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \tag{3.44}$$

To get the discrete time model, one has to sample from the solution of the continuous model with some sampling time T. Furthermore, note that we directly integrate the continuous noise gain $\boldsymbol{B}$ into the discrete random vector $\boldsymbol{w}_k$. This leads to the following matrices for Equation (3.43):

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix} \frac{T^3}{3}\sigma_x^2 & 0 & \frac{T^2}{2}\sigma_x^2 & 0 \\ 0 & \frac{T^3}{3}\sigma_y^2 & 0 & \frac{T^2}{2}\sigma_y^2 \\ \frac{T^2}{2}\sigma_x^2 & 0 & T\sigma_x^2 & 0 \\ 0 & \frac{T^2}{2}\sigma_y^2 & 0 & T\sigma_y^2 \end{bmatrix} \tag{3.45}$$

In addition to the discretized model it is also possible to define the discrete model directly which leads to a different covariance matrix of the noise vector. In this case the model is defined by Equation (3.46) where $\boldsymbol{G}_k$ is the noise gain similarly to $\boldsymbol{B}(t)$ in continuous time. The model directly states that $\boldsymbol{w}_k$ is random discrete acceleration influencing position and velocity according to the equations of motion. Again we consider $\boldsymbol{Q}$ as the covariance of $\boldsymbol{G}_k\boldsymbol{w}_k$ i.e. the covariance of the random vector and its gain.

$$\boldsymbol{x}_{k+1} = \boldsymbol{F}_k\boldsymbol{x}_k + \boldsymbol{G}_k\boldsymbol{w}_k \tag{3.46}$$

With matrices:

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{G} = \begin{bmatrix} \frac{T^2}{2} & 0 \\ 0 & \frac{T^2}{2} \\ T & 0 \\ 0 & T \end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix} \frac{T^4}{4}\sigma_x^2 & 0 & \frac{T^3}{2}\sigma_x^2 & 0 \\ 0 & \frac{T^4}{4}\sigma_y^2 & 0 & \frac{T^3}{2}\sigma_y^2 \\ \frac{T^3}{2}\sigma_x^2 & 0 & T^2\sigma_x^2 & 0 \\ 0 & \frac{T^3}{2}\sigma_y^2 & 0 & T^2\sigma_y^2 \end{bmatrix} \tag{3.47}$$

The nearly CV model is a common choice in people tracking implementation e.g. [BH10], [LGA15], [DBJH15] due to its simplicity. Assuming CV implies the assumption of smooth human motion which is not fulfilled during high acceleration or turns in general. Nevertheless, it often provides sufficiently good forward prediction if correcting detections are frequent enough.

3.3.2 Nearly Constant Acceleration Model

Another popular and rather simple model is the nearly constant acceleration (NCA) model. Similar to the NCV acceleration is assumed to undergo small changes modeled as a Wiener process, hence the model is also called Wiener process acceleration model. To

define the model we first augment the state by the acceleration in x and y direction, i.e. $\boldsymbol{x} = [x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}]^\intercal$. Now we can again state the matrices for Equation (3.40) according to the equations of motion where the noise vector $\boldsymbol{w}(t)$ models the change in acceleration called jerk:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \tag{3.48}$$

Discretization to a model of form 3.43 gives the following matrices:

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix} \frac{T^5\,\sigma_x^2}{20} & 0 & \frac{T^4\,\sigma_x^2}{8} & 0 & \frac{T^3\,\sigma_x^2}{6} & 0 \\ 0 & \frac{T^5\,\sigma_y^2}{20} & 0 & \frac{T^4\,\sigma_y^2}{8} & 0 & \frac{T^3\,\sigma_y^2}{6} \\ \frac{T^4\,\sigma_x^2}{8} & 0 & \frac{T^3\,\sigma_x^2}{3} & 0 & \frac{T^2\,\sigma_x^2}{2} & 0 \\ 0 & \frac{T^4\,\sigma_y^2}{8} & 0 & \frac{T^3\,\sigma_y^2}{3} & 0 & \frac{T^2\,\sigma_y^2}{2} \\ \frac{T^3\,\sigma_x^2}{6} & 0 & \frac{T^2\,\sigma_x^2}{2} & 0 & T\,\sigma_x^2 & 0 \\ 0 & \frac{T^3\,\sigma_y^2}{6} & 0 & \frac{T^2\,\sigma_y^2}{2} & 0 & T\,\sigma_y^2 \end{bmatrix} \tag{3.49}$$

Again one can also define the discrete model given in Equation (3.46) directly where $\boldsymbol{w}_k$ is the acceleration increment at time step k modeled by a discrete time Wiener process.

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{G} = \begin{bmatrix} \frac{T^2}{2} & 0 \\ 0 & \frac{T^2}{2} \\ T & 0 \\ 0 & T \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.50}$$

$$\boldsymbol{Q} = \begin{bmatrix} \frac{T^4\,\sigma_x^2}{4} & 0 & \frac{T^3\,\sigma_x^2}{2} & 0 & \frac{T^2\,\sigma_x^2}{2} & 0 \\ 0 & \frac{T^4\,\sigma_y^2}{4} & 0 & \frac{T^3\,\sigma_y^2}{2} & 0 & \frac{T^2\,\sigma_y^2}{2} \\ \frac{T^3\,\sigma_x^2}{2} & 0 & T^2\,\sigma_x^2 & 0 & T\,\sigma_x^2 & 0 \\ 0 & \frac{T^3\,\sigma_y^2}{2} & 0 & T^2\,\sigma_y^2 & 0 & T\,\sigma_y^2 \\ \frac{T^2\,\sigma_x^2}{2} & 0 & T\,\sigma_x^2 & 0 & \sigma_x^2 & 0 \\ 0 & \frac{T^2\,\sigma_y^2}{2} & 0 & T\,\sigma_y^2 & 0 & \sigma_y^2 \end{bmatrix} \tag{3.51}$$

The NCA model is still a very simple model but compared to the constant velocity it is not limited to nosy straight forward motion. It can also model constant change of velocity appearing as speed increase or change in direction.

3.3.3 Nearly Coordinated Turn Model

A coordinated turn is defined as a turn with constant speed, i.e. the magnitude of the velocity vector $v(t) = v$ and constant angular velocity $\omega(t) = \omega$. Then the equations of motion are given as:

$$\dot{x} = v\cos\theta \tag{3.52}$$
$$\dot{y} = v\sin\theta \tag{3.53}$$
$$\ddot{x} = -\omega v\sin\theta = -\omega\dot{y} \tag{3.54}$$
$$\ddot{y} = \omega v\cos\theta = \omega\dot{x} \tag{3.55}$$

One can define the nearly coordinated turn model (NCT) according to Equation (3.40) using the state $\boldsymbol{x}(t) = [x, y, \dot{x}, \dot{y}]^\intercal$ with the matrices:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\omega \\ 0 & 0 & \omega & 0 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.56}$$

The authors in [BSLK04] suggest to model the noise vector $\boldsymbol{w}(t)$ as zero-mean white Gaussian noise and its covariance $\boldsymbol{Q}$ is a design parameter. Considering that we will work with a discrete time model anyway we choose to define it directly using the solution of $\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t)$ i.e. $\boldsymbol{x}(t) = \exp(At)$. Since we do not know the value of ω we add it to the state $\boldsymbol{x}_k = [x, y, \dot{x}, \dot{y}, \omega]^\intercal$ and the noise vector $\boldsymbol{w}_k = [w_x, w_y, w_\omega]^\intercal$. Then we can define the discrete nearly coordinated turn model according to Equation (3.46) as:

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & \frac{\sin T\omega}{\omega} & \frac{\cos T\omega - 1}{\omega} & 0 \\ 0 & 1 & -\frac{\cos T\omega - 1}{\omega} & \frac{\sin T\omega}{\omega} & 0 \\ 0 & 0 & \cos T\omega & -\sin T\omega & 0 \\ 0 & 0 & \sin T\omega & \cos T\omega & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{G} = \begin{bmatrix} \frac{T^2}{2} & 0 & 0 \\ 0 & \frac{T^2}{2} & 0 \\ T & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{bmatrix} \tag{3.57}$$

$$\boldsymbol{Q} = \begin{bmatrix} \frac{T^4\sigma_x^2}{4} & 0 & \frac{T^3\sigma_x^2}{2} & 0 & 0 \\ 0 & \frac{T^4\sigma_y^2}{4} & 0 & \frac{T^3\sigma_y^2}{2} & 0 \\ \frac{T^3\sigma_x^2}{2} & 0 & T^2\sigma_x^2 & 0 & 0 \\ 0 & \frac{T^3\sigma_y^2}{2} & 0 & T^2\sigma_y^2 & 0 \\ 0 & 0 & 0 & 0 & T^2\sigma_\omega^2 \end{bmatrix} \tag{3.58}$$

One has to be careful since ω can actually be zero and divisions by ω occur in the state transition matrix $\boldsymbol{F}$. Therefore, it is required to use the limit of $\boldsymbol{F}$ where ω approaches zero. One can immediately infer from the coordinated turn assumptions, that this limit corresponds to the NCV model transition matrix in Equation (3.45) augmented with the state transition for ω. The NCT model specifically models turns i.e. motion on circular arcs which for example makes it useful for people tracking at positions where a person is likely to turn right / left into a room from the hallway.

3.4 The Assignment Problem for Data Association

In this section we will discuss the assignment problem and an algorithm commonly used to solve it. The solution to the assignment problem is later used in Section 4.2.4 to assign observed detections to tracked persons, i.e. data association. In general the assignment problem deals with one to one assignment between two sets of equal size where each assignment holds a certain value or cost. The goal is to maximize the sum of values or minimize the overall cost while assigning all elements in the sets.

The description of the assignment problem in this section is based on the description in [PS98]. Formally, the assignment problem can be outlined as follows. We consider two sets $S = \{s_0, \ldots, s_{n-1}\}$ and $T = \{t_0, \ldots, t_{n-1}\}$ with n elements each and $S \cap T = \emptyset$. Now we define the value (cost) of assigning a node $s_i \in S$ to a node $t_j \in T$ by c_{ij}. In the following we assume that all costs are positive integers including zero. Additionally, we define variables x_{ij} where $x_{ij} = 1$ if s_i is assigned to s_j and $x_{ij} = 0$ otherwise. Requiring that no element is assigned more than once leads to the following two constraints:

$$\sum_{i=0}^{n-1} x_{ij} = 1, \quad \forall i \in \{0, \ldots, n-1\} \tag{3.59}$$

$$\sum_{j=0}^{n-1} x_{ij} = 1, \quad \forall j \in \{0, \ldots, n-1\} \tag{3.60}$$

The cost of all assignments is then given as:

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} c_{ij} x_{ij} \tag{3.61}$$

Therefore, to get an optimal assignment one has to maximize (or minimize) Equation (3.61). From this point on we will only consider the case of maximization. It is easy to transform a minimization problem into a maximization problem by subtracting the individual cost from the maximum cost:

$$c_{max} = \max_{ij} c_{ij}^{min} \tag{3.62}$$

$$c_{ij}^{max} = c_{max} - c_{ij}^{min} \tag{3.63}$$

There are two equal modeling approaches. One in terms of matrices ${c_{ij}}_{n \times n}$ and ${x_{ij}}_{n \times n}$ and another in terms of bipartite graphs with disjoint vertex sets S and T. Here, we will consider the approach utilizing bipartite graphs.

A solution to the assignment problem in polynomial time can be obtained by the Kuhn Munkres algorithm based on the papers by Harold W. Kuhn [Kuh55] and James Munkres [Mun57]. It is also commonly referred to as the Hungarian Method since Harold W. Kuhn based his paper on the work of two Hungarians D. Kőnig and J. Egerváry. In

the following we will base the outline of the algorithm in terms of bipartite graphs as provided in the lecture notes of Subhash Suri[1].

First, we define the weighted bipartite graph $G = (V, E)$ with the vertex set $V = S \cup T$ consisting of the two disjoint sets S and T and the set of edges E. Furthermore, we establish that G is complete, which means that for every $s_i \in S$ and every $t_j \in T$ there is an edge $(s_i, t_j) \in E$ with weight c_{ij}. This is easily met since we assume that every pairing is possible. Although, it might add a value of zero, i.e. $c_{ij} = 0$ for some i and j. To develop the algorithm we need some definitions regarding bipartite graphs. A *matching* M is defined as a set of edges where it holds that no two edges contain the same vertex. In the assignment problem we are interested in a weighted *perfect* matching involving all vertices and therefore containing exactly n edges, where its weight is given by $\sum_{(s_i,t_j)\in M} c_{ij}$. Hence, to solve the assignment problem we intend to find a matching with maximum weight. We refer to edges $e \in M$ as *matched* and otherwise as *free*. Additionally, vertices of matched edges are also *matched* and otherwise *exposed*. A path $P = e_1, e_2, \ldots, e_{k-1}, e_k$ in M is called *alternating* if matched and free edges alternate. Furthermore, if both the first vertex and the last one are exposed the alternating path P is called *augmenting*. This is due to the fact that $M' = M\Delta P$, where Δ denotes the symmetric difference i.e. $M\Delta P = M \setminus P \cup P \setminus M$, leads to a larger matching $|M'| > |M|$. In addition to alternating paths, we also define *alternating trees*, which are trees with an exposed vertex as the root and where every path in the tree is an alternating path.

For the solution of the assignment problem we also consider the dual of the assignment problem as outlined in [Kuh55]. For this purpose we define vertex labels by a labeling function $l : V \mapsto \mathbb{N}$. A labeling is called *feasible* if it satisfies the following constraint:

$$l(s_i) + l(t_j) \geq c_{ij}, \quad \forall i, j \in \{0, \ldots, n-1\} \tag{3.64}$$

The dual problem to the assignment problem is then to find labels such that their sum is minimized under the constraint given in (3.64). Using the vertex labeling, one can define a subgraph containing only edges $e = (s_i, t_j)$ where the sum of the vertex labels equals the weight of the edge exactly. $G_l = (V, E_l)$ where:

$$E_l = \{(s_i, t_j) \mid l(s_i) + l(t_j) = c_{ij}\} \tag{3.65}$$

This definition leads to the following key theorem for the Hungarian Method:

Theorem 1. *If l is a feasible labeling following condition* (3.64) *and M is a perfect matching in G_l then M is also a maximum weight matching.*

Proof. Let M' be an arbitrary perfect matching in G and l a feasible labeling. Since by definition of a perfect matching every $v \in V$ is covered exactly once and the weight of

[1] http://www.cs.ucsb.edu/~suri/cs231/Matching.pdf

M' is given as:

$$\sum_{(s_i,t_j)\in M'} c_{ij} \leq \sum_{(s_i,t_j)\in M'} l(s_i) + l(t_j) = \sum_{v\in V} l(v) \tag{3.66}$$

Hence, the sum of weights in a perfect matching is bound by the sum of all vertex labels. Now let M be a perfect matching in G_l then:

$$\sum_{(s_i,t_j)\in M} c_{ij} = \sum_{v\in V} l(v) \tag{3.67}$$

From this it follows that:

$$\sum_{(s_i,t_j)\in M'} c_{ij} \leq \sum_{(s_i,t_j)\in M} c_{ij} \tag{3.68}$$

Therefore, the weight of all edges in M is the maximum achievable, and hence M a maximum weight matching. □

Due to this fact one can omit the weights and instead of searching for a maximum weight matching, search for a perfect matching in G_l. Since we utilize the graph G_l, which depends on the choice of a feasible labeling, a perfect matching in G_l might not be possible due to the lack of available edges. Hence, an approach is needed to improve the labeling l while keeping it feasible. By improving the labeling we mean that the improved labeling function l' should lead to a larger set of edges, i.e. $E_l \subset E_{l'}$. To do this we consider a subset of vertices $S' \subseteq S$ and its neighborhood $N(S') = \bigcup_{s'\in S'} N(s')$ where $N(s') = \{t' \mid (s', t') \in E_l\}$. Additionally, we required that $N(S') \neq T$ because in that case E_l would already hold enough edges for a perfect matching and improving the labeling would not be necessary. For convenience we define the function $w(s_i, t_j) = c_{ij}$ which gives the weight of the edge between nodes s_i and t_j. Then the new labels are computed as follows:

$$\alpha = \min_{s'\in S',\ t'\in T\setminus N(S')} l(s') + l(t') - w(s', t') \tag{3.69}$$

$$l'(v) = \begin{cases} l(v) - \alpha & \text{if } v \in S' \\ l(v) + \alpha & \text{if } v \in N(S') \\ l(v) & \text{otherwise} \end{cases} \tag{3.70}$$

What this does is that we first find the value α which gives the smallest difference in edge weight and corresponding vertex label sum for nodes that do not yet have an edge in E_l. Then we evenly add on one side and subtract on the other which ensures that edges from E_l are still present in $E_{l'}$. Due to the value of α this also ensures that $E_{l'}$ will have at least one additional edge and hence $E_l \subset E_{l'}$.

To apply the Kuhn-Munkres algorithm we first need to compute an initial matching

Algorithm 3.3: Initialize

```
  // compute initial feasible labeling
1 ∀s ∈ S : l(s) = 0;
2 for s ∈ S do
3 |   for t ∈ T do
4 |   |   l(s) = max(l(s), w(s, t));
5 |   |   l(t) = 0;
6 |   end
7 end
  // initially nothing is matched
8 M = ∅;
```

Algorithm 3.4: Compute Slack

```
1 for t ∈ T do
2 |   slack = l(u) + l(t) − w(u, t);
3 |   if slack < slack(t) then
4 |   |   slack(t) = slack;
5 |   end
6 end
```

and vertex labeling. Initialization is given in Algorithm 3.3. We start with an empty matching and compute a feasible labeling by assigning the maximum edge weight to each vertex in S and a label of zero to each vertex in T. The pseudo code to solve the assignment problem is then given in Algorithm 3.6. The outermost while loop ensures that the algorithm iterates as long as the matching is not perfect, i.e. the number of edges in M is equal to the number of vertices in either of the bipartite vertex sets. In the loop we utilize a queue to search for an augmenting path in G_l in a breath first fashion to increase the matching. We start at a free vertex s' which serves as a root of an alternating tree. The sets S' and T' represent the vertices in the alternating tree. Since we start with a vertex $s' \in S$ we initialize $S' = \{s'\}$ and $T' = \emptyset$ (line 3). To be able to reconstruct the path we keep track of every nodes parent. Until we find an augmenting path we dequeue a node p from Q in every iteration. Starting from p we consider all its neighbors with respect to the graph G_l excluding the vertices which are already in the alternating tree (line 9). If the considered neighbor v is not part of a matching we found an augmenting path since we always start with a free vertex, and can therefore augment the matching M. Otherwise we extend the augmenting tree by the vertex v and its matched vertex u (line 23). Additionally, u is added to the queue to continue the search from there (line 21). As discussed before we might not be able to improve the matching in G_l while it is still not perfect in G because of the labeling. Hence, the labeling has to be improved. We do this using the Algorithms 3.4 and 3.5 to calculate slack and update the labels respectively. Utilizing the slack variables we

Algorithm 3.5: Update Labels

```
1  α = ∞;
2  for t ∈ T \ T′ do
3  |  α = min(α, slack(t));
4  end
5  for v ∈ V do
6  |  if v ∈ S′ then
7  |  |  l(v) = l(v) − α;
8  |  end
9  |  if v ∈ T′ then
10 |  |  l(v) = l(v) + α;
11 |  else
12 |  |  slack(v) = slack(v) − α;
13 |  end
14 end
```

can incrementally compute α by updating it for a single node $u \in S'$ every time a new one is added to S' (line 24). The label update is then similar to the formula given in Equation (3.69) but uses the slack variable to compute α. Furthermore, for variables not in T' the slack has to be updated as well since its counterpart gets a label increase of α.

The complexity of Algorithm 3.6 can be given in terms of the number of nodes $n = |S| = |T|$. First consider the outermost loop. We start with $|M| = 0$ and increase the size of the matching by one every iteration since the augmenting path always gives one additional edge, hence the loop is executed n times. Slack computation in Algorithm 3.4 as well as the label updates in Algorithm 3.5 are obviously also bounded by $O(n)$. The for loop, in the worst case, considers every neighbor of S' which is bound by the size of T and hence the for loop is also $O(n)$. In summary the for loop is of complexity $O(n)$ and gets executed $O(n)$ times due to the inner while loop, which in turn is executed $O(n)$ times by the outer while loop, hence giving an overall complexity of $O(n^3)$.

Algorithm 3.6: Kuhn-Munkres Algorithm

Input: $G = (S \cup T, E)$, C
Output: Perfect Matching M

while $|M| \neq |S|$ **do**
  pick $s' \in S$ s.t. s' is free;
  $S' = \{s'\}$, $T' = \emptyset$;
  Q.enqueue(s'), Parent(s') = null;
  $\forall t \in T:$ slack(t) $= \infty$, computeSlack(s');
  not_found = $true$;
  while $Q.size > 0$ && not_found **do**
    $p = Q$.dequeue();
    for $v \in N_l(S') \setminus T'$ **do**
      Parent(v) = p;
      if v *is free* **then**
        $x = v$;
        while $Parent(x) \neq null$ **do**
          $P = P \cup (x, \text{Parent}(x))$;
          $x = \text{Parent}(x)$;
        end
        $M = M \Delta P$;
        not_found = $false$;
        break;
      else
        `// v is matched to u`
        Q.enqueue(u);
        Parent(u) = v;
        $S' = S' \cup \{u\}$, $T' = T' \cup \{v\}$;
        computeSlack(u);
      end
    end
  end
  updateLabels();
end
return M

3.5 CLEAR MOT Metrics

To compare different approaches to tracking multiple people a performance criteria has to be established. In [BS08] multiple object tracking metrics are proposed, called CLEAR MOT, providing criteria for tracking accuracy and precision. Those metrics are for example used in [LBLA16] or [MBM12] which are approaches we indent to compare with. In this section we give a quick reference for those metrics which will be later used to

evaluate the implemented people tracker.

The multiple object tracking precision (MOTP) is defined as follows:

$$\text{MOTP} = \frac{\sum_{i,t} d_t^i}{\sum_t c_t} \tag{3.71}$$

Here the distances d_t^i describe how far off the tracker hypothesis of object i is from the ground truth at time t. In our case of tracking people in 3D space, however assuming that they move on the ground plane since our robot is restricted to a flat surface as well, the distances d_t^i are euclidean distances between person centers. The values c_t correspond to the overall number of matches found by the tracker at time t. Therefore, the MOTP actually gives a mean error estimate in meters. The intention of the metric is to give a measure of how precise the tracker estimates positions. A low MOTP hence, corresponds to high precision because if the tracker is accurate the distances d_t^i between tracked persons and the ground truth are low.

To consider the ability of keeping track of all objects with correct data association the multiple object tracking accuracy (MOTA) is also introduced. Equation (3.72) states how to calculate the MOTA.

$$\text{MOTA} = 1 - \frac{\sum_t m_t + fp_t + mme_t}{\sum_t g_t} \tag{3.72}$$

The MOTA consists of three different failure cases. The number of completely missed objects m_t, the number of false positives fp_t and the number of mismatches mme_t all at time t weighted by the total number of objects present at t annotated by g_t. We count a person as missed if there is a ground truth track but no corresponding one from the tracking algorithm. In the evaluation the Kuhn Munkres algorithm, described in the previous Section, is used to match ground truth tracks and hypothesis tracks from the tracking algorithm. One then has to describe a threshold distance at which tracks are considered valid matches, and otherwise misses. A false positive on the other hand, is a track hypothesis with no matching ground truth track. For example if there are two tracks matched by the Kuhn Munkres algorithm but they are farther away than the threshold, the track hypothesis is considered a false positive whereas the ground truth track leads to a miss. The third failure case are mismatches, which typically occur if two persons are walking near by and the tracking algorithm mistakenly switches the track ids between the persons at some point in time. Those failure cases can be calculated separately to show weaknesses of a certain approach in specific area. Nevertheless, the MOTA gives a good idea of how precise the tracker is overall.

Obviously one has to establish ground truth object correspondences in a dataset first to apply MOTP and MOTA metrics. For real world scenarios one typically has to do this by hand if it is not possible to place special markers on each person. Hence, ground truth annotations might not always be correct as well, but nevertheless it gives a good enough estimate and enables comparison between different tracking approaches.

3.6 Summary

To summarize, we introduced the necessary probability theory needed for Bayesian filtering and particle filtering, as well as those algorithms themselves. In addition, we provided the formal definitions of three commonly used motion models, namely the NCV, NCA and NCT models. In order to later on assign detections to their corresponding persons, we covered the assignment problem and its solution, the Kuhn Munkres algorithm. Furthermore, we introduced the CLEAR MOT metrics from [BS08] in order to establish a criteria to assess the proposed people tracking approach.

In the following chapter we use the fundamental concepts introduced here to outline our approach to people tracking.

CHAPTER 4

Approach

This chapter contains the main contribution of this thesis, namely the particle filter based people tracking approach and a map based motion model. First, the probabilistic model used in the tracking algorithm incorporating several detectors is outlined in Section 4.1. Based on this model the needed components for the people tracking algorithm, i.e. filtering, data association and tracking management, are developed in Section 4.2.

4.1 Probabilistic Model for People Tracking

In this Section we will establish a probabilistic model for the people tracking problem estimating the position of several persons in a 2D map, given detections within the field of view of the robot. For now, we consider estimating the states of T persons given M detections, similar as done in [RN16] for tracking multiple objects with $T = M$. This means that first we assume there are as many persons to track as detections. Dealing with a different number of persons to track and detections will be discussed later. We denote the state of a person i at time k by the random vector $\mathbf{x}_k^i$. Furthermore, detection j at time k is modeled by the random vector $\mathbf{z}_k^j$. For simplicity we restrict the illustrations of the model to $T = M = 2$ persons and detections, but it can be easily generalized to arbitrary T and M. The people tracking problem for $T = M = 2$ can be modeled as a dynamic Bayesian network shown in Figure 4.1. It is assumed that the state of each person $\mathbf{x}_k^i$ evolves over time independently of every other persons state. This assumption might not be true since people often move in groups or at least move in awareness of others. However, in this work we do not consider groups or movement of people relative to each other, hence sticking to this assumption. In addition, as in Bayesian filtering we assume that the state of a person fulfills a first order Markov property, i.e. it depends only on the state at the previous time step. This assumption is not always valid because it requires the state to be complete, meaning it has to incorporate all relevant information of the past which is not always possible e.g. through motion dynamics not considered in

the model. However, Bayesian filtering is typically robust towards such violations [Thr02], which we will apply later on. Due to ambiguity in data association it is not known which detection corresponds to which person, hence both detections depend on both person states. From the DBN we can immediately state the pdf of the joint probability

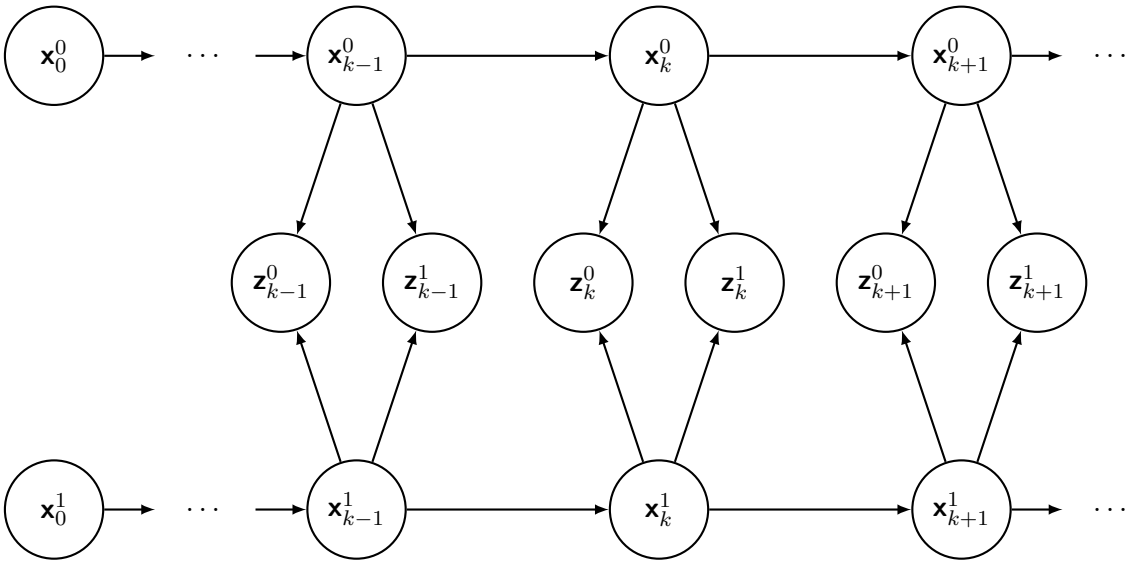


Figure 4.1: Dynamic Bayesian Network for People Tracking

distribution:

$$f_{\mathbf{x}_{0:k}^0,\mathbf{x}_{0:k}^1,\mathbf{z}_{1:k}^0,\mathbf{z}_{1:k}^1}(\boldsymbol{x}_{0:k}^0,\boldsymbol{x}_{0:k}^1,\boldsymbol{z}_{1:k}^0,\boldsymbol{z}_{1:k}^1) = f_{\mathbf{x}_0^0}(\boldsymbol{x}_0^0)f_{\mathbf{x}_0^1}(\boldsymbol{x}_0^1) \prod_{i=1}^{k} f_{\mathbf{x}_i^0|\mathbf{x}_{i-1}^0}(\boldsymbol{x}_i^0|\boldsymbol{x}_{i-1}^0)f_{\mathbf{x}_i^1|\mathbf{x}_{i-1}^1}(\boldsymbol{x}_i^1|\boldsymbol{x}_{i-1}^1) f_{\mathbf{z}_i^0,\mathbf{z}_i^1|\mathbf{x}_i^0,\mathbf{x}_i^1}(\boldsymbol{z}_i^0,\boldsymbol{z}_i^1|\boldsymbol{x}_i^0,\boldsymbol{x}_i^1) \tag{4.1}$$

Where $f_{\mathbf{x}_0^0}(\boldsymbol{x}_0^0)$ and $f_{\mathbf{x}_0^1}(\boldsymbol{x}_0^1)$ are priors of the state. The main problem here is to calculate the conditional probability density of the detections, since there are $M!$ ways to assign detections to persons for $T = M$ persons at each time step and hence, $(M!)^K$ possible assignments for K time steps [RN16]. A common approach to work around this issue is to predetermine the detection assignments by a data association algorithm which is not ideal but can reduce complexity significantly to make people tracking feasible. Using a data association algorithm, assigning detections to specific persons at each time step one can factor the conditional probability density of the detections. This way the DBN reduces to two instances of the one already discussed in Section 3.2 for state estimation. Without loss of generality we assume detection i is assigned to person i by the data association algorithm which can be easily accomplished by labeling the detections accordingly. Then we get:

$$f_{\mathbf{z}_i^0,\mathbf{z}_i^1|\mathbf{x}_i^0,\mathbf{x}_i^1}(\boldsymbol{z}_i^0,\boldsymbol{z}_i^1|\boldsymbol{x}_i^0,\boldsymbol{x}_i^1) = f_{\mathbf{z}_i^0|\mathbf{x}_i^0}(\boldsymbol{z}_i^0|\boldsymbol{x}_i^0)f_{\mathbf{z}_i^1|\mathbf{x}_i^1}(\boldsymbol{z}_i^1|\boldsymbol{x}_i^1) \tag{4.2}$$

Now, for people tracking on a mobile robot we are not interested in the overall joint probability distribution but rather on estimating the state of people given the detections. This means we would like to estimate the conditional pdfs $f_{\mathbf{x}_k^0|\mathbf{z}_{1:k}^0}(\boldsymbol{x}_k^0|\boldsymbol{z}_{1:k}^0)$ and $f_{\mathbf{x}_k^1|\mathbf{z}_{1:k}^1}(\boldsymbol{x}_k^1|\boldsymbol{z}_{1:k}^1)$. Hence, we have a Bayesian filtering problem for each person to track as introduced in Section 3.2.

4.1.1 Detection Sensor Fusion

As depicted in the state of the art review in Chapter 2 there are several detectors available to detect people with different sensors. It is possible to include several detections to correct the belief state estimation of a single person in various ways. The most straight forward approach would be to include all associated detections in the measurement vector $\mathbf{z}_k$. However, this imposes practical issues, namely, one has to dynamically adjust the measurement vector and therefore also the measurement model when adding or removing detectors. Furthermore, detector outputs have to be synchronized and the special case when a person is detected by one detector and not by another handled explicitly in the measurement model. As an alternative, we extend the model to incorporate multiple detections individually, per person tracked. Still, we retain the assumption that all detections are assigned to a person, meaning that if there are T persons to be tracked and D detectors we have TD detections and each tracked person is assigned D detections by the data association. As an example consider the DBN in Figure 4.2 representing a single person detected by two separate detectors. Using this network one can easily

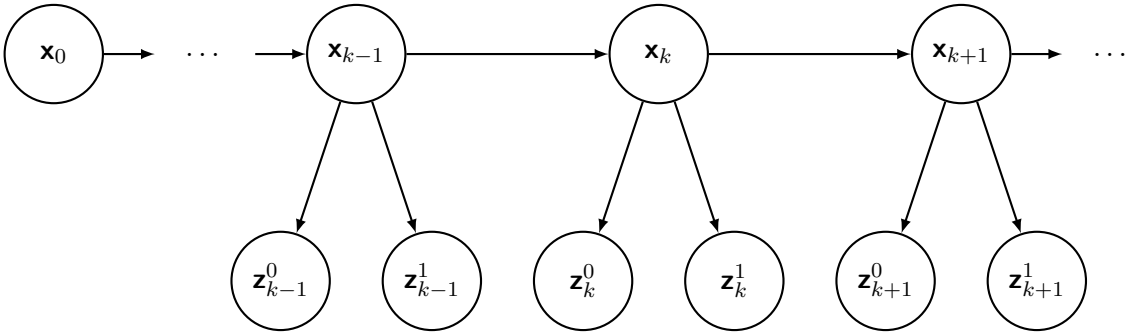


Figure 4.2: Dynamic Bayesian Network representing a single person detected by two detectors

derive the prediction and update step for the Bayes filter similar to Section 3.2. This time Bayes theorem (3.24) has to be applied twice in order to use both measurement pdfs. When generalizing to D detections it simply has to be applied D times.

$$
\begin{aligned}
f_{\mathbf{x}_k|\mathbf{z}_{1:k}^0,\mathbf{z}_{1:k}^1}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}^0,\boldsymbol{z}_{1:k}^1) = \eta \; & f_{\mathbf{z}_k^0|\mathbf{x}_k,\mathbf{z}_{1:k-1}^0,\mathbf{z}_{1:k}^1}(\boldsymbol{z}_k^0|\boldsymbol{x}_k,\boldsymbol{z}_{1:k-1}^0,\boldsymbol{z}_{1:k}^1) \\
& f_{\mathbf{z}_k^1|\mathbf{x}_k,\mathbf{z}_{1:k-1}^0,\mathbf{z}_{1:k-1}^1}(\boldsymbol{z}_k^1|\boldsymbol{x}_k,\boldsymbol{z}_{1:k-1}^0,\boldsymbol{z}_{1:k-1}^1) \\
& f_{\mathbf{x}_k|\mathbf{z}_{1:k-1}^0,\mathbf{z}_{1:k-1}^1}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}^0,\boldsymbol{z}_{1:k-1}^1)
\end{aligned}
\tag{4.3}
$$

Again utilizing conditional independence in the DBN and introducing the recursion using the total probability theorem (3.25) we get:

$$f_{\mathbf{x}_k|\mathbf{z}^0_{1:k},\mathbf{z}^1_{1:k}}(\boldsymbol{x}_k|\boldsymbol{z}^0_{1:k},\boldsymbol{z}^1_{1:k}) = \eta' \; f_{\mathbf{z}^0_k|\mathbf{x}_k}(\boldsymbol{z}^0_k|\boldsymbol{x}_k) f_{\mathbf{z}^1_k|\mathbf{x}_k}(\boldsymbol{z}^1_k|\boldsymbol{x}_k) \int_{\mathbb{R}^N} f_{\mathbf{x}_k|\mathbf{x}_{k-1}}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \; f_{\mathbf{x}_{k-1}|\mathbf{z}^0_{1:k-1},\mathbf{z}^1_{1:k-1}}(\boldsymbol{x}_{k-1}|\boldsymbol{z}^0_{k-1},\boldsymbol{z}^1_{k-1}) d\boldsymbol{x}_{k-1} \tag{4.4}$$

Leading to the recursive prediction and correction steps in equations (4.5) and (4.6).

$$\overline{bel}(\mathbf{x}_k) = \int_{\mathbb{R}^N} f_{\mathbf{x}_k|\mathbf{x}_{k-1}}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) bel(\mathbf{x}_{k-1}) d\boldsymbol{x}_{k-1} \tag{4.5}$$

$$bel(\mathbf{x}_k) = \eta' \; f_{\mathbf{z}^0_k|\mathbf{x}_k}(\boldsymbol{z}^0_k|\boldsymbol{x}_k) f_{\mathbf{z}^1_k|\mathbf{x}_k}(\boldsymbol{z}^1_k|\boldsymbol{x}_k) \overline{bel}(\mathbf{x}_k) \tag{4.6}$$

Therefore, using several detections of a single person can be simply fused using a Bayes filter by incorporating both multiplicatively in the correction step. Please note asynchronously arriving detections can be correctly dealt with using equation 4.6 in this approach. In particular, if only one of the detections arrives, only one density is considered and the other one can simply be replaced by a multiplication with 1.0, meaning it has no influence on the belief state at this time step. Also the detection probability densities can be specified individually depending on the detector characteristics. Hence, this approach gives more freedom in terms of implementation contrary to the measurement vector extension.

4.2 People Tracking Algorithm

Based on the probabilistic model we can state our general approach for the people tracking algorithm. From the model it is clear that the tracking algorithm needs three things:

- A Bayesian filter for every person to track
- A data association approach to assign detections to tracks
- A track management to create new tracks and delete unnecessary ones

An overview of the architecture is shown in Figure 4.3. As one can see, the received detections are first passed to a data association algorithm assigning detections to existing tracks if possible. Then, based on those assignments the track management can decide whether new tracks should be initialized or if obsolete tracks can be deleted. Assigned detections are then passed to the Bayesian filters to update the state estimate of each tracked person.

We first describe the overall algorithm and then go into detail on the individual parts. In the following we use the term *track* representing the belief state of a person and some

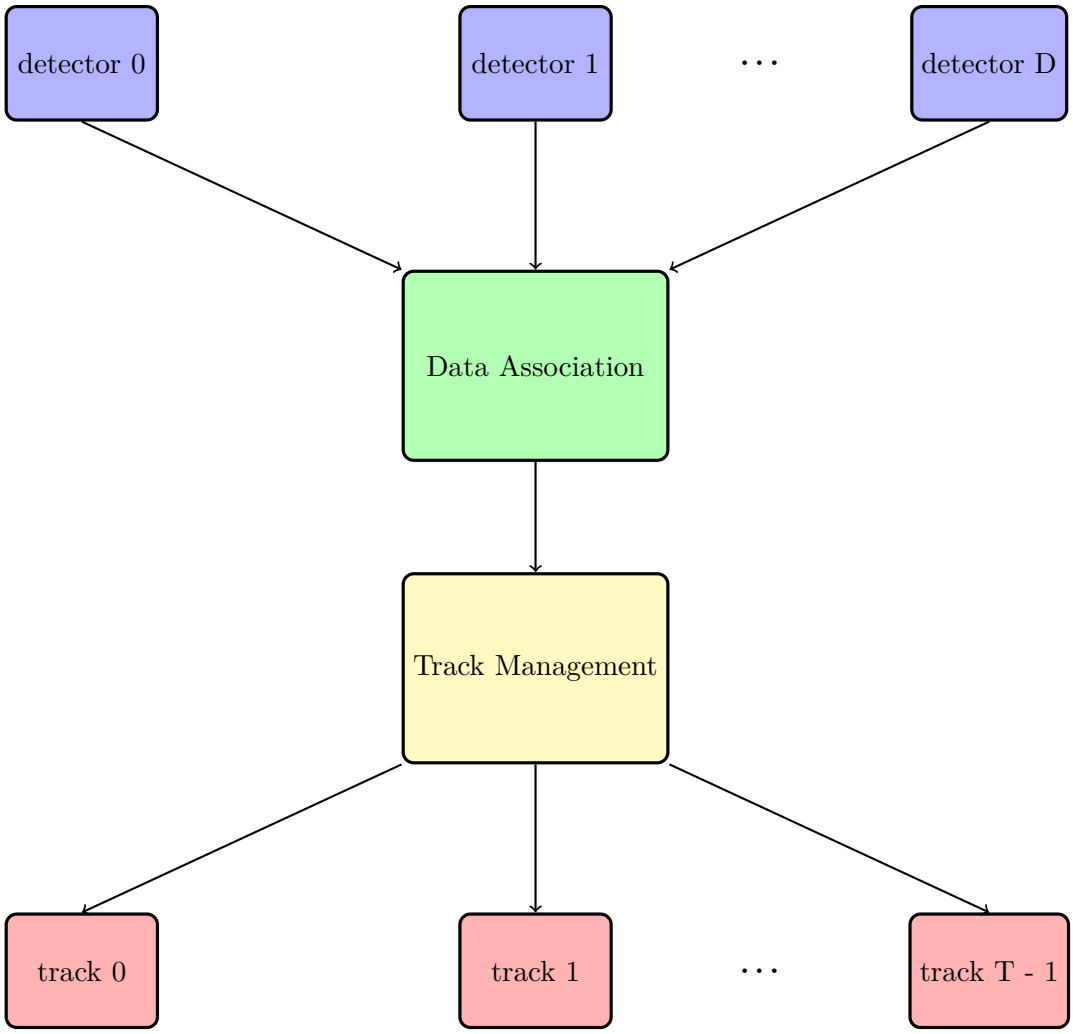


Figure 4.3: This Figure illustrates the general structure for the tracking algorithm. Several detections are received as an input, which are passed through the data association procedure. Based on the assignments, the track management block can decide to create or delete tracks, as well as update existing tracks based on their assigned detections.

additional attributes discussed in later sections. As an input the algorithm needs an array of tracks from the previous time step denoted as $\boldsymbol{T}_{k-1}$. Every track in the array is a tuple $(ID, \boldsymbol{X}, \hat{\boldsymbol{\mu}}_{\mathbf{x}}, \widehat{\boldsymbol{C}}_{\mathbf{x}}, M_C, D_C, V_C, m, m_v)$ with a unique identifier ID. The array $\boldsymbol{X}$ holds all the particles representing the belief state of the track with $\hat{\boldsymbol{\mu}}_{\mathbf{x}}$ and $\widehat{\boldsymbol{C}}_{\mathbf{x}}$ being the mean and covariance of those particles. The variables M_C, D_C and V_C are integer valued counters, whereas m and m_c are boolean flags needed for the track creation and deletion processes discussed in Section 4.2.5. Besides tracks the algorithm gets possibly multiple arrays of detections as an input, one for each detector attached, denoted as $\boldsymbol{Z}_{k-1}^0, \ldots, \boldsymbol{Z}_{k-1}^{D-1}$. The pseudo code is given in Algorithm 4.1. At first the algorithm predicts the belief state of every track from the previous time step to the current one (line 1-4). After forward prediction mean and covariance also have to be updated (line 3) because they are needed for data association. The forward predicted tracks are then used for data association which assigns a detection from every detector to a certain track (lines 5-7). Although, there does not always exist a detection from every detector for every track, it is still possible to place either dummy tracks or dummy detections, depending on which array is larger, such that every track is assigned a detection from each detector. Now that tracks got matched with detections, a correction step can be applied on the belief state of the tracks. The data association procedure provides a detection index for the detection array with index i assigned to track index t represented by $assignment^i[t]$ with a certain cost $assignment\_cost^i[t]$. How the cost is defined will be discussed later in Section 4.2.4. For the moment the cost is a measure on how meaningful a certain assignment is. By restricting the cost by a certain threshold A_{thres} (line 12) only meaningful assignments are used for the correction step (line 13). This procedure is referred to as *gating*. By gating the assignments the possibly introduced dummy tracks or detections are also filtered. If the track at index t with assigned detection $assignment^i[t]$ passed the gate $\boldsymbol{Z}_{ua}[assignment^i[t]]$ respectively $\boldsymbol{T}_{ua}[t]$ can be set to -1 (lines 15 and 16). The values in $\boldsymbol{Z}_{ua}$ and $\boldsymbol{T}_{ua}$ marked with -1 are marked as used and are therefore not considered in track creation or deletion. Otherwise they hold the index of a detection that is considered for track creation (line 19) and the index of a track that is considered for deletion (line 21). Note that it requires only one detection for a track to be corrected and hence only tracks with no detection assigned from any detector are considered to be deleted. Therefore, the same array $\boldsymbol{T}_{ua}$ is used throughout all detection arrays. Mean and covariance are also calculated here (line 14) since they provide improved estimates after the correction step. In the following Sections we go into more detail of each procedure used in the people tracking algorithm.

Algorithm 4.1: People Tracking Algorithm

Input: array of tracks $\boldsymbol{T}_{k-1}$, arrays of detections $\boldsymbol{Z}_k^0, \ldots, \boldsymbol{Z}_k^{D-1}$
Output: array of tracks $\boldsymbol{T}_k$

```
// forward predict every track
```
for $i = 0$ *to* $|\boldsymbol{T}_{k-1}| - 1$ **do**
  $\boldsymbol{T}_k[i].\boldsymbol{X} = \text{Prediction}(\boldsymbol{T}_{k-1}[i].\boldsymbol{X})$;
  $[\boldsymbol{T}_k[i].\hat{\boldsymbol{\mu}}_{\mathbf{x}}, \boldsymbol{T}_k[i].\widehat{\boldsymbol{C}}_{\mathbf{x}}] = \text{CalculateMeanAndCovariance}(\boldsymbol{T}_k[i].\boldsymbol{X})$;
end
```
// apply data association for every detection set
```
for $i = 0$ *to* $|\boldsymbol{Z}_k| - 1$ **do**
  $[assignment^i, assignment\_cost^i] = \text{DataAssociation}(\boldsymbol{T}_k, \boldsymbol{Z}_k^i)$;
end
```
// correction step for every track with assigned detections
```
$\boldsymbol{T}_{ua} = \{0, \ldots, |\boldsymbol{T}_k| - 1\}$;
for $i = 0$ *to* $|\boldsymbol{T}_k| - 1$ **do**
  $\boldsymbol{Z}_T$;
  for $j = 0$ *to* $|\boldsymbol{Z}_k| - 1$ **do**
    $\boldsymbol{Z}_{ua} = \{0, \ldots, |\boldsymbol{Z}_k^i| - 1\}$;
    `// gating`
    if $assignment\_cost^j[i] < A_{thres}$ **then**
      append $\boldsymbol{Z}_k^j[assignment^j[i]]$ to $\boldsymbol{Z}_T$;
      $\boldsymbol{Z}_{ua}[assignment^j[i]] = -1$;
      $\boldsymbol{T}_{ua}[i] = -1$;
      $\text{UpdateTrackMaturity}(\boldsymbol{T}_k[i], \boldsymbol{Z}_k^j[assignment^j[i]])$;
    end
    `// track creation`
    $\text{CreateTracks}(\boldsymbol{Z}_{ua})$;
  end
  $\text{Correction}(\boldsymbol{T}_k[t].\boldsymbol{X}, \boldsymbol{Z}_T)$;
  $[\boldsymbol{T}_k[i].\hat{\boldsymbol{\mu}}_{\mathbf{x}}, \boldsymbol{T}_k[i].\widehat{\boldsymbol{C}}_{\mathbf{x}}] = \text{UpdateMeanAndCovariance}(\boldsymbol{T}_k[i].\boldsymbol{X})$;
end
```
// track deletion
```
$\text{DeleteTracks}(\boldsymbol{T}_{ua})$;

4.2.1 Track Representation

As shown in Section 4.1 we can estimate the state of a person using Bayesian filtering given the detections associated with them. In this work we choose to approximate the Bayes filter with the particle filtering algorithm 3.1 introduced in Section 3.2. The choice of particle filters in contrast to Kalman filters has two reasons. First, particle filters enable the use of nonlinear models describing human motion. Although, in literature [BH10] nonlinear versions of the Kalman filter, i.e. the extended Kalman filter or the unscented Kalman filter have been used, due to the linearization they are by definition not optimal for nonlinear models. Second, particle filters are not restricted to normal distributions enabling the representation of multi-modal distributions. This enables multiple likely locations of a person during forward prediction, for example if a person can move straight ahead or turn into a room or in occlusion scenarios.

Formally, we define the state vector of a person at time k, as in the introduction of basic motion models in Section 3.3, through the 2D position of the person on the floor and his/her velocity split into components in both x and y directions as $\boldsymbol{x}_k = [x_k, y_k, \dot{x}_k, \dot{y}_k]^\intercal$ and the measurement vector for detection d at time k as $\boldsymbol{z}_k^d = [x_k^d, y_k^d]^\intercal$ containing the 2D position for the detection commonly obtained by a detector. As seen in the previous Section 4.2 we need to split the particle filter algorithm introduced in Section 3.2 into a prediction step and a correction step since the forward predicted belief state is required for data association. Note that we use arrays instead of sets for particles to be closer to an actual implementation. Forward prediction given in Algorithm 4.2 gets an array of state vector, weight pairs $(\boldsymbol{x}, w)$ of the previous time $k-1$ $\boldsymbol{X}_{k-1}$ as an input and computing the array of predicted particle weight pairs for time k $\bar{\boldsymbol{X}}_k$. However, low variance resampling as introduced in Section 3.2 is applied first to forward predict as many high weight particles as possible. The pdf, $f_{\mathbf{x}_k|\mathbf{x}_{k-1}}(\boldsymbol{x}_k|\boldsymbol{X}_{k-1}[n].\boldsymbol{x})$ is the state transition probability density defined by the motion model (see Sections 3.3 and 4.2.2) which gives the density for a new particle $\boldsymbol{x}_k$ given the particle $\boldsymbol{X}_{k-1}[n].\boldsymbol{x}$ from the previous time step. Particle weights in this step are only relevant if correction is not applied, otherwise they are overwritten. For tracks that only require forward prediction we therefore assign equal weights to all particles to stay consistent. The weight $\frac{1}{N}$ is chosen here such that the weight of all particles sums up to one. The correction part including several detections is stated in Algorithm 4.3. Here $f_{\mathbf{z}_k^0|\mathbf{x}_k}(\boldsymbol{z}_k^0|\bar{\boldsymbol{X}}_k[n].\boldsymbol{x}_k) \cdot \ldots \cdot f_{\mathbf{z}_k^{D-1}|\mathbf{x}_k}(\boldsymbol{z}_k^{D-1}|\bar{\boldsymbol{X}}_k[n].\boldsymbol{x}_k)$ is defined by detection probability density given by the detection model. How those distributions are obtained is discussed in Section 4.2.2. The algorithm normalizes particle weights such that they sum up to one (line 4 and 6-8). In addition the particles are sorted in a descending fashion according to their weight (line 9) such that one can easily extract the highest weight particle from the array.

To provide a single best estimate of the belief state of a person one can either use the particle with the highest weight, or calculate a sample mean over all particles. Highest weight particles represent the most likely location of the person according to recent measurements but do not give very stable trajectories over time. Hence, the weighted

Algorithm 4.2: Particle Filter Algorithm for People Tracking: Prediction

Input: $\boldsymbol{X}_{k-1}$
Output: $\bar{\boldsymbol{X}}_k$

$\boldsymbol{X}_{k-1} = \text{lowVarianceResampling}(\boldsymbol{X}_{k-1})$;
for $n = 0$ *to* $N-1$ **do**
  sample $\bar{\boldsymbol{X}}_k[n].\boldsymbol{x} \sim f_{\mathbf{x}_k|\mathbf{x}_{k-1}}(\boldsymbol{x}_k|\boldsymbol{X}_{k-1}[n].\boldsymbol{x})$;
  $\bar{\boldsymbol{X}}_k[n].w = \frac{1}{N}$;
end

Algorithm 4.3: Particle Filter Algorithm for People Tracking: Correction

Input: $\bar{\boldsymbol{X}}_k$, $\boldsymbol{z}_{1:k}^0, \ldots, \boldsymbol{z}_{1:k}^{D-1}$
Output: $\boldsymbol{X}_k$

$\eta = 0$;
for $n = 0$ *to* $N-1$ **do**
  $\bar{\boldsymbol{X}}_k[n].w = f_{\mathbf{z}_k^0|\mathbf{x}_k}(\boldsymbol{z}_k^0|\bar{\boldsymbol{X}}_k[n].\boldsymbol{x}_k) \cdot \ldots \cdot f_{\mathbf{z}_k^{D-1}|\mathbf{x}_k}(\boldsymbol{z}_k^{D-1}|\bar{\boldsymbol{X}}_k[n].\boldsymbol{x}_k)$;
  $\eta = \eta + \bar{\boldsymbol{X}}_k[n].w$;
end
// normalization
for $n = 0$ *to* $N-1$ **do**
  $\bar{\boldsymbol{X}}_k[n].w = \frac{\bar{\boldsymbol{X}}_k[n].w}{\eta}$;
end
sort particles descending accoding to their weight in $\bar{\boldsymbol{X}}_k$;
$\boldsymbol{X}_k = \bar{\boldsymbol{X}}_k$;

sample mean approach combined with a covariance matrix provide better estimates for further processing e.g. in a navigation algorithm. Mean and covariance for the particle array $\boldsymbol{X}_k$ are calculated in Algorithm 4.4.

Algorithm 4.4: Calculate Mean and Covariance

Input: $\boldsymbol{X}_k$
Output: $\hat{\boldsymbol{\mu}}_{\boldsymbol{x}_k}, \widehat{C_k}$

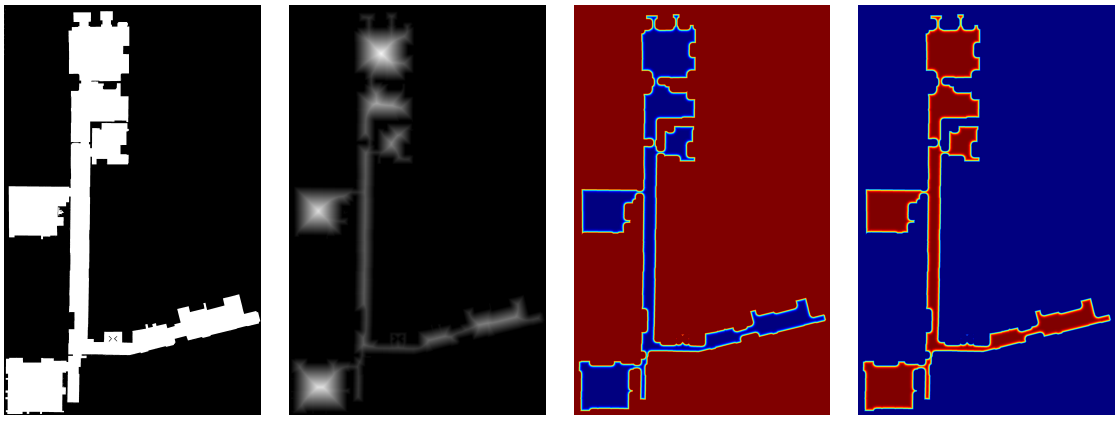
$\hat{\boldsymbol{\mu}}_{\boldsymbol{x}_k} = \sum_{i=0}^{N-1} \boldsymbol{X}_k[i].\boldsymbol{x}_k \boldsymbol{X}_k[i].w_k$;
$\widehat{C_k} = \frac{1}{1-\sum_{i=0}^{N-1}(\boldsymbol{X}_k[i].w_k)^2} \sum_{i=0}^{N-1} \boldsymbol{X}_k[i].w_k(\boldsymbol{X}_k[i].\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_{\boldsymbol{x}_k})(\boldsymbol{X}_k[i].\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_{\boldsymbol{x}_k})^\intercal$;

4.2.2 Heat Map Motion Model

To model the state transition probability density $f_{\mathbf{x}_k|\mathbf{x}_{k-1}}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ we need to model the human motion. Typically used models in literature have already been introduced in Section 3.3 which rely on rather strong simplifications, e.g. that a person moves

with nearly constant velocity or acceleration. We intend to give a more accurate model, especially considering the static environment of the robot defined by the map. We assume that the map is known which enables us to compute parts of the motion model in advance. Furthermore, if the map is static it is possible to utilize historical data of where people most likely move from their current position. Nevertheless, the approach in its essence is also applicable if the robot constructs the map on-line through a simultaneous localization and mapping (SLAM) algorithm, however historical data can obviously not be applied.

Heat Map Construction



(a) Initial grid map (b) Distance transform (c) Likelihood field (d) Inverse likelihood field

Figure 4.4: This Figure visualizes the steps taken in order to construct an initial heat map based on a grid map.

Consider the map of our lab in Figure 4.4a. In a first step we construct a likelihood field of where people are likely to move on the map. We assume people can move on every cell in the map that is not occupied by an object, i.e. the white part of the map. Furthermore, we expect that the likelihood decreases towards a wall, whereas it is impossible to move on pixels which are occupied, i.e. walls or locations outside of the map. The approach taken here is just the opposite to the sensor likelihood field for range finders presented in [TBF05], where it is most likely to measure something near an occupied cell in the map. To construct the likelihood field we first calculate the distance of every pixel to its nearest occupied pixel using a distance transform of the map image given in Figure 4.4b. Assuming that the distances are normal distributed with zero mean and a certain variance σ we construct the likelihood field in Figure 4.4c. Note that the values are

Figure 4.5: Jet Color Map

mapped to colors according to the Jet color map given in Figure 4.5 such that blue means least likely and red most likely, similar to heat maps where hot regions are colored red and cold regions blue. As stated before we need the inverse, hence we subtract all the likelihood values from the maximum to obtain Figure 4.4d. In the following we will refer to the inverse likelihood field as the initial heat map, since it serves as a starting point for historical data. Obtaining the initial heat map is summarized in Algorithm 4.5.

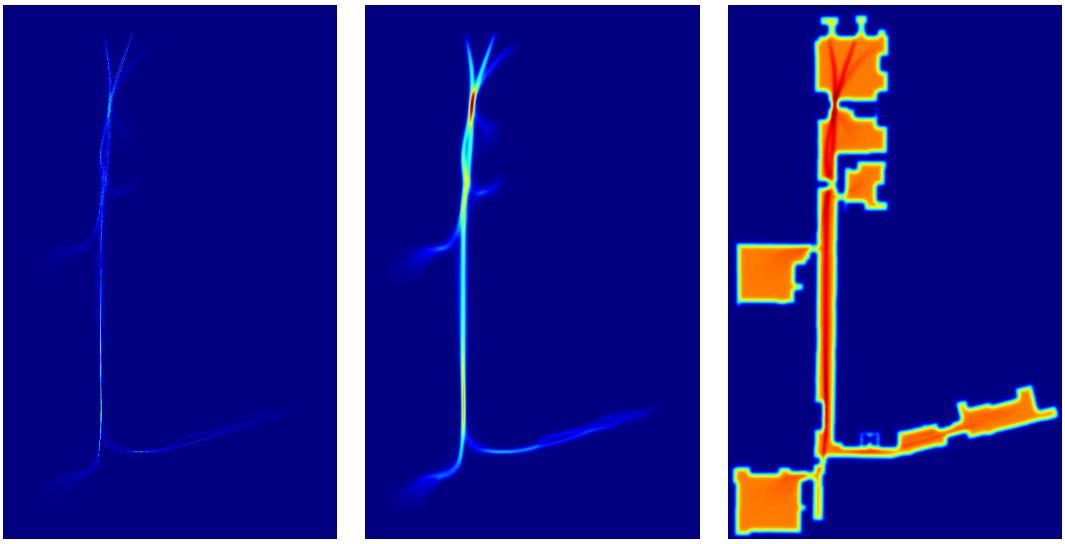
Algorithm 4.5: Compute initial heat map

Input: map M, σ
Output: initial heat map H

$D = \text{distanceTransform}(M)$;
$max = 0$;
for $row = 0$ *to* $M.rows$ **do**
  for $col = 0$ *to* $M.cols$ **do**
    $H[row, col] = \frac{1}{\sqrt{(2\pi\sigma^2)}} \exp\left(-\frac{D[row,col]^2}{2\sigma^2}\right)$;
    if $max < H[row, col]$ **then**
      $max = H[row, col]$;
    end
  end
end
for $row = 0$ *to* $M.rows$ **do**
  for $col = 0$ *to* $M.cols$ **do**
    $H[row, col] = max - H[row, col]$;
  end
end

Now if the robot is often situated in the same environment, it makes sense to incorporate previously observed persons into the heat map. Furthermore, historical data can also be obtained from pedestrian simulations using a microscopic model such as the one provided in [See15]. We adapt the heat map model using the state estimates given by the tracks $\boldsymbol{T}_k$ at time k. We adjust the values of the heat map along the linearly interpolated path each track takes between time step $k-1$ and k. Values are additively adjusted by a certain value p_w. The choice of p_w depends on how important historical data is in comparison to the initial likelihood field. In general, if historical data is recorded long enough, the initial likelihood fades since the values are insignificantly small in comparison to the historical data. To avoid this effect we record the historical data separately and incorporate a normalized version into the initial heat map. Also, keeping the non-normalized version gives the advantage that the historical data is easily extendable at a later point in time, which can then be normalized each time an updated version of the heat map is generated. In Figure 4.6a historical data for the Roblab map acquired using a pedestrian simulation based on the models in [See15] is shown. One can see that most persons walk along the

main hallway, as expected, and less pass towards the rooms. Since people do not always use the exact same paths it makes sense to apply a Gaussian blur to the historical data. Figure 4.6b shows the blurred version with a $\sigma = x$. Incorporating the data into the initial heat map results in Figure 4.6c. One can see that for maps with narrow corridors the initial heat map is a pretty good approximation considering local neighborhoods in the map, which are in the following used for the motion model. Historical data aggregation is



(a) Recorded data (b) Blurred data (c) Integration into initial heat map

Figure 4.6: Heat map historical data

given in Algorithm 4.6. The Algorithm finds corresponding tracks of time steps $k-1$ and k and increments the entries at the coordinates in between the two track observations by p_w (lines 11 to 13). Note that one has to convert real world track coordinates to the corresponding pixel coordinates in the map (lines 5 to 8). Algorithm 4.7 shows how to combine the initial heat map with the historical data. Before the maps are added they have to be normalized to the same range of values.

Algorithm 4.6: Aggregate historical data

Input: $\boldsymbol{T}_{k-1}$, $\boldsymbol{T}_k$, p_w
Output: H

initialize all values in H to 0;
for $i = 0$ *to* $|\boldsymbol{T}_k| - 1$ **do**
  for $j = 0$ *to* $|\boldsymbol{T}_{k-1}| - 1$ **do**
    if $\boldsymbol{T}_k[i].ID == \boldsymbol{T}_{k-1}[j].ID$ **then**
      idx_k^x =convertToMapIndex($\boldsymbol{T}_k[i].\hat{\boldsymbol{\mu}}_{\mathbf{x}}.x$);
      idx_k^y =convertToMapIndex($\boldsymbol{T}_k[i].\hat{\boldsymbol{\mu}}_{\mathbf{x}}.y$);
      idx_{k-1}^x =convertToMapIndex($\boldsymbol{T}_{k-1}[j].\hat{\boldsymbol{\mu}}_{\mathbf{x}}.x$);
      idx_{k-1}^y =convertToMapIndex($\boldsymbol{T}_{k-1}[j].\hat{\boldsymbol{\mu}}_{\mathbf{x}}.y$);
      $P_0 = (idx_{k-1}^x, idx_{k-1}^y)$;
      $P_1 = (idx_k^x, idx_k^y)$;
      for *(a, b) along line from* P_0 *to* P_1 **do**
        $H[a, b] = H[a, b] + p_w$;
      end
    end
  end
end

Algorithm 4.7: Combine historical data and initial heat map

Input: H, H'
Output: H

normalize contents of H and H' to interval $[0, 1]$;
$H = H + H'$;

Motion Model

In order to predict the motion of a person from its current position and velocity, we utilize the local neighborhood of the persons state estimate in the heat map. We consider the next position reachable from the current position, given the current velocity estimate of the person. Note, that we calculate the next position in meters instead of map coordinates and then look up the corresponding cell in the grid map to be able to adapt to different map granularities. Since the person might change its direction we not only consider the position reachable with the current angle θ of the velocity vector, but also the change in direction with a resolution of $\frac{2\pi}{p}$, where p is the number of different directions. The number of meaningful directions depends on the granularity of the grid map and also on the persons velocity. Figure 4.7 shows an example, where we use the current velocity magnitude r, but rotate it by an additional angle θ'. Hence, we obtain an angular velocity

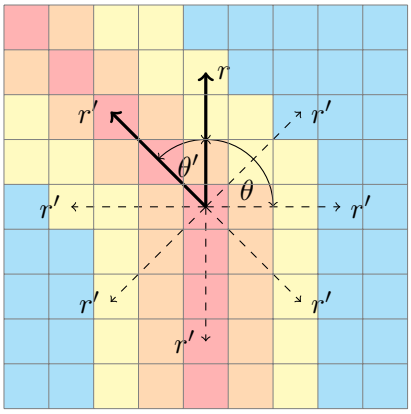


Figure 4.7: Given the current velocity vector r at an angle θ, we consider different changes in direction θ' to end up with a new velocity vector r'.

of $\omega = \frac{\theta'}{dt}$. Considering the additional rotation θ' the possible next velocities are given by:

$$\dot{x}_{next} = r\cos(\theta + \omega dt) = r\cos(\theta + \theta') \tag{4.7}$$

$$\dot{y}_{next} = r\sin(\theta + \omega dt) = r\sin(\theta + \theta') \tag{4.8}$$

And the corresponding position by:

$$x_{next} = x_{curr} + r\cos(\theta + \theta')dt \tag{4.9}$$

$$y_{next} = y_{curr} + r\sin(\theta + \theta')dt \tag{4.10}$$

We then take the value of the heat map at the position (x_{next}, y_{next}), converted into map coordinates, $h(x_{next}, y_{next})$ and define that the probability that the person moves to this point in the map is proportional to this heat map value. Subsuming all of the possible next state, heat map values we can define a discrete probability distribution formed by those values normalized in the interval $[0, 1]$, from which we can draw an angle increment for the velocity vector. Using this angle increment we calculate the forward predicted value of the particle. One important thing to consider here is that using this method the prediction might get stuck in a local minimum if grid map cells in the local environment have similar values. This can cause the prediction to basically spin in place. In order to work around this issue we apply two countermeasures. First, we increase the value of dt in the calculation of the next position in the heat map, such that the likelihood values obtained represent moving in that direction for a longer time period. For simplicity, we set $dt = 1$ giving:

$$x_{next} = x_{curr} + r\cos(\theta + \theta') \tag{4.11}$$

$$y_{next} = y_{curr} + r\sin(\theta + \theta') \tag{4.12}$$

Second, one can define a weight function for the direction probabilities. For example, it is typically more likely that a person keeps its direction, and turning 180° is the least

likely case. A natural way of doing this is to assume that the persons angular change follows a normal distribution $f_{\theta'}(\theta') \sim \mathcal{N}(0, \sigma^2_{\theta'})$ which we use as a weight $w(\theta') = f_{\theta'}(\theta')$, i.e. we multiply the heat map value at (x_{next}, y_{next}) by the pdf value $f_{\theta'}(\theta')$ at θ'. Hence, we obtain that the state transition distribution from which we sample the next state for each particle is defined as follows:

$$f_{\mathbf{x}_k|\mathbf{x}_{k-1}}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) = h(x_{next}, y_{next}) f_{\theta'}(\theta') \tag{4.13}$$

Here, x_{next} and y_{next} are calculated from the previous state vector $\boldsymbol{x}_{k-1}$ using equations (4.11) (4.12) substituting $\boldsymbol{x}_{k-1}.x$ and $\boldsymbol{x}_{k-1}.y$ for x_{prev} and y_{prev} respectively. Using this weight function the model assumes, of course depending on the variance $\sigma^2_{\theta'}$, that slight changes in direction are the most likely, whereas very drastic changes are highly unlikely but still possible.

Algorithm 4.8: Generate Forward predicted sample

Input: particle $\boldsymbol{x}_{k-1}$, real time between step $k-1$ and k dt, angle partitions p, heat map H

Output: particle $\boldsymbol{x}_k$

$angle\_probabilities[p]$;
// person orientation from velocity
$\theta = \text{atan2}(\boldsymbol{x}_{k-1}.\dot{y}, \boldsymbol{x}_{k-1}.\dot{x})$;
$r = \sqrt{\boldsymbol{x}_{k-1}.\dot{x}^2 + \boldsymbol{x}_{k-1}.\dot{y}^2}$;
for $i = 0$ *to* $p - 1$ **do**
  $\theta_{tar} = \theta + \frac{2\pi i}{p}$;
  $x_{next} = \boldsymbol{x}_{k-1}.x + r\cos(\theta_{tar})$;
  $y_{next} = \boldsymbol{x}_{k-1}.y + r\sin(\theta_{tar})$;
  $idx_x = \text{convertToMapIndex}(x_{next})$;
  $idx_y = \text{convertToMapIndex}(y_{next})$;
  $angle\_probabilites[i] = w(\frac{2\pi i}{p}) H[idx_x, idx_y]$;
end
normalize *angle_probabilities* to [0, 1];
draw d from *angle_probabilities*;
$\theta' = \frac{2\pi}{p} d$;
// wrap to range $[-\pi, \pi]$
$\theta_{next} = \theta + \text{atan2}(\sin(\theta'), \cos(\theta'))dt$;
// sample acceleration from Gaussian Distributions
draw $\delta_x \sim \mathcal{N}(0, \sigma_x^2)$;
draw $\delta_y \sim \mathcal{N}(0, \sigma_y^2)$;
$\boldsymbol{x}_k.x = \boldsymbol{x}_{k-1}.x + \boldsymbol{x}_{k-1}.\dot{x}dt + \frac{1}{2}\delta_x dt^2$;
$\boldsymbol{x}_k.y = \boldsymbol{x}_{k-1}.y + \boldsymbol{x}_{k-1}.\dot{y}dt + \frac{1}{2}\delta_y dt^2$;
$\boldsymbol{x}_k.\dot{x} = r\cos(\theta_{next}) + \delta_x dt$;
$\boldsymbol{x}_k.\dot{y} = r\sin(\theta_{next}) + \delta_y dt$;

Generating a new particle according to the heat map motion model is described in Algorithm 4.8. At first, we calculate polar coordinates from the persons velocity vector (lines 2 and 3). Given the polar coordinates we calculate possible next states according to the angular partitioning (lines 5 to 7) and then determine the angle probabilities using the heat map values at the next state weighted according to the change in angle (line 10). Afterwards, we normalize the angle probabilities such that they form a pmf (line 12). To create a particle sample we draw a value from this pmf and calculate the corresponding angle θ' which is normalized such that we take the shortest possible path on the circle (lines 14 to 21). Using the next state angle θ_{next} given by the previous angle θ and the change θ' times dt (line 22) we can calculate the new particle state $\mathbf{x}_k$ according to the laws of motion (lines 25 to 28). Note that similar to the nearly constant velocity model we model acceleration by zero mean Gaussian noise (lines 23 and 24) since we do not have any information of how much a person accelerates at a given point in time.

4.2.3 Detection Models

The purpose of the detection model is to describe how a detection is generated from the state of a person. Since a detector does not always perfectly capture the world, the detection model is inherently probabilistic. In the case of people detection the detector typically gives the 2D location of the person on the ground with the addition of zero mean Gaussian noise modeling the possible error in position. Hence, a general discrete time detection model can be described by the following equation:

$$\mathbf{z}_k = \boldsymbol{H}\mathbf{x}_k + \mathbf{v}_k \tag{4.14}$$

Here $\mathbf{x}_k$ is, as used before, the random variable describing the state of a person, $\mathbf{z}_k$ represents the detection, $\boldsymbol{H}$ is the observation matrix extracting the measurement from the state and $\mathbf{v}_k \sim \mathcal{N}(0, \boldsymbol{C}_{\mathbf{z}_k})$ is a noise vector. Since the 2D position of the person is already part of its state it is sufficient to use a linear time invariant model with the observation matrix:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.15}$$

During the particle filter correction step in Algorithm 4.3 we utilize the likelihood of a certain detection given the state of a particle to estimate the weight, i.e. the error in sampling from the motion model prior, which is characterized by the pdf $f_{\mathbf{z}_k|\mathbf{x}_k}(\boldsymbol{z}_k|\boldsymbol{x}_k)$. Using the detection model we can predict a detection $\hat{\boldsymbol{z}}_k$ given the belief state $\boldsymbol{x}_k$ of the single particle with $\hat{\boldsymbol{z}}_k = \boldsymbol{H}\boldsymbol{x}_k$. This is a realization, or sample, of the random variable $\hat{\mathbf{z}}_k$ which we will use in the following for a consistent derivation. Furthermore, we know that the received detection is normally distributed with covariance $\boldsymbol{C}_{\mathbf{z}_k}$ and mean $\boldsymbol{\mu}_{\mathbf{z}_k}$. An example is given in Figure 4.8 where the covariance is visualized as 99% ellipse. To measure how far they are apart it is natural to utilize the Mahalanobis distance. The

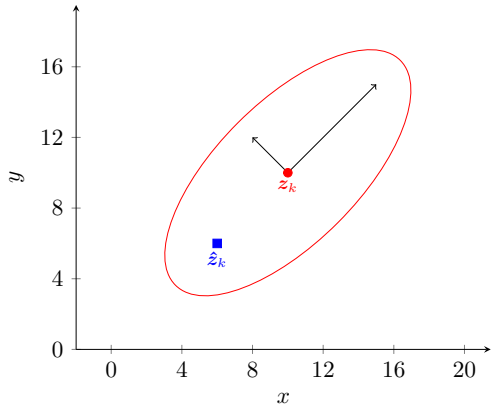


Figure 4.8: Predicted Detection $\hat{\boldsymbol{z}}_k$ and Observed Detection $\boldsymbol{z}_k$

Mahalanobis distance is defined as follows:

$$\mathsf{d}_{\hat{\mathbf{z}}_k} = \sqrt{(\hat{\mathbf{z}}_k - \boldsymbol{\mu}_{\mathbf{z}_k})^\intercal \boldsymbol{C}_{\mathbf{z}_k}^{-1} (\hat{\mathbf{z}}_k - \boldsymbol{\mu}_{\mathbf{z}_k})} \tag{4.16}$$

The intuition behind the Mahalanobis distance is that it gives the distance in terms of standard deviations along the eigenvectors, drawn as arrows in Figure 4.8, of the covariance matrix $\boldsymbol{C}_{\mathbf{z}_k}$ from the mean $\boldsymbol{\mu}_{\mathbf{z}_k}$. To compute the weights we need to give the pdf of the distance values. The squared distances can be represented by a sum of standard normal variables and hence are χ^2 distributed.

Proof. First, we factor $\boldsymbol{C}_{\mathbf{z}_k} = \boldsymbol{A}\boldsymbol{A}^\intercal$ using the Cholesky decomposition.
Furthermore, we substitute $\mathbf{y} = \boldsymbol{A}^{-1}(\hat{\mathbf{z}}_k - \boldsymbol{\mu}_{\mathbf{z}_k})$ in the formula for the squared Mahalanobis distance:

$$\begin{aligned}
\mathsf{d}^2_{\hat{\mathbf{z}}_k} &= (\hat{\mathbf{z}}_k - \boldsymbol{\mu}_{\mathbf{z}_k})^\intercal (\boldsymbol{C}\boldsymbol{C}^\intercal)^{-1} (\hat{\mathbf{z}}_k - \boldsymbol{\mu}_{\mathbf{z}_k}) \\
&= \mathbf{y}^\intercal \boldsymbol{I} \mathbf{y}
\end{aligned}$$

The mean and covariance of $\mathbf{y}$ are calculated as follows:

$$\begin{aligned}
\boldsymbol{\mu}_{\mathbf{y}} &= \mathrm{E}\{\mathbf{y}\} \\
&= \mathrm{E}\{\boldsymbol{A}^{-1}(\hat{\mathbf{z}}_k - \boldsymbol{\mu}_{\mathbf{z}_k})\} \\
&= \boldsymbol{A}^{-1}(\mathrm{E}\{\hat{\mathbf{z}}_k\} - \boldsymbol{\mu}_{\mathbf{z}_k}) \\
&= 0 \\
\boldsymbol{C}_{\mathbf{y}} &= \mathrm{E}\{\mathbf{y}\mathbf{y}^\intercal\} \\
&= \mathrm{E}\{(\boldsymbol{A}^{-1}(\hat{\mathbf{z}}_k - \boldsymbol{\mu}_{\mathbf{z}_k}))(\boldsymbol{A}^{-1}(\hat{\mathbf{z}}_k - \boldsymbol{\mu}_{\mathbf{z}_k}))^\intercal\} \\
&= \boldsymbol{A}^{-1}\mathrm{E}\{(\hat{\mathbf{z}}_k - \boldsymbol{\mu}_{\mathbf{z}_k}))(\hat{\mathbf{z}}_k - \boldsymbol{\mu}_{\mathbf{z}_k})^\intercal\}(\boldsymbol{A}^{-1})^\intercal \\
&= \boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{A}^\intercal(\boldsymbol{A}^{-1})^\intercal \\
&= \boldsymbol{I}
\end{aligned}$$

Hence, the variables of $\mathbf{y}$ are standard normal distributed and $\mathsf{d}^2_{\hat{\mathbf{z}}_k} = \mathbf{y}^\intercal \boldsymbol{I} \mathbf{y} = \sum_{i=0}^{n-1} \mathsf{y}_i^2$ is therefore χ^2 distributed with n degrees of freedom. □

So we can obtain the pdf for the squared distances between detection estimates from a belief and a received detection through the χ^2 distribution with two degrees of freedom. Hence we obtain:

$$f_{\mathbf{z}_k|\mathbf{x}_k}(\boldsymbol{z}_k|\boldsymbol{x}_k) = f_{\mathsf{d}^2_{\hat{\mathbf{z}}_k}}(d^2_{\hat{\boldsymbol{z}}_k}) = \begin{cases} 0 & \text{if } d^2_{\hat{\boldsymbol{z}}_k} < 0 \\ \frac{1}{2} e^{-\frac{d^2_{\hat{\boldsymbol{z}}_k}}{2}} & \text{if } d^2_{\hat{\boldsymbol{z}}_k} \geq 0 \end{cases} \tag{4.17}$$

A plot of the density is given in Figure 4.9. One can see that the density decreases as

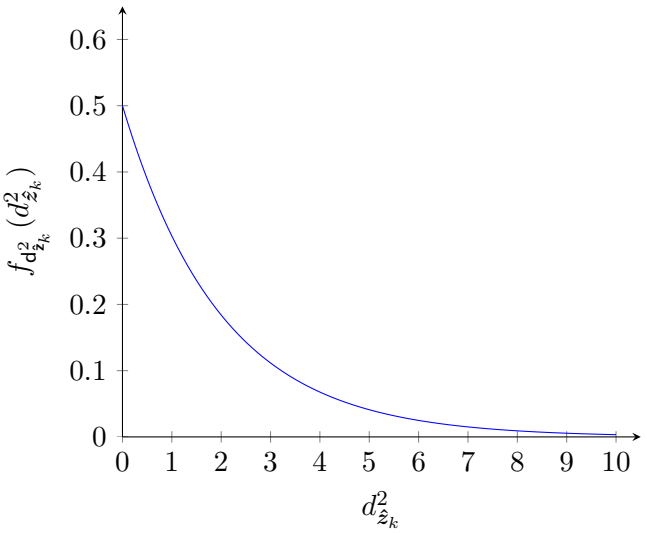
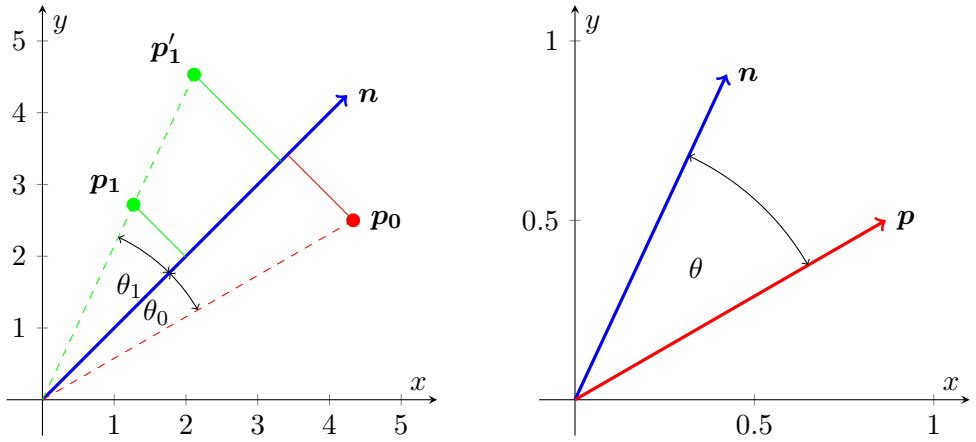


Figure 4.9: χ^2 Distribution with two degrees of freedom for the squared Mahalanobis distance

the Mahalanobis distance increases which is intuitively what we expect.

RGB Sensor Model

The sensor model for RGB camera based detectors differs from the general model since RGB cameras do not provide any depth information and hence the detector cannot provide a good estimate of the persons position. Nevertheless, one can utilize the inverse camera matrix to obtain the ray on which the person can be found. In the following we assume such a ray, projected onto the ground plane is provided by the detector. In theory, one has to consider an infinite amount of possible positions along the direction vector and compare those to the predicted measurement given by a particle. Using a single distance measure, e.g., the normal distance of the particle position to the ray leads to wrong results, namely as the particle distance from the camera origin increases the



(a) Normal distance vs. angular distance (b) Normalized direction vectors

Figure 4.10: On the left, we compare angular and normal distance. Since $\boldsymbol{n}$ is considered an infinite direction vector, angular distance is a better measure for matching detections and tracks. On the right, we show a normalized versin of the direction vector and the vector towards the detection in order to calculate θ.

normal distance also increases if the particle does not already lie on the ray itself. A good measure of whether the predicted detection in form of a particle corresponds to the observed detection is to consider the angular difference between the the ray and a second ray given by the camera origin and the particle position. To illustrate this consider Figure 4.10 where $\boldsymbol{p_0}$ and $\boldsymbol{p_1}$ represent particles and $\boldsymbol{n}$ represents the detection ray, i.e. $\boldsymbol{n}$ has no specific end point. Regarding normal distance, $\boldsymbol{p_1}$ is closer to $\boldsymbol{n}$ than $\boldsymbol{p_0}$, however this is not true anymore if we shift $\boldsymbol{p_1}$ towards $\boldsymbol{p_1'}$ since the angle between $\boldsymbol{p_0}$ and $\boldsymbol{n}$ is actually smaller than between $\boldsymbol{p_1}$ and $\boldsymbol{n}$. Hence, detections occurring closer to the camera origin always have an advantage if the normal distance is utilized. Therefore, we utilize the angle between the detection ray and the particle ray. Figure 4.10b shows an example where $\boldsymbol{n}$ and $\boldsymbol{p}$ are normalized direction vectors for the detection and the particle respectively. The angle θ can then be obtained from those vectors using the dot product:

$$\theta = \arccos(\boldsymbol{n} \cdot \boldsymbol{p}) \tag{4.18}$$

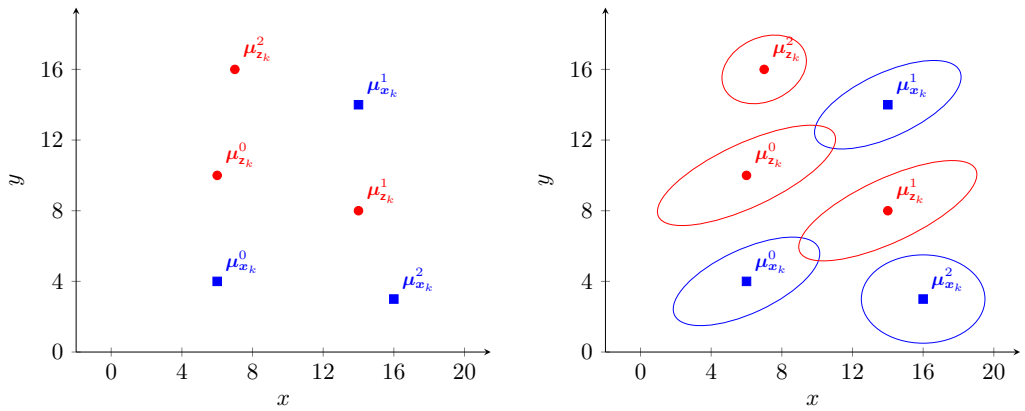
For particle weighting we then assume that the angle is normal distributed with zero mean $\theta \sim \mathcal{N}(0, \sigma_\theta)$ and hence obtain the following density function for the RGB detector model:

$$f_{\mathbf{z}_k|\mathbf{x}_k}(\boldsymbol{z}_k|\boldsymbol{x}_k) = f_\theta(\theta) = \frac{1}{\sqrt{2\pi\sigma_\theta^2}} \exp(-\frac{\theta^2}{2\sigma_\theta^2}) \tag{4.19}$$

4.2.4 Data Association

In the beginning of Section 4.2 we mentioned that the tracking algorithm needs a data association approach to assign detections to tracks. In general there exist approaches that do not rely on a one to one assignment but can assign a detection to multiple tracks in a probabilistic fashion as done with joint probability data association (JPDA) [BSDH09] or consider multiple assignment scenarios through multi hypothesis tracking (MHT) [Rei78][CH96], see also Chapter 2. All those approaches have in common that they are computationally more complex than a one to one assignment obtained through the Munkres algorithm discussed in Section 3.4 which can solve the assignment problem in $O(n^3)$ where n is the number of pairs to match. Since particle filters are used in this work which require significant resources we decided to stick to a one to one assignment for data association to keep overall computational complexity in check. In the following we describe the choice of cost, required by the Munkres algorithm, for assigning a specific track to a certain detection.

Overall, the approach can be classified as a nearest neighbor approach assigning the closest detection and track to one another. We consider two distance measures for the cost of assigning a track to a detection. The most obvious choice is to use a simple euclidean distance between the detection center and the single best estimate of the track represented by its mean. The disadvantage of the euclidean distance is that it completely neglects any statistical information about the tracks and detections. To work around that one can utilize the Mahalanobis distance as we already did for the detection model in Section 4.2.3. The Mahalanobis distance not only takes the track and detection mean into account but also their covariances. Though, application of the Mahalanobis distance assumes that the distributions of tracks and detections are fully characterized by their mean and covariance and hence hypothesized as normal distributions which actually might not be the case. Consider the example in Figure 4.11. On the left hand side



(a) Euclidean distance (b) Mahalanobis distance

Figure 4.11: Data Association Example

track and detection means are drawn without covariances. Tracks are marked by boxes and detections by dots. Considering euclidean distances one would match the following pairs of tracks and detections: (0, 0), (1, 2), (2, 1). In contrast on the right hand side considering covariances it makes more sense to match (0, 1) and (1, 0) while leaving track 2 and detection 2 unmatched for further processing in the track management system.

In contrast to the application in the detection model, where weighting is performed for each particle individually, meaning a particle is a single point in the distribution, we consider the Gaussian estimate of the track belief before the weighting $\hat{\mathbf{x}}_k^i = \overline{bel}(\mathbf{x}_k^i)$ based on all particles with mean $\widehat{\boldsymbol{\mu}}_{\mathbf{x}_k^i}$ and covariance $\widehat{\boldsymbol{C}}_{\mathbf{x}_k^i}$. Therefore, instead of using the detection covariance $\boldsymbol{C}_{\mathbf{z}_k^j}$ in the Mahalanobis distance calculation we need to calculate the actual covariance of the difference between predicted detection $\hat{\mathbf{z}}_k^i$ and observed detection $\mathbf{z}_k^j$.

$$\mathbf{z}_k^j = \boldsymbol{H}\mathbf{x}_k^i + \mathbf{v}_k \tag{4.20}$$

$$\hat{\mathbf{z}}_k^i = \boldsymbol{H}\hat{\mathbf{x}}_k^i \tag{4.21}$$

$$\tilde{\mathbf{z}}_k^{ij} = (\mathbf{z}_k^j - \hat{\mathbf{z}}_k^i) \tag{4.22}$$

$$= \boldsymbol{H}(\mathbf{x}_k^i - \hat{\mathbf{x}}_k^i) + \mathbf{v}_k \tag{4.23}$$

Note that it is assumed that the mean of $\tilde{\mathbf{z}}_k^{ij}$ is zero, meaning that it is expected that the predicted detection matches the observed detection on average.

$$\boldsymbol{C}_{\tilde{\mathbf{z}}_k^{ij}} = \mathrm{E}\{(\mathbf{z}_k^j - \hat{\mathbf{z}}_k^i)(\mathbf{z}_k^j - \hat{\mathbf{z}}_k^i)^\intercal\} \tag{4.24}$$

$$= \mathrm{E}\{(\boldsymbol{H}(\mathbf{x}_k^i - \hat{\mathbf{x}}_k^i) + \mathbf{v}_k)(\boldsymbol{H}(\mathbf{x}_k^i - \hat{\mathbf{x}}_k^i) + \mathbf{v}_k)^\intercal\} \tag{4.25}$$

$$\begin{aligned} = {} & \boldsymbol{H}\mathrm{E}\{(\mathbf{x}_k^i - \hat{\mathbf{x}}_k^i)(\mathbf{x}_k^i - \hat{\mathbf{x}}_k^i)^\intercal\}\boldsymbol{H}^\intercal + \mathrm{E}\{\mathbf{v}_k\mathbf{v}_k^\intercal\} \\ & + \boldsymbol{H}\mathrm{E}\{(\mathbf{x}_k^i - \hat{\mathbf{x}}_k^i)\mathbf{v}_k^\intercal\} + \mathrm{E}\{\mathbf{v}_k(\mathbf{x}_k^i - \hat{\mathbf{x}}_k^i)^\intercal\}\boldsymbol{H}^\intercal \end{aligned} \tag{4.26}$$

Since we assume that the detection noise is independent of the state prediction noise it follows that both $\mathrm{E}\{(\mathbf{x}_k^i - \hat{\mathbf{x}}_k^i)\mathbf{v}_k^\intercal\}$ and $\mathrm{E}\{\mathbf{v}_k(\mathbf{x}_k^i - \hat{\mathbf{x}}_k^i)^\intercal\}$ are zero. Hence, we get:

$$\boldsymbol{C}_{\tilde{\mathbf{z}}_k^{ij}} = \boldsymbol{H}\widehat{\boldsymbol{C}}_{\mathbf{x}_k^i}\boldsymbol{H}^\intercal + \boldsymbol{C}_{\mathbf{z}_k^j} \tag{4.27}$$

Using this covariance we can define the Mahalanobis distance between the mean of a track and a detection. We use the squared version here to penalize errors higher and to later define a gate according to the χ^2 distribution.

$$\mathsf{d}^2_{\tilde{\mathbf{z}}_k^{ij}} = (\hat{\mathbf{z}}_k^i - \mathbf{z}_k^j)^\intercal \boldsymbol{C}^{-1}_{\tilde{\mathbf{z}}_k^{ij}}(\hat{\mathbf{z}}_k^i - \mathbf{z}_k^j) \tag{4.28}$$

For an optimal assignment we want to minimize the overall distance in matchings. Since we described a maximum matching version of the Munkres algorithm in Section 3.4 we convert the distances as follows and put them in a matrix $\boldsymbol{U} = (u_{ij})$:

$$\mathsf{d}^2_{max} = \max_{ij} \mathsf{d}^2_{\tilde{\mathbf{z}}_k^{ij}} \tag{4.29}$$

$$u_{ij} = \mathsf{d}^2_{max} - \mathsf{d}^2_{\tilde{\mathbf{z}}_k^{ij}} \tag{4.30}$$

Algorithm 4.9: Data Association

Input: $\boldsymbol{T}$, $\boldsymbol{Z}$
Output: *assignment*, *assignment_cost*

1 $max = 0$;
2 **for** $i = 0$ *to* $|\boldsymbol{T}| - 1$ **do**
3 **for** $j = 0$ *to* $|\boldsymbol{Z}| - 1$ **do**
4 $\hat{z} = \boldsymbol{HT}[i].\hat{\boldsymbol{\mu}}_{\mathbf{x}}$;
5 **if** $\boldsymbol{Z}.type == RGB$ **then**
6 construct normalized vector $\boldsymbol{z}$ from camera origin towards $\hat{z}$;
7 $\boldsymbol{U}[i,j] = \arccos(\boldsymbol{n} \cdot \boldsymbol{z})$;
8 **else**
9 $\boldsymbol{U}[i,j] = (\hat{z} - \boldsymbol{Z}[j].\mathbf{z}_k^j)^\intercal \boldsymbol{C}^{-1}_{\tilde{\mathbf{z}}_k^{ij}} (\hat{z} - \boldsymbol{Z}[j].\mathbf{z}_k^j)$;
10 **end**
11 **if** $max < \boldsymbol{U}[i,j]$ **then**
12 $max = \boldsymbol{U}[i,j]$;
13 **end**
14 **end**
15 **end**
16 **for** $i = 0$ *to* $|\boldsymbol{T}| - 1$ **do**
17 **for** $j = 0$ *to* $|\boldsymbol{Z}| - 1$ **do**
18 $\boldsymbol{U}_{max}[i,j] = max - U[i,j]$;
19 **end**
20 **end**
21 augment matrix to square size;
22 $assignment = MaxCostAssignment(\boldsymbol{U}_{max}[i,j])$;
23 $assignment\_cost = \boldsymbol{U}$;

As in Section 4.2.3 we again consider RGB camera based detectors as a special case. However, in contrast to the measurement model we not only consider one particle but the whole distribution. Similar to the detection model we construct a ray from the camera origin through the predicted detection $\hat{\mathbf{z}}_k^i$ and calculate the angular difference θ serving as the distance measure between predicted detection and received detection.

The data association procedure is then summarized in Algorithm 4.9 which is used in the tracking algorithm. The algorithm takes an array of tracks and an array of detections. First we compute the squared Mahalanobis distance (lines 2-10) and fill the matrix $\boldsymbol{U}$ while computing the maximum. Then we transform the minimization problem into a maximization one (lines 11-15) using the matrix $\boldsymbol{U}_{max}$. This is then passed to the Munkres algorithm (line 17) which computes maximum cost assignment. Note that we return the matrix $\boldsymbol{U}$ as the assignment cost since we utilize the χ^2 distribution of the Mahalanobis distance for the gating. To define a gating threshold we first define that with probability p a valid detection is inside the gate and hence valid detections fall outside of

the gate with probability $1 - p$. Now consider that the cdf $F_{\mathsf{x}}(A_{thres})$ is the probability that the random variable x is below the threshold A_{thres}, i.e. $\mathsf{x} \leq A_{thres}$. Therefore, the inverse cdf $F_{\mathsf{x}}(p)^{-1}$ gives a value A_{thres} for which $\mathsf{x} \leq A_{thres}$ with probability p. If we choose $p = 0.99$ and by applying the corresponding gating threshold A_{thres} 99% of true detections are inside the gate.

4.2.5 Track Management

Previously, we assumed that there exists a certain amount of tracks in the system which are mapped to detections through the data association. However, we neglected that if there are more detections than tracks it can be the case that another person entered the vicinity of the robot that needs to be tracked and hence a new track has to be initialized. Contrary, there could be tracks that are not matched to any detection by the data association, meaning that a person might have moved out of the robots field of view. Obviously, since we are not dealing with perfect sensors, tracks should not be deleted immediately if there are no detections associated with them in the current cycle and likewise new tracks should not be created from every newly occurred detection due to possible false positives. The track initiation and deletion logic here is based on approaches taken in the publications [BH10] [LGA15].

In the people tracking algorithm 4.1 we use an array of tracks $\boldsymbol{T}_k$ which holds the tracks of the current time step k. For track management we will make use of the variables M_C and D_C in the track tuple $(ID, \boldsymbol{X}, \hat{\boldsymbol{\mu}}_{\mathbf{x}}, \widehat{\boldsymbol{C}}_{\mathbf{x}}, M_C, D_C, V_C, m, m_v)$.

Track Creation

First, we consider the creation of tracks from detections that are not assigned to an existing track by the data association, i.e. they did not pass the gating threshold for any track. The new track is created immediately from the detection, but considered immature and hence not published by the tracking system. To create a track from a detection we initialize a particle filter with particles distributed around the detection according to the detection pdf $f_{\mathbf{z}_k}(\boldsymbol{z}_k)$. We assume that the detection is normal distributed around the value $\boldsymbol{z}_k$ and the covariance $C_{\mathbf{z}_k}$ given by the detector, hence $\mathbf{z}_k \sim \mathcal{N}(\boldsymbol{z}_k, C_{\mathbf{z}_k})$. Additionally, it is assumed that a detection vector $\boldsymbol{z}_k = [x_k, y_k]^\intercal$ only holds the 2D position which is the case for detectors used in this work. Therefore, we can only initialize the 2D position of the state vector using the detection and set the velocities to noise sampled from a zero mean normal distribution. The initialization is given in Algorithm 4.10. After the initial particle set is created (lines 2-6), we also calculate the mean and covariance (line 7) to get an initial estimate. Furthermore, the variables M_C, D_C V_C are initialized to zero (lines 8-10). Finally we add the newly created track to the track array $\boldsymbol{T}_k$ at the current time step (line 11).

All newly created tracks start as immature tracks, to reach maturity we distinguish between two cases. First, a track becomes mature if it is matched with a detection in

Algorithm 4.10: Create Tracks

Input: $\boldsymbol{Z}_{ua}$, $\boldsymbol{Z}_k$, $\boldsymbol{T}_k$

for $i = 0$ *to* $|\boldsymbol{Z}_{ua}| - 1$ **do**
  if $\boldsymbol{Z}_{ua}[i] \neq -1$ **then**
    create new track T;
    for $n = 0$ *to* $N - 1$ **do**
      sample $T.\boldsymbol{X}[n].[\boldsymbol{x}.x, \boldsymbol{x}.y]^\intercal \sim \mathcal{N}(\boldsymbol{Z}_k[\boldsymbol{Z}_{ua}[i]].\hat{\boldsymbol{\mu}}_{\mathbf{z}}, \boldsymbol{Z}_k[\boldsymbol{Z}_{ua}[i]].\widehat{\boldsymbol{C}}_{\mathbf{z}})$;
      $T.\boldsymbol{X}[n].\boldsymbol{x}.\dot{x} = \delta_{\dot{x}}$;
      $T.\boldsymbol{X}[n].\boldsymbol{x}.\dot{y} = \delta_{\dot{y}}$;
    end
    $[T.\hat{\boldsymbol{\mu}}_{\boldsymbol{x}}, T.\widehat{\boldsymbol{C}}]$ = CalculateMeanAndCovariance($T.\boldsymbol{X}$);
    $T.M_C = 0$;
    $T.D_C = 0$;
    $T.V_C = 0$;
    $T.m = false$;
    $T.m_v = false$;
    append T to $\boldsymbol{T}_k$;
  end
end

several cycles of the tracking algorithm. Those cycles are not necessarily consecutive, as long as the track is not deleted in the meantime. The reason for this is that depending on the frame rate of the detectors, detections of a certain person might not occur at every cycle. This ensures that the track reached a certain stability before it is considered to track a real person and can therefore be output by the tracking algorithm as a mature track. To decide whether the track reached maturity we increase the value of M_C every time it is associated with a detection and call it mature if the counter reached a certain threshold *promotion_cycles*.

The second case is that it is often beneficial to only consider visually confirmed tracks if a camera is available, which means that the tracked person was detected through a camera based detector, since laser based detection can lead to a lot of false positives in cluttered environments (see Chapter 2). Visual confirmation is also modeled by a counter V_C which is increased if a camera based detection is associated with the track. The track is said to be visually confirmed if the counter value is greater than zero, i.e. $V_C > 0$. In contrast to the maturity counter, the visual confirmation counter can also be decreased in case a non-camera-based detection is matched with the track. This ensures that false positives from a laser based detector are not visually confirmed forever if the track is assigned a camera based detection once. To prevent that the track is immediately removed from the mature set if a non-camera-based detection occurs directly after a camera-based one, the counter is always increased by two and decreased by one. The

procedure to decide maturity of tracks is stated in Algorithm 4.11.

Algorithm 4.11: Update Track Maturity

Input: T, $\boldsymbol{z}$
if $T[n].M_C \geq promotion\_cycles$ **then**
  $T[n].m = true$;
else
  $T[n].M_C = T[n].M_C + 1$;
end
if $\boldsymbol{z}.type == visual$ **then**
  $T[n].V_C = T[n].V_C + 2$;
else
  $T[n].V_C = T[n].V_C - 1$;
end
if $T[n].V_C > 0$ **then**
  $T[n].m_v = true$;
else
  $T[n].m_v = false$;
end

Track Deletion

Deletion of a track also depends on whether it is assigned a detection or not in several cycles. We again employ a counter D_C which is increased in every cycle a track does not receive an update through an associated detection. Tracks are then deleted if this counter reaches a certain threshold. For track deletion we distinguish between the threshold for immature tracks *deletion_cycles_im* and mature tracks *deletion_cycles_m*, where the threshold for immature tracks is typically smaller, i.e. *deletion_cycles_im* < *deletion_cycles_m*. The deletion threshold parameter for mature tracks can also be considered as the time for which occluded persons are still tracked. Keeping track of an occluded person for a longer period of time enables the tracker to immediately match the track again with a detection, if the person comes back into the field of view of the robot, instead of creating a new track which has to reach maturity again. Track deletion is summarized in Algorithm 4.12.

Algorithm 4.12: Delete Tracks

Input: $\boldsymbol{T}_{ua}$, $\boldsymbol{T}_k$

for $i = 0$ *to* $|\boldsymbol{T}_{ua}| - 1$ **do**
  if $\boldsymbol{T}_{ua}[i] \neq -1$ **then**
    if $\boldsymbol{T}_k[\boldsymbol{T}_{ua}[i]].m == true$ **then**
      if $\boldsymbol{T}_k[\boldsymbol{T}_{ua}[i]].D_C \geq deletion\_cycles\_m$ **then**
        delete $\boldsymbol{T}_k[\boldsymbol{T}_{ua}[i]]$;
      else
        $\boldsymbol{T}_k[\boldsymbol{T}_{ua}[i]].D_C = \boldsymbol{T}_k[\boldsymbol{T}_{ua}[i]].D_C + 1$;
      end
    else
      if $\boldsymbol{T}_k[\boldsymbol{T}_{ua}[i]].D_C \geq deletion\_cycles\_im$ **then**
        delete $\boldsymbol{T}_k[\boldsymbol{T}_{ua}[i]]$;
      else
        $\boldsymbol{T}_k[\boldsymbol{T}_{ua}[i]].D_C = \boldsymbol{T}_k[\boldsymbol{T}_{ua}[i]].D_C + 1$;
      end
    end
  end
end

CHAPTER 5

Implementation

In order to evaluate the people tracking approach introduced in Chapter 4 we implemented the tracking algorithm, including the motion models introduced in Section 3.3 and the heat map creation algorithm as ROS *packages*. ROS is a broad collection of libraries and tools designed for robotics software used in robotics research. The benefit of ROS is that it helps to encapsulate software for a specific purpose into a package, which can then be used as a part of a larger application, enabling reuse of certain packages. Packages in ROS typically contain one or more executables called *nodes*. In addition, ROS defines interfaces between such nodes through *messages*. In the following we describe the implementation of the tracking algorithm and the heat map creation algorithm as ROS nodes in Section 5.1. Furthermore, we discuss ROS messages serving as an interface between people detection ROS nodes and the tracking node. Afterwards, we discuss the robot hardware setup used for our experiments, including sensors and people detection packages in Section 5.3.

All implemented packages as well as auxiliary packages mentioned in this Chapter are available on Github[1].

5.1 People Tracking Package

The `tuw_people_tracking` contains the people tracking ROS node which serves as an interface between the tracking algorithm and other ROS nodes, specifically people detection nodes. It receives detections and publishes tracks. As a common interface for detections and tracks we utilize the `tuw_object_msgs` package, which defines a message for detected objects, `ObjectDetection.msg`. What follows is an overview of the important message fields:

[1] `https://github.com/tuw-robotics`

- **header**: Holds information about the time stamp of the detection and the coordinate frame to which object poses are relative to.
- **type**: Holds the objects type, which are persons in our case.
- **sensor_type**: Holds the type of sensor used in the detection algorithm. This is important to enable visual confirmation.
- **objects**: Holds an array of objects defined by their pose, i.e. position and orientation, their twist, i.e. linear and angular velocity, and the corresponding covariance matrices.

A key thing to note here is that received detections have to be transformed into a common coordinate frame such that they are comparable. In our case it is convenient to transform all detections into the map frame, because representing tracks in the map frame makes working with the heat map motion model easier. Furthermore, the people tracking node enables the use of the `rqt_reconfigure` package to dynamically adjust the tracking parameters. The people tracking node is implemented as a subclass of the actual tracking implementation which is in contrast completely independent of ROS such that it can in principle be used in another framework. Overall, the tracker is implemented according to the Algorithms proposed in Section 2.2. For the heat map representation we utilize the `grid_map`[2] package which comes with a lot of convenient functionality. For the Kuhn Munkres Algorithm used in data association we utilize the implementation provided by the C++ library `dlib`[3]. For visualization of detections and tracks we adapted the plugins for the ROS visualization tool called `rviz` provided by the `spencer_people_tracking` [4] package. Persons can be visualized by the package in multiple ways, for consistency we chose to use 3D bounding boxes with the addition of 99% covariance ellipses and an arrow pointing in the direction of the persons linear velocity. An example visualization is shown in Figure 5.1.

5.2 Heat Map Package

The `tuw_tracking_heatmap` implements the creation of an initial heat map and the integration of historical data as described in Section 4.2.2. First, we utilize the conversion functionality of the `grid_map` package to transform a map of the environment provided by the `map_server` package into a grid map and furthermore into an OpenCV image[5]. OpenCV provides convenient functionality to create the initial heat map using a distance field transform on the map image. Furthermore, OpenCV is used for visualization of the heat maps since it is less resource hungry than visualizing them through the `grid_map rviz` plugin. To record historical data we also utilize the `ObjectDetection.msg`

[2] `https://github.com/ethz-asl/grid_map`
[3] `http://dlib.net/`
[4] `https://github.com/spencer-project/spencer_people_tracking`
[5] `https://opencv.org/`

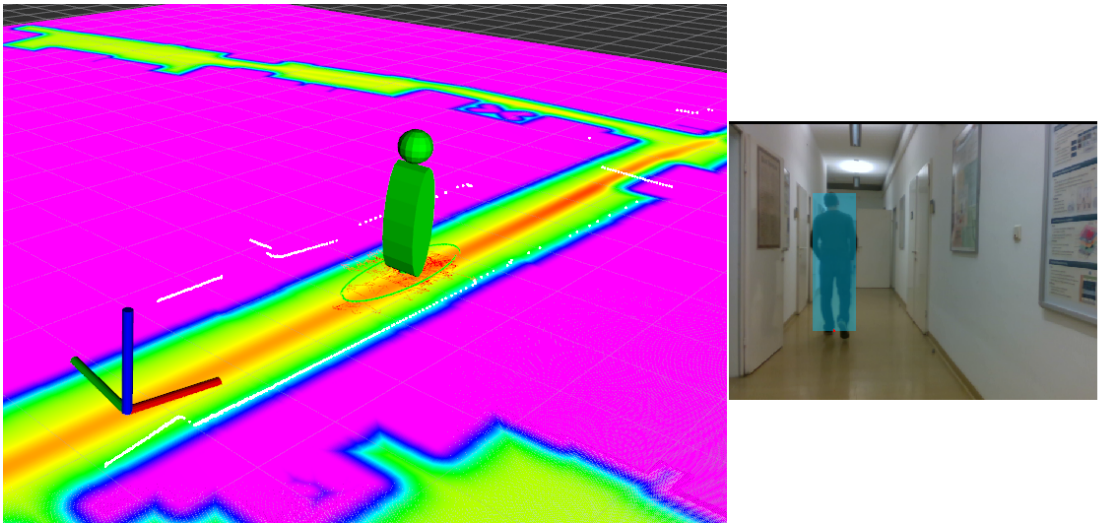


Figure 5.1: Example visualization showing a tracked person in our lab including the underlying heat map on the left. The small red arrows below the person visual mark the corresponding particles for the track. On the right one can see a person being detected in an image annotated by a bounding box.

type such that the heat map node can subscribe to detected or tracked persons from both, a simulation or the people tracking node. The received person positions are then transformed into map coordinates and the corresponding cell in the grid map incremented as described in Algorithm 4.6. The `grid_map` package enables the use of several layers in one map, which is convenient to hold several versions of the heat map. Specifically, we keep the initial likelihood field based heat map, the non blurred and non normalized version of the heat map incorporating historical data and the whole heat map containing the normalized addition of the initial map and the blurred historical data. In order to save the heat map and use it in the tracking algorithm the heat map node exports the heat map into a so called *bag* file, which saves the map as a ROS message. The people tracking node then uses the heat map from the bag file for the heat map motion model.

5.3 Robot Setup

For the tracking experiments we use a Pioneer 3DX robot shown in Figure 5.2 equipped with a Hokuyo laser scanner and an Intel RealSense ZR300 depth camera. Close-up views of the camera and the laser scanner are shown in Figure 5.3. The laser scanner is mounted 0.325 m above the ground such that it can observes the legs of walking people. The Hokuyo laser scanner detects objects at distances ranging from 0.02 m up to 5.6 m. The camera is placed at a height of 1.3 m from the ground intended to observe at least the upper body of close persons. The ZR300 observes a field of view of 75° x 41.5° x 68° for RGB imaging and 70° x 46° x 59° for depth images. The camera is operated with

Figure 5.2: Pioneer 3DX robot evaluation platform

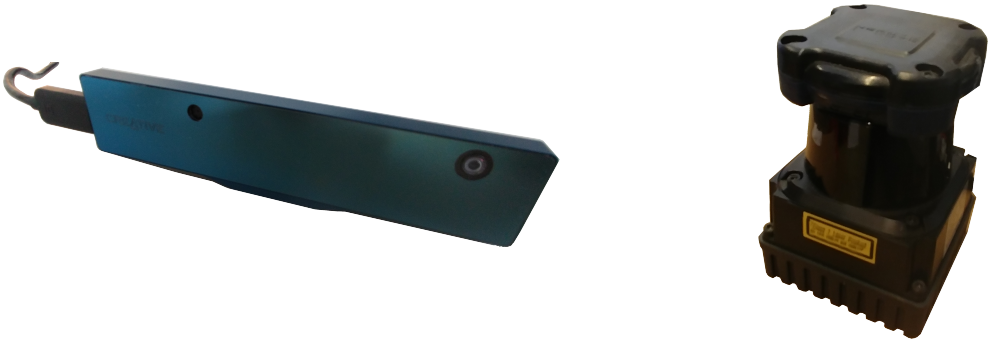
color resolution of 640 x 480 at 30 fps and depth resolution of 480 x 360 at 30 fps. The robot is equipped with an NVIDIA Jetson TX2 developer board, which is capable of running the tracking algorithm on the CPU and additionally provides a GPU utilized by RGB based people detection algorithms.

5.3.1 Detectors

In terms of detection algorithms we use available ROS nodes adapted to publish object detection messages. All detectors used provide only information about a persons location including a covariance estimate, but do not provide velocity or orientation. To compensate for the weaknesses of each sensor, mentioned in Section 2.1, we use three different detectors, one based on monocular RGB images, one based on depth images and one based on data from the laser scanner. The following detector packages are used:

- `leg_detector`[6]: This official ROS package implements a leg detector algorithm based on 2D laser scans similar to what is described in [AMB07]. It clusters points in the laser scan into blobs and identifies which such blobs correspond to legs.

[6] `http://wiki.ros.org/leg_detector`



(a) Intel ZR300 Camera (b) Hokuyo Laser Scanner

Figure 5.3: Separate Figures showing the camera, on the left, and laser scanner, on the right, used on the Pioneer 3DX robot.

Matched pairs of legs are published as persons. Since the leg detector relies on laser scans the published people locations are pretty accurate. Although, the leg detector is prone to false positives in office environments due to tables and chairs being mistakenly classified as persons.

- `ros_deep_learning`: This package provides a ROS wrapper for a deep learning pedestrian detector which is available as a pretrained model from NVIDIA[7]. The detector outputs bounding boxes in the image, of which we use the lowest point to compute the ray mentioned in Section 4.2.3. The detection is pretty accurate from our experience, including people at further distances. However, due to the monocular detection we do not obtain any depth information of how far away a detected person actually is.
- `realsense_people_detection`[8]: Intel provides a people detection implementation for the ZR300 camera based on depth images which we wrapped into a ROS node. Due to the depth range constraints this detector can only detect persons in the range of 0.3 m to 3.0 m, but is very accurate.

[7] `https://github.com/dusty-nv/jetson-inference`
[8] `https://github.com/IntelRealSense/realsense_samples_ros`

CHAPTER 6

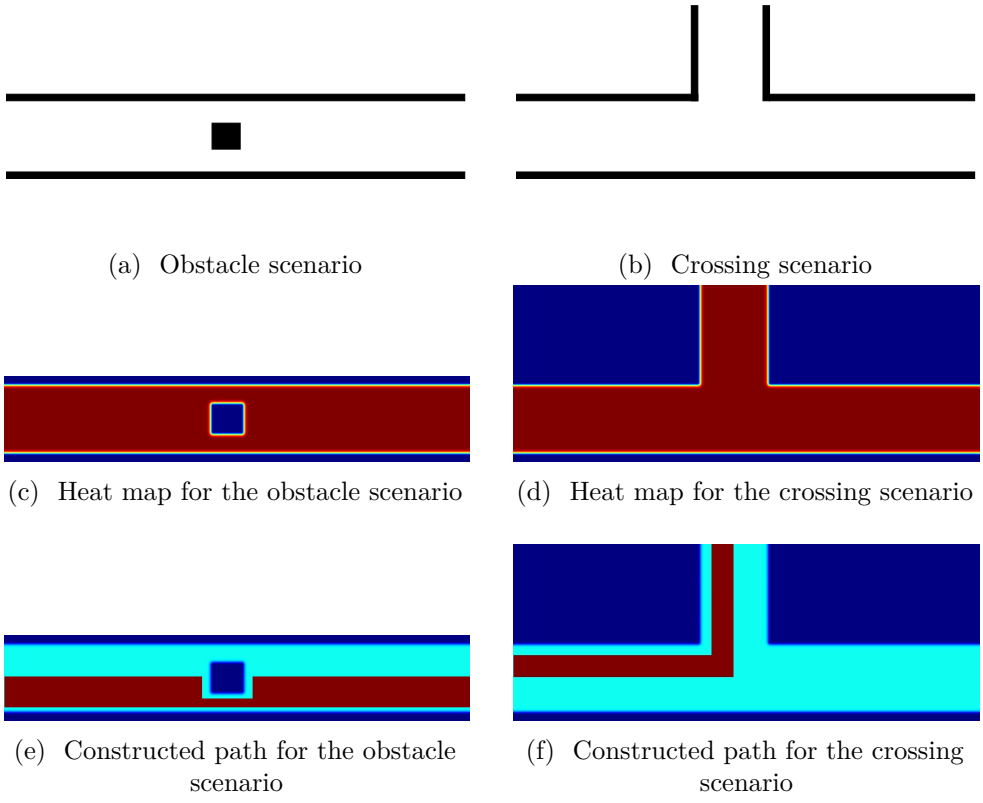
Results

In this chapter we evaluate the prototype implementation of the people tracking approach presented in this thesis. First, we compare commonly used motion models for people tracking described in Section 3.3 with the introduced heat map motion model from Section 4.2.2. We test pure forward prediction without any detections in different map scenarios. Furthermore, we compare the overall approach including detections to state of the art people tracking utilizing the CLEAR MOT metrics discussed in Section 3.5. In Section 6.1 we discuss the conducted experiments and elaborate on our choice of experiments. Results regarding the motion model comparison are discussed in Section 6.2 and results on overall tracking performance in Section 6.3 respectively.

6.1 Experimental Setup

The experiments discussed here are supposed to fulfill two purposes. First, we intend to give the best motion model for pure forward prediction. Comparing forward prediction performance and hence, prior distributions of particles, is important because one wants to use the least amount of particles possible and therefore, not waste to many of them representing unlikely belief states. Furthermore, a solid forward prediction implicitly improves occlusion handling, since predicting a persons position even without obtaining detections, due to the person being occluded, gets more accurate. Properly predicting the location of occluded persons, for example positively influences human aware navigation of the robot.

Secondly, we are of course interested in overall tracking performance. Good performance is defined by minimizing the distance of the trackers belief state of a person to the ground truth and furthermore, minimize the number of missed persons, the number of false positives and the number of mismatches. The CLEAR MOT metrics cover all



(a) Obstacle scenario

(b) Crossing scenario

(c) Heat map for the obstacle scenario

(d) Heat map for the crossing scenario

(e) Constructed path for the obstacle scenario

(f) Constructed path for the crossing scenario

Figure 6.1: Test maps for comparing motion model forward prediction

of those cases and are therefore used to assess the overall performance of the presented people tracking approach.

6.1.1 Comparison of Forward Prediction

To compare forward prediction behavior of different motion models we created artificial map scenarios. We consider forward prediction on a straight corridor with an obstacle present in the center as shown in Figure 6.1a and crossing into another corridor given in Figure 6.1b. The maps are chosen this way since we would like to show that keeping the map in mind leads to a more meaningful distribution of particles, which can therefore lead to a lower number of particles required. Furthermore, we want to emphasize that without detections there are often multiple meaningful paths available for a person to take. Hence, one expects that prior particle distributions cover all of those paths to a certain degree while neglecting highly improbable locations like walls in the static map. In the test setup we initialize a certain number of particles N, specified in the test case, at the starting position and then let the particle filter forward predict the particle states

in fixed time intervals dt. The particles are normally distributed with covariance matrix $\boldsymbol{C}_{init}$ where the initial velocities are completely determined by zero mean Gaussian noise. To assess the heat map motion model we create a likelihood field based heat map for each of the maps in Figure 6.1 according to Algorithm 4.5. In addition, we create heat maps with constructed paths that are more likely to show that the motion model correctly considers those as well.

6.1.2 Comparison of Tracking Performance

In order to assess overall tracking performance of our approach we record data into bag files for certain scenarios. Using recorded data instead of live tracking serves the purpose that annotating ground truth is easier and also results are reproducible, neglecting the inherent randomness of the probabilistic approach. We also test the motion models considered in the previous Section, i.e. constant velocity and coordinated turn. For the scenarios we chose to include two sequences, one where the robot is static and people walk in and out of its field of view and a second one where the robot drives along the corridor of our lab. The heat map utilized here is the one shown in Figure 4.6c introduced in Section 4.2.2. In the following we will refer to the scenarios as static robot scenario, which is about 70 sec long, containing eight tracks and moving robot scenario with similar length and 5 tracks. To annotate ground truth in both scenarios and to evaluate performance using the CLEAR MOT metrics we use the tools provided in the `spencer_people_tracking`[1] package, which gives the advantage that we can compare our results to the ones obtained by the authors of [LBLA16] and [ML12], although their test data sets are not available to us. In order to determine good parameter configurations we run the test cases with different parameters and compare them according to their CLEAR MOT performance values. In terms of detectors we use all three detectors described in Section 5.3 to capture as much of the environment as possible. One thing to note is that, due to restrictions in field of views and range, care must be taken in ground truth annotation to only consider persons actually track-able by the robot.

6.2 Forward Prediction Comparison Results

In what follows we compare the forward prediction results on the maps given in Figure 6.1. For the obstacle map in Figure 6.1a we chose to compare the heat map motion model to the nearly constant velocity motion model, since the scenario is mostly straight forward movement. In the other scenario featuring the crossing shown in Figure 6.1b the heat map motion model is compared to the coordinated turn motion model, such that both can take the possible turn into account. We refrain from comparison with the nearly constant acceleration motion model since it does not give meaningful results in longer forward predicting scenarios without detections. We use the parameters in Table 6.1 for the comparison plots. Tracks are initialized at position (3.0, 1.0) in the test maps with velocities sampled from $\mathcal{N}(0, \boldsymbol{C}_{init})$. We forward predict 150 particles with a constant time

[1] `https://github.com/spencer-project/spencer_people_tracking`

N	σ_x	σ_y	$\boldsymbol{C}_{init}$	p	dt
150	0.1	0.1	diag(0.05)	8	0.25

Table 6.1: Particle Filter and Motion Model Parameters

step of $dt = 0.25$ and plot the particles in the map at times 0, $35dt$, $70dt$, $105dt$ and $140dt$.

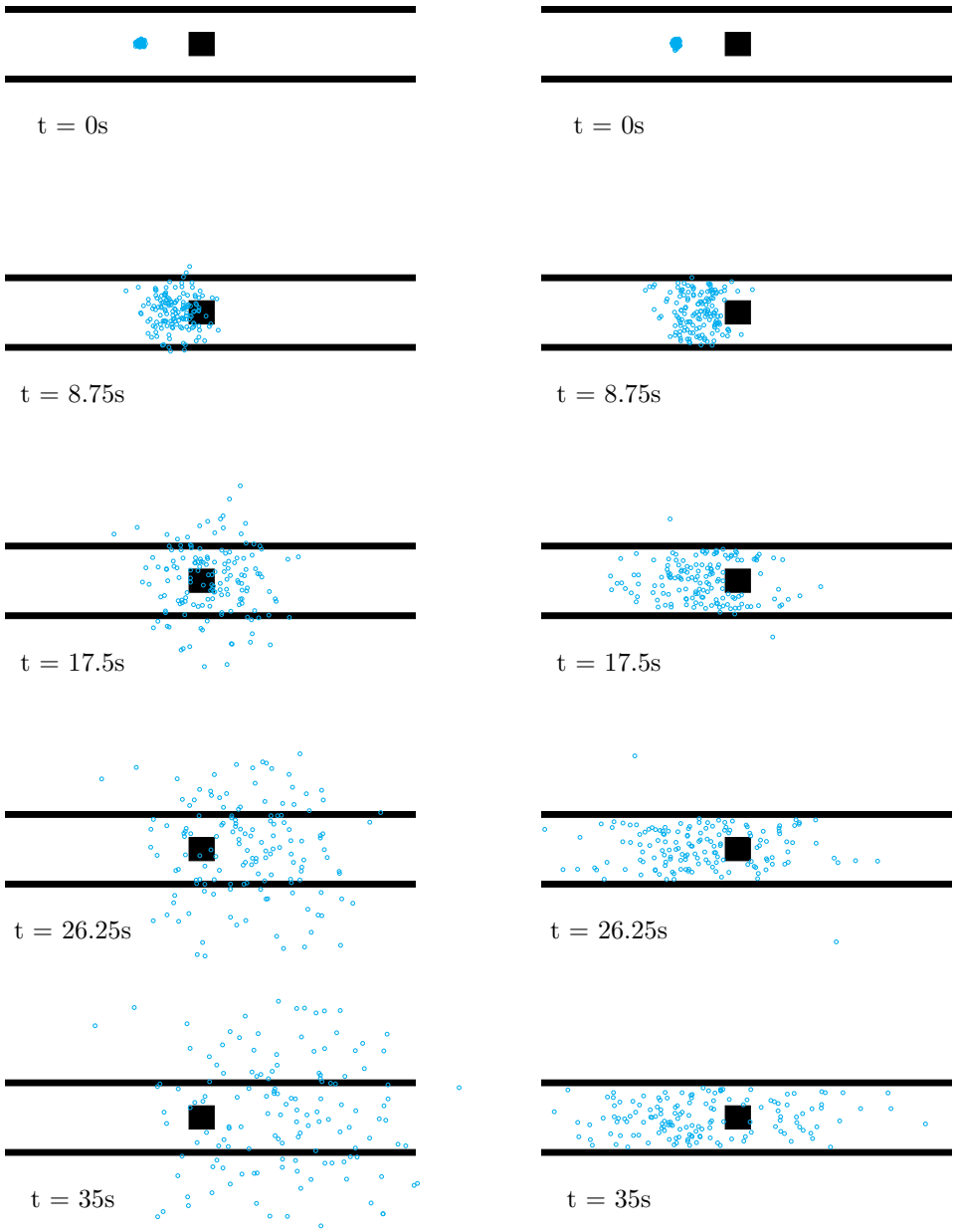
First, consider Figure 6.2 where we compare the constant velocity forward prediction on the left, Figure 6.2a, and heat map forward prediction on the right, Figure 6.2b, using the initial heat map in the obstacle scenario given in Figure 6.1c. One can observe that starting in the second plot at $35dt$ the behavior is still pretty similar, although in the constant velocity prediction a small number of particles already moves outside of the map. As time increases it is obvious that in the heat map motion model case, particles are more condensed and represent rather likely positions in the map, whereas the constant velocity particles move forward keeping their direction as expected, and hence fail to represent a meaningful distribution for a persons position after some time. Furthermore, using the heat map motion model the static object in the middle of the map can successfully be avoided by the forward prediction. Considering, that the constant velocity model behaves similar during small time steps one can argue that frequently received detections justify the use of such a prediction model. However, in scenarios where no detections are available for some time the heat map model provides a more accurate representation of the distribution. One disadvantage of the heat map motion model that can be observed in the experiment is that if the starting position is already likely, a high amount of particles floats around its local neighborhood. This behavior is due to the fact that the heat map motion model considers changes in all directions. The counter measure to apply weights to reward keeping the direction, which is already applied here as introduced in Section 4.2.2, could in principle be changed to reward moving forward even more, however there is a tradeoff in respecting heat map probabilities and ensuring forward movement. Another thing to keep in mind here is that since there is no detection, the person can adjust its velocity and therefore close-by-locations still have to be represented by the distribution.

Similar behavior can be observed in the crossing scenario comparing the coordinated turn forward prediction with the heat map model shown in Figure 6.3. Even though the coordinated turn model, plotted in Figure 6.3a, is able to predict some particles to move in an arc towards the branch in the crossing, it still is by design not able to accurately predict the long term path needed to either move towards the branch or stay on the straight path. Contrary, due to respecting the underlying map, the heat map motion model is able to predict particles along both paths, although as mentioned before there are still more particles representing close-by-locations than locations further away from the initial position.

Finally, consider the results for artificially created likely paths shown in Figures 6.1e and 6.1f. Figure 6.4 gives the plotted particle distributions. One can observe that the

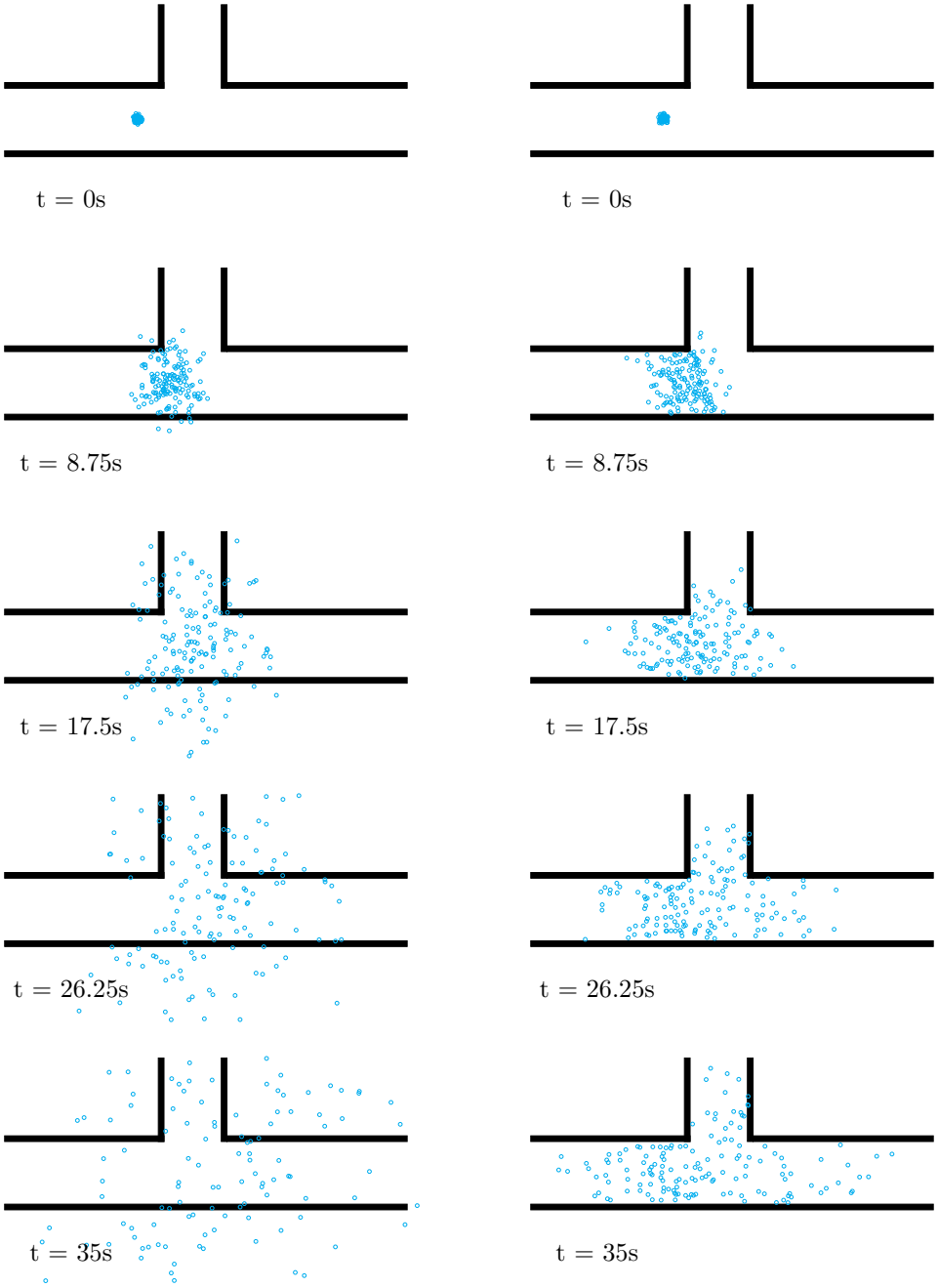
majority of particles follows the introduced paths in both scenarios. In the obstacle scenario, Figure 6.4a particles are more likely to pass under the obstacle as expected. Considering the crossing scenario, Figure 6.4b clearly shows that from the start particles move towards the likely path and take the branching path in the crossing. Since we did not reduce the likelihood of other locations to zero, there are still some particles covering them as expected.

In summary, we can say that if static map information is available, the heat map motion model provided more accurate forward prediction than typically used constant velocity or coordinated turn models. However, if detections are available frequently enough, and occlusions are not likely to happen, the simple models are still a good alternative.



(a) Constant Velocity Forward Prediction (b) Heat Map Forward Prediction

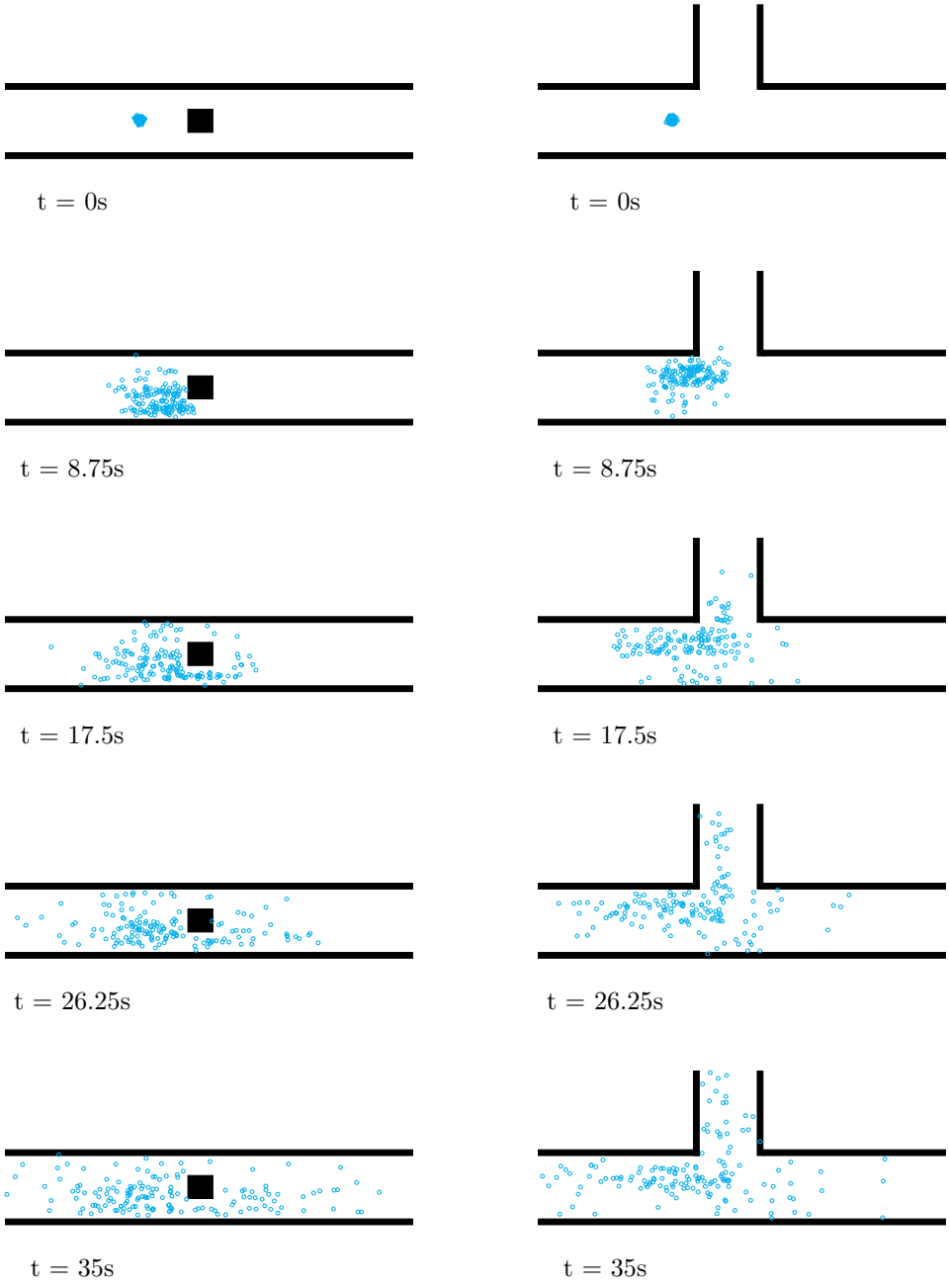
Figure 6.2: Constant Velocity motion model on the left in comparison with the heat map motion model on the right in the obstacle scenario



(a) Coordinated Turn Forward Prediction

(b) Heat Map Forward Prediction

Figure 6.3: Coordinated Turn motion model on the left in comparison with the heat map motion model on the right in the crossing scenario



(a) Obstacle scenario (b) Crossing scenario

Figure 6.4: Heat map motion model forward prediction for both scenarios with constructed paths

6.3 Tracking Performance Comparison Results

In this section we show the results for overall tracking performance using three different motion models, namely CV, CT and the proposed heat map motion model. An example screenshot of the tracking visualization in *rviz* is shown in Figure 6.5.

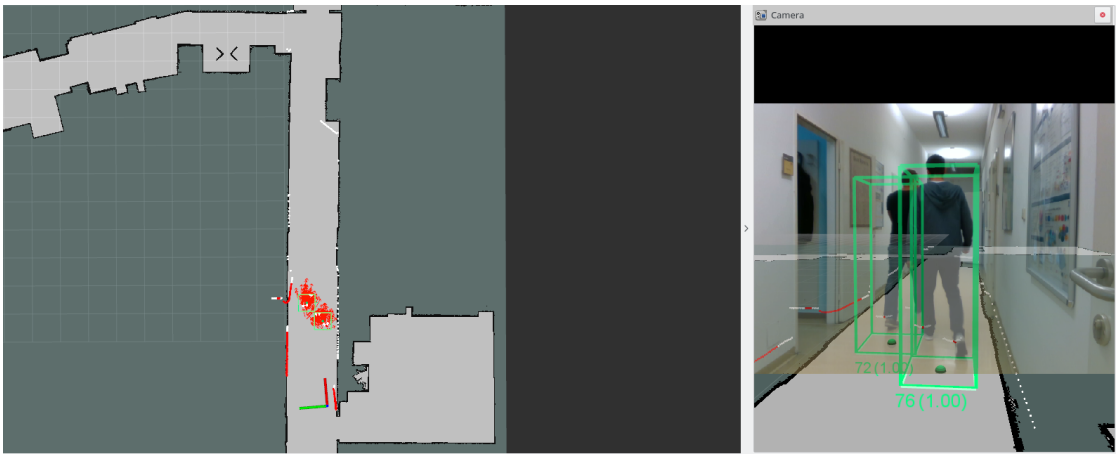


Figure 6.5: Example screenshot of the tracking visualization in *rviz*

6.3.1 Tracking Performance with static Robot

We tested several parameter configurations for each motion model, of which some are summarized in Table 6.2. We chose the configurations here, such that the best performing configuration for each motion model is shown and additionally, the parameter influence can be discussed. The parameters varied here are: the number of particles N, the forward prediction time dt, the standard deviations $\sigma_{\theta'}$ and σ_{ω} for the heat map and CT motion model respectively, the number of deletion cycles for mature and immature tracks d.c.(m) and d.c.(im) after which a track with no corresponding detection is deleted, the promotion cycles p.c. denoting the number of corresponding detections a track has to receive to be considered mature and the gating threshold for data association A_{thres}. For A_{thres} Table 6.2 holds two values, 3.22 which corresponds to 80% probability that a true detection is inside the gate and 9.21 which corresponds to 99% probability. One parameter that is fixed for all configurations is the standard deviation for linear motion noise σ_x and σ_y because it turned out to be a reasonable value for all motion models tested here. The corresponding results to Table 6.2 are given in Table 6.3. In addition, Figure 6.6 shows the best MOTA results with its corresponding MOTP for each motion model. As defined in Section 3.5 we consider the MOTA and the MOTP as tracking metrics. Additionally, the tables states the individual failure rates which are part of the MOTA criteria. Bold entries in the table mark the best value obtained for each motion model. Note that except for the MOTA, lower values mean better performance. Also keep in mind that the MOTP obtained is the euclidean distance between ground truth

case	N	motion model	dt	σ_x, σ_y	$\sigma_{\theta'}$ / σ_ω	d.c.(m)	d.c.(im)	p.c.	A_{thres}
1	100	heat map	0.3	0.1	1.0	30	5	3	9.21
2	100	heat map	0.3	0.1	1.0	60	10	3	9.21
3	100	heat map	0.3	0.1	0.75	30	5	3	9.21
4	50	heat map	0.3	0.1	1.0	30	5	3	9.21
5	100	heat map	0.3	0.1	1.0	30	5	5	9.21
6	100	heat map	0.4	0.1	1.0	30	5	3	9.21
7	100	heat map	0.3	0.1	1.0	30	5	3	3.22
8	100	CV	0.3	0.1	-	30	5	3	3.22
9	100	CV	0.3	0.1	-	60	10	3	3.22
10	50	CV	0.3	0.1	-	30	5	3	3.22
11	100	CV	0.3	0.1	-	30	5	5	3.22
12	100	CV	0.4	0.1	-	30	5	3	3.22
13	100	CV	0.3	0.1	-	30	5	3	9.21
14	100	CT	0.4	0.1	0.1	30	5	3	3.22
15	100	CT	0.4	0.1	0.1	60	10	3	3.22
16	100	CT	0.4	0.1	0.05	30	5	3	3.22
17	50	CT	0.4	0.1	0.1	30	5	3	3.22
18	100	CT	0.4	0.1	0.1	30	5	5	3.22
19	100	CT	0.3	0.1	0.1	30	5	3	3.22
20	100	CT	0.4	0.1	0.1	30	5	3	9.21

Table 6.2: Test configurations for the static scenario

tracks and the matched track in meters. Through the bold entries one can immediately

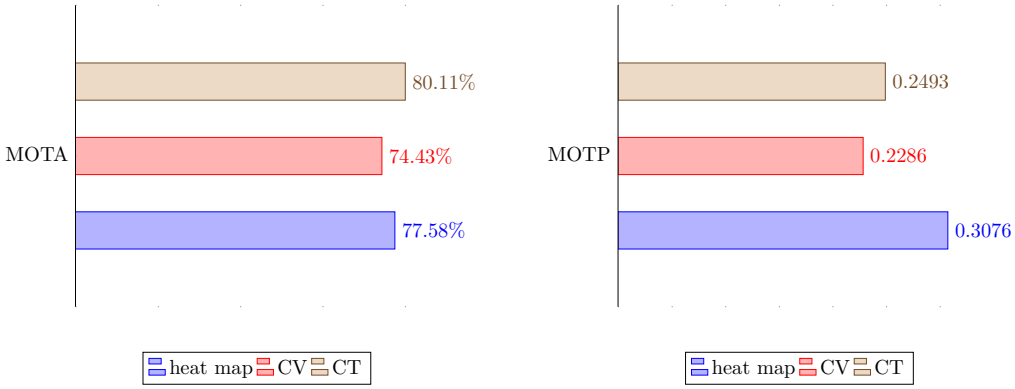


Figure 6.6: Test result plots for the static scenario, showing the best case for each motion model

see that there is not always a clear overall winner, except for the CT motion model, which also overall performs best compared to the other motion models. Now we discuss the influence of the individual parameters in comparison to the first and best performing

case	fp rate	mismatch rate	miss rate	MOTA	MOTP
1	8.43%	0.28%	**13.70%**	**77.58%**	0.3076
2	20.11%	0.36%	15.19%	64.34%	0.3349
3	6.93%	0.21%	17.26%	75.60%	**0.2807**
4	10.22%	0.21%	22.57%	67.00%	0.3280
5	8.63%	0.48%	20.81%	70.08%	0.3625
6	9.03%	0.65%	18.79%	71.53%	0.3579
7	**6.91%**	**0.14%**	16.60%	76.36%	0.2988
8	**7.27%**	0.14%	18.16%	**74.43%**	**0.2286**
9	15.37%	**0.07%**	**15.65%**	68.90%	0.2478
10	7.77%	0.27%	24.74%	67.22%	0.3032
11	7.91%	0.25%	26.23%	65.61%	0.2535
12	8.86%	0.47%	20.90%	69.77%	0.2846
13	10.21%	0.22%	27.10%	62.47%	0.2769
14	**7.24%**	**0.07%**	**12.57%**	**80.11%**	**0.2493**
15	16.06%	0.40%	19.44%	64.10%	0.3311
16	7.43%	0.14%	15.58%	76.84%	0.2713
17	10.77%	0.41%	29.05%	59.77%	0.3062
18	7.96%	0.35%	20.58%	71.11%	0.2748
19	5.85%	0.24%	20.47%	73.44%	0.3179
20	7.32%	0.33%	25.22%	67.13%	0.2993

Table 6.3: Test results for the static scenario

entry in terms of MOTA for each motion model.

First, consider the increase in deletion cycles, i.e. test cases 2, 9 and 15. For each model the main impact is that it significantly increases the false positive rate. This is as expected since a false positive occurs if there is a hypothesis track with no matching ground truth track, which is often the case if we keep tracks without corresponding detections alive for a higher number of tracking cycles. Here, one must be careful in accessing the tracking performance because the increase in deletion cycles also helps to cope with a lack of proper detections or occlusions. Hence, even though the increase in deletion cycles lowered the MOTA here, such a configuration might still generalize better to other situations.

The second parameter we consider is the standard deviation for the angular change $\sigma_{\theta'}$ in the heat map motion model occurring in test case 3. With $\sigma_{\theta'} = 0.75$ the forward prediction of the heat map motion model is much more focused on very small angular changes. Since the scenario is a narrow corridor where people do not heavily change direction one would expect this model to perform better overall, which it does except for the miss rate. An increased miss rate without an increase in false positives means that there actually is no hypothesis at some point for a ground truth track. A possible reason for that is a failed data association leading to early deletion of a track.

Similarly, we examine the standard deviation change for the rotational velocity ω in the CT model in test case 16. Here we can see that the peaked distribution leads to slightly worse results overall, but again the miss rate is increased more than the false positive rate, possibly for the same reason.

Reducing the number of particles to 50 affects the tracking performance for all motion model, which is exactly what one would expect. Interestingly, the CT motion model seems to perform the worst with less particles. The main problem again seems to be incorrect data association leading to a high miss rate for all three motion models, which occurs more often if the motion model prior is not accurate enough.

The effect of the number of promotion cycles is fairly obvious for all motion models. Although, more promotion cycles lead to more stable tracks, it actually delays track promotion and hence, leads to a higher miss rate.

Regarding the forward prediction time dt, our experiments showed that the best value here is not equal for every motion model. While the heat map and the CV model both perform best with dt = 0.3 s, the CT motion model benefits from a slightly higher forward prediction time of dt = 0.4 s. Test cases 6, 12 and 19 show that a slight increase or decrease respectively immediately leads to worse performance throughout.

Finally, we take a look at the gating threshold A_{thres}, where again one value does not perform equally good for all motion models tested. As one can observe, for the heat map motion model a higher threshold value is preferred, whereas the CV and CT model benefit from a lower threshold.

Overall, we can say that all three motion models performed similar, however both the proposed heat map motion model and the CT model have an edge over the more simple CV motion model. The CT motion model also outperformed the proposed heat map motion model. If we compare our results to the ones in [LBLA16], the tracking performance of our particle filter based approach is similar suggesting that the approach taken is valid, however due to the difference in test scenarios and detectors used, care must be taken in direct comparison. In comparison with the approach in [ML12] our MOTA and false positives are slightly better than what they achieved on the RGB-D people dataset [SA11], wheres our MOTA results are worse than what the authors reported for their own datasets. One thing to keep in mind here is that such results heavily depend not only on the dataset, but also on the detection algorithms used.

6.3.2 Tracking Performance with moving Robot

As mentioned previously, we did not only test the static tracking performance, but also recorded a dynamic scenario where the robot is moving around. Since the general effect of parameters does not change whether the robot is static or dynamic, we here only give

results of one good performing set of parameters for each motion model. Parameters are stated in Table 6.4. Corresponding results are given in Table 6.5. In addition, the MOTA and MOTP are plotted in Figure 6.7. For the dynamic scenario, a bit of parameter

case	N	motion model	dt	σ_x, σ_y	$\sigma_{\theta'}$ / σ_ω	d.c.(m)	d.c.(im)	p.c.	A_{thres}
1	100	heat map	0.25	0.1	0.5	30	5	3	3.22
2	100	CV	0.3	0.1	-	30	5	3	3.22
3	100	CT	0.25	0.1	0.1	30	5	3	3.22

Table 6.4: Test configurations for the dynamic scenario

adjustment had to be done, especially concerning forward prediction time, because of the moving robot. In case of the heat map motion model we also observed that a reduction of the standard deviation $\sigma_{\theta'}$ for the angular change resulted in quite a performance improvement. As one can see from the results, the performance difference between

case	fp rate	mismatch rate	miss rate	MOTA	MOTP
1	2.51%	**0.00%**	13.93%	83.56%	0.2595
2	**1.93%**	0.07%	**13.30%**	**84.70%**	**0.2570**
3	2.99%	0.07%	15.63%	81.32%	0.2854

Table 6.5: Test results for the dynamic scenario

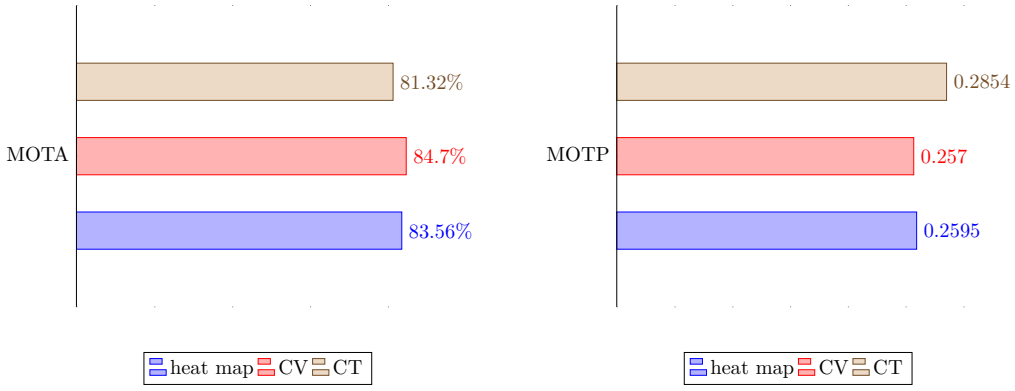


Figure 6.7: Test result plots for the dynamic scenario

motion models according to the CLEAR MOT metrics is pretty small. Interestingly, the CV model provided the overall best performance here, although by quite a small margin.

As we have seen in Section 6.2 the heat map motion model has significantly better forward prediction without the presence of detections. Hence, in order to test this effect we repeat the tests for the dynamic scenarios and drop some of the detections. The most accurate detections in terms of 3D locations are provided by the laser based leg detector, therefore we decided to drop two out of three detection frames using the ROS

drop tool. Furthermore, we only dropped one out of three detections for both camera based detections since dropping two third resulted in mostly non existing tracks because of the visual confirmation needed. In order to maintain the tracks and not delete them between detections we increased the deletion cycles for mature and immature tracks to 90 and 30 respectively. The results are given in Table 6.6. The MOTA and MOTP for those results are plotted in Figure 6.8. As one can see, having a lower amount of

case	fp rate	mismatch rate	miss rate	MOTA	MOTP
1	**4.78%**	0.07%	**22.79%**	**72.36%**	0.3165
2	38.71%	0.07%	30.97%	30.26%	**0.2852**
3	20.29%	0.07%	26.97%	52.67%	0.3577

Table 6.6: Test results for the dynamic scenario with a reduced number of detections

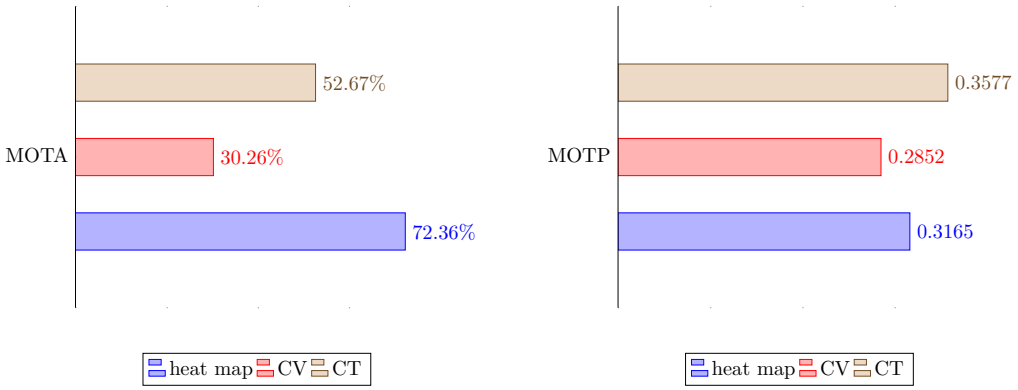


Figure 6.8: Test result plots for the dynamic scenario with a reduced number of detections

accurate detections results in worse tracking performance overall for all three motion models. However, the heat map motion model is able to maintain significantly lower false positive and miss rates due to its superior forward prediction. To conclude, we can say that depending on the detection type, as well as detection frequency the heat map motion model can be useful to maintain better tracking performance. Nonetheless, if detections occur frequently enough, classic motion models lead to similar performance. One more thing to note here is that, in case one intents to keep tracks alive for longer outside of the robots field of view, for example to still consider them in the navigation algorithm, the heat map motion model can also provides a more accurate representation of the track.

6.4 Summary

In this chapter we presented the evaluation results of the proposed people tracking approach, including heat map motion model. We established, that in comparison to

the CV and CT motion model, the heat map model provides more accurate forward prediction. Overall tracking performance was examined both with a static robot as well as with the robot moving through the environment. Results show that, if detection rates are small, the heat map motion model outperforms the CV and CT model, whereas high detection rates lead to similar performances of all three models.

The next chapter concludes the thesis. We give a short summary of the thesis and then discuss possible future work in order to improve the proposed approach.

CHAPTER 7

Conclusion

This chapter concludes the thesis. We first give a summary of the people tracking approach, including the heat map motion model proposed in this thesis and the experimental results. In addition we provide some thoughts on possible future work to improve the tracking approach and also in utilizing the capability of particle filters in representing multi modal distributions to provide more information to navigation algorithms.

7.1 Summary

In this work we proposed a particle filter based approach to track people from a mobile robot platform. We first discussed the general probabilistic model for including different detectors. Then the tracking algorithm was described, including the modeling of detector input, the nearest neighbor based data association strategy applied as well as our approach to create new tracks and deleting obsolete tracks. In addition, a major focus was in developing the heat map motion model, which incorporates static map information as well as historical data of people observed in an environment. Our results first of all show that forward prediction using the heat map motion model outperforms classic approaches using CV or CT motion models. In case of overall tracking performance, we saw that if ground truth annotation is done with respect to what the robot can actually observe and therefore track, all three motion models perform similarly well, given that detections come in frequently to correct forward prediction. When detection frequency was reduced, the superior forward prediction quality of the heat map motion model lead to much better tracking performance, showing the advantages of the proposed motion model. We not only tested with a static robot, but also had the robot move around in the environment tracking people. The results show, that the proposed tracking approach performs similarly well if the robot is moving.

7.2 Future Work

In the results we saw that wrong data association can cause a high number of track misses, hence it seems worth to consider more advanced strategies such as MHT instead of the simple nearest neighbor data association. Another improvement might be to incorporate detections when sampling new particles in the motion model, similar to what is done in FastSLAM 2.0 [MTKW03]. For detectors obtaining accurate world coordinates, like RGB-D based approaches this might lead to more accurate posteriors and hence, more accurate tracking. This can especially be worth for scenarios where people suddenly accelerate and the motion model does not keep up, due to modeling acceleration through small noise terms. Although, we used particle filters to represent people posteriors in this work, we still only output mean and covariance, because common evaluation tools as well as commonly used navigation approaches can only deal with single best estimates. Here, one can make use of multi modal distributions to output several possible time locations of people, especially in case of occlusion, to provide more data for safe navigation around humans. One strategy could for example be to consider a certain amount of particles with the highest weights and output them as possible persons that have to be considered. Furthermore, it would be interesting to include our people tracking approach into a navigation framework to investigate its impact on moving in populated environments.

List of Figures

1.1 The people tracker integrates several detection algorithms to give a good estimate of a person's current position and velocity in the environment. . 3

2.1 This Figure shows a detected person using a leg detector based on [AMB07]. In red one can see the laser scan. The yellow dot marks a detected person with the 99% covariance matrix visualized by the circle. 8
2.2 This Figure shows a person detected in an image by a deep learning based pedestrian detector. 9
2.3 This Figure shows shows a color image on the left with its corresponding depth map on the right obtained using an Intel RealSense ZR300 camera. In the depth image the closer an object is to the camera, the brighter it appears. 10

3.1 Example of 2D Gaussian pdf 19
3.2 Dynamic Bayesian Network for Filtering 22

4.1 Dynamic Bayesian Network for People Tracking 38
4.2 Dynamic Bayesian Network representing a single person detected by two detectors 39
4.3 This Figure illustrates the general structure for the tracking algorithm. Several detections are received as an input, which are passed through the data association procedure. Based on the assignments, the track management block can decide to create or delete tracks, as well as update existing tracks based on their assigned detections. 41
4.4 This Figure visualizes the steps taken in order to construct an initial heat map based on a grid map. 46
4.5 Jet Color Map 46
4.6 Heat map historical data 48
4.7 Given the current velocity vector r at an angle θ, we consider different changes in direction θ' to end up with a new velocity vector r'. 50
4.8 Predicted Detection $\hat{\boldsymbol{z}}_k$ and Observed Detection $\boldsymbol{z}_k$ 53
4.9 χ^2 Distribution with two degrees of freedom for the squared Mahalanobis distance 54

4.10 On the left, we compare angular and normal distance. Since $\boldsymbol{n}$ is considered an infinite direction vector, angular distance is a better measure for matching detections and tracks. On the right, we show a normalized versin of the direction vector and the vector towards the detection in order to calculate θ. 55
4.11 Data Association Example 56

5.1 Example visualization showing a tracked person in our lab including the underlying heat map on the left. The small red arrows below the person visual mark the corresponding particles for the track. On the right one can see a person being detected in an image annotated by a bounding box. 65
5.2 Pioneer 3DX robot evaluation platform 66
5.3 Separate Figures showing the camera, on the left, and laser scanner, on the right, used on the Pioneer 3DX robot. 67

6.1 Test maps for comparing motion model forward prediction 70
6.2 Constant Velocity motion model on the left in comparison with the heat map motion model on the right in the obstacle scenario 74
6.3 Coordinated Turn motion model on the left in comparison with the heat map motion model on the right in the crossing scenario 75
6.4 Heat map motion model forward prediction for both scenarios with constructed paths 76
6.5 Example screenshot of the tracking visualization in *rviz* 77
6.6 Test result plots for the static scenario, showing the best case for each motion model 78
6.7 Test result plots for the dynamic scenario 81
6.8 Test result plots for the dynamic scenario with a reduced number of detections 82

List of Tables

6.1 Particle Filter and Motion Model Parameters 72
6.2 Test configurations for the static scenario 78
6.3 Test results for the static scenario 79
6.4 Test configurations for the dynamic scenario 81
6.5 Test results for the dynamic scenario 81
6.6 Test results for the dynamic scenario with a reduced number of detections 82

List of Algorithms

3.1 General Particle Filter Algorithm as in [Thr02] 23
3.2 Low Variance Sampling Algorithm as in [Thr02] 24
3.3 Initialize 32
3.4 Compute Slack 32
3.5 Update Labels 33
3.6 Kuhn-Munkres Algorithm 34

4.1 People Tracking Algorithm 43
4.2 Particle Filter Algorithm for People Tracking: Prediction 45
4.3 Particle Filter Algorithm for People Tracking: Correction 45
4.4 Calculate Mean and Covariance 45
4.5 Compute initial heat map 47
4.6 Aggregate historical data 49
4.7 Combine historical data and initial heat map 49
4.8 Generate Forward predicted sample 51
4.9 Data Association 58
4.10 Create Tracks 60
4.11 Update Track Maturity 61
4.12 Delete Tracks 62

Bibliography

[AGLB08] K. O. Arras, S. Grzonka, M. Luber, and W. Burgard. Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities. In *2008 IEEE International Conference on Robotics and Automation*, pages 1710–1715, May 2008.

[AGR+16] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, June 2016.

[AM15] I. Ardiyanto and J. Miura. Human motion prediction considering environmental context. In *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*, pages 390–393, May 2015.

[AMB07] K. O. Arras, O. M. Mozos, and W. Burgard. Using boosted features for the detection of people in 2D range data. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3402–3407, April 2007.

[AMGC02] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, Feb 2002.

[AS08] S. Ali and M. Shah. *Floor Fields for Tracking in High Density Crowd Scenes*, pages 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[BBCT05] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun. Learning motion patterns of people for compliant robot motion. *Internationl Journal of Robotics Research*, 24:31–48, 2005.

[Ber99] N. Bergman. *Recursive Bayesian Estimation: Navigation and Tracking Applications*. Linköping studies in science and technology: Dissertations. Department of Electrical Engineering, Linköping University, 1999.

[BH10] N. Bellotto and H. Hu. Computationally efficient solutions for tracking people with a mobile robot: an experimental evaluation of bayesian filters. *Autonomous Robots*, 28(4):425–438, 2010.

[BS08] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):246309, May 2008.

[BSDH09] Y. Bar-Shalom, F. Daum, and J. Huang. The probabilistic data association filter. *IEEE Control Systems*, 29(6):82–100, Dec 2009.

[BSLK04] Y. Bar-Shalom, X. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. Wiley, 2004.

[CH96] I. J. Cox and S. 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, Feb 1996.

[DBJH15] C. Dondrup, N. Bellotto, F. Jovan, and M. Hanheide. Real-time multisensor people tracking for human-robot spatial interaction. In *International Conference on Robotics and Automation (ICRA) - Workshop on Machine Learning for Social Robotics*, 2015.

[DdFG01] A. Doucet, N. de Freitas, and N. Gordon. *An Introduction to Sequential Monte Carlo Methods*, pages 3–14. Springer New York, New York, NY, 2001.

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[FEP+16] D. Fischinger, P. Einramhof, K. Papoutsakis, W. Wohlkinger, P. Mayer, P. Panek, S. Hofmann, T. Koertner, A. Weiss, A. Argyros, and M. Vincze. Hobbit, a care robot supporting independent living at home: First prototype and lessons learned. *Robotics and Autonomous Systems*, 75:60 – 78, 2016. Assistance and Service Robotics in a Human Environment.

[FHM02] A. Fod, A. Howard, and M. A. J. Mataric. A laser-based people tracker. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 3, pages 3024–3029, 2002.

[Fou17] O. S. R. Foundation. Robot operating system. http://www.ros.org, June 2017. Accessed: 29.05.2017.

[Gri17] K. Grill. Lecture notes in probability theory and stochastic processes for informatics (in german). `https://institute.tuwien.ac.at/fileadmin/t/mathstoch/upload/wsi.pdf`, January 2017.

[HFMV02] D. Helbing, I. J. Farkas, P. Molnar, and T. Vicsek. Simulation of pedestrian crowds in normal and evacuation situations. *Pedestrian and evacuation dynamics*, 21(2):21–58, 2002.

[HM14] F. Hlawatsch and G. Matz. Lecture Notes in Signal Processing 2. Institute of Telecommunications Vienna University of Technology, October 2014.

[JML14] O. H. Jafari, D. Mitzel, and B. Leibe. Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5636–5643, May 2014.

[KLF+02] M. Kleinehagenbrock, S. Lang, J. Fritsch, F. Lomker, G. A. Fink, and G. Sagerer. Person tracking with a mobile robot based on multi-modal anchoring. In *Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication*, pages 423–429, 2002.

[KPAK13] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch. Human-aware robot navigation: A survey. *Robot. Auton. Syst.*, 61(12):1726–1743, December 2013.

[Kuh55] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[KUS03] P. Konstantinova, A. Udvarev, and T. Semerdjiev. A study of a target tracking algorithm using global nearest neighbor approach. In *Proceedings of the 4th International Conference Conference on Computer Systems and Technologies: E-Learning*, CompSysTech '03, pages 290–295, New York, NY, USA, 2003. ACM.

[LBLA16] T. Linder, S. Breuers, B. Leibe, and K. O. Arras. On multi-modal people tracking from mobile platforms in very crowded and dynamic environments. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5512–5519, May 2016.

[LFH+03] L. Liao, D. Fox, J. Hightower, H. Kautz, and D. Schulz. Voronoi tracking: location estimation using sparse and noisy sensor data. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 1, pages 723–728 vol.1, Oct 2003.

[LGA15] T. Linder, F. Girrbach, and K. O. Arras. Towards a robust people tracking framework for service robots in crowded, dynamic environments. In *Assistance and Service Robotics Workshop (ASROB-15) at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[LJ01] X. R. Li and V. P. Jilkov. A survey of maneuvering target tracking—part III: Measurement models. 2001.

[LJ03] X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking. part I. dynamic models. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4):1333–1364, Oct 2003.

[LSTA10] M. Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras. People tracking with human motion predictions from social forces. In *2010 IEEE International Conference on Robotics and Automation*, pages 464–469, May 2010.

[MBM12] M. Munaro, F. Basso, and E. Menegatti. Tracking people within groups with RGB-D data. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2101–2107, Oct 2012.

[ML12] D. Mitzel and B. Leibe. Close-range human detection and tracking for head-mounted cameras. In *Proceedings of the British Machine Vision Conference*, pages 8.1–8.11. BMVA Press, 2012.

[MTKW03] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI.

[Mun57] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[OTWD04] S. M. Oh, S. Tariq, B. N. Walker, and F. Dellaert. Map-based priors for localization. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2179–2184 vol.3, Sept 2004.

[PR09] V. Prisacariu and I. Reid. fastHOG - a real-time GPU implementation of HOG. Technical Report 2310/09, Department of Engineering Science, Oxford University, 2009.

[PS98] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications Inc., 1998.

[RDGF15] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[Rei78] D. B. Reid. An algorithm for tracking multiple targets. In *1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes*, pages 1202–1211, Jan 1978.

[RF16] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[RN16] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach, Global Edition*. Pearson Education Limited, 2016.

[SA11] L. Spinello and K. O. Arras. People detection in RGB-D data. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[SBFC03] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers. People tracking with mobile robots using sample-based joint probabilistic data association filters. *The International Journal of Robotics Research*, 22(2):99–116, 2003.

[See15] S. Seer. *A unified framework for evaluating microscopic pedestrian simulation models*. PhD thesis, Technische Universität Wien, Karlsplatz 13, 1040 Wien, Österreich, October 2015.

[SL11] P. Sudowe and B. Leibe. Efficient use of geometric constraints for sliding-window object detection in video. In *International Conference on Computer Vision Systems (ICVS'11)*, 2011.

[SMC04] M. Scheutz, J. McRaven, and G. Cserey. Fast, reliable, adaptive, bimodal people tracking for indoor environments. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 2, pages 1347–1352 vol.2, Sept 2004.

[TBF05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. MIT Press, 2005.

[TC05] E. A. Topp and H. I. Christensen. Tracking for following and passing persons. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2321–2327, Aug 2005.

[Thr02] S. Thrun. Particle filters in robotics. In *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*, 2002.

[Thr10] S. Thrun. Toward robotic cars. *Commun. ACM*, 53(4):99–106, April 2010.