FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Mobile Robotik

Sichere und Effiziente Autonome Navigation in Anwesenheit von Menschen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Informatik

eingereicht von

Klaus Buchegger, BSc
Matrikelnummer 1126329

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Wolfgang Kastner
Mitwirkung: Univ.Ass. Dipl.-Ing. Dr.techn. Markus Bader

Wien, 14. Februar 2018

Klaus Buchegger	Wolfgang Kastner

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Mobile Robotics

Safe and Efficient Autonomous Navigation in the Presence of Humans

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Computer Engineering

by

Klaus Buchegger, BSc
Registration Number 1126329

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Wolfgang Kastner
Assistance: Univ.Ass. Dipl.-Ing. Dr.techn. Markus Bader

Vienna, 14th February, 2018

Klaus Buchegger	Wolfgang Kastner

Erklärung zur Verfassung der Arbeit

Klaus Buchegger, BSc
Marktstraße 13, 76726 Germersheim

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, 14. Februar 2018

Klaus Buchegger

Kurzfassung

Damit fahrerlose Fahrzeuge autonom in einer mit Menschen geteilten Umgebung navigieren können, müssen die menschlichen Bewegungen und sozialen Richtlinien (z.B. links überholen) berücksichtigt werden, um eine sichere und effiziente Navigation zu gewährleisten. Diese Arbeit stellt einen prädiktiven Bewegungsregler vor, der auf Menschen reagiert, um die Fahrzeugtrajektorie auf der Steuerungsebene mit einer hohen Aktualisierungsrate zu optimieren. Die Vorhersage von zukünftigen Positionen von Personen ermöglicht es dem System, eine Trajektorie mit diesen Vorhersagen zu optimieren, was eine Sequenz von Motorsteuerungen für eine sanft ausgeführte Bewegung ergibt. Die Vorhersage basiert auf menschlichen Bewegungsdaten, die aus der Beobachtung des Umfeldes gelernt werden. Eine modellprädiktive Regelung wird um Beschränkungsfunktionen erweitert, die eine sichere Entfernung zu Menschen und ihren vorhergesagten zukünftigen Positionen sicherstellen. Weiters um Kostenfunktionen, die Trajektorien bevorzugen, die den sozialen Richtlinien entsprechen. Die Verbesserungen wurden bezüglich Fahrzeit, Weglänge und Mindestabstand zu Personen entlang der geplanten Trajektorien statistisch ausgewertet. Zusätzlich wurde analysiert, wie gleichmäßig die geplanten Trajektorien in Bezug auf Geschwindigkeit und Orientierung waren, da plötzliches Bremsen, Beschleunigen oder ruckartige Drehungen Menschen in unmittelbarer Nähe irritieren könnten. Auf diese Weise konnte gezeigt werden, dass der entwickelte Regler wesentlich besser ist als ein Regler der menschliche Bewegungen nicht beachtet.

Abstract

In order to enable driverless vehicles to navigate autonomously in an environment shared with humans, the human movements and social guidelines (e.g. passing side preferences) have to be considered, to ensure both safe and efficient navigation. This work presents a predictive, human-aware motion controller which optimizes the vehicle trajectory at the control level with a high update rate. Predicting future positions of persons allows the system to optimize a trajectory around those predictions, yielding a sequence of motor controls for a smooth executed motion. The prediction is based on human movement data, learned from observing an environment. A model predictive controller is enhanced with constraint functions that ensure a safe distance to humans and their predicted future positions, and with cost functions that favor trajectories that satisfy social guidelines. The improvements were statistically evaluated using simulation runs in terms of travel duration, path length, and minimum distance to persons along the path. Additionally, we also analyzed how steady the planned trajectories were, in terms of velocity and orientation, as sudden brakes, accelerations or jerks to some direction could irritate humans in close proximity. This way, we are able to show that the developed motion controller performs significantly better than a controller without human-awareness.

Contents

CHAPTER 1

Introduction

In the course of this work, we present the approach of a new navigational planner that allows a robot to navigate amongst humans through an environment. The planner provides trajectories for a robot, such that it moves efficiently around detected persons, while also trying to make the persons feel comfortable in the presence of the robot by adhering to social conventions. This chapter outlines the problem definition of our work, the expected results as well as the taken methodological approach. Chapter 2 gives an overview of related work concerning autonomous navigation in the presence of humans. In Chapter 3, we explain in detail the general problems in human-aware navigation and the associated techniques and approaches used to solve those problems. The implementation of our approach is described in Chapter 4, where we show our proof of concept, followed by the experimental results we achieved with our implementation in Chapter 5. Finally in Chapter 6, we give a summary of our work, the problems we faced and a potential outlook on future work.

1.1 Problem Definition

Human-aware robot navigation is the task of letting a driverless vehicle move through an environment in which humans are present [KPAK13]. This means that the robot has to react properly to the humans and to ensure that no human will be harmed by the robot.

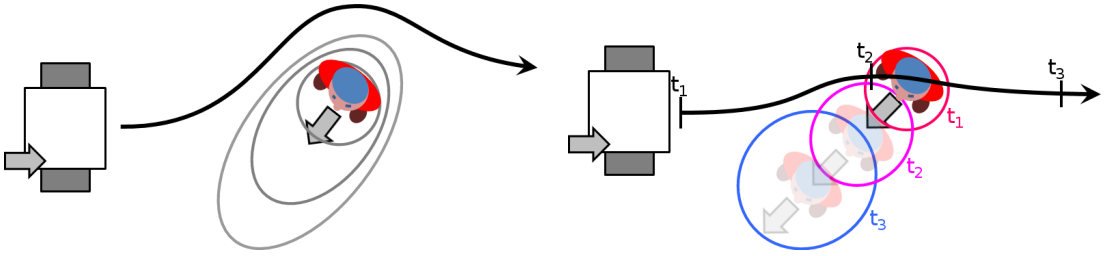
In classic robot navigation, the navigation task is split into two parts. First, the so called *global planner*, that has a prerecorded static map of the environment and tries to find a path from the robot's position to a target position. Typically, such a static map contains information about the position of walls, cabinets, tables, etc. Second, the *local planner*, which collects data from a variety of sensors such as sonars, laser rangefinders or cameras and selects an intermediate trajectory along the global path. In contrast to the global planner, the local planner satisfies the robots movement constraints and dynamics (e.g. limited acceleration or minimum turn radius) and reacts to dynamic changes of the

evironment (e.g. closed doors, obstacles temporarily blocking the path) [RKC98]. *Global planners* usually update their path every few seconds or even only once per defined target and use algorithms such as A* or modified versions of A* [CKB+17]. *Local planners* on the other hand, select trajectories multiple times per second, and are classically based on techniques such as the dynamic window approach (DWA) [FBT97], model predictive control (MPC) [PL15], or simple PID controllers [PL15].

This classic approach can be extended to become human-aware. To efficiently navigate amongst humans, driverless vehicles have to detect human movements and to predict where the people are moving. Therefore, the robots need a motion model that describes human movements. Additionally, these robots need rules to specify how robots are supposed to react to near humans, e.g. what distance to keep, on which side to pass. Another important aspect is whether the human-aware navigation should be handled by the *global* or *local planner*. A *global planner* could predict the movement of all detected persons for the full duration it will take the robot to move to the target. Even though the construction of such long predictions for all persons can be computationally expensive, the low update rate of *global planners* allows such computations. However, since avoiding humans is a safety requirement, a higher update rate might be preferable. Such a high rate could be achieved by incorporating the human-awareness into the *local planner* which then reacts to humans along the shorter, intermediate trajectory instead of along the full path to the robot's target.

1.2 Aim of the Work

The aim of this work is to provide a new approach for a *local planner* for driverless vehicles, that takes into account the movement patterns of humans when searching for an optimal trajectory through an environment. We focus on the *local planner*, or *control level*, as reacting to moving persons is safety critical and needs quick responses. Additionally, predicting human movements over a longer time period is prone to be erroneous, as the intention of the humans cannot be identified by a robot. Therefore, limiting the prediction to a shorter, localized horizon will provide more reliable results. The importance of the prediction is depicted in Figure 1.1. Figure 1.1a shows a robot following a path from left to right, that avoids a detected human by passing to the top. Here, the path is chosen based on a social cost field around the person, expanded along the moving direction of the person. This cost field tells the planner which areas should be avoided, but does not predict the human movement [KSF09]. In Figure 1.1b, the robot is now following a path that is based on movement prediction, in addition to a social cost field. For every timestep t_n, the future position of the person is predicted, and the social cost field is moved to that future position. This allows the planner to generate a path that is split into the same time segments t_n, for each point in time only avoiding the corresponding cost field. As a result, the path reduces the detour to the top by crossing through the person's initial position, as the planner knows the area will no longer be occupied, once the robot reaches that position [KHGB15].



(a) Without human predictions (b) With human predictions

Figure 1.1: Comparison of human-aware navigation approaches.

The human motion model will predict movement of humans based on previously recorded observations in a statistical map. This map contains information that describes for every position, in which direction most people moved when they were observed in that position. With this information, a prediction can be incrementally constructed. This will be an improvement over implementations, that assume constant velocity and orientation for the human movement. Based on this prediction, the robot can then evaluate trajectories with respect to the expected human discomfort for all involved humans. For this purpose, social cost functions will be defined, that describe areas the robot shouldn't enter, similar to the approach of Kollmitz et al. [KHGB15]. Additionally, the movement of the robot should be restricted, in order to follow behaviours that can be predicted by humans. For example, the robot should avoid sudden changes in velocity or movement direction.

As working environment for an implementation of our proposed approach, the Robot Operating System (ROS)[1] will be used. ROS is a modular framework that already contains tools that can be utilized for simple robot movements, self-localization, global planning, static obstacle avoidance, etc. The proposed implementation will extend these functionalities, in order to avoid humans efficiently, without disturbing them.

The implementation is intended to work both in a simulation[2] and on a real Pioneer P3-DX® robot[3]. The result is expected to outperform existing navigation algorithms that only deal with static obstacles, both in movement efficiency as well as concerning human discomfort. Additionally, the use of a statistical motion model will yield more realistic predictions compared to the constant velocity assumption used in publications such as [KHGB15].

[1] ROS: http://ros.org
[2] Simulation will be done with Gazebo, http://gazebosim.org
[3] Pioneer P3-DX: http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx

1.3 Methodological Approach

The scientific contribution of this work will be to study the impact of the statistical prediction of human walking on autonomous navigation. This approach will be analyzed and compared to state-of-the-art strategies both in simulation and in real-world scenarios.

1.3.1 Literature Review

Existing literature and publications will be analyzed in order to find a starting point for the implementation. As human-aware navigation is an actively researched topic, there certainly will be sufficient information available about what kind of movements proved most beneficial when the robot interacts with humans. This collected data will later be used to tune the implementation parameters. Additionally, existing publications may contain commonly used evaluation criteria as well as possible benchmark scenarios, such that the implementation can be properly compared with other solutions.

1.3.2 Navigation Framework

The navigation framework, that is currently used by the work group, treats every detected obstacles as a static object. Therefore, the framework has to be enhanced, such that the movement of possible dynamic obstacles, like humans or other moving robots, will be taken into account during the calculation of the local trajectory. With this enhancement, the robot will be able to find a trajectory that efficiently navigates through moving obstacles. To keep a safe distance to detected person, a constraint function will be implemented, such that a trajectory can only be valid if at any point in time the distance to the closest person is above a certain limit.

1.3.3 Prediction Algorithm for Human Movement

When a human person is detected, the robot has to predict the movement. Usually, it is simply assumed that the person moves with constant velocity, because accurate prediction is complicated, and a topic of its own. In our approach, the prediction will be based on previously recorded statistical data about movement in a specific environment. Based on this data, the future position of the human will depend on where other people usually moved from their current position.

1.3.4 Social Aspects in Human-Aware Navigation

In addition to the pure collision avoidance of persons, we will also consider social aspects, which should increase the social acceptance of a robot navigating in the presence of humans. Currently, the public is not accustomed to the presence of freely roaming robots, therefore when introducing such, it is important that the presence of the robot will be tolerated or even welcomed. To achieve this, the robot should behave similar to humans,

for example by following the same social rules in walking traffic.

1.3.5 Implementation and Evaluation

Finally, the performance of the implementation will be analyzed, both in simulation as well as with a real robot, according to previously researched evaluation criteria. It will also be compared to the performance of the current state of the navigation framework, and possibly other published implementations, if the used scenarios can be reproduced in our environment.

1.4 Summary

In this chapter, we outlined the problem of human-aware navigation of driverless vehicles. Furthermore, we presented our approach to solve this navigation problem, by predicting human movements and finding constraints that result in safe and efficient trajectories around humans. In the next chapter, we present different state of the art approaches and related work.

CHAPTER 2

State of the Art

Recently, driverless vehicles are being tested in different environments. Some of those environments, such as hospitals [KW91], train stations [WBV+14] contain freely moving humans, that may or may not want to interact with such robots. Additionally, industrial robotic workstations experience a trend towards collaborative interactions between robot and humans. In both cases, either with mobile or static robots, the robots need to be designed in a way such that the humans around it are safe and may not be harmed by the robot's actions.

Different approaches have been taken, in order to ensure safety. Traver et al. [TdPPF00] propose a robot design which estimates the potential danger of the robot posture, depending on the position of the robot's arm. In order to reduce the danger, the arm is moved away from a nearby person, accounting for the person's position, movement direction and head orientation, such that the person can be aware of the robot's movement. De Luca et al. [LASHH06] developed a collision detection strategy, which in case of an unavoidable collision quickly detects where the collision happened, and allows to react by moving the robot away from the impact point. Their approach does not depend on external sensing, as they introduced a collision detection signal based on joint positions, velocities, and torques, to determine which part of the robot was involved in the collision. This allows them to reduce the damage caused by a potentially unavoidable collision. Flacco et al. [FKLK12] proposed a collision avoidance strategy which uses repulsive vectors to influence the robot motion. Depending on the distance between the robot and a person, repulsive vectors are computed which are then processed into a smooth robot movement to avoid collisions.

While the above publications mainly revolve around static robot workstations such as in an industrial setup, their insights can also be applied to mobile robotics. However, when deploying a robot into an environment with humans that potentially unaccustomed to driverless vehicles, more has to be considered than just physical safety. In order to make the robot move in a way that is neither intrusive nor particularly inefficient, researchers

have focused on different aspects that can improve the acceptability amongst humans while still completing the robot's task without much delay. Kruse et al. [KPAK13] classified many publications on human-aware navigation into the categories "comfort", "naturalness" and "sociability". In their definitions, comfort means to reduce annoyance and stress for humans in interaction with robots. Naturalness is the attempt to increase the similarity of low-level behavior patterns between robots and humans. And sociability is achieved by giving the robot high-level cultural conventions to follow.

The approach of Kollmitz et al. [KHGB15] can be categorized as focusing on sociability. They predict the future position of humans assuming a constant velocity model and place a social cost field in front of the persons. This is done for each predicted timestep as a layer in the cost map. The cost field represents different rules for the robot, for example it should not invade the personal space 1.2 m around the human, and passing maneuvers should be performed on the right side. Using this cost map, they optimize the planned path with an A* algorithm with respect to social constraints, execution time, static environment constrains and path distance. Bennewitz et al. [BBCT05] on the other hand centered their work on naturalness, by learning human motion patterns. An expectation maximization algorithm clusters trajectories between points of interest, and generates hidden Markov models (HMMs) for them. They then use an HMM for each person in the environment to predict their individual movements. Those predictions are incorporated into a time dependant A* algorithm that discounts the cost of cells, according to the probability that a person will be in this cell at a given point in time. In their research, Feil-Seifer et al. [FSM11] focused on human comfort. They developed a guidance robot, that leads a person to a given target. The main point is, that according to the human's distance and speed, the reactive controller of the robot should adapt such that the person can dictate the movement speed and distance to the robot. Their results show, that compared to let the robot simply move to the target, the human was significantly closer to the target once it reached the position, especially when the human moved slower than the robot's normal speed.

Predicting the future movement of a detected human significantly increases the benefit of human-aware navigation. Once the robot has an estimation for human movements, the robot can smoothly plan suitable trajectories around humans without the need for radical replanning when sensor measurements of changed human movements arrive. Prediction in the field of human-aware navigation can be split into two general approaches, namely based on reasoning or on learning. Reasoning based approaches usually predict humans similar to how robot trajectories are selected. Those approaches range from very simplified assumptions of constant velocity and direction to more elaborate models that take obstacles and other humans into account. Hoeller et al. [HSMS07] improve upon linear prediction by reasoning with potential fields. Obstacles such as walls are assigned a repulsive potential, while possible targets get an attracting potential. The predicted person then walks towards such a target and avoids walls. In their detection and tracking approach, Luber et al. [LSTA10] predict human movement by using a social forces model. Similar to the potential field approach, each obstacle as well as other

humans in the environment apply forces to a person, resulting in a predictable walking path. However, the social forces influence the persons in a different way, as the direction of the force vectors and the person's orientation are taken into account. The person's future orientation and velocity is predicted by summing up the directed forces weighted in relation to the person's current orientation, such that forces directly opposing the person are stronger than forces perpendicular to the movement direction. The resulting force is then applied to the person to predict the future movement accordingly. In contrast to reasoning based approaches, learning based approaches rely on collected data samples to increasingly improve the predictions. Bennewitz et al. [BBCT05] learn motion patterns from previously observed humans and predict by reproducing one of the learned trajectories. In their work, they assume people don't randomly walk around, but rather follow similar trajectories, depending on the task they are performing. The recorded trajectories are therefore clustered into patterns using an expectation maximization algorithm. Afterwards, detected persons are predicted with the help of hidden Markov models, derived from the patterns.

CHAPTER 3

Human-Aware Navigation

In this chapter, the concepts of some of the algorithms used in human-aware navigation are explained. Human-aware navigation is a combination of human-robot interaction and motion or path planning for robots. In the absence of humans or other moving obstacles, a robot can usually plan a simple global path avoiding static obstacles by using a prerecorded map, and react to the errors of such a map by locally adapting the path, as soon as an unexpected static obstacle is detected. However, if moving obstacles join the robot in the environment, the planning task becomes more dynamic. Not only has the robot to locally react to detected to changes, but in order to move efficiently it is necessary to predict the movement of all obstacles. While moving obstacles like other robots might be accurately predicted by communicating with them, navigating amongst humans poses an even greater challenge. Even for humans it's hard to precisely identify another humans' intentions to predict movement patterns. Just consider two persons approaching each other on a sidewalk. Sometimes, in such a scenario, there is a short moment of misunderstanding, where both persons try to move to the same side. For a robot, it's even harder, as it cannot computationally interpret all those subtle signs persons give each other, such as mimic, posture or eye contact. An additional difficulty is, that the reaction of humans to driverless vehicles, moving freely in public spaces, is not sufficiently studied, yet. However, by examining situations in which selected humans share space with robots already, such as workstations in industrial plants, some key aspects can be identified. Those aspects have to be considered, and provide a starting point for further research, when designing a robot for the use in environments shared with arbitrary persons. In the following, we describe several features which are currently researched in order to improve the human acceptance of the presence of a robot, mainly concerning safety and sociability of a robot. Afterwards, we illustrate the necessary steps, which make a robot human-aware, namely the detection and tracking of persons, the prediction of their walking paths, and the planning of trajectories around persons.

3.1 Safety Aspects

Whenever humans interact with machines, or reside in the vicinity of machines, such as industrial robot arms, autonomous cars, or smaller mobile robots, there is the risk of physical harm. Naturally, the designers of such machines have to ensure safety, especially if humans are intended to be close. While there are different kinds of harms, which have to be prevented, such as physical or psychological harms [LFS+17], we focus on the physical harms. Depending on the shape and motion of a robot, physical harm can be inflicted in different ways. Direct collision of robot and person is usually the first situation which comes to mind, but puncturing or grazing as well as clamping the person between the robot and another object have to be considered. While 100% safeness might not be achieved due to the unpredictability of all involved factors, a mobile robot should be designed to minimize the chance of a collision with a person. Additionally, it is also necessary for the robot to be prepared in the event of a collision, such that the damage to the person is reduced as much as possible.

3.1.1 Pre-Collision

A challenge in safety is to prevent such harmful situations before they can occur. In autonomous navigation, this can be achieved by planning paths around humans such that they are never closer than a specified distance. However, since an environment cannot be perfectly known by a robot, and unexpected situations might occur, such as a person suddenly appearing right in front of the robot, the robot must be able to rapidly react and stop, or otherwise avoid contact. In order to achieve this, the sensor measurement update rate must be high, and planning procedure must quickly come up with a reaction. Common navigation strategies are to deny the robot to enter an area around a person to prevent a collision, or to slow the robot down when coming closer to a person, such that the planner has more time to react to unexpected situations. Predicting the most likely walking path of a person can additionally improve the capabilities of keeping an appropriate safety distance.

3.1.2 Post-Collision

As mentioned before, it might not always be possible to prevent contact. Especially in cases where the robot is intended to closely interact with a person. For robots assisting an elderly or for collaborative industrial robots, keeping distance is not an option. Therefore, it is equally important to plan how to react in the event of a contact. The robot must be able to differentiate between intended and unintended contact, whether the person tries to interact with the robot, or the robot harms the person. In the following, we limit to unintended contact, as the focus of our work is on navigation amongst humans, and not the interaction with them. Depending on the type of contact, it might be necessary to devise different post-collision strategies. While in the case of a direct collision it might be suitable to completely stop the robot, doing the same when the robot clamped a person to a wall might cause further harm, so it would be better to slowly move away from a

clamped person. As in preventing, for reacting to unintended contact, it is necessary to detect such unintended contact quickly, and plan further movement. Additionally, to preemptively reduce the damage done by a collision, it might be worth considering to reduce the momentum, by either generally limiting the robot's velocity, or by keeping the mass as low as possible.

3.2 Social Aspects

Beyond the safety aspects which focus on avoiding the chance and damage of a crash, there are several other aspects which can be considered when navigating close to humans. Namely, so called social aspects, also known as psychological safety aspects, which are concerned with reducing the stress or discomfort of humans in the vicinity of robots. A small selection of those aspects we were concerned with is presented in the following, however, due to limited computational resources and sensor capabilities only some were dealt with in this work, the others are left open for future work and improvement on the framework. While the safety aspects mentioned above are necessary to reduce physical harm done to persons, the social aspects presented in the following are aimed at making the presence of a robot socially acceptable. Currently, humans are not accustomed to mobile robots roaming freely in crowded spaces, such as offices, hospitals or train stations. Therefore, when such a robot is introduced to the public, it should not only technically be safe, but the persons around the robot should also feel safe, such that the persons tolerate or even support the presence of such a robot. The idea behind designing such a robot is that it should up to a certain degree be similar to a human person, by behaving in the same fashion, or at least by adhering to the same social rules.

3.2.1 Proxemics and Comfortable Distances

In proxemics, the study of how humans perceive the space around them, Hall defined four different areas, circularly around the person's center [Hal69]. Starting with the intimate space, up to a distance of 45 cm, in which embracing physical contact is expectable and the voice is limited to a whisper, which is usually reserved to partners. From 45 cm to 120 cm distance resides the personal space, which is used for interactions with family members and friends, as everyone inside is still at arm's length, and can be touched or grabbed. For direct conversations with acquaintances and strangers, the social space is suitable, ranging from 120 cm to 360 cm, additionally, at the upper end, around 300 cm distance, people don't feel obliged to interact with each other anymore, but can still choose to do so. The forth area is the public space, starting from 360 cm, which is no longer used for direct interactions. While the definition of those spaces was based on interactions between persons, and their distances were defined studying American test subjects, it can still be used as a guideline for human-robot-interactions in general, but one should be aware of possible cultural differences. On the one hand it seems clear, that a robot navigating amongst humans should never enter the intimate space, on the other hand it might not always be possible to stay in the public space in order to be

non-intrusive. So it should still be possible that a robot navigates past a person in a narrow hallway, as long as it does not have to enter the intimate space. Huettenrauch et al. [HEGT06] suggest that human-robot-interactions should take place in the personal space, whereas the intimate space should be reserved for special care robots which are intended to touch the person for support. Hall stated in his work that the different spaces and their sizes around a person depend on the intention of that person, so it might be reasonable to assume smaller radii, when two persons, or a person and a robot simply pass each other.

3.2.2 Visibility

Another branch of research is concerned with the visibility of a robot. The path should be planned in a way, such that at every point the robot is visible to a detected person, for example by avoiding driving behind the person or choosing paths with walls blocking the view. This way the robot makes sure not to surprise a person by suddenly appearing in the field of view, close to the person, and also the person might perceive the robot as more predictable as the movement can be watched. In order to achieve such a traceable path, the planner has to prioritize areas which can easily be seen by the person, for example by only moving the eyes, and penalize areas which the person can only see if they turn the head or move around. Additionally, when moving towards a location occupied by a person, the planner should choose a path which frontally approaches the person, rather than just getting as close as possible. Sisbot et al. [SMUAS07] propose an approach to such a visible robot. However, since it is necessary to not only detect a person, but also to know in what direction the head is turned, and where the eyes are looking at, considering visibility in the implementation is beyond of the scope of this work.

3.2.3 Passing Preference

Whenever a robot drives past a person, it is worth considering on which side to pass. Consider again the sidewalk scenario with two people approaching each other, while at times there is this moment of misunderstanding, most of the time the two persons agree on a passing side without communication. Moussaid et al. [MHG+09] showed that in some countries and cultures, like France, the preferred side seems to be to pass other persons on the right side, while in others, such as Japan, passing others on the left is preferred. Surprisingly, the side does not depend on the traffic laws, as in Great Britain people also prefer to walk on the right side. Nevertheless, depending on the culture or country a mobile robot is operating in, such preferences could be incorporated into the path planning. For example, in European countries, the planner could prioritize trajectories that deviate to the right when passing an approaching person. This could also be applied to overtaking a slow walking person, by deviating to the left, in order to pass the person on their left side. In order to avoid confusion, in the following the passing preference will be stated from the point of view of the passed person, such that in the approaching scenario the robot passes on the person's left-hand side, and also

overtakes on the person's left-hand side. However, since the passing side is simply a preference, the robot shouldn't be forced to stop if the preference can't be satisfied. If for example the approaching person walks closely to a wall, leaving no space to pass on the preferred side, while leaving more than enough space on the other side, the robot should still choose to pass the person, instead of waiting until the person is gone. Chen et al. [CELH17] recently presented a small mobile robot which respects such passing preferences by introducing social norms into their planner, obtained by deep reinforcement learning. Another example for such an approach is given by Kollmitz et al. [KHGB15], in their work they use a social cost function, a three dimensional Gaussian shifted to one side, to make passing on the preferred side cheaper for the robot.

3.2.4 Crowd Interaction

Another considered aspect was how the robot should behave around groups of people. Consider a platform in a train station, populated with many people waiting for an arriving train. Most likely, the robot would not be able to pass such a crowd while still satisfying the safety distance. One way to handle a situation like this could be similar to how supply carts are operated, with some sort of rotating light the robot could inform everyone of its presence and hope that enough people will move aside. Lidoris et al. [LRWB09] developed a robot which was able to navigate through Munich by interacting with pedestrians and asking for directions. In a similar manner, a robot trying to navigate through a crowded area could interact with persons blocking the path by verbally asking them to move aside. Additionally, if the people don't move aside to make way, the robot could also choose to slowly squeeze through the crowd, for example by using a soft, pressure sensitive casing to avoid pushing too hard. This behaviour is similar to how a person walks through a dense crowd, as intended to be done by Seer et al. in the project TransportBuddy[1]. However, as such behaviours would require additional sensing and human-robot interaction capabilities which we were not able to add to our simulation or real world experiments at this time, it is out of scope of this work. Furthermore, we believe that for people to accept a robot closely passing through a crowd, they must become more accustomed to the sheer presence of such a robot first, leaving this aspect to future improvements of our approach.

3.2.5 Legibility and Predictability

Finally, for autonomous navigation amongst people it should not only be figured out how the robot has to react to the presence of a person, it is also important to consider how the person will react to the presence of the robot. When a robot is approaching a person, the person will most likely try to keep distance. In order to make it easier for the person to react, the robot's intended movement should be recognizable. Kruse et al. [KBGK12] proposed that a robot navigating amongst humans should behave as if it were a person. They analyzed walking behaviours of two persons crossing each other's path in order to

[1]TransportBuddy, FFG project submitted by AIT, TU Wien, Blue Danube Robotics, bkm and DS Automation, https://www2.ffg.at/verkehr/projekte.php?id=1423

find characteristics which can be replicated by a mobile robot. They concluded that using a static social cost model might not be sufficient to react predictably in every situation and that a dynamic social costs should be used which adapt to the scenario. While some of the above aspects such as the visibility indirectly increase the legibility of the robot, the robot could also directly increase it by giving surrounding persons information about its intentions and planned movement. For example, the robot could verbally inform persons about its target, or warn someone who might be close to the planned trajectory. Another way would be to visually show the planned moving direction by turn signals, or by projecting the trajectory onto the ground. In our approach, we design our robot to be legible by humans, by selecting local trajectories that result in human like behaviour, such as passing on the culturally preferred side in certain scenarios. Additionally, we aim to avoid sudden changes in velocity or direction, such that it easier to predict the robot's movement.

3.3 Detection and Tracking

In order to be able to treat humans as such, the system needs a way to distinguish them from other (static) obstacles. Such a detection is commonly obtained by using laser range finders or cameras. In the scan of a laser range finder, which is frequently equipped on mobile robots for obstacle detection, it is possible to identify the shape of the legs of a person [FHM02]. However, solely using such leg detections tends to be error-prone. Linder et al. [LBLA16] report false positive rates of a simple 2D laser detector of above 200%. Therefore, such a leg detector can be paired with a camera based detection which leads to better results. The image obtained from a camera can be scanned for the shape of a torso, to detect a person [DT05]. In order to accurately locate a detected person, the camera image can be combined with laser range finder measurements, or with depth information of a stereo camera, to determine the distance to the detected person. Beyond detection and location of a person, it might also be of interest to identify a located person, for example by using face recognition [Man16]. In navigation, such information could be used to more accurately determine potential targets of a person, allowing for a better reaction. Additionally, in robot-human interaction, knowing the identity of a person could enable individual-related interactions.

Once a human person is detected and located in a measurement cycle, it can also be of interest to match the person to a detection in the next measurement cycle, to keep track of that person. By tracking persons, the detected locations can be combined into individual trajectories for each person, and knowing such a past trajectory can be useful to predict a person's target, or even identify the intention. The tracking is closely connected to predicting walking paths of a person. By predicting possible future locations of a person, the detection of the next measurement cycle can be associated to the previous one, which is necessary to keep track. Typically, the models used for tracking a person are similar to the approaches for predicting walking paths, outlined in the next section. Linder et al. [LBLA16] developed a framework which allows integration of different detection and tracking approaches into ROS. In their comparison of different approaches, they

concluded that the most important factor to successful tracking is a stable detection, with a low false positive rate, and sophisticated association models are only useful if they don't limit the computational resources for the detection.

3.4 Predicting Walking Paths

Predicting the walking paths of humans is a beneficial step in human-aware navigation, as this allows the robot to reason about possible future positions of the humans when planning a path through the environment. As shown in Figure 1.1, with this temporal knowledge the trajectories can be planned to be more efficient. However, since a person's intentions can't be fully captured by a sensor system, predicting the correct path is an inherently difficult task. Therefore, there are many different approaches on how to predict human walking behavior. The approaches can be grouped into two different classes, based on reasoning or on learning.

In reasoning based prediction the behaviour is obtained manually defining a model, or functions that describe how a person reacts to the environment. The simplest, and a widely used approach is to assume constant velocity and constant direction of movement (see Equation (3.1)), such that the person walks in a straight line. $\dot{\boldsymbol{x}}$ is the positional change of the person, consisting of the change of x and y coordinates, and $\boldsymbol{v}$ is the velocity vector of the person.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{v} \tag{3.1}$$

While this model is useful in open spaces or for a short temporal horizon, it is clear that, as soon as obstacles are involved, the predicted person can quickly end up in unrealistic positions (e.g. in a wall). But in many cases this prediction is sufficient, and the advantage of such a simple approach is that it is easy to implement and requires only little computational resources. The constant velocity approach is often extended by some kind of obstacle handling, such as using potential fields to let the predicted person move away from walls (see Equation (3.2)), as done by Kessler et al. [KSG12]. The vector from the obstacle potential field F_o is then combined with the constant velocity vector to determine the next position, similar to how charged particles behave in an electric field.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{v} + \boldsymbol{F_o}(\boldsymbol{x}) \tag{3.2}$$

Similar approaches replace the potential field with social forces, which repel the person's prediction from walls or other persons [LSTA10].

In contrast to reasoning, the approaches based on machine-learning don't require such model definitions. The concept here is, that the robot can learn how persons walk through the environment by observing them. This means, that such approaches usually get more accurate over time, as more observations are obtained. A popular method to evaluate the collected data is to use the Expectation Maximization Algorithm, as done by Bennewitz et al. [BBT02], where the trajectories are grouped into clusters. Each trajectory starts in

a random cluster, and for every observed point the likelihood of the trajectory fitting into that cluster is calculated, checking if another cluster is more likely, and afterwards the parameters of the cluster are improved using the new trajectory. Aside from this short term prediction for intermediate trajectories, other approaches consider predicting future targets or way points for a person (e.g. the exit, junctions, or someone's office). Knowing where the person's target is can significantly improve the prediction, especially when junctions are along the path. At a junction the predicted person has to move one way or another, which without further knowledge can only be predicted randomly. However, when the target is known, the shortest (or fastest) path to that target can be planned to improve the decision at each junction. Thompson et al. [THK09] propose an approach which identifies intermediate navigation targets in a map, based on observed walking trajectories. Their model then uses those targets to predict walking paths from a person's current position, to all feasible targets in the vicinity.

Our approach tries to combine some of the above mentioned features. We use a potential field F_o for static obstacles repelling persons, and a second potential field F_l which is based on learned walking patterns to attract predicted person. The learned walking patterns are represented by a grid in which the value of each cell stands for the frequency of persons who were detected walking through this cell. An example of such a grid is shown in Figure 3.1. The assumption of our prediction model is, that locations where

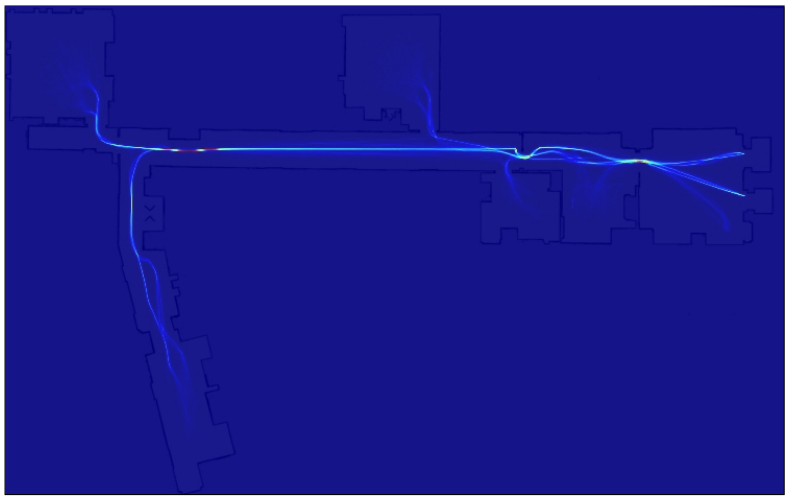


Figure 3.1: Grid representation of learned walking patterns.

many people have been observed, are more likely to attract new persons than locations where (almost) none have been observed before. Therefore, we shape our second potential field such that the orientation of the prediction is drawn towards attracting locations. In contrast to the constant orientation approaches, our model not only consists of the person's position $\boldsymbol{x}$ but also the orientation θ. The vectors of both fields are combined to get a vector pointing towards the most probable future position. In our model, we then predict the change in orientation such that in one second the person turns towards this

future position (see Equations (3.3) and (3.4)). Here the arcus tangens function $tan^{-1}(\boldsymbol{x})$ of a vector is used as a shorthand for the arcus tangens of the fraction of the x and y coordinates of the vector.

$$\dot{\boldsymbol{x}}(t) = v * \begin{pmatrix} cos(\theta) \\ sin(\theta) \end{pmatrix} \tag{3.3}$$

$$\dot{\theta}(t) = tan^{-1}(\boldsymbol{F_o}(\boldsymbol{x}) + \boldsymbol{F_l}(\boldsymbol{x})) \tag{3.4}$$

3.5 Global or Local Planner

A core design decision in human-aware navigation is at what planning level the human evasion should be handled. Typically, the navigation task is split into a global and a local planner [RKC98]. The global planner usually has a map of the environment and tries to find a path from the current position through the environment to the target by applying algorithms such as A*. As the map of the global planner tends to be a static representation of the environment which was recorded at some time, such a map might not accurately show the current state of the environment. Therefore, a global plan can only be used as a rough guideline through the environment. In order to react to unmapped changes, such as a now closed door, or moved chairs, a second planning layer is used, the local planner which is also called reactive planner. As long as the environment looks like the prerecorded map, the local planner is usually just in charge of following the global path, which can be achieved by using a simple PID controller. But as soon as some changes are detected, the local planner also has to modify the globally planned path and find a local trajectory that avoids new obstacles while still roughly following the global plan. To achieve that, the local planner needs knowledge about the local environment, provided by range-scanners such as sonars or lasers, to then use techniques like the dynamic window approach [FBT97] or model predictive control [CB04] to compute feasible local trajectories. By using such a layered planning architecture, both layers can operate at different update frequencies. For the global plan, it is necessary to evaluate large areas of the map, which can be computationally expensive. The task of the local planner on the other hand is usually comparably easy, for example by using a PID controller. Therefore, the global planner typically operates on a low update rate of once every few seconds, or even only once per assigned target, while the local planner reaches update rates of several hundreds of times per second.

Depending on the design goal, the human-awareness can be incorporated into either level, or even both. The global planner could predict the movement of all detected humans in the environment for the complete time the robot would need to drive to the target. Kollmitz et al. [KHGB15] use such an approach in a single layered planner, where the time dependent A* algorithm computes an optimal path towards the target while accounting for the detected and predicted positions of persons in a time dependent multi-layered map. However, since they combined both planning layers into one, the update rate is a compromise of about 2Hz. As avoiding collisions with humans is safety critical, and both detection and prediction of humans tend to be error-prone, a higher update rate might

be desired, especially when close to humans. Therefore, in our opinion, evading humans should be handled by a fast local planner, which knows the surroundings and predicts the human walking path only for a short temporal horizon. While the resulting evasive trajectory might not be optimal, due to the limited knowledge of the local planner, it can be sufficiently good to solve the problem, and by recomputing tens of times per second, unexpected changes can be handled faster. However, the human-awareness can also be useful in the global planner, as by knowing which areas of the environment are frequently occupied by humans, the global path can be planned precautionary avoiding such areas, and use regions which tend to be free. This way the potential necessity of reacting to human presence can be reduced.

3.6 Model Predictive Control

Amongst the multitude of different planning approaches, this work is based on the principle of Model Predictive Control (MPC), also called receding horizon predictive control (RHPC) [CB04]. The core structure of an MPC is shown in Figure 3.2.

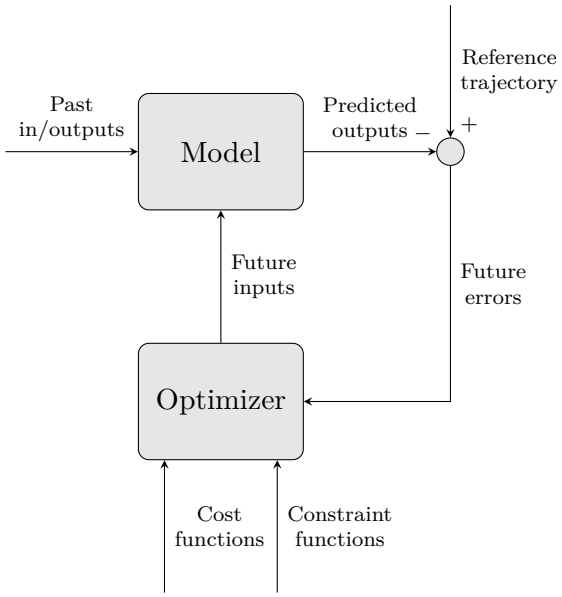


Figure 3.2: Components of a generic MPC system, as described by Camacho et al. [CB04].

The basic idea is, to compute the optimal input sequence for a system, such that the output of the system closely follows a reference trajectory, while the input is subject to constraints and costs. Given a sequence of inputs from the current time t up to a limited time horizon $t+N$, the model of the system predicts the resulting outputs. Those predicted outputs are compared to the reference trajectory and, depending on the current error, the optimizer computes improved future inputs which satisfy the given constraints

functions, in order to minimize the cost functions. While the input sequence consists of N future inputs, only the first input is applied to the system at the current time t, for the next time step the whole computation is repeated, such that an input sequence from $t+1$ to $t+1+N$, based on new information, is generated. Since the computation of the future input sequence is highly based on the output predicted by the system model, it is necessary to obtain a model which is an accurate representation of the real system. In the context of robot navigation, this means a good model of the drive kinematics for the robot needs to be formulated. For our Pioneer P3-DX®, a differential drive model is used. The constraint functions are defined to specify limitations such as maximum angular velocity of the wheels, or a minimum distance to obstacles or humans. The cost functions are used to let the optimizer choose a trajectory that minimizes the remaining distance to the target at time $t+N$.

3.7 Summary

In this chapter, we introduced several aspects of human-aware navigation, concerning both physical and psychological safety. We also discussed the difficulties in detecting, tracking, and ultimately predicting movements of persons and how to approach those difficulties. Finally, we explained the difference between the global and local planning level, and why we think avoiding persons should be handled on the local level, by using techniques as the outlined MPC. We describe the implementation of the presented concepts in the upcoming chapter.

CHAPTER 4

Implementation

In this chapter, we explain how we implemented our human-aware navigation approach. The following sections give a short introduction to the core framework, which we call MPN and to the Robot Operating System (ROS) which was used. Afterwards, we present the algorithms used for learning human walking patterns and predicting such, as well as how the robot was designed to react to predicted humans. The implementation was written in C++ and as an evaluation platform a P3-DX® robot was used in indoor environments.

4.1 The MPN Framework

The MPN framework developed by Todoran [Tod18] solves the task of autonomous robot navigation based on MPC. Inputs for a kinematic model of the robot platform are optimized with respect to cost and constraint functions so the robot finds an optimal path to a designated target. The framework was written in C++ and is embedded into the Robot Operating System (ROS) that allows running the framework both in a simulated environment and on real world robots. This framework was used as a starting point for the implementation of this work, where we extended the provided implementation in order to make the robot human-aware.

A core aspect of the framework is, that the constraint and cost functions can be easily extended to experiment with different optimization approaches. Different platform models, such as for differential or Ackermann drive have already been formulated and are exchangeable. When necessary, those models can be adapted, or new models can be defined for different platforms, however since such models are typically changed less often than the optimization problem itself, the current model formulation is highly optimized. In order to simplify the formulation of the optimization problem, the framework provides a clean, templated interface for the definition of cost functions or hard and soft (in-)equality constraints.

Prior to our improvements, the system state consisted of the robot's position, orientation, velocity and acceleration. The movement restrictions such as maximum velocity or acceleration of the wheels were formulated as constraint functions. Moreover, constraints and cost functions for navigational purposes were added such that the robot moves efficiently towards the target and evades obstacles. The core cost function is based on a velocity cost map. As the values for each position in such a map represent how much time is needed to reach the global target from that position, the local trajectory should always end in the cheapest reachable position (with the lowest value). To avoid hitting walls or other static obstacles a minimum distance to any detected obstacles was a sufficient constraint. The optimizer then tries to find the best local trajectory by searching the cheapest reachable position, which is still valid under the constraints, for every point in time up to a specified temporal horizon. Note that the optimizer is only able to find the best trajectory if the cost functions are convex and therefore have only one minimum, as the optimizer could otherwise get stuck in a local minimum.

During the course of our implementation, we extended the state model, to integrate the prediction of detected persons, as explained in Section 4.4. Afterwards, we defined a constraint for the minimum required distance to the predicted persons, and improved the velocity cost map such that the cost function prioritizes passing a person on his/her left side, as described in Section 4.5.

4.2 Introduction to ROS and our Setup

The Robot Operating System (ROS) is a meta-operating system which supports code reuse for research and development in robotics. A typical ROS system consists of several so-called nodes (processes), each responsible for a special task. Those nodes, which can potentially run distributed over different machines in a network, share information by exchanging messages over the ROS communication infrastructure. A simple example can be seen in Figure 4.1. A motion planning node receives sensor measurements from a laser rangefinder node, computes a movement vector and sends it to a motor driver node which translates the vector into motor control commands to actuate in either the real world or in simulation. Such a simple setup can randomly drive through an environment without hitting static obstacles. For simulation purposes, ROS also provides an interface to the simulation software Gazebo. In our simple example this means that the sensor readings are provided by the simulation, and the computed wheel movements are then executed in the simulation. Another handy tool of ROS is Rviz, which can visualize all messages exchanged in the ROS system, such as the laser rangefinder measurements, the robot's position, or a map of the environment.

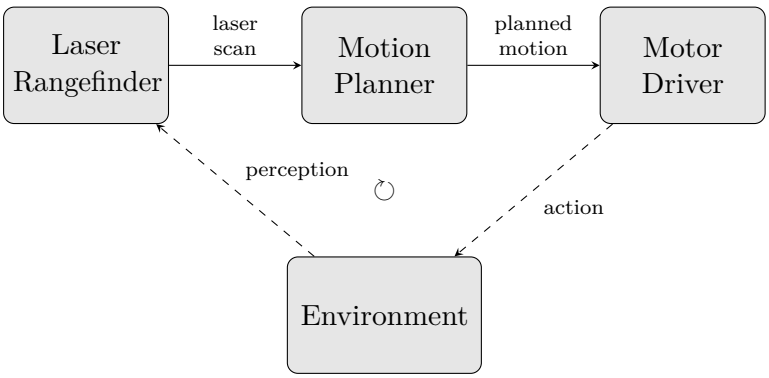


Figure 4.1: Simple ROS system, consisting of a laser rangefinder, motion planner and a motor driver, the interface to the real world or simulation environment is indicated by the dashed arrows.

Our setup consists of several ROS nodes, as can be seen in the simplified schematic in Figure 4.2. Every rectangle represents a node and the arrows show the communication network, labeled with the message sent along the connection. For simplicity some very generic nodes such as the localization, cameras, laser scanners and motor drivers are omitted from the diagram, and important messages to or from those nodes are shown by dashed arrows. The blue nodes were extended and improved during the implementation of this work.

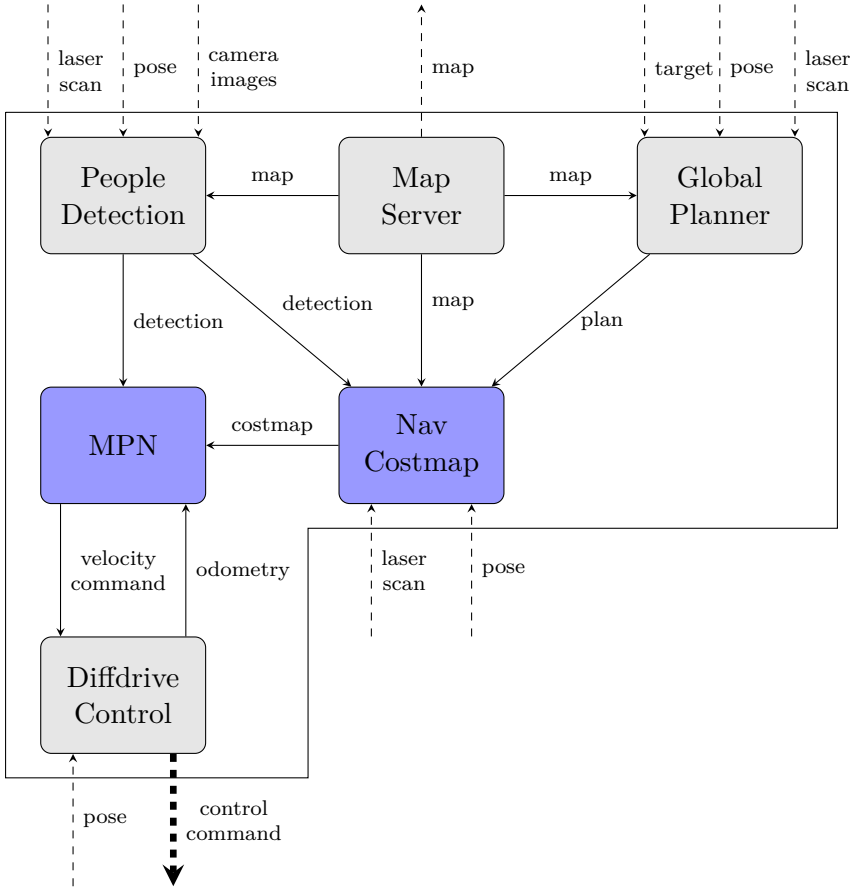


Figure 4.2: Schematic view of our ROS setup.

The **map server** knows and publishes a static *map* of the environment to the **global planner**, the **people detection**, the **nav costmap** as well as to a localization node. With this *map*, the estimated *pose* and *laser scan* information, the **global planner** computes a crude *path* to a given *target* for the **nav costmap**. The **people detection** node looks for patterns in the *camera images* and the *laser scan* to detect persons and determines their position in the *map* relative to the robot's *pose*. These *detections* are then sent to the **nav costmap** node and to the **MPN** node. In the **nav costmap** node, a local velocity *costmap* is calculated based on the static map, laser scans and the global plan, using the Fast Marching Method [GÁGM15]. For this velocity *costmap*, an intermediate target is set to the position furthest along the global plan, which is still inside a local area with a radius of a few meters. Starting from this intermediate target, free space is set to the maximum velocity, whereas around obstacles the velocities are dampened gradually down to zero. Additionally, using the people *detections*, an area around every person is slightly slowed down, in order to accommodate for social preferences. Using this velocity *costmap*, the people *detections* and the kinematic model

of the robot the **MPN** node optimizes a local trajectory according to several constraints using MPC and passes an immediate *velocity command* to the **diffdrive control** node. In the **diffdrive control** the *velocity command* is converted into individual wheel *control commands* using the robot's differential drive control model, and additionally the robot's odometry information is passed back to the **MPN** node. These *control commands* are then sent to the motor driver node which then makes the robot finally move.

4.3 Learning Walking Patterns From Observations

Our prediction model is based on learned walking patterns. In order to preprocess observations, we implemented an additional ROS node, that keeps track of detected persons. This is done by taking a map of the current environment, and storing in every cell the count of detected persons at the corresponding location. An example of this representation can be observed in Figure 4.3.

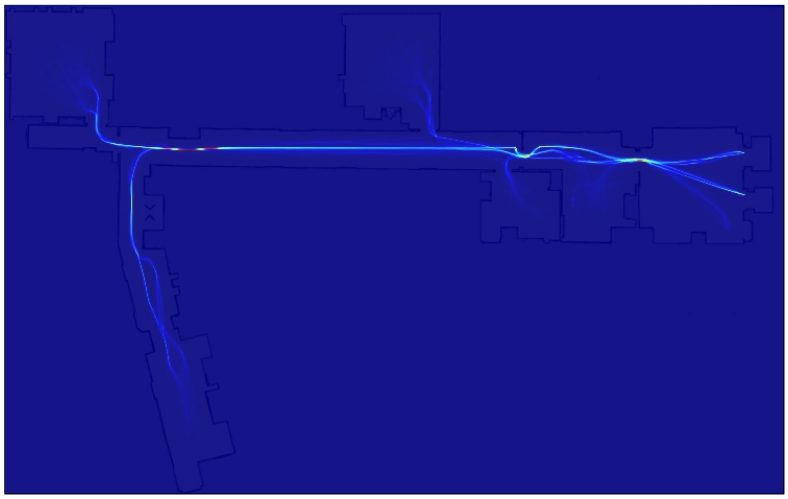


Figure 4.3: Grid representation of learned walking patterns.

With this map we wanted to predict the future pose of detected persons by assuming that they will walk towards highly visited or passed locations. The idea was, that once enough people have been observed, lanes might be recognized, which people typically follow, such as from an office room to the exit of the building. So for every position of a person, we checked the surrounding cells, and set the change of orientation in our state to turn towards the cell with the highest detection count. In order to avoid getting stuck by circling around a local maximum, we limited the searched area to a cone in front of the person. A first success was achieved with this approach, but to be computationally feasible, we had to precompute the search for the best cell. Since we still wanted to limit the search to a cone we computed the vector pointing to the best cell for four different possible directions of movement.

While the above approach was promising, we further improved the prediction by not only taking into account the cell with the highest value, but rather treating every cell as an attractor and weighting them by the count. So every cell surrounding a position exerts an attraction force onto a person in that position, and the combined force vector determines in what direction the person will be predicted to turn. This way, we were able to provide a more reasonable prediction in case of several parallel lanes, as groups of cells with a low count have a similar effect as a single cell with a high count. So instead of strictly following the strongest lane, the prediction follows the general moving direction of those parallel lanes. Moreover, the improvement reduced the complexity of the search area, since a single cell is no longer a local maximum where the prediction could get stuck. The search area could be reduced to a simple triangle or semi-circle, instead of a polygon which excludes the current position. Moreover, if there are several cells with the same high value, the first approach simply picks the first encountered, while the force approach accounts for all of them by adding up their attracting forces. Additionally, the resulting vector fields show a finer gradation, as in the above approach groups of cells tended to point towards the same target which is no longer the case. The algorithm used to calculate the force is presented as pseudo-code in Algorithm 4.1. The illustration in Figure 4.4 shows an exemplary force vector generated by Algorithm 4.1, where the current orientation is the gray arrow, the attracting force is the green arrow and the search area is indicated by the yellow polygon. As in the first approach, the vector fields are precomputed for four different directions of movement, as iterating over the heatmap is computationally expensive.

Algorithm 4.1: Vector field computation using attracting forces.

Input: A 2D grid representation $\mathbf{M}$ of the recorded paths
Output: Vector fields $\mathbf{V}$ following the recorded paths for a set of predefined *directions*

for each *direction* d **do**
  for each $(x_M, y_M) \in \mathbf{M}$ **do**
    $p \leftarrow$ polygon area $\in \mathbf{M}$ in *direction* d in front of (x_M, y_M)
    $\mathbf{F} \leftarrow (0, 0)$
    for each $(x_p, y_p) \in p$ **do**
      $\mathbf{F} \leftarrow \mathbf{F} + (x_p - x_M, y_p - y_M) * \mathbf{M}(x_p, y_p)$
    end
    $\mathbf{F} \leftarrow \frac{\mathbf{F}}{|\mathbf{F}|}$
    $\mathbf{V}_d(x_M, y_M) \leftarrow \mathbf{F}$
  end
end
return $\mathbf{V}$

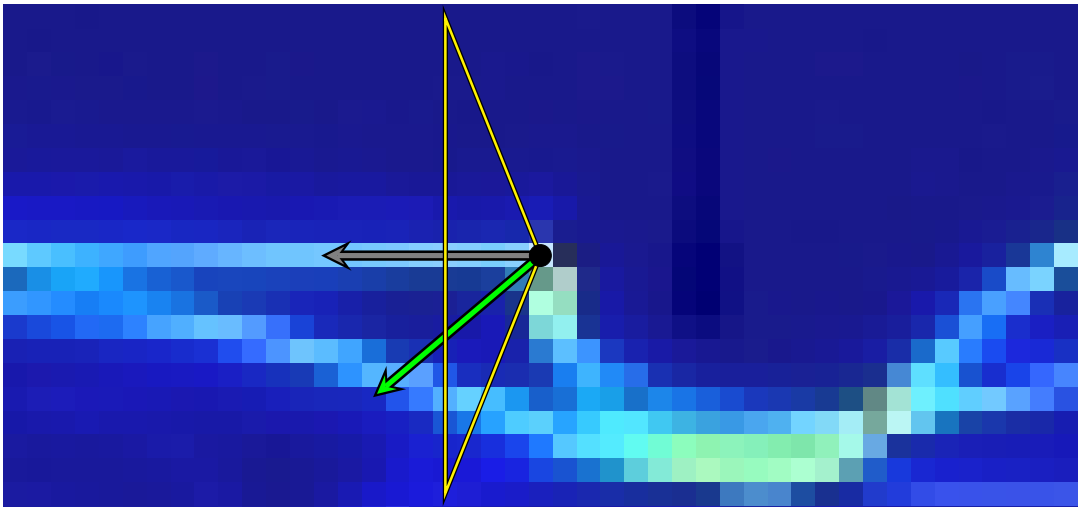
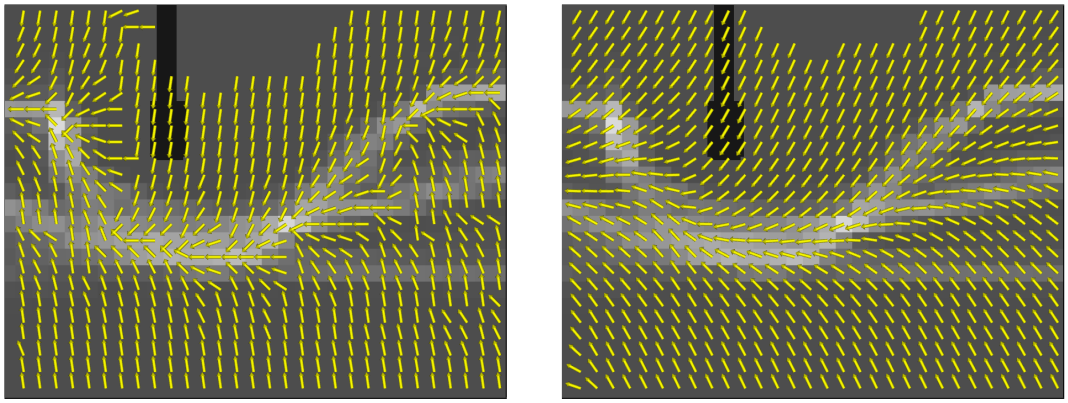


Figure 4.4: Determining the force vector (green arrow) for the current walking direction (gray arrow) by iterating over all cells in the yellow polygon.

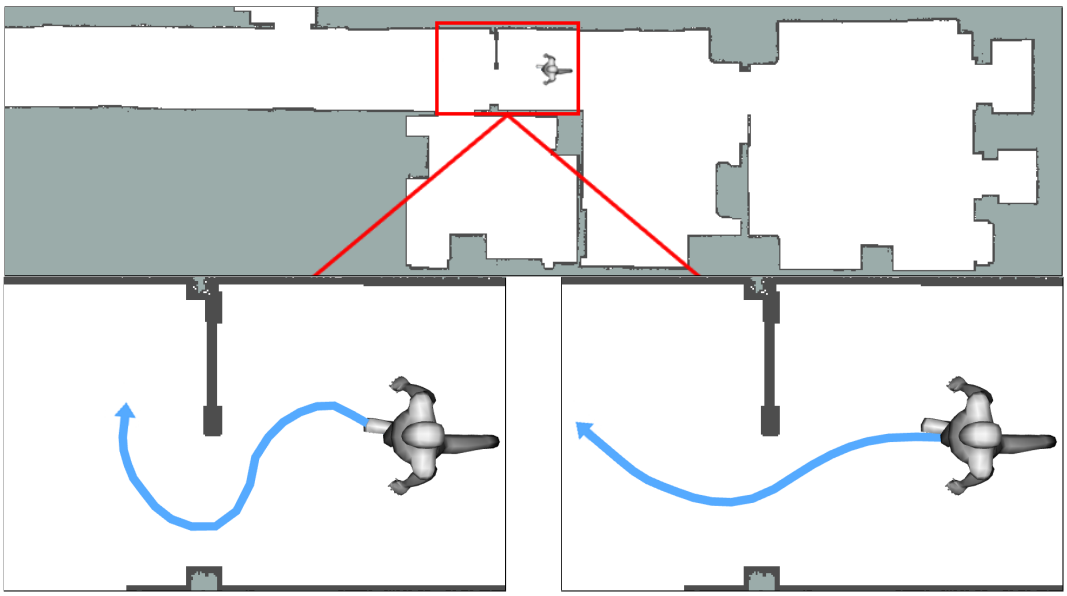
Comparing both approaches in Figure 4.5 shows that both provide reasonable vectors in areas where people walked on an almost identical line (here in the center of the pictures), but when there are several lanes close to each other (here on the right and left borders of the pictures), the second approach captures the general moving direction better. The initial approach predicts the person to walk to the most occupied lane first, and follow that lane, while the improved version only slightly drifts towards that lane.



(a) First approach, attracting towards the cell with the highest count in the polygon.

(b) Second approach, accounting for all cells in the polygon.

Figure 4.5: Comparison of the resulting attracting vector fields for walking to the westwards. The first approach follows the strongest line while the second approach follows the general moving direction.



(a) First approach, attracting towards the cell with the highest count in the polygon.

(b) Second approach, accounting for all cells in the polygon.

Figure 4.6: Comparison of the resulting paths for both approaches in the office scenario.

The predicted paths resulting from those vector fields can be compared in Figure 4.6. While both approaches correctly predict the person to avoid the wall and walk through the door, the first approach is less intuitive, as the person is predicted to walk to the stronger lane both before and after passing the door, resulting in larger turns. In scenarios with one concentrated lane, both approaches worked equally well.

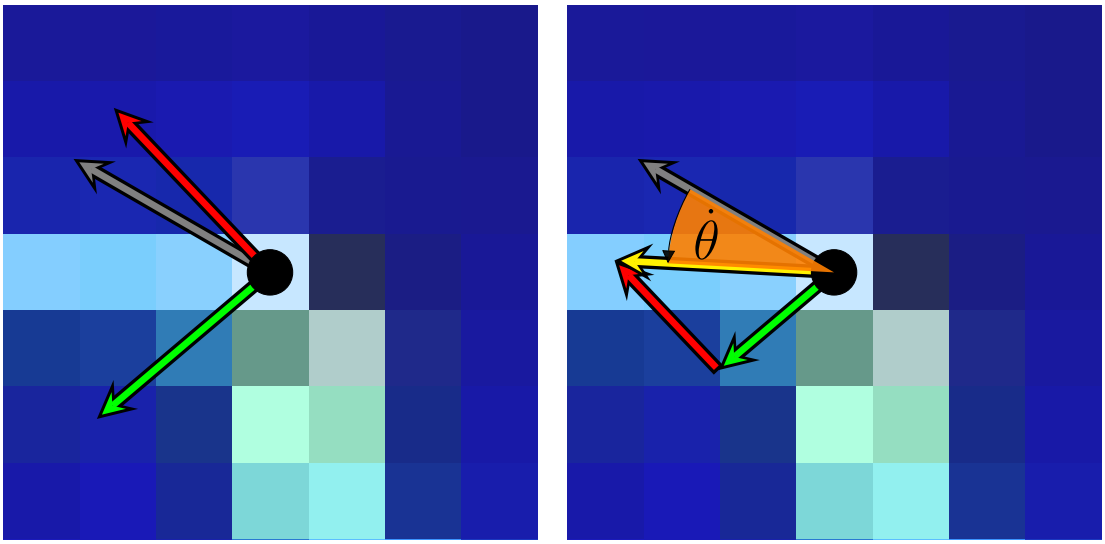
In order to also provide a reasonable prediction in areas for which the heatmap doesn't have any information yet, we combined our approach with a simple potential field for repelling persons from obstacles similar to the work of Kessler et al. [KSG12]. Depending on the total count of peoples observed in the polygon, we scale the impact of the potential field, such that in areas with low observation count the prediction mainly depends on the potential field, whereas in areas where many people have been seen the potential field is ignored.

4.4 Prediction With Extended State and Model

In order to efficiently integrate the prediction of detected persons into our framework, the system state was extended by a vector of person states. Where each person's state is itself a vector $\boldsymbol{p}$, representing the person's coordinates x and y, the orientation angle θ and the walking speed v (4.1). Additionally the derivative of the person states $\dot{\boldsymbol{p}}$ was defined (4.2). The derivatives of each state variable are shown in Equations (4.3-4.6). The positional change depends on the velocity and orientation (4.3, 4.4). The change in orientation is obtained from our learned data. Our model is based on the assumption that persons move towards locations where many people have been seen before, so each location attracts with an intensity proportional to the number of peoples observed at that specific location. The change in orientation is therefore defined by the difference between the current orientation θ and the orientation facing the most attracting location nearby θ_{att}, scaled by a global proportional parameter P (4.5). Additionally, we assume that the person walks with constant speed (4.6) and is based on the detected velocity (4.7).

$$\boldsymbol{p} = (x, y, \theta, v) \tag{4.1}$$
$$\dot{\boldsymbol{p}} = (\dot{x}, \dot{y}, \dot{\theta}, \dot{v}) \tag{4.2}$$
$$\dot{x} = v * cos(\theta) \tag{4.3}$$
$$\dot{y} = v * sin(\theta) \tag{4.4}$$
$$\dot{\theta} = P * (\theta_{att} - \theta) \tag{4.5}$$
$$\dot{v} = 0 \tag{4.6}$$
$$v_0 = v_{detected} \tag{4.7}$$

As evaluating the recorded data by iterating through the shown polygon is computationally expensive, we chose to precompute the attraction vectors. Therefore, we limited the orientation of the search polygon to the four cardinal directions, saving the resulting vector fields. On runtime, for every position of a person, the vector fields enclosing the person's orientation are then queried, and the vectors are combined proportionally, such that the cardinal direction closer to the person's orientation is weighted higher. Figure 4.7 illustrates the computation described in Algorithm 4.2. Additionally the change in direction is limited for the first t_{conf} seconds, as for that time we have more confidence in the current orientation of the person, with increasing prediction time the initial orientation of the person gets less useful and we shift towards our vector field.



(a) Current orientation (gray), and attracting force vectors for walking west (green) and north (red).

(b) Target orientation (yellow), generated by combining force vectors for walking west (green) and north (red). Change of orientation $\dot{\theta}$ is the angle between current (gray) and target (yellow) orientation

Figure 4.7: Determining $\dot{\theta}$ based on learned data. The background is an image section of the complete map, the color represents the count of detected persons for each cell, ranging from blue (low) to white (high).

Algorithm 4.2: Calculation of $\dot{\theta}$.

Input: Vector fields $\mathbf{V}$ following the recorded paths for a set of predefined *directions* and person's current position and orientation (x, y, θ) at relative time t in the future.

Output: Predicted change of orientation $\dot{\theta}$ for the next second.

$(dir_1, dir_2) \leftarrow$ cardinal directions enclosing person's orientation θ
$p \leftarrow$ percentual angle of θ between dir_1 and dir_2
$\boldsymbol{F_1} \leftarrow \boldsymbol{V}_{dir_1}(x, y)$
$\boldsymbol{F_2} \leftarrow \boldsymbol{V}_{dir_2}(x, y)$
$\boldsymbol{F} \leftarrow \boldsymbol{F_1} * p + \boldsymbol{F_2} * (1 - p)$
$\dot{\theta} \leftarrow tan^{-1}(\boldsymbol{F})$
if $t < t_{conf}$ **then**
  return $\frac{t}{t_{conf}} * \dot{\theta}$
else
  return $\dot{\theta}$
end

By integrating the person states into the system state, the framework now calculates the prediction for every state of the robot that is evaluated during the optimization. This however means that for any point in time, this can potentially result in computing the prediction multiple times (for different possible robot states at that point in time). The prediction currently only depends on the detection, the learned data and the time, which means recalculating the prediction for a specific point in time is unnecessary. Since it was not relevant for the computational performance, it was still implemented like this, such that the prediction could be extended by some kind of human-robot-interaction in the future. For example, it might be reasonable to assume that a person will move away from the robot slightly, even if the robot is making way. Additionally, if the interactions are quantifiable, constraint and cost functions could be added, in order to prioritize less invasive trajectories. Such reactions of humans to the robot are out of the scope of this work, however.

4.5 Safety Constraint and Social Aspects

In order to let our robot react to detected humans accordingly, we designed both a strict constraint to avoid getting to close to a person, as well as a social cost field such that the robot will pass a person on the preferred side. A schematic of both is shown in Figure 4.8, where the black dot is a detected person with the orientation indicated by the black arrow. The red circle indicates the area around the person which has to be avoided at all cost, whereas the orange Gaussian ellipsoid represents the social cost field which should let the robot pass the person on his/her left side, if possible.

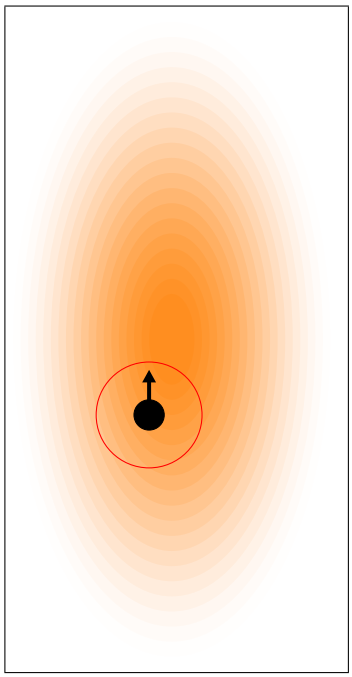


Figure 4.8: Social cost field and safety constraint around a detected person (black dot) facing north (black arrow). The red circle shows the safety distance constraint which should never be violated. The orange gradient shows the social cost field with the maximum cost in the center of the elliptical field. The field's center is shifted to the right and front of the detected person to model passing side preferences.

The main constraint added to the optimization is to keep a minimum euclidean distance to all detected persons. This constraint marks every trajectory invalid, that at some point in time is closer than $dist_{min}$ to any predicted person. As stated by Huettenrauch et al. [HEGT06], the minimal distance should always be outside of the intimate space (45 cm), and preferably at the border of the personal space (1.2 m). (4.8) shows our safety constraint f, where (x_r, y_r) is the robot's position at a specific point in time t and (x_p, y_p) is a person's position at the same time t. (4.9 - 4.12) show the gradients $\frac{\partial f}{\partial x_r}$, $\frac{\partial f}{\partial y_r}$, $\frac{\partial f}{\partial x_p}$ and $\frac{\partial f}{\partial y_p}$ of the constraint function which are needed for the optimizer to find the

minimum.

$$f = \min_{\forall (x_p, y_p) \in persons} \sqrt{(x_r - x_p)^2 + (y_r - y_p)^2} > dist_{min} \tag{4.8}$$

$$\frac{\partial f}{\partial x_r} = \frac{x_r - x_p}{\sqrt{(x_r - x_p)^2 + (y_r - y_p)^2}} * \bar{e}_{x_r} \tag{4.9}$$

$$\frac{\partial f}{\partial y_r} = \frac{y_r - y_p}{\sqrt{(x_r - x_p)^2 + (y_r - y_p)^2}} * \bar{e}_{y_r} \tag{4.10}$$

$$\frac{\partial f}{\partial x_p} = \frac{x_r - x_p}{\sqrt{(x_r - x_p)^2 + (y_r - y_p)^2}} * \bar{e}_{x_p} \tag{4.11}$$

$$\frac{\partial f}{\partial y_p} = \frac{y_r - y_p}{\sqrt{(x_r - x_p)^2 + (y_r - y_p)^2}} * \bar{e}_{y_p} \tag{4.12}$$

Since this constraint is a hard limitation, it is important to define an appropriate minimal distance. Clearly it is important to stay at least out of the intimate space, but increasing the safety radius around persons further can cause unnecessarily large detours, or might even force the robot to stop in a narrow hallway even though it might be possible to pass just outside the intimate space. For our proof of concept, it was sufficient to use a minimum distance which could be manually adjusted to experiment with different values. In addition to the pure minimum distance to all persons, our approach also allows us to account for both the braking distance of the robot and the increasing uncertainty of the person's predicted path. The safety radius is increased proportionally to the robot's velocity at an evaluated position (by $dist_{braking}$), and to the time that will have passed until the robot reaches that position (by $dist_{uncertainty}$). In order to use the safety constraint in the MPN framework, we defined a constraint function as shown in Algorithm 4.3. As the framework allows slight error margins for the result of the combined constraints, the result of each constraint has to be scaled accordingly (by $scaling_{error}$) in order to allow individual constraint error margins. Additionally we also define the gradient of the safety constraint, which is the sum of equations (4.9) to (4.12) for the closest person. To increase the efficiency of the planned trajectories, it is worth considering dynamically adapting minimum distances. Depending on the shape of the environment, the robot could increase the minimum distance in large open spaces and decrease it for narrow hallways. Alternatively, if there is enough time to reevaluate a trajectory, the planner could initially look for a collision-free trajectory with a large safety distance, and if there are none to be found, decrease the distance and repeat the search. This way the robot could maximize the safety distance while still staying efficient and avoiding hard stops, it is however out of the scope of this work.

In terms of social aspects of human-aware navigation, we looked into passing side preferences, as studied by Moussaid et al. [MHG+09], who concluded that in European countries it is typically preferred to pass other persons on their left side. Similar to Kollmitz et al. [KHGB15], we defined a social cost field around persons, which is shaped asymmetrically, to prioritize trajectories on the left-hand side, over the right-hand side.

Algorithm 4.3: Safety constraint to ensure minimum distance to all persons.

Input: Robot position and velocity (x_r, y_r, v_r) and all detected person positions (x_p, y_p) at t seconds into the future.

Output: How far outside of the safety area this robot position is

$dist_{min} = \infty$
for each *person* (x_p, y_p) **do**
  $dist = \sqrt{(x_r - x_p)^2 + (y_r - y_p)^2}$
  if $dist < dist_{min}$ **then**
    $dist_{min} \leftarrow dist$
  end
  $result \leftarrow dist - dist_{safety} - v_r * dist_{braking} - t * dist_{uncertainty}$
  return $result * scaling_{error}$
end

This social cost field was then integrated into our velocity map, computed by the Fast Marching Method, as done by Kessler et al. [KSG12]. Unlike in the work of Kessler et al., our robot is not driving at constant velocity, therefore the social cost field around the person prediction could not be embedded into a single velocity map layer. It would have been necessary to compute several layers, for different points in time and, considering future work for human-robot-interactions, also for different robot trajectories. Since this wasn't computationally feasible, we decided to limit the social cost field to the detected position of the person. Inside such a field, the maximum velocity is reduced (see Figure 4.9), such that the area is avoided, if there is enough space. The endpoint of the local trajectory follows the gradient descent of the velocity map, so the trajectory will only be planned to end inside the social cost field if there is no area with higher velocity in the vicinity. By extending the field further to the right than to the left (from the point of view of the person), the optimizer will prefer a deviation to the left. When the trajectory is then re-optimized as time progresses, the optimizer will try to find a better trajectory locally close to the old trajectory. In many cases, this was sufficient, to give the optimizer a slight nudge towards the preferred passing side.

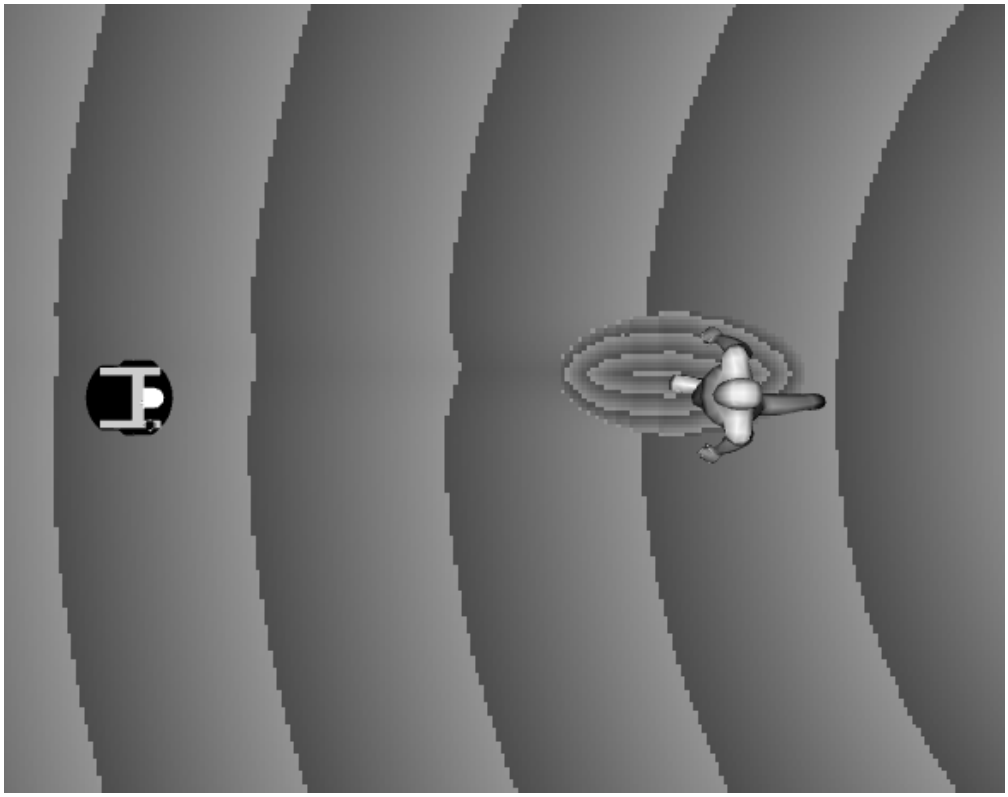


Figure 4.9: Social cost field embedded in the velocity map, an ellipse around a detected person is "slowed down" to accommodate for the passing side preference.

4.6 Summary

In this chapter, we provided implementation details for our prediction strategy, safety constraint and social cost. We explained thoroughly how a person is predicted and how the robots finds a trajectory that both avoids physical contact and satisfies a social passing side preference. The evaluation of the implementation is presented in the next chapter.

CHAPTER 5

Experimental Results

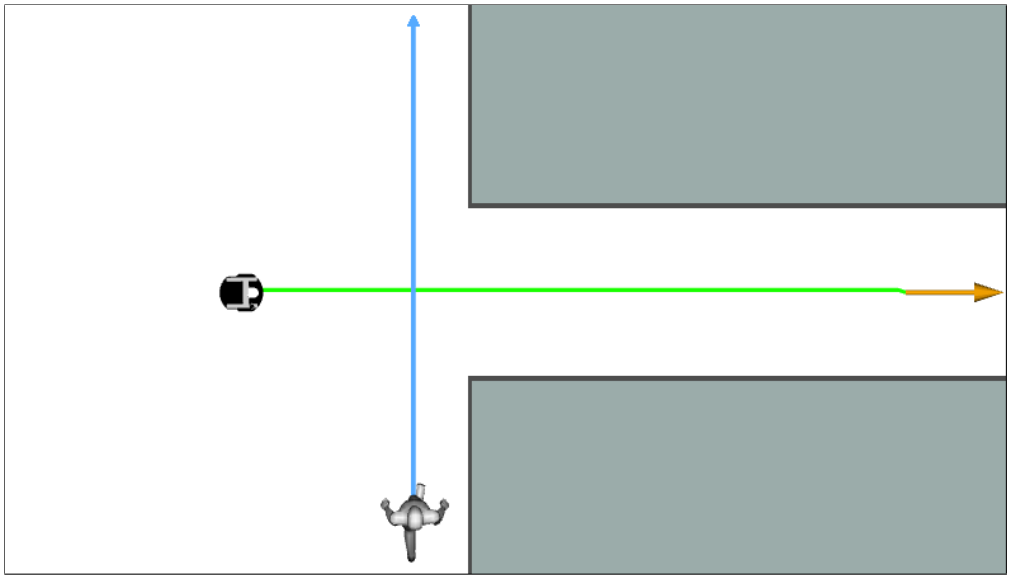
In this chapter, the experimental results are presented in detail and compared with state-of-the-art implementations. First, we outline our testing environment and the used scenarios, then we demonstrate the results of our simulations followed by the real world experiments. Finally, we analyze how our improved MPN implementation performs compared to other approaches and also how our prediction approach improves over a constant velocity and orientation assumption.

5.1 Environments and Scenarios

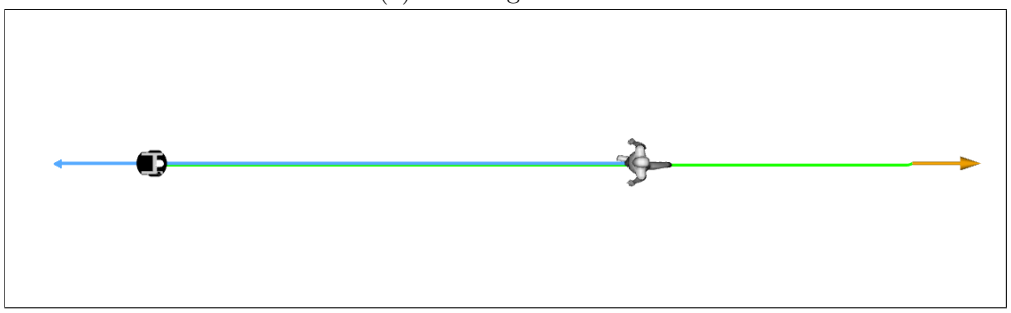
Two different environments were used to test this work in simulated and real world experiments.

At first, the implementation was tested in Gazebo[1], a simulation tool capable of simulating sophisticated physical dynamics. In this simulation, we created both constructed scenarios to test special cases such as crossings, as well as a more general scenario which was a model of the office of the Automation Systems Group at TU Wien. The simulated robot was a model of a differential-drive Pioneer P3-DX®. Human movement was either hard-coded (as for the special test cases), or created by a person flow simulation of the Austrian Institute of Technology[2]. Additionally Rviz was used to visualize the simulated environment, the sensor readings, and the predicted and planned trajectories. Afterwards, tests were performed in our real world environment.

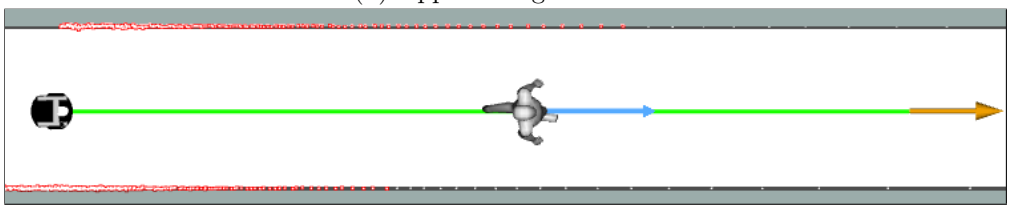
[1] Gazebo: http://http://gazebosim.org
[2] AIT mPed+: http://www2.ffg.at/verkehr/projekte.php?id=726



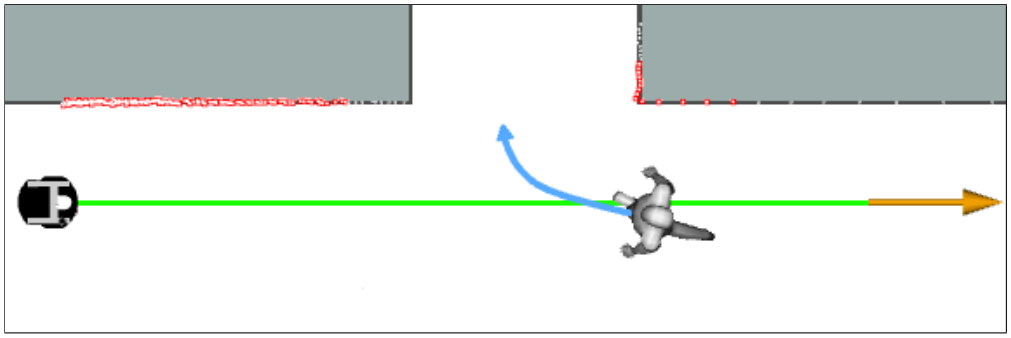
(a) Crossing scenario.



(b) Approaching scenario.



(c) Following scenario.



(d) Approaching scenario with turn.

Figure 5.1: Different constructed simulation scenarios visualized in Rviz.

Examples of the different constructed, special case simulation scenarios can be observed in Figure 5.1. A crossing scenario can be seen in Figure 5.1a, where a human crosses the robot's path at a ~90° angle. In Figure 5.1b the robot and a human approach and pass each other. Figure 5.1c shows the robot driving behind a slow walking person in a narrow hallway. In the last constructed scenario, shown in Figure 5.1d, a person first approaches the robot, but takes a turn to the right in front of the robot. The simulated office scenario is presented in Figure 5.2. Finally, Figure 5.3 shows a real world scenario, where the Pioneer P3-DX® and a person approach each other.

Figure 5.2: Office simulation scenario visualized in Rviz.

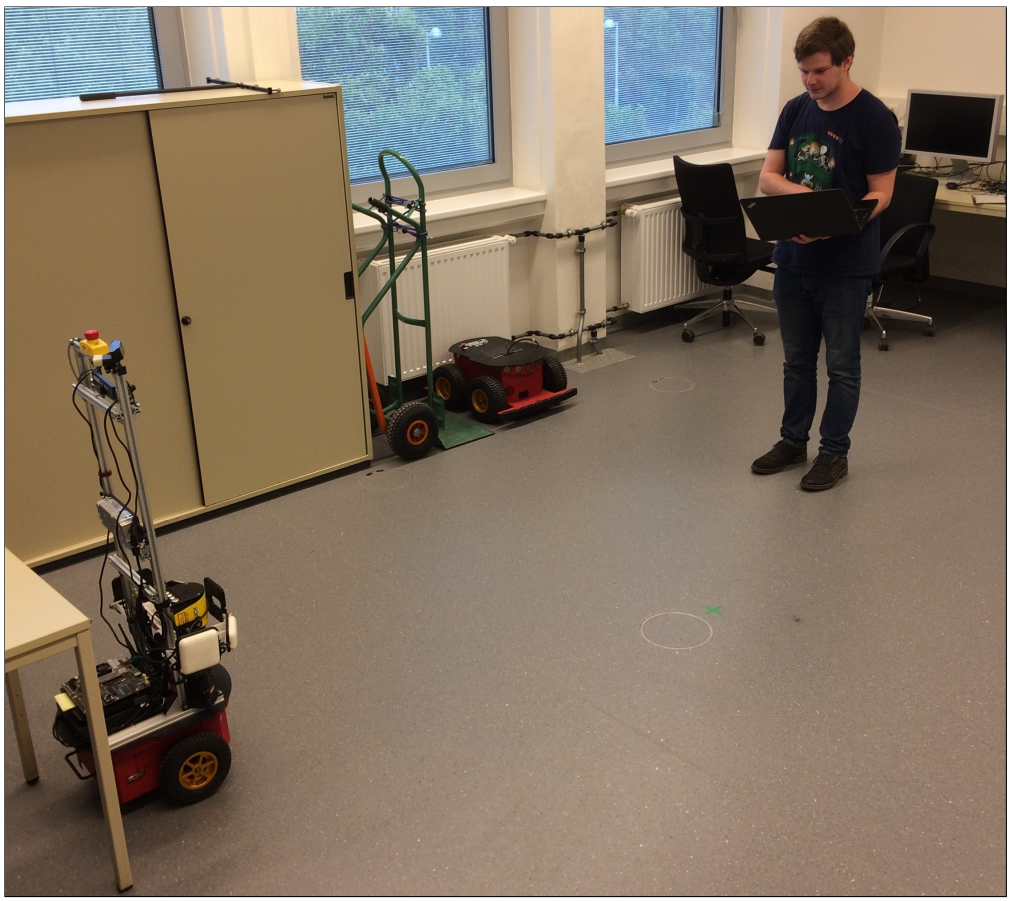


Figure 5.3: Real world experiment, robot and participant are approaching each other.

To evaluate the performance of our improved planner in the scenarios, we compare it to the previous version, without our human awareness, and to other published results. Our chosen performance criteria are the execution time, the resulting path length, and the minimum distance to persons during the maneuver, as those properties can be found in publications, and thus allow us to directly compare our results to others. Execution time and path length characterize the efficiency of the chosen path, while the minimum distance measures the safety aspect as well as the human comfort. Additionally, we recorded the velocities and orientations during the maneuver, in order to identify sudden jerks or stops.

5.2 Simulation Experiments

Testing the implementation in a simulation allows us to experiment in a controllable and perfectly measurable environment. During the development, it also allowed us to easily try new implemented features, and since human interaction is the core aspect, it was also important to perform initial tests without the risk of harming persons. Gazebo was chosen as the simulation tool, as it can be used with ROS and the physical simulation is suitable for our Pioneer P3-DX® model. To test our improved implementation, the person was simulated by publishing a detection message with the current pose. For the old version, additionally a cylinder was added in the simulation, such that the person was visible for the simulated robot in the laser scan. We didn't use this cylinder for the new version, since the people detection module would have filtered the leg detections in the laser scan anyway.

The first tested scenario was the crossing of the robot and a person. The robot started in an open room to the south and the target was to enter a hallway to the north. Meanwhile a person initially stood north-east of the robot, close to the wall, and then walked west in a straight line to pass between the robot and the entrance to the hallway. The setup is shown in Figure 5.1a. For our repeated tests, the robot started at the coordinates $(-1.23, -1.7)$ and faced north, and the person's initial position was $(1.0, 0.0)$ facing west. The target set for the robot was at $(-1.23, 7.0)$ again facing north, and the person moved in a straight line westwards. For each test run, the person's movement speed was set to a constant value between 0.3 m/s and 0.7 m/s, the robot's maximum speed was limited to 4.0 rad/s (or 4.0 m/s, given the wheel diameter of 20 cm) and acceleration to 2.0 rad/s^2. While those speeds were relatively low, compared to typical human walking speed of 1.4 m/s [BBHK06], they were chosen referring to the work of Kollmitz et al. [KHGB15]. Person and robot movements were started manually in short succession to add noise to the experiment. Travel duration, path length and minimum distance to the person were measured, starting when the robot's y-coordinate went above -1.68 and up until it passed 4.0.

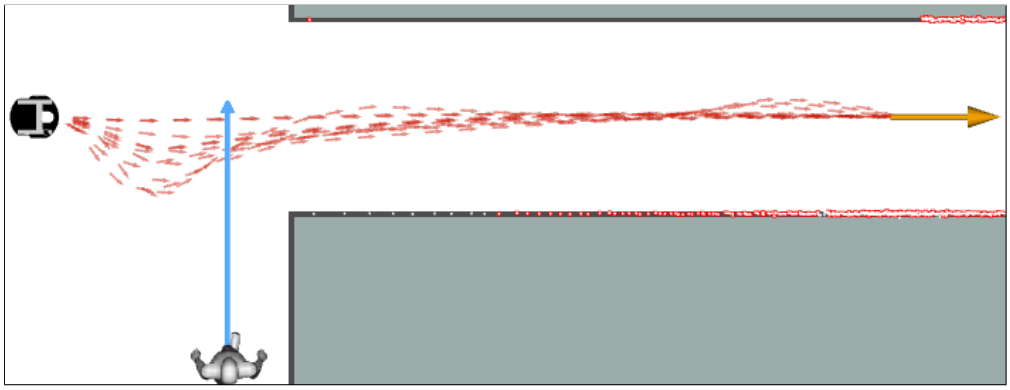


Figure 5.4: Resulting paths in the simulated crossing scenario.

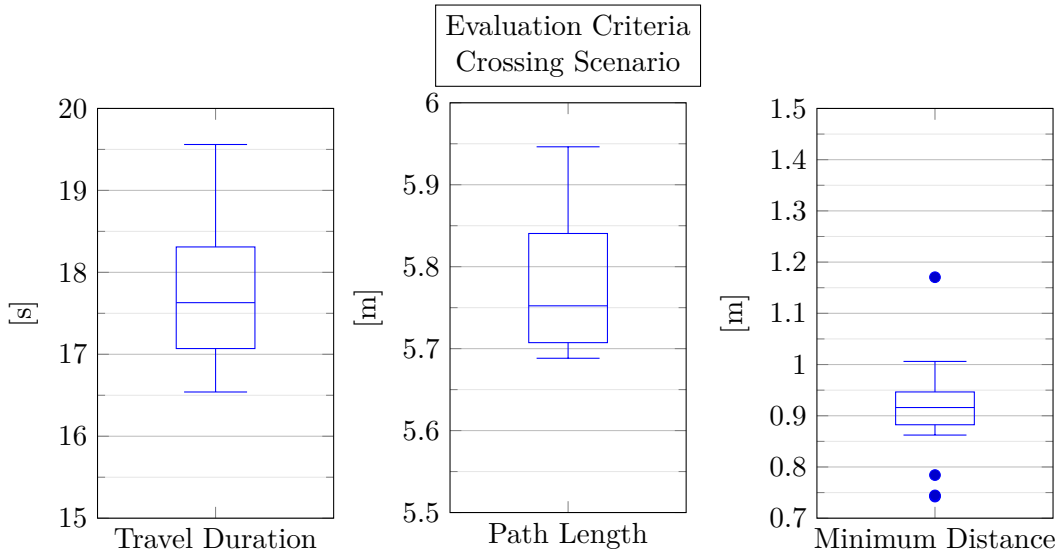


Figure 5.5: Measurements of path lenght, travel duration and minimum distance of our implementation in the crossing scenario.

In the simulation runs, the robot always passed behind the person, as shown in Figure 5.4. The measurements of the evaluation criteria are summarized in Figure 5.5. The robot needed between 16.54 s and 19.56 s to drive 5.68 m to 5.94 m and the minimum distance during the maneuver ranged from 0.86 m to 1.01 m with some outliers at 1.17 m, 0.74 m and 0.79 m, all satisfying the safety distance constraint of 0.7 m. In all runs the robot passed behind the person, depending on which trajectory the optimizer explored first, the planner either chose a straight line or deviating to the bottom. The deviation depended on how long the person took to pass the entrance to the hallway. The faster the person was out of the way, the smaller was the deviation. The planner chose that deviation in order to pass the person earlier and thus it was able to drive at a higher speed. When the planner chose a straight line as its trajectory, the robot accelerated slower, to let the person pass first. There was, however, no noticeable difference in travel duration between deviating to the bottom or driving straight and accelerating slower, as the higher speed made up for the increased path length. This explains why the planner used both strategies, as both were equally good. The travel duration only depended on how much time the person needed to walk past the entrance. When the person's walking speed was high enough such that the person passed the entrance before the robot could get to close, the chosen trajectory was a straight line and the robot was able to accelerate to full speed quickly, leading to a lower travel time. Figure 5.6 shows an exemplary velocity and orientation profile of the robot. As the robot predicted the person's future position, there was never the necessity of slowing down or stopping, only slightly before the robot passed the person the acceleration was reduced a bit such that the safety constraint remained satisfied. Additionally, when deviating to the bottom, the robot rotated smoothly without any sudden jerks.

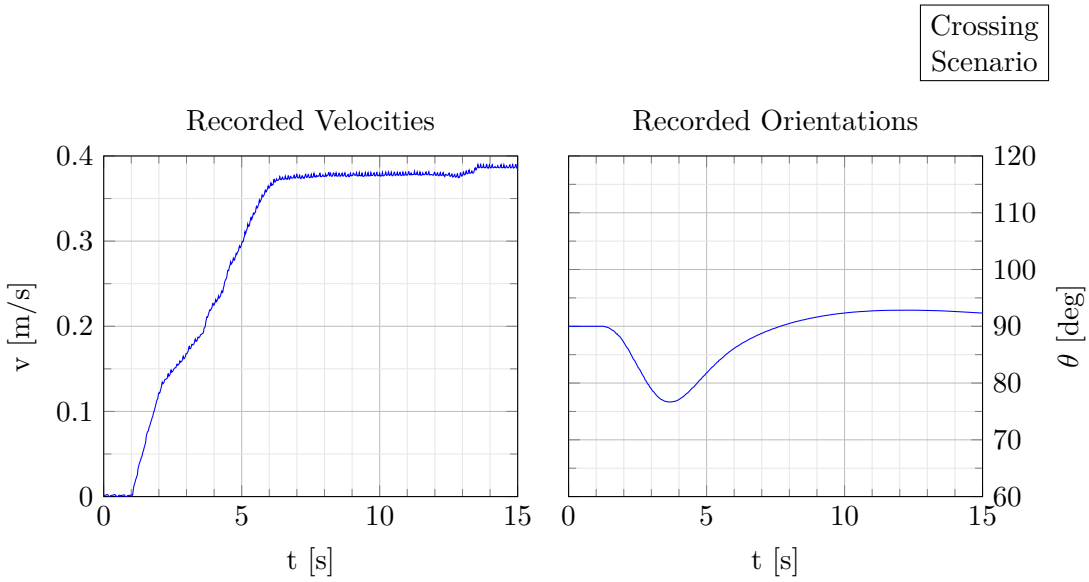


Figure 5.6: Example velocity and orientation progression of our implementation in the crossing scenario. After 1.5 s the robot starts turning with slightly decreased acceleration to safely pass the person. After 3.5 s the robot is behind the person and starts turning back with normal acceleration.

The second tested scenario was the approaching and passing of the robot and a person. Starting left, the robot moved to the right, whereas the person started in right and moved to the left in a straight line, as shown in Figure 5.1b. As there was enough space to either side, the robot had to choose on which side to pass the approaching person. Similar to the first scenario, the person and the robot started in a fixed position and moved towards a target with varying movement speeds of the person. The robot's initial position was $(-2.0, -7.0)$ and for the person it was $(5.0, -7.0)$. Movement for the robot was again limited to 0.4 rad/s and the person's speed ranged from 0.3 m/s to 1.5 m/s. This time, the measurement was started once the robot's x-coordinate passed -1.98 and ended when it reached 5.0.

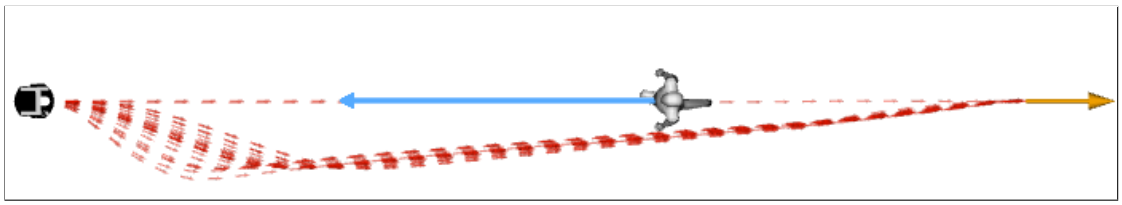


Figure 5.7: Resulting paths in the simulated approaching scenario.

In this scenario, the robot always respected our implemented preference of passing a person to the left side (from the person's point of view), all paths were chosen to pass

the approaching person south, as can be seen in Figure 5.7. A summary of the recorded measurements can be observed in Figure 5.8. Here, the robot needed between 21.26 s and 22.10 s to travel 7.11 m to 7.17 m while keeping a minimum distance to the person between 0.711 m and 0.73 m. The outliers were the cases in which the person walked with a speed of 0.8 m/s to 1.0 m/s. In those cases, the robot didn't have enough time to move aside and therefore only kept a minimum distance of about 0.69 m, this is however still within the error margin of the optimizer, therefore the planner chose to follow through with those trajectories, driving 7.2 m to 7.24 m in 22.10 s to 22.32 s. There was also one case where the robot wasn't able to evade the person, who was walking at 1.5 m/s. Since there was no way the robot could have satisfied the safety constraint, with the maximum velocity limited to 0.4 m/s, the robot waited for the person to pass, before driving towards the target in a straight line. Overall, the path length and travel duration increased with increasing walking speed of the person, while the kept safety distance decreased. The exemplary velocity and orientation profile in Figure 5.9 show that, again, the robot didn't need to perform a sudden brake and turned away from the person smoothly, without any sudden jerks.

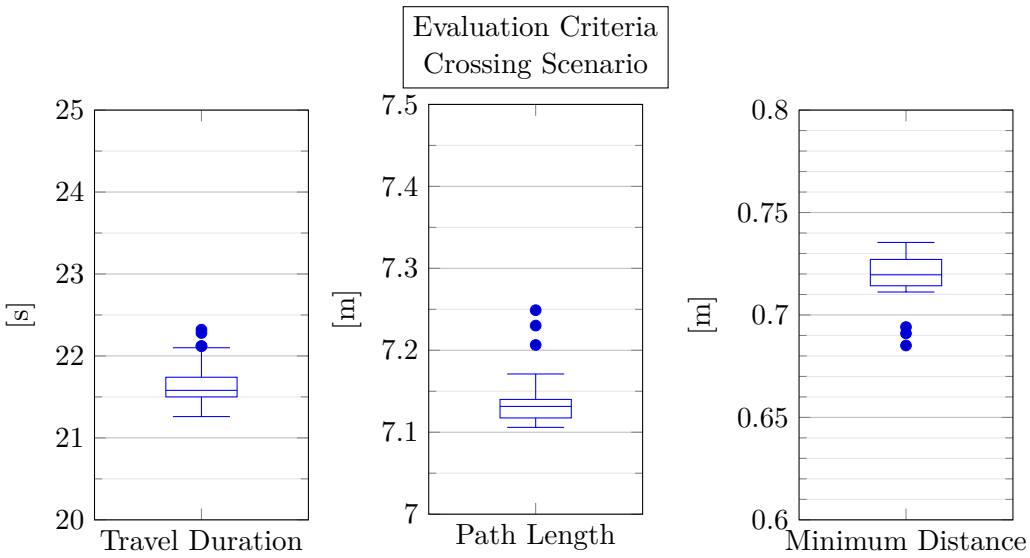


Figure 5.8: Measurements of path length, travel duration and minimum distance of our implementation in the approaching scenario.

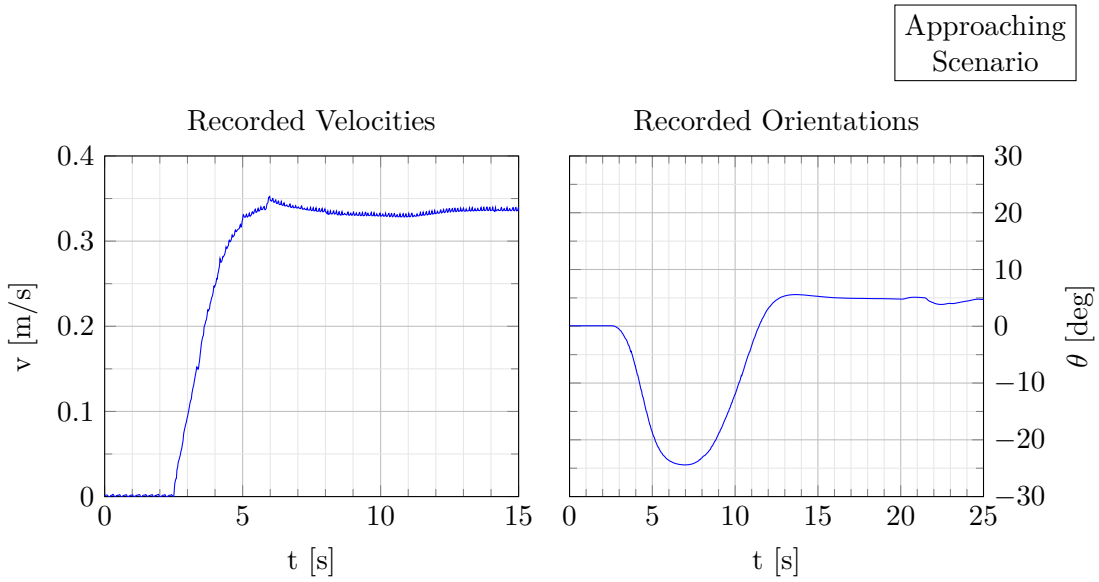


Figure 5.9: Example velocity and orientation progression of our implementation in the approaching scenario.

In our third simulated scenario, the robot started in a hallway behind the person, and both were moving to the same direction. The person's movement speed was limited to be 0.15 m/s, which was slower than the robot's maximum speed of 0.4 m/s, and the hallway was to narrow for the robot to overtake the person. This time, the walking path was not hardcoded to be a perfectly straight line, but rather simulated by the used people flow simulation. Therefore, at times, the person deviated a few centimeters to either side, just like real person's who most likely won't take the perfectly optimal path towards the target. This way, we performed our initial tests on how the implementation responds to such imperfect paths.

In the tests of our new implementation, the robot managed to slowly follow the person. As the robot approached the person with enough space between them, it accelerated towards its maximum speed, however, as it got closer, the speed was gradually reduced. Since the person was detected and predicted to walk at a speed of 0.15 m/s in a straight line, the robot adapted its speed such that it settled down just below the walking speed, at about 0.13 m/s, as can be seen in Figure 5.10.

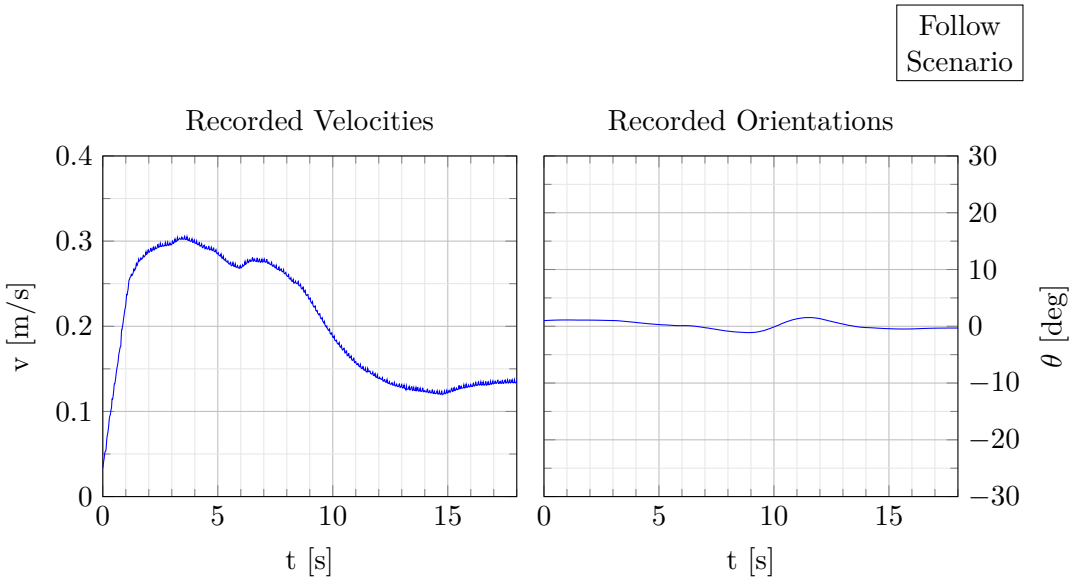


Figure 5.10: Example velocity and orientation progression of our implementation in the follow scenario.

Even though the person blocked the robot's path, predicting the walking motion allowed to robot to plan a trajectory of continuous movement. At one point, starting after about 10 s, the recorded orientation in Figure 5.10 shows that the robot slightly turned. This results from the person's walking path, which was not a perfectly straight line in this scenario, and optimizer trying to find a better solution where the robot could move towards the target faster. Since our passing preference implementation dictates that the person should be passed on her/his left side, the optimizer explored a possible trajectory where the robot had to start turning left. However, since the robot had no way of passing the person, it turned back to directly facing the target.

Finally, in our last scenario, we tested the combination of our trajectory constraints with the prediction strategy. In contrast to the above scenarios, where the person walked in an (almost) perfectly straight line, in this scenario the person took a turn to enter a hallway. Initially, the robot and the person approached each other, similarly to the second scenario, but before they both meet, the person turned right, out of the robot's path. The walking path was provided by the used person flow simulation which showed slight variations between the runs, but the paths were roughly the same. At first, we recorded several simulated persons taking such paths, such that our prediction learned them. Then we ran the scenario, letting the person start at $(-17.2, -3.0)$ with the target being $(0.0, -0.5)$ and the robot started at $(-17.2, 3.0)$ driving to the person's initial position. In each run, the person's walking speed was set to a constant value between 0.12 m/s and 0.40 m/s and the person and robot were again started manually in short succession.

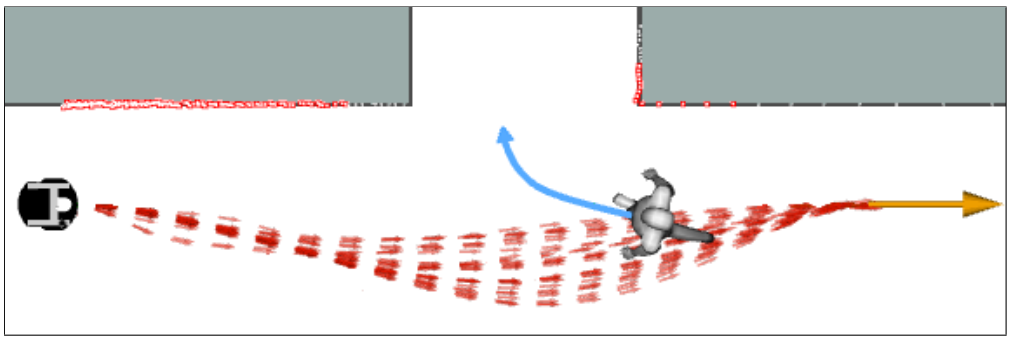


Figure 5.11: Resulting paths in the simulated approaching scenario with turn.

In our experimental runs, the robot predicted the person to take a turn to enter the hallway and depending on the person's walking speed, the robot detoured a little to pass the person at an appropriate distance. The resulting paths can be observed in Figure 5.11, again the robot is shown in its initial position with each sequence of red arrows representing one chosen path. Here, we chose to display the person not in the initial position, but rather at a point where the predicted path (blue arrow) shows the turn towards the hallway. The corresponding summary of the evaluation criteria is presented in Figure 5.12. During the runs we observed that the robot's travel duration, path length and the minimum distance to the person all depended on the walking speed of the person. A faster walking speed meant that the person entered the hallway earlier, where he/she was out of the way of the robot, which resulted in a greater minimum distance between them. Additionally, for faster walking speeds the robot had to detour less, resulting in a shorter travel duration. The exemplary velocity and orientation profiles in Figure 5.13 show, that the robot again was able to smoothly accelerate and turn, without the need of sudden brakes or jerks. In all cases, the robot correctly predicted the person's path, which allowed the robot to efficiently drive towards the target.

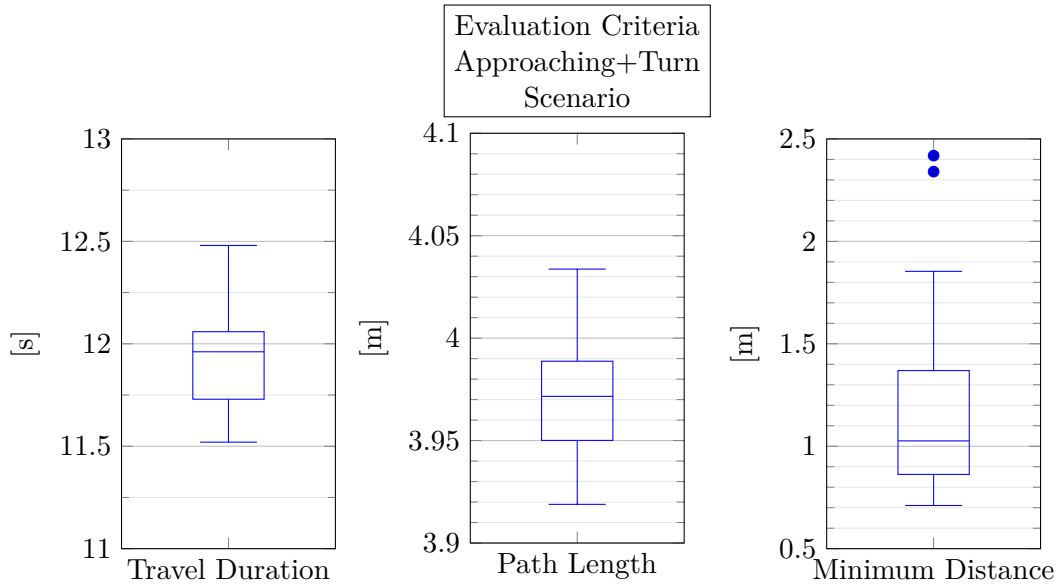


Figure 5.12: Measurements of path length, travel duration and minimum distance of our implementation in the approaching scenario with turn.

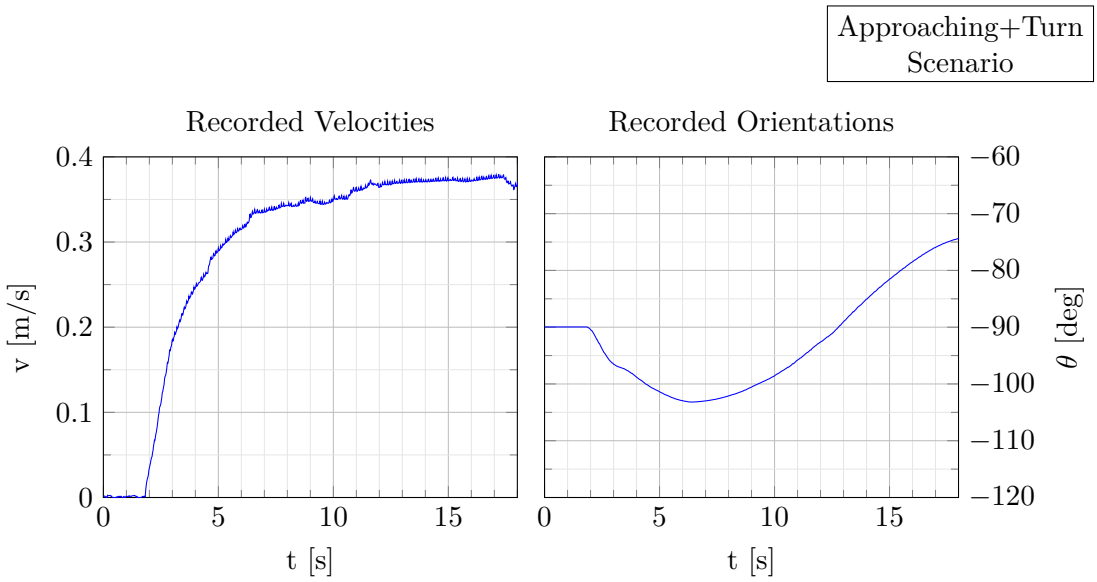


Figure 5.13: Example velocity and orientation progression of our implementation in the approaching scenario with turn.

5.3 Real World Experiments

In comparison to the simulated experiments, the tests on the Pioneer P3-DX® robot in the office showed weak points in the current hardware setup. The initial intent was to let the robot drive through the office on a normal workday, to observe how the evasion process works in real life, and to determine a value for the safety distance parameter that feels comfortable for the participating persons. However, due to limitations of the detection and tracking of persons, we quickly noticed that it was not yet feasible to test in such an environment with persons unaccustomed to driverless vehicles. The main issue with the detection was the limited range and field of view, as the depth images of the used camera could only be used in a range from about 0.5 m to 3 m and in a cone of 59° in front of the robot. Due to this limitation, the robot couldn't start calculating evasive trajectories sooner than when the person was at a distance of maximum 3 m in front of the robot. With typical human walking speeds around 1.4 m/s [BBHK06] and safety distances between 0.5 m and 1.2 m the robot simply didn't have enough time to evade the approaching human, which resulted in a stop until the person passed the robot. Even the walking speed limits of around 0.4 m/s, used in our simulation were too fast.

So we resorted to a special case scenario, where we tested how the robot interacts with a few selected persons following restrictions, such as reduced walking speed or approach angle. The participants were instructed to walk very slowly, with roughly 0.1 m/s and approach the robot directly in front, to give the robot more time to drive away from the predicted human walking path. With these restrictions, the robot was able to successfully plan and follow a trajectory that satisfied a minimum safety distance between 0.5 m and 1.2 m. However, the resulting trajectory was not as efficient as in our simulations. Even with such slow walking speed, the robot didn't have much time to smoothly evade at a shallow angle. It had to immediately turn away at a much greater angle as soon as the person was detected.

An example of such a scenario can be seen in Figure 5.3, where the person was instructed to walk to the robot's initial position, and the robot's target was set behind the person's initial position. The resulting behavior is shown in Figure 5.14 and Figure 5.15. The robot's trajectory contains a sharp turn to evade the person, similar to the simulated runs with high person walking speeds.

Another finding was, that slight instabilities in the detected velocity vector of a person caused large errors in a constant velocity and orientation prediction model, whereas our model stabilized the orientation. Due to the knowledge that persons previously walked in a straight line from one end of the room to the other, the orientation of the prediction was turned towards this line. In Figure 5.14, it can be seen that the detected orientation erroneously points to the top left, but due to our prediction approach, the person is correctly predicted to walk towards the robot's initial position.

During our tests in this scenario, we tried different safety distance parameter values between 0.25 m and 2 m and asked the participants how comfortable they felt during the test. The expectation was, that the robot should at least respect the personal space,

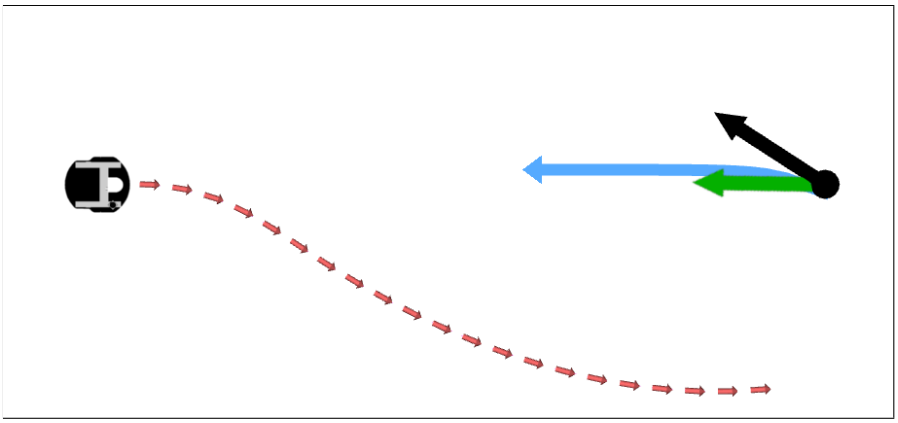


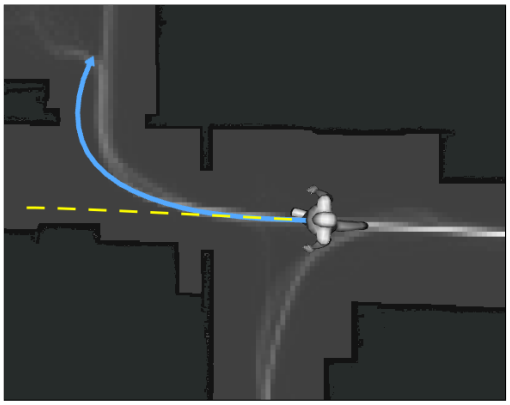
Figure 5.14: Exemplary person prediction and resulting robot path in real world experiment. The robot path is shown by the sequence of red arrows, the person's correct orientation by the green arrow, the wrongly detected orientation by the black arrow and the predicted walking path by the blue arrow.

Figure 5.15: Real world experiment, robot passes an approaching person.

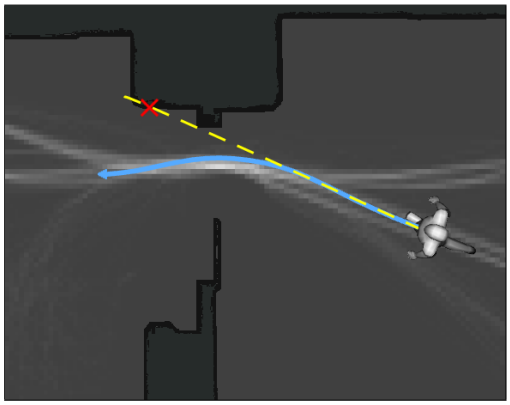
which is usually defined up to 1.2 m [Hal69] around a person. However, in our scenario the participants started feeling comfortable as soon as the robot passed them outside of the intimate space (up to 0.45 m). This could be explained by the fact that the adept participants were used to interactions with the robot, and that the walking and driving speeds were very slow.

5.4 Improved Prediction

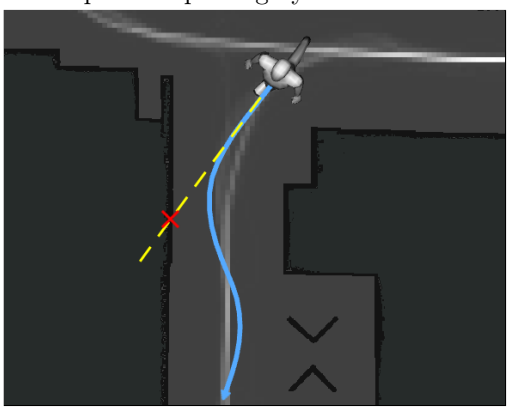
In almost all above experiments, the person moved in a straight line, which perfectly fits the constant velocity assumption. To further test the performance of our person prediction approach with learned data, we simulated a more general environment of the office of the Automation Systems Group at TU Wien.



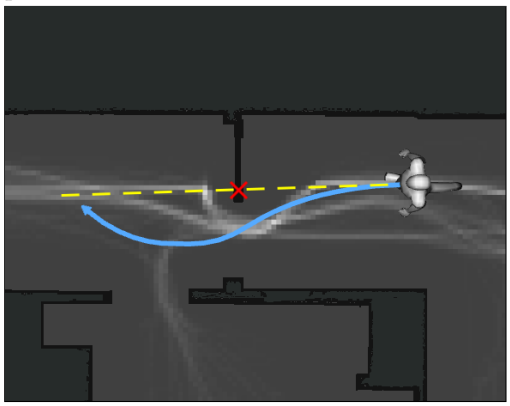
(a) Person is correctly predicted to walk into the room at the top left, while the constant model predicts passing by the entrance.



(b) Person is correctly predicted to walk through the door, while the constant model predicts a crash into the wall.



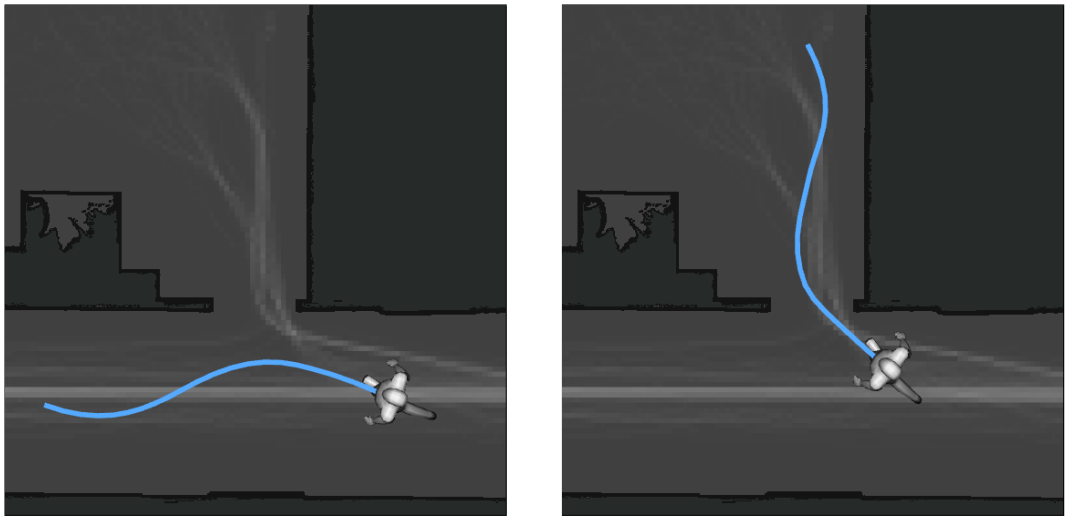
(c) Person is correctly predicted to take a turn and follow the hallway, while the constant model predicts a crash into the wall.



(d) Person is correctly predicted to walk through the open half of the door, while the constant model predicts a crash into the door.

Figure 5.16: Prediction of a person in different situations. The blue paths show the prediction of our approach and the dashed, yellow lines are predictions of a constant model. With the exception of one case, the constant model would always predict a crash, marked by a red cross.

In this environment, we used a person flow simulation of the AIT to quickly generate data samples which were learned by our system. We created a scenario in which persons



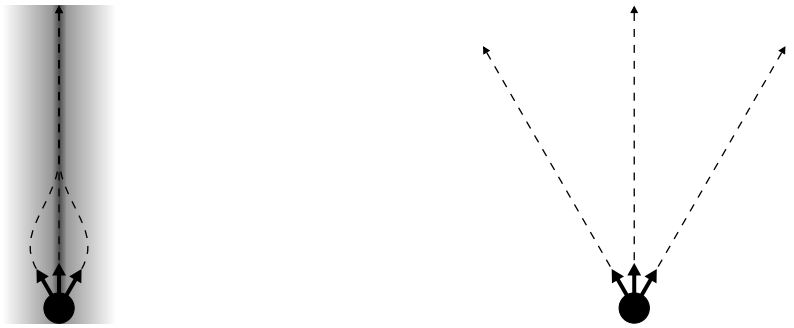
(a) At first the person is predicted (wrongly) to continue walking down the hallway.

(b) Once the person turns towards the room entrance, the prediction is corrected to enter the room.

Figure 5.17: Performance of the prediction approach in a situation where the person has a choice which direction to go.

deployed by the flow simulation walk through the office are predicted by the robot. Since the classic constant velocity and orientation assumption performs poorly in situations where the predicted person can't walk in a straight line, such as approaching a wall, we focused on such situations to check how our approach improves the constant assumption. In the case of our office environment, relevant situations were when a person came from the stairs to enter the office, or whenever a person left a room. Additionally, we made sure that also in cases where a constant assumption would be correct, our approach performs equally well by looking at the predictions of persons walking straight through the hallway. Exemplary predicted paths for different situations can be observed in Figure 5.16, where the blue lines show the path which was predicted by our new approach. In comparison, the dashed, yellow lines show the result of a constant velocity and orientation assumption, and the red crosses show collision point of the constant prediction. During our tests, we noticed that our prediction works best if the person's walking path is limited by the environment, such as in the entrance area where all persons roughly take the same path. If there are choices which path to take, for example whether to go left or right after exiting a room, we always predict the direction where more people have been seen, and therefore predict wrongly whenever a person takes a less likely path. However, when a person takes a turn which differs from our prediction, the prediction is quickly recalculated and adapted to this change. For example, in Figure 5.17a the person is first predicted to walk down the hallway, as that was more likely than entering the room. But as soon as the person turns towards the entrance of the room, our prediction is corrected to enter the room, as shown in Figure 5.17b.

In the real world experiments, we noticed that our prediction provides a way to stabilize uncertainties of the detection, compared to a constant velocity and orientation assumption. With a constant model, we were not able to properly predict the walking path of a detected person, as the detected orientation vector jumped in every measurement cycle. In our experiment, the jumps were about 30° to the left and the right of the actual orientation. Therefore, the predicted paths jumped as well, resulting in unstable behaviour, because the robot first planned to pass the person by turning right, but when the detected orientation jumped in the same direction, the robot had to replan the trajectory, leading to confusing behaviour. After we recorded a person walking the arranged path, in order to create a simple heatmap, the stability greatly improved. Now, even if the detected orientation jumped, the prediction quickly turned back towards the most likely path, such that the robot didn't have to replan the trajectory. Figure 5.18 schematically shows the predictions of an unstable detection, for a constant model, and for our model with a simple heatmap, indicated by the gray background shading.



(a) Our model (b) Constant model

Figure 5.18: Comparison of detection instability handling. When the detected person walks close to a previously recorded path, our model (a) bends the prediction (dashed arrows) towards such a path, even if the detected orientation is off by a few degrees. A constant model (b) experiences a significant error.

Due to the faced complications with the detection framework as mentioned in the previous section, we were not able to test the prediction in a larger scale real world scenario. The limited range of the detection, with the used cameras, was not feasible to set up experiments in a bigger environment. However, the results of our test in the small room were promising, and given a detection framework with greater range, the prediction should perform just as well as in the simulation.

5.5 Comparison With Existing Implementations

To compare our approach to the previous MPN implementation, without human-awareness, we performed the same tests in the three simulated scenarios and the real world scenario.

Using the old implementation in the first simulated scenario, we noticed that in general, the faster the person moved, the greater the kept safety distance was, but up until 0.5 m/s the robot always violated the minimum requirement of 0.7 m. From 0.3 m/s to 0.4 m/s, the robot stopped right in front of the person, once the person got too close. Due to our simplified person simulation, this resulted in a crash, as the person just continued in a straight line. From 0.4 m/s to 0.6 m/s, the robot halted to the left of the person and continued once the way was free again, but still the safety distance was violated. The only case where the robot didn't stop was at (and above) 0.7 m/s, here however the robot chose a very inefficient path. As the person was detected entering the robot's path from the left, the robot detoured to the right, but since the person continued moving, the detour increased until the robot reached the point where it turned around and continued towards the hallway. The paths of our test runs can be seen in Figure 5.19.

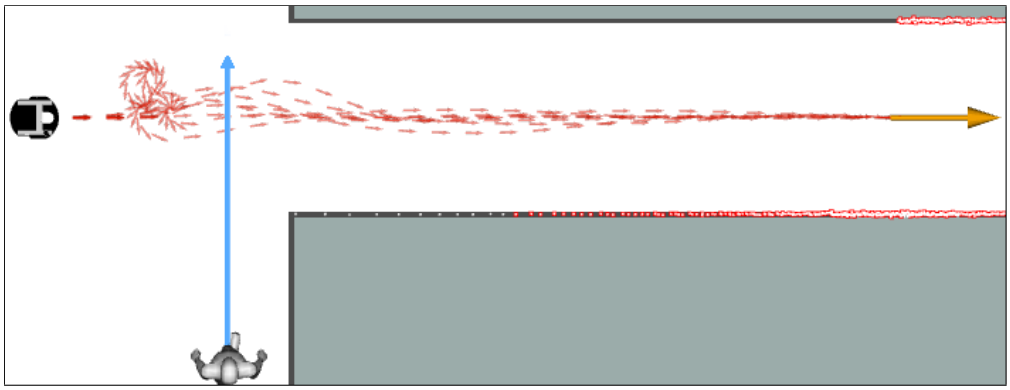


Figure 5.19: Resulting paths of the old implementation in the crossing scenario. The robot spins around, failing to pass in front of the person.

A comparison of the measured evaluation criteria can be seen in Figure 5.20. While the path length was similar to our improved version in most cases and even better in some, the travel duration was always higher, and the safety distance constraint got violated. The shorter paths were the result of the robot not planning to drive around the person until it was too late, so the robot just stopped, and continued driving later. In the velocity graph in Figure 5.21, it can be seen that while at first accelerating faster than in our new approach, the robot had to stop abruptly, as it got too close to the person. Only after the person passed by, the robot accelerated again to continue towards the target. In our approach, we were able to prevent the need for such sudden stops, which results in a movement pattern which is easier to predict for the involved persons. Additionally, right before the robot came to a hard stop it quickly jerked westwards trying to avoid hitting the person.

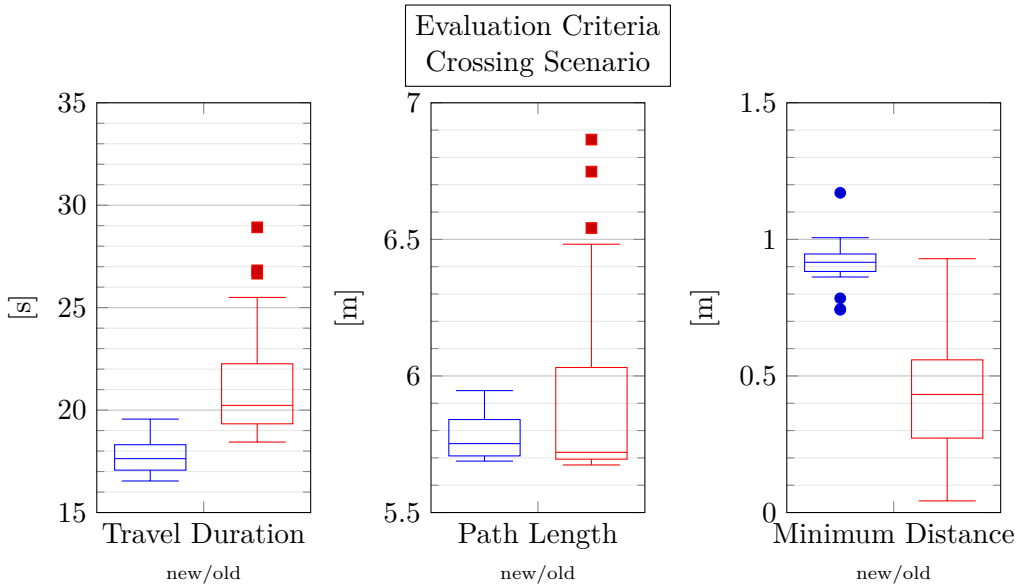


Figure 5.20: Comparison of measurements of travel duration, path length and minimum distance of the old implementation with our new implementation in the crossing scenario.

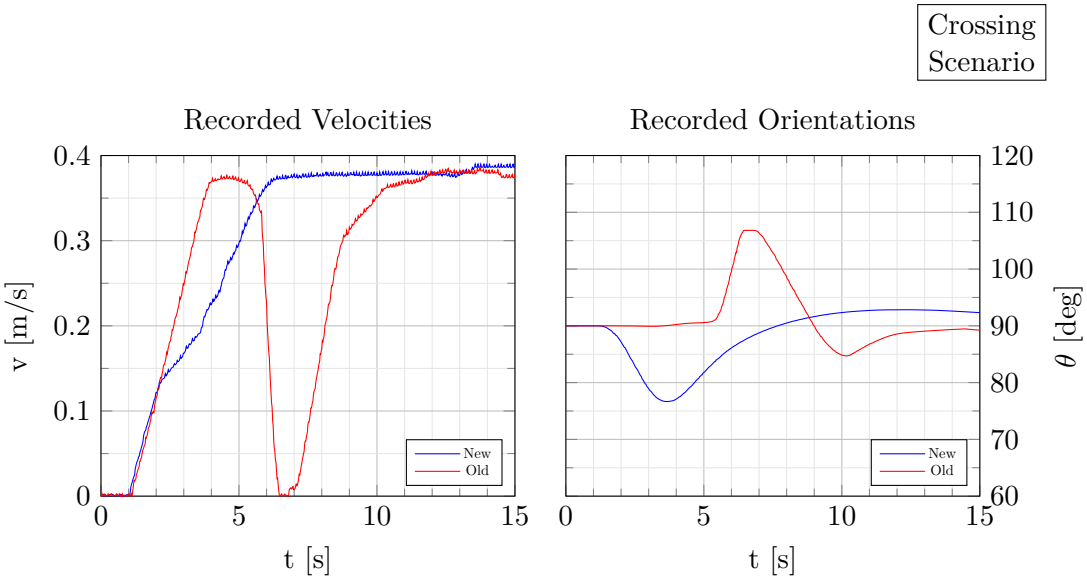


Figure 5.21: Velocity and orientation comparison in the crossing scenario. The old version accelerates faster, but has to perform an emergency brake with sharp turn to the left after 5.5 s. The improved approach starts to turn early, after 1.5 s and decreases the acceleration so the robot can pass the person safely.

In the second scenario, without human-awareness, the robot wasn't able to pass the person without stopping. Up to 0.6 m/s walking speed of the person, the robot always stopped at some point to either side as it got to close to the person. With walking speeds above 0.6 m/s, the robot stopped while still being in front of the person, which again caused a crash with our simulated person. The chosen paths are presented in Figure 5.22.

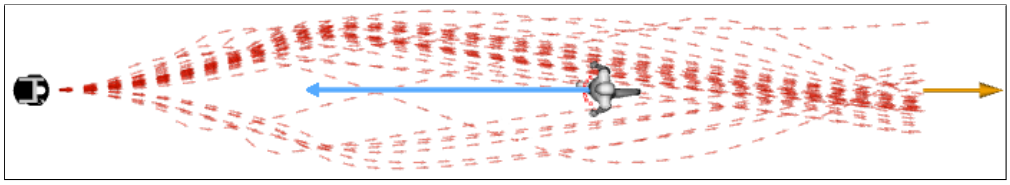


Figure 5.22: Resulting paths of the old implementation in the approaching scenario. The robot passes the person on both sides.

It can be seen that in contrast to our improved approach, the robot passed on either side of the person. As with our new approach, the execution time and path length didn't depend on the person's walking speed, only the kept distance decreased with increasing walking speed. Figure 5.23 shows the velocity and orientation profile for an exemplary run of this scenario, where the robot stopped beside the person. After about 11 s the robot got too close to the person, and therefore had to stop, or at least slow down drastically and quickly turn to the right. Once the person passed the robot and the minimum distance constraint was again satisfied, the robot continued driving towards the target. Figure 5.24 summarizes the comparison of the old and our new approach. While the path length was roughly the same as with our new approach, with 7.1 m to 7.2 m, the travel duration was significantly longer in most cases, and the safety distance got violated in almost all runs. The travel duration ranged from 20.7 s to 28.7 s, as the stopping and accelerating usually caused a delay of a few seconds. In a few runs, the robot reached the target faster than with our new approach, however, it still violated the safety constraint and had to brake close to the person, but only for a brief moment, since the person walked out of the range quickly, so the robot could continue driving before it came to a halt. Only in the first run, where the person walked slowest, with just 0.3 m/s, the robot was able to keep a distance of at least 0.71 m, in all other runs it got too close to the person, and in the runs below 0.4 m minimum distance the person crashed into the robot, which was stopping just in front.

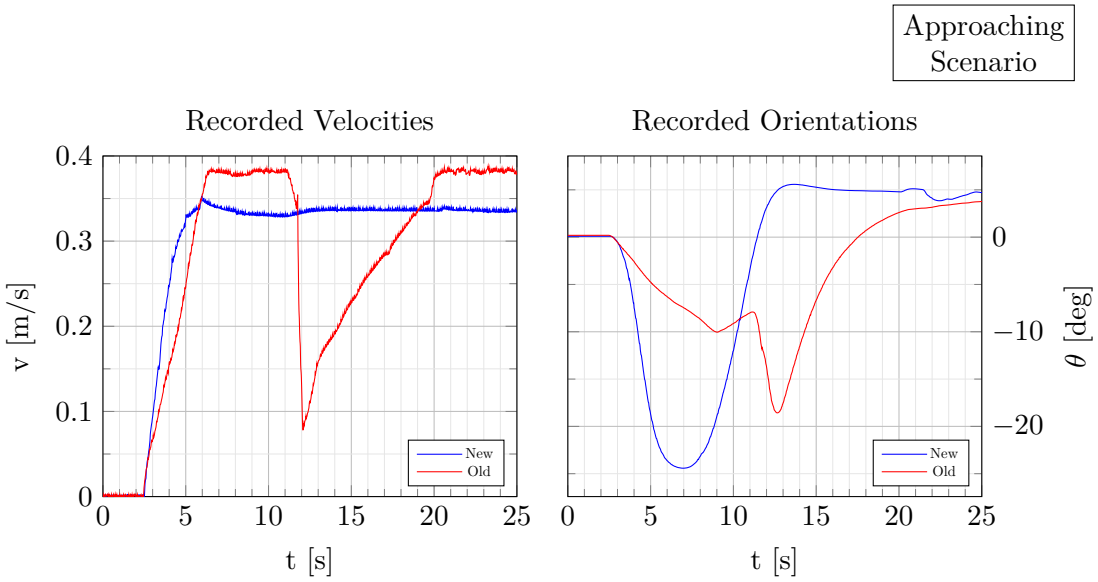


Figure 5.23: Velocity and orientation comparison in the approaching scenario. The old version accelerates to a higher maximum speed, but has to brake drastically after 11 s. Both approaches start turning after 3 s, but the turning angle of the old version is too shallow, and the robot has to take a sharp turn during the emergency brake to avoid crashing into the person.

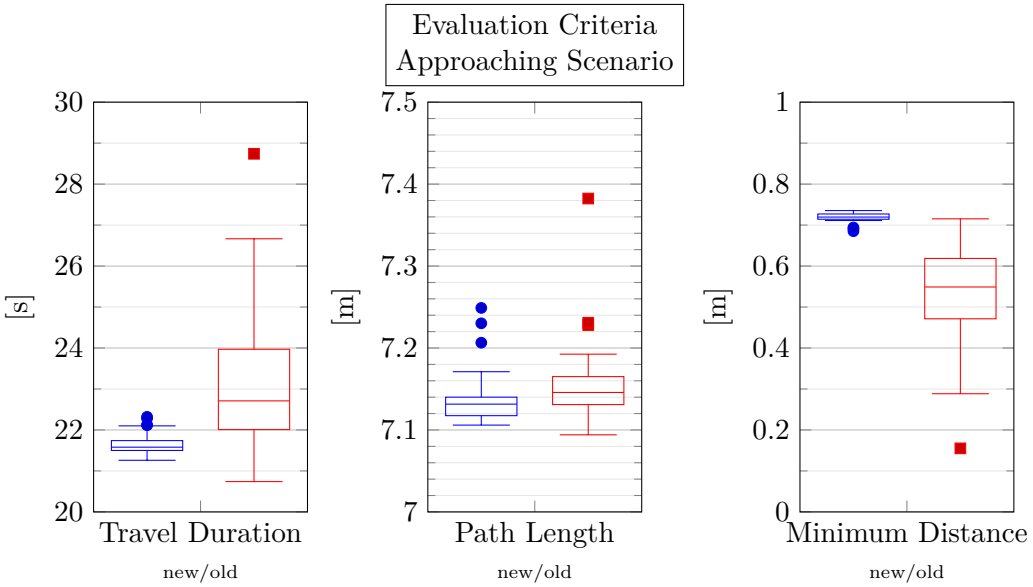


Figure 5.24: Comparison of measurements of travel duration, path length and minimum distance of the old implementation with our new implementation in the approaching scenario.

For the third scenario, where the robot had to follow a person walking slowly in a hallway, the old implementation showed unintuitive results. The robot started driving towards its target, but as soon as the person appeared in the laser rangefinder measurements, the robot abruptly stopped and waited for a while. When the person was out of range again, the robot continued driving, but only to stop again after a few meters, when it was closer to the person again. Corresponding velocity and orientation profiles in direct comparison to the results of our new approach can be found in Figure 5.25. This behaviour happens, since the person is treated as a static obstacle, and the planner sees that this static obstacle is completely blocking the path. Once the path is blocked, it is not feasible to move closer to the obstacle, and since no other path to the target can be found, the robot stops. When the person moves out of the local view, the path towards the target is free again, and the planner selects a new trajectory. But since the robot is driving faster than the person walks, the stopping and accelerating continues until the robot reaches the target. In terms of orientation, the robot stayed on a almost straight line for both the old and our new implementation. While this behaviour is safe, as the robot never got close to the detected person, it is clearly not efficient, whereas our approach showed much more promising results.

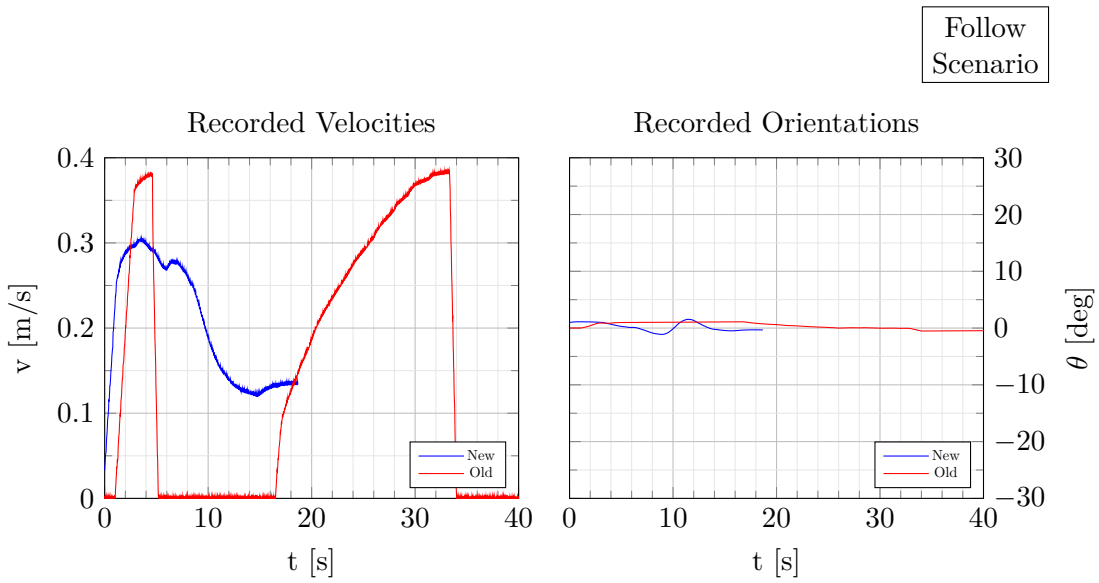


Figure 5.25: Velocity and orientation comparison in the following scenario. The old version starts accelerating quickly, after the person leaves the detection range at 1 s, but as the person is detected again after 5 s the robot stops abruptly. After some waiting, the robot continues only to stop again. With our improved approach, the robot first moves closer to the person, but in order to keep the required distance it slows down and follows the person. After 19 s the robot already reached the target, while the old version required more than 40 s.

As we finally tested the old MPN implementation in our real world scenario, we got very similar results to our improved version. The robot was able to pass the approaching person safely, and only had to stop for a brief moment once. However, as seen in the results of the second simulated scenario, the old implementation performed reasonably well at lower walking speeds of the person, and since the person was limited to 0.1 m/s the movement between sensor readings was small enough for the robot to react in time.

Additionally, when compared to the work of Kollmitz et al. [KHGB15], our approach yielded similar improvements over a static planning approach without human-awareness. In the crossing scenario, their robot was able to pass behind the crossing person, with little detour, by slowing down or waiting for the person to pass. We weren't able to directly compare the experimental results, as their setup wasn't explained thoroughly enough, but in our tests we noticed that slowing down and driving in a straight line wasn't always the fastest way to pass the person. In some cases, our robot detoured slightly to pass behind the person earlier, allowing for a faster moving speed and less deceleration and acceleration, which might be an improvement over their approach. Moreover, our approach computationally outperformed the work of Kollmitz et al., as we were able to update our plan at frequencies of 10-50 Hz, while they were limited to 2 Hz. This higher frequency allows our robot to react to potential prediction errors much faster. It has to be noted that in the time that passed since Kollmitz et al. published their approach, computational power of modern processors increased, however not by a factor of 5-25.

CHAPTER 6

Conclusion

In this final chapter, we first summarize our work, the concept of human-aware navigation, our intention, approach, proof-of-concept implementation and our results. Afterwards we discuss the issues we faced during the course of this work, in developing the approach and implementing it. We show that on one hand our results are an important step towards safe robot navigation in vicinity of humans, but that on the other hand more work still needs to be done to improve safety and performance. Therefore, we finish off with suggestions and ideas how to further improve our approach and implementation in the future.

6.1 Summary

In this work, we made a mobile robot human-aware, such that it can safely operate in environments crowded with humans. This means that we developed a path planner for a driver-less vehicle that lets it navigate through those environments, while avoiding contact with persons, and respecting social rules such as passing side preferences. In order to become human-aware, our approach detects persons and predicts their walking paths in order to preemptively plan collision avoiding trajectories. The prediction is based on learned data, to determine the most likely future position of a person, depending on where many people have been observed before. With this prediction, our MPC based optimization can search optimal velocity sequences such that the robot efficiently reaches its goal, while still satisfying our safety constraint and minimizing social cost. The safety constraint is the Euclidean distance to all detected persons at any point in time along the planned trajectory. The social cost we used tells the optimizer to prefer trajectories which pass a person on his/her left side, instead of the right side, as this is the general passing side preference in European culture. By choosing trajectories that satisfy that constraint and cost, our mobile robot efficiently navigates through the environment, keeps

a safe distance to persons, and tries to behave predictably, such that the persons don't feel irritated, disturbed or threatened by the robot's presence.

In order to provide a proof-of-concept, we implemented our approach by extending our existing navigation framework, which we call MPN, written in C++. Before our improvements, the framework was capable of finding efficient trajectories in static environments, but our experiments showed that it behaved poorly in the presence of moving obstacles, such as walking persons. We added a generic person detection module to our ROS setup and embedded our prediction approach into the MPN. Additionally, we used the provided interface for constraint definitions to formulate our safety constraint, and adapted the framework's cost function, such that it also accounts for the passing side preference. The resulting, new framework was then tested and compared to the old version in several scenarios, both in simulation as well as on a real robot system. In these experiments, we showed that with our improvements, the robot can now smoothly navigate around humans, minimizing the need for sudden stops when it gets too close to a person, and respects our social rule of passing on a person's left side, if possible. Additionally, we compared our new prediction approach to the commonly used constant velocity and orientation assumption, to understand where our approach performs better, and where it still needs improvements. We concluded that it works best, if the person's movement is restricted by the environment, such as walking around a corner in a hallway, where it significantly outperforms a constant model, which would predict the person to walk into a wall. In terms of computational performance, our implementation is able to update the robot's trajectory at 10-50 Hz, enabling the robot to quickly react to unforeseen human movements, which additionally provides safety.

6.2 Challenges and Downsides

During or work, we faced several challenges. First of all, due to their nature, it is very difficult to accurately predict the walking paths of persons. Currently, computers are not capable of reasonably interpreting a person's gestures, facial expressions, postures or all other subtle features which would indicate the person's intention. Without knowing what a person intends, predicting the person's walking path is a mostly random guess. While in some situations such guesses are reasonable enough, for example that, a person rather takes a turn than walking into a wall, there is always the possibility of a sudden change in direction, an abrupt stop, or other unpredictable actions. Therefore, we chose an prediction strategy which roughly follows the main flow, which works well in cases where no choices are involved. However, if for example a person leaves a room, we will always predict the person to walk towards the area which was occupied more in the past, if the person chooses to walk the other way, our prediction will be wrong at first and it will be corrected once the person starts turning to the less likely direction.

In terms of performance, our approach has the problem, that iterating over the recorded grid map, which stores the previously observed person's positions, is computationally expensive. In our proof-of-concept implementation, it was not feasible to evaluate the grid

map at runtime. Our idea was to determine the change in orientation of the predicted person, by examining an area in front of the person, at every step in the prediction. With our update frequency, this would mean that 10 to 50 times per second, for every predicted control point the grid map would have to be queried for every checked input sequence. During our initial tests, we realized that those grid map accesses quickly became a performance bottleneck, therefore we had to find a trade-off between performance and prediction accuracy. We decided to partially precompute the change in orientation by using search areas oriented in the four cardinal directions, such that for every possible position, we save the most likely future orientation for the four cases where the person walks north, east, south or west. At runtime we then linearly interpolate between those four orientations, according to the person's movement direction. In some scenarios, such as taking a 90° turn at a T-junction, this interpolation sometimes caused unintuitive bends or wobbles in the predicted walking path.

Another performance problem was the integration of the passing side preference into the velocity cost map. Currently, we use a single layer which represents the distance, in terms of travel duration, a cell is from the target. The velocities are dampened around a person's detected position, such that passing on the wrong side takes longer. However, since we only have a single layer, for the current point in time, the prediction of the person is not incorporated into the velocity cost map. This implementation was sufficient to give the optimizer a slight nudge to the preferred side, but in more complex situations with turns in the predicted path, this might not work equally well.

6.3 Future Work

Our work was an initial step in making a mobile robot human-aware. While we achieved promising results in our simulated experiments, there is still much work to be done, such that a robot can efficiently and safely drive amongst humans in public spaces.

First of all, the people detection of the used Pioneer P3-DX® robot needs to be improved, such that the system can detect persons at greater distances. Once this has been achieved, our simulated experiments can be replicated with real persons in order to further evaluate our implementation.

As our prediction strategy involves a strong trade-off between performance and accuracy, by precomputing the attraction vectors, future improvements of our implementation might use different algorithms to determine the change in orientation more efficiently. Additionally, our approach cannot model choices between possible paths, as it will always predict the path with more observed persons. If persons' targets could be identified, the learned data could be grouped into paths for different targets, such that the model will predict the most chosen path that leads to the approximated targets. In environments such as an office, where the robot will meet a set of person repeatedly, the targets could be determined by identifying a detected person. However, in public spaces such as a train station, it might be more reasonable to randomly guess a target, depending on where the person is coming from. Another problem of our prediction model is, that it doesn't

account for the robot's position. As mentioned in Section 4.4, the MPN framework is structured such that human-robot interactions could easily be integrated into the prediction, however, those interactions are not sufficiently studied, yet. Currently, most researchers assume that humans react to the presence of a robot in a similar fashion to how they react to other persons, which might turn out to be incorrect. Therefore, it is important to observe how humans interact with robots to derive a model. Integrating such a sophisticated human-robot interaction behaviour into the model could greatly improve the accuracy of the prediction, and therefore also the robot's resulting path.

In Section 3.1 and Section 3.2, we discussed important aspects which the navigational planner should consider as it is searching for a valid trajectory. While we developed an approach to respect safety distance, passing side preference, and legibility, there are several aspects still open for future extensions of our approach. On one hand, visibility could be integrated into the MPN framework in terms of constraints and cost functions. On the other hand, crowd interactions could be handled by a new state in the internal state-machine, which gets activated whenever the robot has to stop due to persons blocking its path.

Summarizing, our work showed, that human-aware navigation is achievable on the local planner or control level. We were able to provide an implementation that not only safely avoids humans, but also follows social guidelines. While most of our work was shown in simulation, it is reasonable to assume that similar results are achievable in real world experiments, when a sufficiently stable detection with increased range can be used. This would then, for example, allow the deployment of assistance robots in crowded public spaces.

List of Figures

List of Algorithms

Bibliography

[BBCT05] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun. Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research*, 24(1):31–48, 2005.

[BBHK06] R. C. Browning, E. A. Baker, J. A. Herron, and R. Kram. Effects of obesity and sex on the energetic cost and preferred speed of walking. *Journal of Applied Physiology*, 100(2):390–398, 2006.

[BBT02] M. Bennewitz, W. Burgard, and S. Thrun. Using em to learn motion behaviors of persons with mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 502–507 vol.1, 2002.

[CB04] E. Camacho and C. Bordons. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2004.

[CELH17] Y. F. Chen, M. Everett, M. Liu, and J. P. How. Socially aware motion planning with deep reinforcement learning. *CoRR*, abs/1703.08862, 2017.

[CKB+17] I. Chaari, A. Koubaa, H. Bennaceur, A. Ammar, M. Alajlan, and H. Youssef. Design and performance analysis of global path planning techniques for autonomous mobile robots in grid environments. *International Journal of Advanced Robotic Systems*, 14(2):1729881416663663, 2017.

[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.

[FBT97] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, Mar 1997.

[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.

[FKLK12] F. Flacco, T. Kröger, A. D. Luca, and O. Khatib. A depth space approach to human-robot collision avoidance. In *2012 IEEE International Conference on Robotics and Automation*, pages 338–345, May 2012.

[FSM11] D. Feil-Seifer and M. Matarić. People-aware navigation for goal-oriented behavior involving a human partner. In *2011 IEEE International Conference on Development and Learning (ICDL)*, volume 2, pages 1–6, Aug 2011.

[GÁGM15] J. V. Gómez, D. Álvarez, S. Garrido, and L. Moreno. Fast methods for eikonal equations: an experimental survey. *CoRR*, abs/1506.03771, 2015.

[Hal69] E. Hall. *The Hidden Dimension: Man's Use of Space in Public and Private*. Doubleday anchor books. Bodley Head, 1969.

[HEGT06] H. Huettenrauch, K. S. Eklundh, A. Green, and E. A. Topp. Investigating spatial relationships in human-robot interaction. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5052–5059, Oct 2006.

[HSMS07] F. Hoeller, D. Schulz, M. Moors, and F. E. Schneider. Accompanying persons with a mobile robot using motion prediction and probabilistic roadmaps. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1260–1265, Oct 2007.

[KBGK12] T. Kruse, P. Basili, S. Glasauer, and A. Kirsch. Legible robot navigation in the proximity of moving humans. In *2012 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 83–88, May 2012.

[KHGB15] M. Kollmitz, K. Hsiao, J. Gaa, and W. Burgard. Time dependent planning on a layered social cost map for human-aware robot navigation. In *Proc. of the IEEE European Conference on Mobile Robots (ECMR)*, Lincoln, UK, 2015.

[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.

[KSF09] R. Kirby, R. Simmons, and J. Forlizzi. Companion: A constraint-optimizing method for person-acceptable navigation. In *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pages 607–612, Sept 2009.

[KSG12] J. Kessler, J. Strobel, and H.-M. Gross. *Avoiding Moving Persons by Using Simple Trajectory Prediction and Spatio Temporal Planning*, pages 85–96. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[KW91] S. J. King and C. F. R. Weiman. Helpmate autonomous mobile robot navigation system. In *Proc.SPIE*, volume 1388, pages 1388 – 1388 – 9, 1991.

[LASHH06] A. D. Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger. Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1623–1630, Oct 2006.

[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.

[LFS+17] P. A. Lasota, T. Fong, J. A. Shah, et al. A survey of methods for safe human-robot interaction. *Foundations and Trends® in Robotics*, 5(4):261–349, 2017.

[LRWB09] G. Lidoris, F. Rohrmuller, D. Wollherr, and M. Buss. The autonomous city explorer (ace) project - mobile robot navigation in highly populated urban environments. In *2009 IEEE International Conference on Robotics and Automation*, pages 1416–1422, May 2009.

[LSTA10] M. Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras. People tracking with human motion predictions from social forces. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, Anchorage, USA, 2010.

[Man16] B. Mandal. Face recognition: Perspectives from the real world. In *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1–5, Nov 2016.

[MHG+09] M. Moussaïd, D. Helbing, S. Garnier, A. Johansson, M. Combe, and G. Theraulaz. Experimental study of the behavioural mechanisms underlying self-organization in human crowds. *Proceedings of the Royal Society of London B: Biological Sciences*, 276(1668):2755–2762, 2009.

[PL15] L. Pacheco and N. Luo. Testing pid and mpc performance for mobile robot local path-following. *International Journal of Advanced Robotic Systems*, 12(11):155, 2015.

[RKC98] E. Rimon, I. Kamon, and J. F. Canny. *Local and Global Planning in Sensor Based Navigation of Mobile Robots*, pages 112–123. Springer London, London, 1998.

[SMUAS07] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5):874–883, Oct 2007.

[TdPPF00] V. J. Traver, A. P. del Pobil, and M. Perez-Francisco. Making service robots human-safe. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, volume 1, pages 696–701 vol.1, 2000.

[THK09] S. Thompson, T. Horiuchi, and S. Kagami. A probabilistic model of human motion and navigation intent for mobile robot path planning. In *2009 4th International Conference on Autonomous Robots and Agents*, pages 663–668, Feb 2009.

[Tod18] G. Todoran. Optimal local path-planning and control for mobile robotics. Master's thesis, TU Wien, 2018.

[WBV+14] A. Weiss, M. Bader, M. Vincze, G. Hasenhütl, and S. Moritsch. Designing a service robot for public space: An "action and experiences" - approach. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, HRI '14, pages 318–319, New York, NY, USA, 2014.