# Perception and Learning for Mobile Robots in Populated Environments

Marina Kollmitz

Technische Fakultät
Albert-Ludwigs-Universität Freiburg

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Betreuer: Prof. Dr. Wolfram Burgard

# Perception and Learning for Mobile Robots in Populated Environments

Marina Kollmitz

Dissertation zur Erlangung des akademischen Grades Doktor der Naturwissenschaften
Technische Fakultät, Albert-Ludwigs-Universität Freiburg

| | |
|---|---|
| Dekan | Prof. Dr. Rolf Backofen |
| Erstgutachter | Prof. Dr. Wolfram Burgard |
| Zweitgutachter | Prof. Dr. Thomas Brox |
| Tag der Disputation | 14. Mai 2021 |

# Zusammenfassung

Roboter führen heutzutage hauptsächlich repetitive Aufgaben im industriellen Kontext aus und agieren in speziell auf sie zugeschnittenen Anlagen. Doch die Nachfrage nach flexiblen Servicerobotern, die in unmittelbarer Nähe zum Menschen eingesetzt werden können, steigt stetig. Dieser Trend geht mit einer Vielzahl von Herausforderungen einher. Einsatzgebiete, die auf den Menschen ausgerichtet sind, wie Wohnbereiche, Büros, Krankenhäuser oder Flughäfen, sind oft unstrukturiert und verändern sich mit der Zeit. Außerdem treffen Roboter in Wohnbereichen und öffentlichen Räumen auf eine Vielzahl von Menschen mit unterschiedlichen Bedürfnissen und Präferenzen. Die meisten dieser Menschen haben wenig Erfahrung mit Robotern. Zusätzlich müssen die Anschaffungs- und Wartungskosten von Servicerobotern um Größenordnungen niedriger sein als für Industrieroboter, um für einen großflächigen Einsatz wirtschaftlich zu sein. Diese Herausforderungen erfordern ein höheres Maß an Autonomie und Flexibilität und verlangen, dass Serviceroboter sich kontinuierlich an ihre Umgebung anpassen und auf die Menschen, mit denen sie interagieren, eingehen.

In dieser Doktorarbeit werden Perzeptionsmethoden und Ansätze des maschinellen Lernens für den Einsatz mobiler Roboter in auf Menschen ausgelegten Arbeitsbereichen vorgestellt. Zunächst präsentieren wir eine Bildverarbeitungsmethode zur Erkennung von Menschen, die diese zusätzlich anhand ihrer Mobilitätshilfen unterscheidet. Mithilfe der erweiterten Personenerkennung können Roboter individuelle Einschränkungen und Anforderungen von Menschen wahrnehmen und sie so bedürfnisgerecht unterstützen. Des Weiteren stellen wir ein Ganzkörpersensorikkonzept vor, mit dem mobile Roboter Kollisionen und Interaktionskräfte wahrnehmen können. Aufgrund der hohen Dynamik und fehlenden Struktur ihrer Arbeitsbereiche und einer typischerweise eingeschränkten Sensorabdeckung können viele mobile Serviceroboter Kollisionen und unbeabsichtigte Kontakte mit Hindernissen oder Personen nicht vollständig ausschließen. Unser Sensoraufbau basiert auf einem zentral montierten Kraft-Momenten-Sensor und ermöglicht es mobilen Robotern, solche Kontakte wahrzunehmen und angemessen auf sie zu reagieren, um Schäden zu vermeiden. Aufbauend auf dem Kraftwahrnehmungskonzept stellen wir einen lernbasierten Ansatz vor zur Vorhersage von Kollisionen in 2D-Umgebungskarten, die auf Daten planarer Laserscanner basieren. Unser Ansatz interpretiert unerwünschte Kollisionen als Trainingsbeispiele, sodass mobile Roboter mit der Zeit verbesserte Umgebungsmodelle erlernen können. Abschließend untersuchen wir, wie Menschen einem kraftnachgiebigen mobilen Roboter soziale Navigation beibringen können, indem sie ihn entlang ihrer gewünschten Trajektorien schieben. Basierend auf der Interaktion passt der Roboter seine Navigationsfunktion mithilfe von Inverse Reinforcement Learning an. Da kraftnachgiebige Steuerung auch für Laien einfach umzusetzen ist und kein externes Steuergerät benötigt wird, kann jeder in Reichweite des Roboters mit ihm interagieren.

Die vorgestellten Ansätze wurden in umfangreichen Experimenten mit besonderem Fokus auf realitätsnahe Szenarien evaluiert. Die Experimente zeigen, wie mobile Roboter, die die spezifischen Bedürfnisse und Anforderungen von Menschen wahrnehmen können, diese besser unterstützen können. Sie bestätigen weiterhin, dass mobile Roboter durch die gezeigten Ansätze bessere Umgebungsmodelle erlernen und auf die individuellen Vorlieben von Menschen eingehen können. Diese Arbeit ist somit ein wichtiger Schritt auf dem Weg zu flexiblen, autonomen Servicerobotern, die robust und zuverlässig in auf den Menschen ausgerichteten Arbeitsbereichen agieren können.

# Abstract

Robots nowadays mostly perform repetitive tasks in factory settings specifically designed for them. However, the demand for flexible service robots that work in close proximity to humans is steadily increasing. This trend comes with a variety of challenges. Human-centered environments, such as homes, offices, hospitals, or airports, are often unstructured and may change over time. Furthermore, robots in populated environments encounter a variety of people with different needs and requirements who typically have little experience with robots. Lastly, the acquisition and maintenance costs of service robots need to be orders of magnitude lower to be applicable for wide-area use. These challenges demand increased levels of autonomy and flexibility and require service robots to continuously adapt to the environment and people they interact with.

This thesis presents advanced perception and learning techniques for mobile robots in human-centered environments. First, we present a people detection framework that distinguishes them according to their mobility aids. With our framework, robots can reason about people's individual needs and requirements to provide appropriate assistance. We further present a whole-body sensory concept that enables mobile robots to perceive collisions and interaction forces. Due to sensor limitations and the unstructured and dynamic nature of populated environments, most mobile robots cannot entirely rule out collisions or unintended contacts with obstacles or people. Our setup is based on a central force-torque sensor and enables mobile robots to perceive such contacts and react appropriately to avoid harm and damage. Building upon the force perception, we propose a learning-based method to predict collisions in 2D occupancy maps from planar laser rangefinders. Our method turns undesired collision events into training examples for mobile robots to learn improved environment models over time. Finally, we investigate how people can teach a robot socially compliant navigation by pushing it along their desired trajectories. The robot updates its navigation function based on the interaction via inverse reinforcement learning. Since force control is easy for non-expert users and does not require an external control device, everyone in reach of the robot can interact with it.

We performed extensive experiments with a particular focus on real-world scenarios. The experiments showcase how mobile robots that reason about people's special needs can provide better assistance and confirm that our approaches enable mobile robots to learn improved environment models and adapt to people's individual preferences. The methods in this thesis constitute important steps towards flexible, autonomous service robots that can robustly and reliably operate in human-centered environments.

# Acknowledgments

I am very grateful for all the help and support from my supervisor, colleagues, friends, and family, without whom this thesis would not have been possible. First, I want to thank my supervisor Wolfram Burgard for the opportunity to pursue my PhD in the Autonomous Intelligent Systems lab at the University of Freiburg. Thank you for your guidance, helpful discussions and consultation, your ideas, for sharing your expertise, and for the equipment and funding you provided. I am also grateful for the opportunity to travel to international conferences to present my work and connect to many inspiring researchers from the robotics community.

I further want to thank my coauthors Daniel Büscher, Torsten Koller, Andres Vasquez, Tobias Schubert, Andreas Eitel, and Joschka Boedecker. Thank you for all the hard work you put into our papers, your ideas, and the fruitful discussions. It was a pleasure collaborating with you. I was also fortunate to work with many students during my time at the University of Freiburg who did student projects, their Bachelor's or Master's thesis under my supervision, or worked with me as a Hiwi. I enjoyed working with all of you, and it was great to see your devotion and creativity. I want to give special thanks to Lukas Klein, Andres Vasquez, and Archit Bansal, who helped me with my experiments, the robot hardware and software, and worked with me on the NaRKo project.

I want to thank my colleagues at the AIS for the inspiring discussions, friendly chats and laughs we had together. Special thanks to my office mates Tim Caselitz, Chau Do, Camilo Gordillo, Alexander Schiotka, Shengchao Yan, and Jingwei Zhang for generating a very positive and connected office atmosphere. Thinking back to our anecdotes and inside jokes always makes me laugh. I want to thank Andreas Wachaja for the many times he helped with my hardware and electronics-related issues. Furthermore, thank you to Susanne Bourjaillat and Michael Keser for their administrative support and help with computer problems. Another big thank you goes to Daniel Büscher, Jan Kollmitz, Johannes Meyer, and Jannik Zürn for giving constructive comments and suggestions to earlier versions of this thesis. Also, I want to thank all my colleagues, friends, and the students and volunteers who took part in the robot experiments for their time and effort.

Finally, I want to thank my friends and family who supported me in the happy and the challenging times. Thank you to my parents, who always believed in me and gave me the confidence to follow my strengths and interests. Most importantly, I want to thank my husband Jan for always being there for me and motivating me along the way. Thank you for the many discussions we had over my research problems and the many great

suggestions and ideas you provided, for nudging me to give my best, and laughing with me no matter what.

This work has been partially supported by the German Federal Ministry of Education and Research (BMBF), contract number 01IS15044B-NaRKo.

# Contents

# Chapter 1

# Introduction

Robots are nowadays primarily employed in factories where they are physically separated from their human coworkers for safety reasons. Starting in 1969, when General Motors first automated its Ohio car manufacturing plant with Unimate welding robots [120], industrial robots increasingly dominated the shop floors. With $338$ robots per $10{,}000$ employees in Germany in 2018 [59], they play a crucial role in the manufacturing industry. Industrial robots relieve human workers of strenuous labor and fuel economic growth and wealth through their efficiency, accuracy, and speed. The increased production yields justify the substantial acquisition costs and setup efforts.

In contrast to industrial robots, service robots typically operate in areas not specifically designed for them, requiring increased flexibility and adaptiveness. They perform tasks for humans at different degrees of autonomy and often operate with and in close proximity to them. While service robots for professional and personal use had a smaller global annual sales value in 2018 than their industrial counterparts – 12.8 billion compared to 16.5 billion US-Dollars [59, 60] – the International Federation of Robotics (IFR) expects their sales value to more than triple from 2018 to 2022 [60]. With the growing demand, robots will increasingly make their way into our everyday life.

Since the first mass-produced domestic robot Roomba became commercially available in 2002 [62], cleaning robots and robotic lawnmowers took over chores in many private households [60]. In professional settings, service robots relieve humans from mundane and repetitive tasks like industrial robots have for decades. They perform deliveries in hospitals and homes for the elderly [21] and free time for the medical staff to engage with patients. Furthermore, robots help nurses lift patients into and out of their hospital beds [24], thus easing their physical effort. Mobile service robots guide visitors through public areas, such as museums [19, 141], shopping malls [40, 64], or airports [143]. In an aging society, service robots could further assist elderly people in maintaining a self-determined lifestyle. Furthermore, robots in factory settings that can work in collaboration with human workers allow for more flexible factory designs, automation at smaller lot sizes, and customization. All above examples show that service robots have great potential to facilitate and enhance our personal life in the same way industrial robots transformed human labor in the manufacturing industry.

## 1.1 Challenges for Robots in Populated Environments

The recent development towards flexible robots that operate in human-centered environments entails a variety of novel challenges. Now that humans and robots coexist in shared spaces, service robots need to be aware of people and interact appropriately with them. Even service robots that do not intend to provoke people's reactions – meaning the interaction is only passive – will still influence them by their presence. While industrial robots were exclusively operated by experts, service robots interact with a variety of people with different backgrounds and abilities. Thus, the interaction needs to be as easy and intuitive as possible to enable successful communication. In addition, a service robot that performs tasks for humans needs to be aware of their individual needs and abilities to provide appropriate assistance.

Industrial robots operate physically separated from humans, but service robots that operate in direct proximity need novel concepts to ensure human safety. Moving people pose an additional challenge because robots have to continually adapt their environment model and anticipate people's future actions for safe operation. Nevertheless, collisions with obstacles or people can often not entirely be avoided in dynamic environments, especially when people do not notice a navigating robot or cannot understand what it is doing. Service robots should be able to perceive such accidental contacts and react appropriately to them.

Perceived safety is another essential criterion to ensure that humans accept robots around them. People should never feel irritated or even threatened by robots around them. Thus, the actions of the robot not only need to be safe; Humans also need to perceive them as safe and comfortable. The perceived safety and comfort are of paramount concern in domestic applications because people's homes represent a private and intimate space. To ensure human comfort, robot behavior needs to be predictable and easily understandable. Depending on the application and task, people additionally expect service robots to obey higher-level social norms and cultural conventions.

In contrast to industrial robots, most service robots operate in environments not designed for them. While some service robots use specifically installed beacons, like floor markers in logistics applications [3, 132, 134] or electric wires that act as barriers for robotic lawnmowers [14], it is neither desirable nor feasible to install additional structures for all robot tasks. Populated environments are also typically unstructured and may change over time, thus demanding a rising level of autonomy. At the same time, the acquisition and setup costs of robots for domestic or public use need to be orders of magnitude lower than for industrial robots to be attractive for wide-area use. The cost limitation requires the robot to handle incomplete environment information since covering the entire robot workspace with sensors is not cost-effective. As mass products, we can also no longer rely on experts to set up every robot for its site of operation. Instead, robots need to be adaptive and robust to enable their employment in very diverse and changing environments. In contrast to industrial robots, service robots need to continuously adapt to their environment and the people they interact with to successfully fulfill their tasks.

## 1.2 Contributions

This thesis presents novel perception and learning techniques to overcome some of the challenges mobile robots face in populated environments. We investigate the perception of people and reason about their individual needs by classifying them according to the mobility aids they use. Furthermore, this thesis examines how mobile robots can perceive and learn from interaction forces. We show how robots with incomplete sensor coverage can learn better environment models from collision events. Further, we enable service robots to learn people's desired robot navigation behavior through physical interaction. This section outlines the thesis and summarizes its main contributions.

### Perception of People and Their Mobility Aids

Robots in populated environments encounter people with diverse abilities and requirements. People perception is a well-studied field, but most approaches neglect people's individual needs. Chapter 2 presents a novel method to detect people in images and classify them according to the mobility aids they use. In addition to the perception of people, our approach reasons about people's walking impairments and thus enables service robots to provide appropriate assistance. Our approach uses a state-of-the-art image detection framework based on a deep convolutional neural network. For robots that interact with people in a shared environment, locating them in images alone is insufficient. Therefore, we extend the framework by a 3D centroid regression output to predict the positions of people in the world reference frame. Our probabilistic class, position, and velocity tracking module accounts for occlusions and false detections. We further collected and annotated a novel dataset with image bounding boxes and 3D centroids of people from five mobility aids classes: person in a wheelchair, person with a walking frame, person using crutches, person pushing another person in a wheelchair, and pedestrian. Finally, a human-robot experiment with a mobile robot guiding participants either to the stairs or to the elevator of an office building, depending on the perceived mobility aids, showcases a typical application where knowledge of potential walking impairments is essential.

### Perception of Interaction Forces

Most service robots cannot fully rule out collisions with the environment or people due to imperfect people detections, incomplete sensory information, or the dynamic nature of populated environments. At the same time, physical interaction between people and service robots has great potential for intuitive communication. In Chapter 3, we present a novel sensory concept that allows mobile robots to perceive interaction forces. The sensory concept is based on one 6 DoF force-torque sensor mounted between the robot base and a floating shell, thus providing force sensitivity to all exposed robot parts. We further introduce a learning-based force filter to distinguish between external forces and disturbance forces caused by the motion of the mobile platform itself. Our sensory concept combined with the force filter enables the robot to perceive interaction forces during autonomous motion, allowing it to respond appropriately to contacts with people and the environment for safer and more compliant operation. Chapters 4 and 5 further show how mobile robots can learn from interaction forces to improve their navigation behavior in populated environments.

### Learning From Collisions With Obstacles

Cluttered and dynamically changing environments and incomplete sensor information can cause service robots to collide with the environment. While the perception of interaction forces allows robots to react to collisions and avoid severe damage, they are still undesirable events robots should try to avoid. However, if a collision still occurs, one can try to maximize the benefit from this event. Chapter 4 demonstrates how robots can regard collision events as training examples to learn improved environment models, rather than just passively reacting to them. The approach targets mobile robots with planar laser rangefinders, which are popular for indoor robot localization because of their high accuracy but can only perceive one slice of the environment. Depending on the mounting position, they cannot perceive the full extent of some obstacles, including chairs and tables, where often only the legs are visible. Our approach is based on a convolutional neural network and predicts the obstacle footprints from 2D occupancy maps derived from laser rangefinders. It operates on local features and can thus generalize to unseen environments and repeating structures. In this context, we present a novel dataset with collision data from simulated environments. We further combine the simulated examples with new, real-world data to improve the navigation performance over time. The approach allows robots to learn from collisions with the environment in a self-supervised fashion, preventing future collisions and allowing them to adapt to the environment.

### Learning From Human Force Feedback

Mobile robots operating in shared environments must consider human safety and comfort. Defining and devising socially appropriate robot behavior depends on many factors, and manually adjusting the behavior for every robot and site of operation is not cost-effective. At the same time, robots in public spaces encounter people with very little experience with robots. In Chapter 5, we propose to learn socially compliant navigation behavior through physical interaction. Here, people can correct the behavior of a navigating mobile robot by physically pushing it along a trajectory they would prefer. The robot interprets the corrected trajectories as demonstrations and employs inverse reinforcement learning (IRL) to recover the underlying reward function representing the desired behavior. Many previous approaches used force feedback for teaching robot manipulators, but this is the first work investigating physical interaction in the mobile robotics context. Since physical interaction does not require an external control device, everyone in reach of the robot can communicate with it. A user study suggests that the gesture of pushing the robot along is intuitive and easy for non-expert users. Our approach allows for intuitive communication with mobile robots, allowing them to continuously adapt to the individual preferences of the people they interact with.

The approaches in this thesis provide service robots with extended perception capabilities, allowing them to attend to people's individual needs and preferences. Furthermore, this thesis presents new ways for service robots to continuously improve their navigation behavior during autonomous operation, enabling them to adapt to the environment and the people they interact with.

## 1.3 Publications

Parts of this thesis have been published in international peer-reviewed journals and conferences. The publications are listed in chronological order below.

- A. Vasquez, M. Kollmitz, A. Eitel, and W. Burgard. Deep detection of people and their mobility aids for a hospital robot. In *IEEE European Conference on Mobile Robots (ECMR)*, pages 1–7, 2017. `https://doi.org/10.1109/ECMR.2017.8098665`, ©2017 IEEE.

- M. Kollmitz, D. Büscher, T. Schubert, and W. Burgard. Whole-body sensory concept for compliant mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5429–5435, 2018. `https://doi.org/10.1109/ICRA.2018.8460510`, ©2018 IEEE.

- M. Kollmitz, A. Vasquez, A. Eitel, and W. Burgard. Deep 3D perception of people and their mobility aids. *Robotics and Autonomous Systems (RAS)*, 114:29–40, 2019. `https://doi.org/10.1016/j.robot.2019.01.011`, ©2019 Elsevier.

- M. Kollmitz, D. Büscher, and W. Burgard. Predicting obstacle footprints from 2D occupancy maps by learning from physical interactions. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 10256–10262, 2020. `https://doi.org/10.1109/ICRA40945.2020.9197474`, ©2020 IEEE.

- M. Kollmitz, T. Koller, J. Boedecker, and W. Burgard. Learning human-aware robot navigation from physical interaction via inverse reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11025–11031, 2020. `https://doi.org/10.1109/IROS45743.2020.9340865`, ©2020 IEEE.

## 1.4 Collaborations

Parts of the work presented in this thesis resulted from joint work with other researchers. As the supervisor of this thesis, Wolfram Burgard contributed ideas and suggestions to all of its parts. The remaining collaborations are outlined in the following.

- Chapter 2 builds upon the Master's thesis of Andres Vasquez [146], which was supervised by the author of this thesis together with Andreas Eitel, resulting in a conference publication [147]. Andres Vasquez primarily worked on the initial detection method presented in [147] and the Mobility Aids dataset. This thesis focuses on the revised detection method presented in the successive journal publication [74]. The author of this work is the main contributor to the revised detection method and the probabilistic people tracking module. Andreas Eitel provided counseling and advice for both detection methods, helped with the experiment design, and placed the work in the research context. Large parts of Chapter 2 have been previously published in [74], and minor parts follow [147].

- Tobias Schubert and Daniel Büscher contributed ideas and advice for the work presented in Chapter 3, resulting in a conference publication [73]. Daniel Büscher further helped with the model-based Kalman Filter design, where he and the author of this thesis had equal contribution to the derivation of the mathematical formulation. Tobias Schubert contributed to the sensory concept design and derived the calculation of the force impact point in closed form. Large parts of Chapter 3 have been previously published in [73].

- Chapter 4 is the result of joint work with Daniel Büscher, resulting in a conference publication [75]. Daniel Büscher provided consultation and ideas and helped with the design of the approach. In particular, he inspired the idea to train a binary classifier on occupancy map patches. The Bachelor's thesis of Mikel Cortes [25], supervised by the author of this thesis, first addressed the research problem. However, the thesis at hand does not draw from this initial work. Large parts of Chapter 4 have been previously published in [75].

- Torsten Koller contributed ideas and general consultation to the work presented in Chapter 5, resulting in a conference publication [76]. He further assisted with developing and implementing the inverse reinforcement learning method, contributed to the reward function design, and helped with the design and execution of the human-robot experiments. Joschka Boedecker provided suggestions and consultation. Large parts of Chapter 5 have been previously published in [76].

# Chapter 2

# Perception of People With Mobility Aids

**Robots operating in populated environments, such as hospitals, office environments, or airports, encounter a large variety of people. Some of them have an advanced need for cautious interaction because of their advanced age or motion impairments. To provide appropriate assistance and support, robot helpers require the ability to recognize people and their potential requirements. In this chapter, we present a people detection framework that distinguishes people according to the mobility aids they use. Our framework uses a deep convolutional neural network for detecting people in image data. For interactive mobile robots, it is essential to perceive people in the 3D world reference frame in addition to locating them in image data. To this end, we add a 3D centroid regression output to the network to predict the Cartesian positions of people. Our probabilistic class, position, and velocity tracker accounts for false detections and occlusions. We evaluate our approach on a dedicated dataset recorded with our mobile robot containing five classes: pedestrian, person in a wheelchair, pedestrian pushing a person in a wheelchair, person using crutches, and person using a walking frame. Our evaluation confirms that our approach can reliably detect people and their mobility aids and predict their 3D centroids. In a person guidance scenario with our mobile robot, we showcase how the perception of mobility aids can improve the level of assistance that service robots can provide.**

## 2.1 Introduction

Robot helper systems aim to assist people in everyday tasks. In health care applications, robots can help nurses lifting patients into and out of their hospital beds [24] and support the treatment of elderly people with dementia [149]. Mobile robotic assistants can further be employed for rehabilitation [34], carry out delivery tasks in hospitals [10, 21, 36, 126, 137], accompany and assist healthcare professionals [139], and support elderly people [48, 113].

The desired assistance and guidance offered by robot helper systems is often subjective and depends on many factors, like the person's physical condition. People with walking aids tend to walk slower than people without motion impairments; thus, robots should adapt their velocity when guiding people with walking aids [100]. Besides, knowledge of motion impairments can help robots to better anticipate and predict the motion of pedestrians [38], which is crucial for socially compliant navigation [72, 78, 136]. The use of walking aids further entails certain limitations, such as the inability to use stairs and special requirements for door and doorway clearances, elevators, or bathroom facilities [133]. This chapter presents a novel people detection framework that allows mobile robots to reason about those special requirements by perceiving people and their mobility aids. Substantial parts of the ideas, results, figures, tables, and text presented in this chapter have been previously published in [74], ©2019 Elsevier, `https://doi.org/10.1016/j.robot.2019.01.011`. Section 1.4 outlines the author's contribution to this work.

We address the main challenges for perceiving people according to their mobility aids with deep neural network-based detection and probabilistic position and class estimation. Our system estimates the mobility aids classes, 3D positions, velocities, and the tracked motion paths of people. The framework comes in two variants: depth and RGB. The depth only variant targets high privacy demands, while the RGB only framework provides improved detection performance for non-critical applications. Our perception system has three main components: deep learning-based 3D object detection, 3D position and velocity tracking, and probabilistic class estimation. The object detection module takes depth or RGB images as input and outputs 2D image bounding boxes and the 3D centroid for each box, as depicted in Figure 2.1. The probabilistic position and class estimation modules resolve occlusions and provide a probability distribution over the mobility aids classes for each detection, taking the previous observations into account. Our work presents the first perception system that distinguishes people in image data according to their mobility aids, thus enabling robots to provide better assistance to physically impaired or elderly people. Our approach is designed for mobile robots and relies on odometry information for tracking.

The work presented in this chapter builds upon the Faster R-CNN [119] framework, a popular object detection method for 2D bounding box prediction. We extended the original Faster R-CNN with an additional network output that estimates the 3D centroid depth of each bounding box. Related works rely on depth data in addition to RGB images to predict the 3D centroids of the estimated bounding boxes [102, 131, 147, 154]. However, those approaches are limited by the range of the depth sensor. People who are visible in the RGB but not the depth image due to the limited depth range cannot be detected. Our

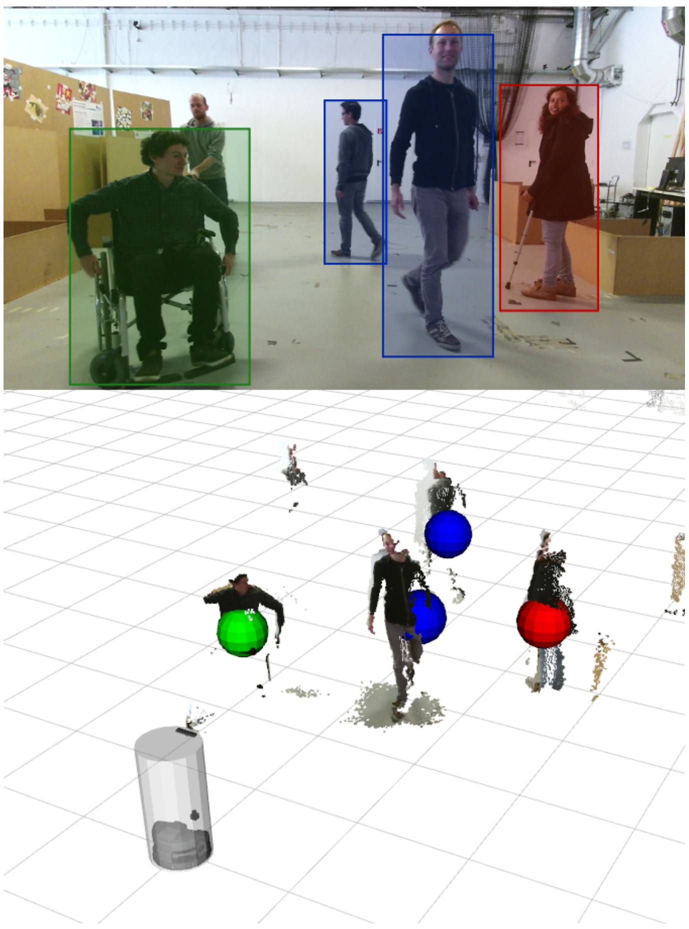

**Figure 2.1:** Our perception framework detects people in 2D images and categorizes them according to the mobility aids they use, as shown on the top. It further estimates each person's 3D centroid from the 2D image, visualized at the bottom as colored spheres. The displayed point cloud only serves as a reference and is not used by our approach.

system overcomes these limitations and is flexible with respect to the employed sensor modality: RGB or depth images. Our detection framework further incorporates a probabilistic position, velocity, and class estimation module to account for occlusions or false detections. It combines an extended Kalman filter to track each person's position and velocity and a hidden Markov model to estimate the mobility aids class of each filter.

This chapter further outlines our Mobility Aids dataset with over $17{,}000$ annotated RGB-D images at $960 \times 540$ pixel resolution. The dataset provides the annotated 2D image bounding boxes, categorized according to five mobility aids classes: pedestrian, person in a wheelchair, pedestrian pushing a person in a wheelchair, person using crutches, and person using a walking frame. The dataset further provides 3D centroid depth labels for each annotated 2D bounding box and robot odometry. We collected the dataset with a Kinect 2 camera mounted on a mobile platform in the Faculty of Engineering facilities of the University of Freiburg and a hospital in Frankfurt. The dataset and a ROS package of our perception system are publicly available at `http://mobility-aids.informatik.uni-freiburg.de`. The webpage also contains a video of the perception results obtained using our approach.

We evaluate our detection framework on our Mobility Aids dataset to demonstrate the object detection and the 3D centroid regression performance. We further analyze the performance of the individual framework components and their respective contributions to the overall system. Finally, we test our approach on our mobile robot in a guidance scenario to show that it can be successfully deployed in the physical world and that it enables robots to provide appropriate assistance to people according to their physical impairments.

## 2.2 Related Work

In the last two decades, deep learning methods have surpassed classical, feature-based machine learning techniques in various domains, like automatic speech recognition and image classification [2]. Large-scale annotated datasets, like ImageNet [29], have enabled and fueled the success of deep learning algorithms for image classification tasks. In 2012, Krizhevsky *et al.* [77] achieved a substantial decrease in image classification error rates on the ImageNet LSVRC classification contest [121] by using a deep convolutional neural network (CNN). Girshick *et al.* [45] built upon the great success of CNNs for image classification by leveraging deep learning for object detection, which aims at producing object bounding boxes in an image in addition to predicting object class labels. In their R-CNN framework [45], the authors used a CNN to extract rich features from promising image patches they obtained from selective search [144] and classified them with linear SVMs. The successor Fast R-CNN [44] used fully connected network layers instead of SVMs and accelerated the detection by passing the full image through the CNN and pooling the regions of interest from the resulting feature map.

Our initial mobility aids detection work [147] used the Fast R-CNN framework. Instead of using selective search for proposal generation, we generated the proposals by clustering point clouds from depth images to increase detection speed. The work presented in this chapter builds upon the Faster R-CNN framework [119], which improves both detection performance and speed compared to Fast R-CNN by generating proposals with a region proposal network. Other object detection frameworks like R-FCN [26], YOLO [118], YOLOv2 [117], or the Single Shot Detector (SSD) [88] exhibit even faster inference speeds than Faster R-CNN but tend not to reach its detection accuracy [58]. However, it is likely possible to adapt and use those frameworks in a similar way for the mobility aids detection task for scenarios that demand higher detection rates.

Pedestrian detection is of paramount importance for human safety in the autonomous driving domain. Dollár *et al.* [31] and Brunetti *et al.* [16] give an overview of past pedestrian detection approaches. Ošep *et al.* [108] argued for the importance of 3D pose information for pedestrian detection in traffic scenes and proposed a 2D-3D Kalman filter approach for combined tracking in image and world space. Our method predicts the distances of people from the camera from image data to enable 3D detection and tracking in the world reference frame. Li *et al.* [83] proposed a pedestrian detection method based on Fast R-CNN, and Zhang *et al.* [155] combined deep region proposal generation and feature extraction with boosted forests for pedestrian classification. Other approaches use depth and vision data from stereo cameras to detect pedestrians in traffic scenes [8, 35]. While we focus on people detection for indoor mobile robots, perceiving mobility aids would certainly be advantageous for autonomous cars in urban traffic since mobility aids influence people's motion and speed.

In the indoor robotics context, researchers have used various sensors for the people perception task. Darrell *et al.* [27] and Muñoz-Salinas *et al.* [103] detected people in stereo camera images. In 2007, Arras *et al.* [4] introduced a people detector that classified leg shapes from planar laser rangefinder data. While planar laser rangefinders typically have a wide field of view, the data is very sparse, and distinguishing legs from other round shapes like poles is challenging in cluttered environments. Related works have therefore combined laser rangefinder-based people detection with other sensor modalities. Dondrup *et al.* [32] proposed a people detection and tracking system fusing laser rangefinder-based people detections and an RGB-D upper body detector in a probabilistic people tracking framework. Similarly, Linder *et al.* [86] integrated multiple people detection modules processing laser rangefinder, depth, and RGB data to detect and track people in crowded public spaces, such as airports.

Similar to our initial mobility aids detection work [147], Zhang *et al.* [154] generated object proposals by clustering point clouds generated from depth images. They processed the object proposals by multiple depth and RGB detectors using manually-designed features. In contrast, we operated on depth images only and classified the proposals with a Fast R-CNN framework. Munaro and Menegatti [102] also generated object proposals by segmenting height maps from depth images and processed the corresponding RGB image regions with a HOG-based classifier. While segmenting point clouds from depth data can significantly speed up the detection process [147], it is not possible to detect people that are visible in the RGB but not in the depth images due to the limited range of the sensor.

Jafari *et al.* [61] addressed the limited depth range of RGB-D sensors and the fact that people close to the robot are only partially visible in the camera image by processing depth images for close range and RGB images for far range people detection. Spinello and Arras [131] also combined an RGB and a depth detector to address the limitations of both modalities. In later work, the authors used a Kalman filter-based multi hypothesis tracking module [91] to process the detections further. Mees *et al.* [98] trained a deep neural network for weighting the predictions of an RGB and a depth-based people detector, depending on environmental conditions such as lighting and the limited depth range. Our work is flexible regarding the input modality since it can process either RGB or depth images and predicts the 3D centroids of people from image data directly. After our mobility aids papers [74, 147] were published, Linder *et al.* [87] introduced a detection framework that, similarly to our later work [75], adapted a deep neural network to regress the 3D centroids of people. In contrast to our work, they fused RGB and depth sensor data inside the network. It would be interesting to extend our approach in future work following a similar paradigm for applications where both depth and RGB data are available.

In contrast to the approaches above, we address the multi-class people detection problem of distinguishing people according to their mobility aids. Other multi-class people detection methods focus on recognizing a person's gender [85] or on person attribute recognition [15, 124, 135]. Oliveira *et al.* [105] segmented human body parts from RGB images, but they did not distinguish different person categories. Beyer *et al.* [12] proposed a wheelchair detector, and Weinrich *et al.* [151] distinguished people without walking aids from people in a wheelchair, using crutches, or with a walker. Both methods operated on planar laser rangefinder data. To the best of our knowledge, our work is the first that addresses the detection and distinction of people according to mobility aids categories from image data. Furthermore, we made our Mobility Aids dataset publicly available. While various datasets exist for detecting people in indoor environments or urban traffic scenes [30, 98, 99, 102, 156], ours is the first to include mobility aids category labels in addition to bounding box annotations and 3D centroid labels.

## 2.3 People Perception Framework

Our perception framework processes 2D images, either RGB or depth, and estimates the mobility aids category and the position and velocity of people in a world reference frame. It consists of three modules, as depicted in Figure 2.2. Section 2.3.1 introduces the detection stage of our framework. The probabilistic position, class, and velocity estimation stages are described in Section 2.3.2.

### 2.3.1 People Detection With Faster R-CNN

Our object detection module builds upon Faster R-CNN [119], an object detection network that runs in an end-to-end manner from image input to object detection output. This work uses the open-source Detectron implementation [46] of Faster R-CNN. Faster R-CNN uses a region proposal network (RPN) to generate regions of interest (RoIs) in the image, followed by a region-based convolutional neural network to classify the RoIs and regress their bounding boxes. Both network modules share their convolutional layers, resulting in a compact and fast end-to-end system. Furthermore, the resulting network

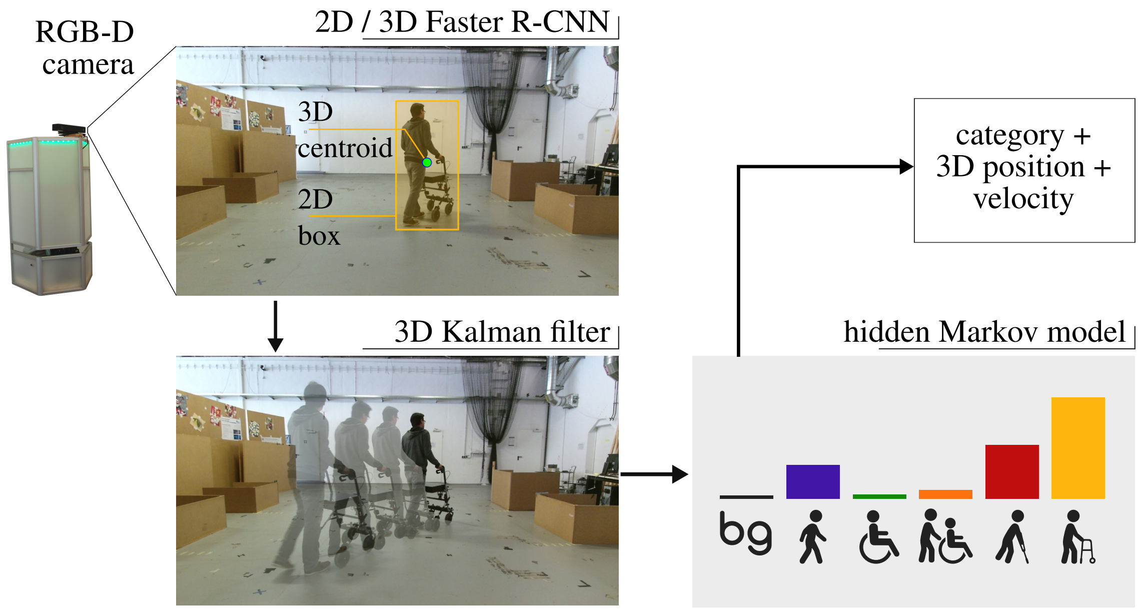

**Figure 2.2:** Our perception system operates on 2D images (RGB or depth). We input the images into a Faster R-CNN network that regresses 2D bounding boxes and 3D centroids of people. We track the resulting detections with a Kalman filter for 3D position and velocity estimation. Further, we employ a hidden Markov model for filtering the mobility aids category predictions over time.

shares convolutions across region proposals, which means that the convolutional feature maps for the input image are computed only once and then used to evaluate each proposal with a RoI pooling procedure.

Our mobility aids detection framework extends the original Faster R-CNN by an additional output that estimates the distance between the 2D image bounding box centroid and the camera, which we in the following refer to as centroid depth. We further adapt the loss function for training the network to incorporate the additional network output. The proposal generation, classification, and 2D bounding box regression functionalities are identical to the original Faster R-CNN. This section describes the resulting network and specifies the training parameters.

The input of our mobility aids detector is a three-channel image, either RGB or color-encoded depth, in the following referred to as DepthJet [33]. In its original form, Faster R-CNN outputs 2D bounding box coordinates and the detection scores for each object category. For robotics, however, knowledge about the locations of people in the image plane is often not sufficient. They need to perceive where people are in the world coordinate frame to interact with them. To this end, we added a new output to the network – the centroid depth regression, as highlighted in red in Figure 2.3. This output estimates the Cartesian distance ${}^{\mathrm{c}}z_{\mathrm{p}}$ between the camera and the person's 3D centroid along the

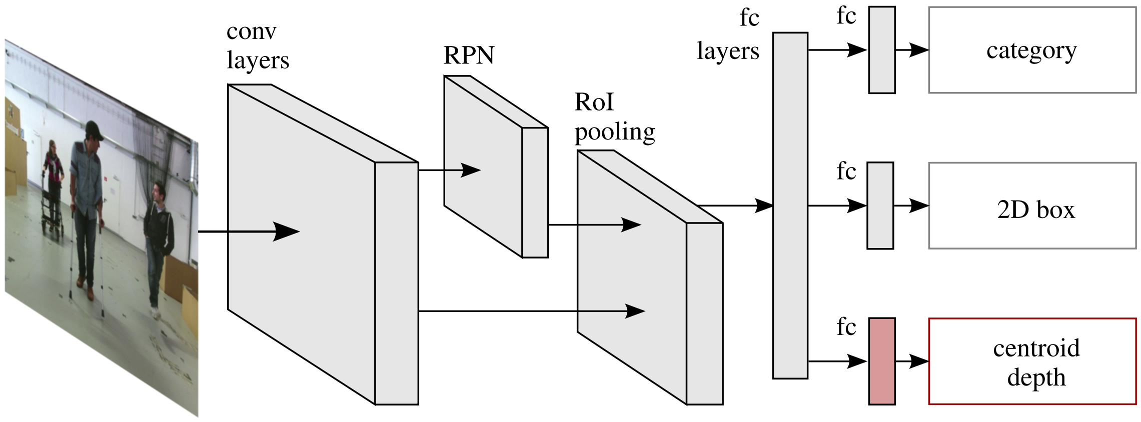

**Figure 2.3:** Faster R-CNN architecture with Region Proposal Network (RPN). For each proposal, we predict the mobility aids category, the 2D image bounding box, and the centroid depth of the bounding box.

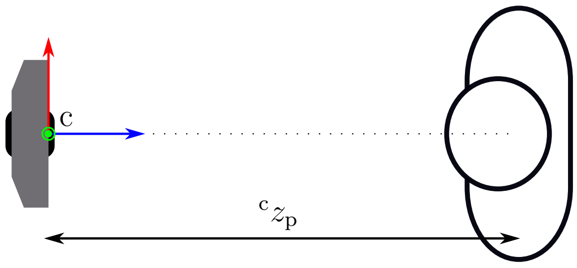

**Figure 2.4:** The centroid depth ${}^{\mathrm{c}}z_{\mathrm{p}}$ is the $z$-coordinate of the person $\mathrm{p}$ in the camera reference frame $\mathrm{c}$.

$z$-coordinate of the camera pointing into the image, as depicted in Figure 2.4. The centroid depth regression layer is arranged after the last fully connected layer of the network and outputs a single real-valued number for each proposal, using linear neuron activations. The 3D centroid of a person in the camera reference frame can be computed from the estimated centroid depth ${}^{\mathrm{c}}z_{\mathrm{p}}$, following the pinhole camera model and using the intrinsic camera calibration, by

$$ {}^{\mathrm{c}}x_{\mathrm{p}} = \left( \frac{{}^{\mathrm{i}}x_{\mathrm{p}} - o_x}{f_x} \right) {}^{\mathrm{c}}z_{\mathrm{p}} \tag{2.1} $$

$$ {}^{\mathrm{c}}y_{\mathrm{p}} = \left( \frac{{}^{\mathrm{i}}y_{\mathrm{p}} - o_y}{f_y} \right) {}^{\mathrm{c}}z_{\mathrm{p}}, \tag{2.2} $$

with the optical center $(o_x, o_y)$ and the focal lengths $(f_x, f_y)$, both in pixels. The centroid of the person in the image plane $\left({}^{\mathrm{i}}x_{\mathrm{p}}, {}^{\mathrm{i}}y_{\mathrm{p}}\right)$ is the center of the regressed 2D bounding box. The left superscript $\mathrm{i}$ denotes the image reference frame.

During training, the network jointly optimizes a multi-task loss containing the object classification and bounding box regression loss, the region proposal loss, and the centroid depth regression loss. Similar to the bounding box regression loss in the original Faster R-CNN, we use the robust loss function $\mathcal{L}_{1,\mathrm{s}}$ (smooth L1) [44] for the centroid depth regression loss

$$\begin{aligned} \mathcal{L}_z &= w_z \sum_{k=1}^{N} \hat{p}_z^{(k)} \mathcal{L}_{1,\mathrm{s}} \left( {}^{\mathrm{c}}z_{\mathrm{p}}^{(k)} - {}^{\mathrm{c}}\hat{z}_{\mathrm{p}}^{(k)} \right), \qquad (2.3) \\ \text{with } \mathcal{L}_{1,\mathrm{s}}(a) &= \begin{cases} 0.5a^2 & \text{if } |a| < 1 \\ |a| - 0.5 & \text{otherwise.} \end{cases} \end{aligned}$$

The number of training examples is $N$ and the ground truth centroid depth of the $k^{\text{th}}$ training example is denoted by ${}^{\mathrm{c}}\hat{z}_{\mathrm{p}}^{(k)}$. It is only available for proposals with non-background object labels, and we use the label $\hat{p}_z^{(k)} \in \{0, 1\}$ to deactivate the centroid depth regression loss for proposals without centroid depth labels. We obtained the centroid depth labels from the depth image of our RGB-D camera, which has a maximum range of $\approx 8\,\mathrm{m}$. For proposals outside that range, we deactivate the centroid depth regression loss. Please refer to Section 2.4.1 for the centroid depth labeling procedure. The weight factor $w_z$ balances the depth regression and image detection performances. We found that a value of $10$, similar to the 2D bounding box regression weights, produces satisfactory depth regression results without significantly diminishing the image detection performance.

We trained our people detection framework with stochastic gradient descent with a momentum of $0.9$ and a weight decay of $0.0001$. The minibatch size was $2$ images per GPU and we trained on $2$ GPUs simultaneously. The total number of RoIs per training minibatch was $1024$. We used the $2000$ best-scored RPN proposals for training and a learning rate of $0.0025$, reduced with a factor of $0.1$ after $30{,}000$ and again after $40{,}000$ iterations. We adopted the warm-up scheme presented in [47] and gradually ramped up the learning rate for the first $500$ iterations with a linearly increasing factor, starting from $1/3$ and increasing to $1$. After the warm-up phase, the learning rate returned to the original learning rate schedule.

We trained three different network architectures: the ResNet-50 [54] architecture as a strong but complex backbone and VGG-CNN-M [22] and GoogLeNet-xxs [98] as faster, more lightweight networks. All networks were trained for $60{,}000$ iterations and pretrained on ImageNet.

At test time, we pass the input image through the Faster R-CNN network and apply class-wise non-maximum suppression to the output bounding boxes. We choose the detections above a class threshold as positive detections and pass them to the probabilistic people tracker presented in the following. The class thresholds depend on the network, input modality, and mobility aids class, and we choose them as the detection confidences where the class precisions correspond to the class average precisions.

### 2.3.2 Probabilistic Position, Velocity, and Class Estimation

The detection stage provides a set of coordinates for each detected person of the form $({}^{\mathrm{i}}x_{\mathrm{p}}, {}^{\mathrm{i}}y_{\mathrm{p}}, {}^{\mathrm{c}}z_{\mathrm{p}}, c')$, where $c'$ is the predicted mobility aids category. Our probabilistic position, velocity, and class estimator computes the belief $\mathrm{Bel}(\mathbf{x})$ of the person state $\mathbf{x} = ({}^{\mathrm{w}}x_{\mathrm{p}}\ {}^{\mathrm{w}}y_{\mathrm{p}}\ {}^{\mathrm{w}}z_{\mathrm{p}}\ {}^{\mathrm{w}}\dot{x}_{\mathrm{p}}\ {}^{\mathrm{w}}\dot{y}_{\mathrm{p}}\ c)^T$, where $({}^{\mathrm{w}}x_{\mathrm{p}}, {}^{\mathrm{w}}y_{\mathrm{p}}, {}^{\mathrm{w}}z_{\mathrm{p}})$ denote the position of the person and $({}^{\mathrm{w}}\dot{x}_{\mathrm{p}}, {}^{\mathrm{w}}\dot{y}_{\mathrm{p}})$ her or his velocity on the ground plane, both in a fixed world coordinate frame with the $z$-axis aligned with the negative gravity vector. The mobility aids class $c$ includes pedestrian, person in a wheelchair, pedestrian pushing a person in a wheelchair, person using crutches, person using a walking frame, and background. The estimator combines two modules: An extended Kalman filter (EKF)-based multi-tracking module and a hidden Markov model (HMM)-based class estimation module. The estimator manages one track for each perceived person that performs both EKF and HMM filtering.

The EKF state $\mathbf{x}_{\mathrm{K}} = ({}^{\mathrm{w}}x_{\mathrm{p}}\ {}^{\mathrm{w}}y_{\mathrm{p}}\ {}^{\mathrm{w}}z_{\mathrm{p}}\ {}^{\mathrm{w}}\dot{x}_{\mathrm{p}}\ {}^{\mathrm{w}}\dot{y}_{\mathrm{p}})^T$ is five-dimensional and uses a constant velocity motion model with the state transition matrix

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{2.4}$$

where $\Delta t$ is the filter time interval. For the process noise, we adopt the piecewise white noise model described by Labbe [79] and assume the system dynamics $f$ are disturbed by constant piecewise noise $\mathbf{w}$:

$$f(\mathbf{x}_{\mathrm{K}}) = \mathbf{F}\mathbf{x}_{\mathrm{K}} + \mathbf{w}\mathbf{\Gamma} \tag{2.5}$$

We model $\mathbf{w}$ by accelerations $a_x$ and $a_y$ and a height error $\sigma_z$, which are independent:

$$\mathbf{w} = \begin{pmatrix} a_x & 0 & 0 \\ 0 & a_y & 0 \\ 0 & 0 & \sigma_z \end{pmatrix} \tag{2.6}$$

The noise gain $\mathbf{\Gamma}$ describes how the noise $\mathbf{w}$ propagates into the state space and is modeled by

$$\mathbf{\Gamma} = \begin{pmatrix} \Delta t^2/2 & 0 & 0 \\ 0 & \Delta t^2/2 & 0 \\ 0 & 0 & 1 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \end{pmatrix}. \tag{2.7}$$

The process noise covariance $\mathbf{Q}$ is then calculated by

$$\mathbf{Q} = \mathbf{\Gamma} \mathbf{w} \mathbf{w}^T \mathbf{\Gamma}^T. \tag{2.8}$$

For the correction step, we integrate the observations $\mathbf{z} = ({}^{\mathrm{i}}x_{\mathrm{p}} \; {}^{\mathrm{i}}y_{\mathrm{p}} \; {}^{\mathrm{c}}z_{\mathrm{p}})^T$ from the detection stage. We first need to design the measurement function

$$h(\mathbf{x}_{\mathrm{K}}) = \mathbf{z}. \tag{2.9}$$

It maps the state to the measurement and, since the state is given in a fixed world coordinate frame, requires to transform the state first to the camera and secondly to the image frame. The transformation between the camera and the fixed world coordinate system is determined by

$${}^{\mathrm{c}}\mathbf{T}_{\mathrm{w}} = ({}^{\mathrm{w}}\mathbf{T}_{\mathrm{r}} \cdot {}^{\mathrm{r}}\mathbf{T}_{\mathrm{c}})^{-1}, \tag{2.10}$$

where ${}^{\mathrm{w}}\mathbf{T}_{\mathrm{r}}$ denotes the homogeneous transformation matrix between the world coordinate frame and the robot base, which we obtain from odometry, and ${}^{\mathrm{r}}\mathbf{T}_{\mathrm{c}}$ is the known homogeneous transformation matrix between the robot base and the camera. We transform the 3D position from the world reference frame to the camera frame by

$$\begin{pmatrix} {}^{\mathrm{c}}x_{\mathrm{p}} \\ {}^{\mathrm{c}}y_{\mathrm{p}} \\ {}^{\mathrm{c}}z_{\mathrm{p}} \\ 1 \end{pmatrix} = {}^{\mathrm{c}}\mathbf{T}_{\mathrm{w}} \begin{pmatrix} {}^{\mathrm{w}}x_{\mathrm{p}} \\ {}^{\mathrm{w}}y_{\mathrm{p}} \\ {}^{\mathrm{w}}z_{\mathrm{p}} \\ 1 \end{pmatrix}. \tag{2.11}$$

Afterward, we project ${}^{\mathrm{c}}x_{\mathrm{p}}$ and ${}^{\mathrm{c}}y_{\mathrm{p}}$ into the image plane, using the intrinsic camera calibration and following Eq. (2.1) and Eq. (2.2), by

$${}^{\mathrm{i}}x_{\mathrm{p}} = \frac{{}^{\mathrm{c}}x_{\mathrm{p}}}{{}^{\mathrm{c}}z_{\mathrm{p}}} f_x + o_x \tag{2.12}$$

$${}^{\mathrm{i}}y_{\mathrm{p}} = \frac{{}^{\mathrm{c}}y_{\mathrm{p}}}{{}^{\mathrm{c}}z_{\mathrm{p}}} f_y + c_y. \tag{2.13}$$

The measurement function is non-linear with respect to the state $\mathbf{x}_{\mathrm{K}}$, which means that we need to employ an extended Kalman filter and use the Jacobian $\mathbf{H}$ of $h$ in the correction step.

We obtain the observation noise $\mathbf{R}$ experimentally by analyzing the Faster R-CNN detections $\mathbf{z}$ and the corresponding ground truth image bounding box and centroid depth labels $\hat{\mathbf{z}}$ in our test dataset 1, which is used for the evaluation of the detection part of the framework and to tune the tracking module, see Section 2.4.1. To this end, we compile a data matrix

$$\mathbf{A} = [\mathbf{z}^{(1)} - \hat{\mathbf{z}}^{(1)}, \dots, \mathbf{z}^{(N)} - \hat{\mathbf{z}}^{(N)}], \tag{2.14}$$

where $N$ is the number of training examples and the superscript specifies the index of the detection-label pair, and specify the observation noise as the covariance of $\mathbf{A}$.

For the data association between tracks and observations, we consider the pairwise Mahalanobis distances

$$\begin{aligned} \delta_{ij}^2(t) &= \mathbf{v}_{ij}^T(t)\mathbf{S}_{ij}^{-1}(t)\mathbf{v}_{ij}(t), \\ \text{with } \mathbf{v}_{ij}(t) &= \mathbf{z}_i(t) - h(\bar{\mathbf{x}}_{\mathrm{K},j}(t)) \\ \text{and } \mathbf{S}_{ij}(t) &= \mathbf{H}_j(t)\mathbf{P}_j(t)\mathbf{H}_j^T(t) + \mathbf{R}, \end{aligned} \tag{2.15}$$

where $\bar{\mathbf{x}}_{\mathrm{K}}(t)$ and $\mathbf{P}(t)$ are the predicted state mean and covariance at time $t$, and the observation and track indices are $i$ and $j$. We only pair observations if the Mahalanobis distance is below a fixed threshold $\delta$. Furthermore, we enforce a maximum Euclidean distance $d$ between detections and position hypotheses. If for one observation both thresholds are satisfied for multiple tracks, we pair it to the one with the highest probability density value

$$p_{ij} = \frac{1}{\sqrt{(2\pi)^3 \mathrm{det}\mathbf{S}_{ij}(t)}} \exp\left(-\frac{1}{2}\mathbf{v}_{ij}^T(t)\mathbf{S}_{ij}^{-1}(t)\mathbf{v}_{ij}(t)\right). \tag{2.16}$$

If multiple observations are paired with one track at time $t$, we update it successively with all paired observations. If an observation was not paired, a new track is initialized. Finally, if there is no observation for a track, we perform a prediction without observation update.

Each track also has one HMM associated with it for estimating the class $c$ of the tracked person, according to

$$p(c_t \mid c'_{1:t}) = \eta p(c'_t \mid c_t) \sum_{c_{t-1}} p(c_t \mid c_{t-1}) p(c_{t-1} \mid c'_{1:t-1}). \tag{2.17}$$

It models the probability that a tracked person belongs to class $c_t$ given the past observations $c'_{1:t}$. The factor $\eta$ normalizes the distribution. The measurement model $p(c'_t \mid c_t)$ connects the hidden with the observed variable for time step $t$. The HMM further assumes that the class $c_t$ can randomly change from one time step to the next, represented by the transition model $p(c_t \mid c_{t-1})$. In a hospital, a possible transition could be person with crutches $\rightarrow$ pedestrian $\rightarrow$ person in a wheelchair, for a patient who has just finished physiotherapy and hands over the crutches to return to her or his wheelchair. We initialize the HMM with a uniform prior $p(c_1)$ over all categories.

The softmax output of deep neural network classifiers like Faster R-CNN could be interpreted as $p(c_t \mid c'_t)$. However, training with one-hot encoded labels results in very peaky distributions and over-confident estimates. Therefore, we do not employ the network scores directly for filtering. Instead, we analyzed the detections and labels for the test dataset 1 to statistically determine the underlying probability distributions. The num-

**Table 2.1:** Default threshold values of the estimation module.

| threshold | value |
|---|---|
| max. Euclidean distance $d$ | $1.0\,\mathrm{m}$ |
| max. Mahalanobis distance $\delta$ | $7.815$ |
| max. position uncertainty $\sigma_{xy}$ | $4.0\,\mathrm{m}$ |

ber of detections for one class given a certain label determines the measurement model $p(c'_t \mid c_t)$ of the HMM. It corresponds to the normalized confusion matrix of our classifier. The transition model $p(c_t \mid c_{t-1})$ is given by the number of transitions from one class to another, compared to the total number of transitions. Due to the limited amount of examples in the dataset, we might not observe all class transitions and confusions, even if they are possible. Therefore, we assign small probabilities to all unobserved but possible transitions and false detections, using a Dirichlet prior.

The EKF determines the data association for the HMM. If there is no observation $c_t$ for a track at time step $t$, we treat it as a background detection in the HMM. The position, velocity, and class estimator removes tracks with a standard deviation in position above a threshold $\sigma_{xy}$. Furthermore, tracks where the background class is dominant are deleted. Table 2.1 summarizes the threshold values we used during the experiments.

## 2.4 Experiments

We performed a set of experiments to evaluate our mobility aids detection framework and its components. This section introduces our Mobility Aids dataset in Section 2.4.1. We evaluate the performance of the Faster R-CNN detection module in Section 2.4.2 and assess our entire framework in Section 2.4.3. Section 2.4.4 presents a real-robot scenario where the robot uses our perception pipeline to give special assistance in a person guidance task.

### 2.4.1 Mobility Aids Dataset

We recorded RGB and depth images of people of the five mobility aids categories and robot odometry using our mobile Festo Robotino platform Canny, equipped with a Kinect 2 camera mounted $1\,\mathrm{m}$ above the ground and capturing images at $15$ frames per second. The robot was controlled by a notebook computer running ROS (Robot Operating System). Images were collected in the Faculty of Engineering facilities of the University of Freiburg and in a hospital in Frankfurt.

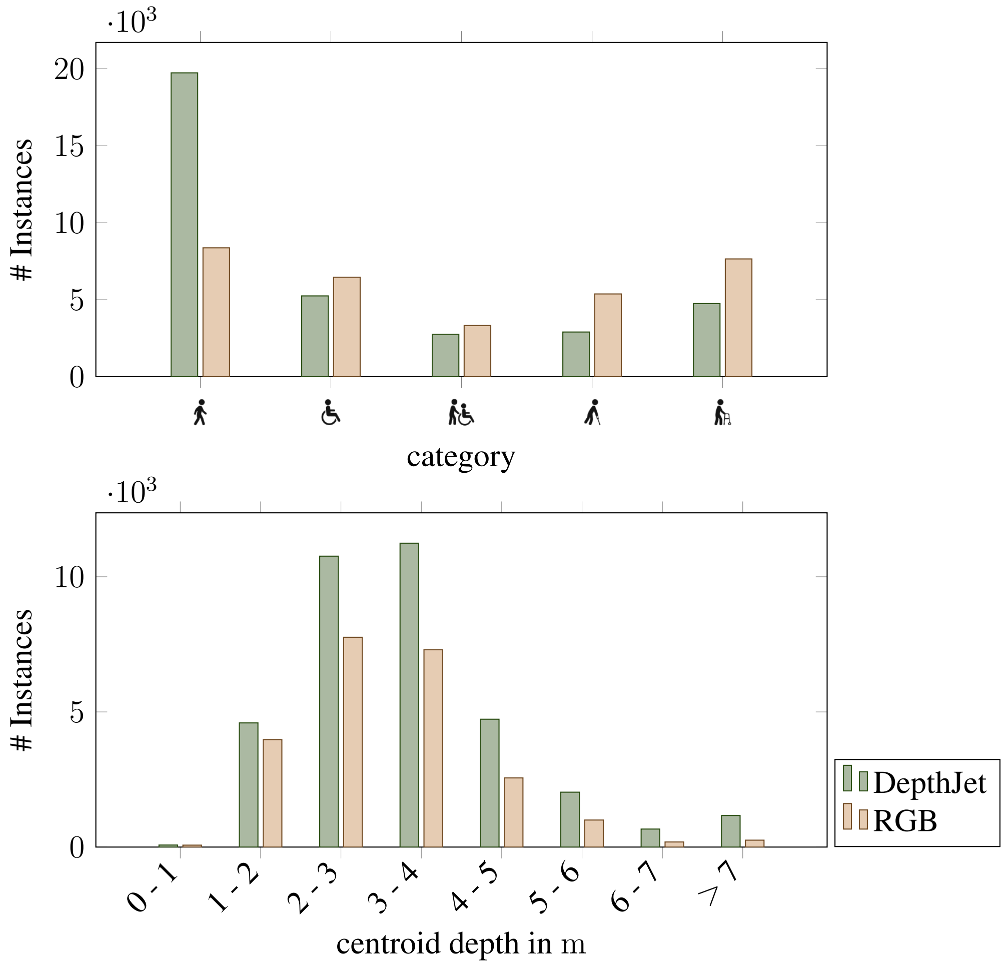

**Figure 2.5:** Number of instances per mobility aids category (top) and per centroid depth interval (bottom) in the Mobility Aids dataset.

We divided the dataset into subsets for training and testing. We used two test sets to evaluate the performance of the approach. Test set 1 contains 4317 frames, and we used it to assess the detection methods and tune the parameters of the tracking module (Section 2.3.2). We used test set 2 for testing our method in combination with our probabilistic position, velocity, and class estimation module. It contains a total of 1801 frames merged from four video sequences, and, in contrast to test set 1, the ground truth labels also consider occluded objects.

We manually annotated the RGB images and the depth images in DepthJet color encoding on 2D bounding box level. Afterward, we labeled the centroid depths of the 2D bounding boxes. The centroid depth labeling procedure is explained in the following section. Figure 2.5 summarizes the number of instances for each mobility aids category and different centroid depth intervals in the dataset for DepthJet and RGB. Note that some images inside the hospital were only recorded in DepthJet because of privacy concerns.

Furthermore, some people visible in the RGB images are not visible in the DepthJet images because of the limited depth range of the camera.

**Centroid Depth Labeling Procedure**

We only considered the DepthJet bounding boxes for the centroid depth labeling because they definitely include depth information describing the person. We converted each depth image to a point cloud and removed the ground plane. Then we extracted the part of the point cloud that belongs to the 2D bounding box. Finally, we clustered the extracted part and calculated the 3D centroid of each cluster as the mean of all points belonging to the cluster.

Now we need to decide which cluster best represents the person. For both test sets, an experimenter selected the most representative clusters by hand. For the training split, we used a simple heuristic to choose the best cluster from the following criteria:

- cluster size,
- distance between cluster center and bounding box center in the image,
- and depth of cluster centroid.

The cluster size and distance criteria ensure that the heuristic selects a dominant cluster. The cluster depth criterion favors clusters closer to the camera to avoid choosing parts of the background.

For all clusters in a bounding box, we calculated scores

$$\begin{aligned} s_{n,\mathrm{j}} &= \frac{n_\mathrm{j}}{\sum_\mathrm{j} n_\mathrm{j}} && (2.18)\\ s_{xy,\mathrm{j}} &= \frac{\sqrt{({}^\mathrm{i}x_\mathrm{j} - {}^\mathrm{i}x_\mathrm{p})^2 + ({}^\mathrm{i}y_\mathrm{j} - {}^\mathrm{i}y_\mathrm{p})^2}}{\sqrt{w_b^2 + h_b^2}} && (2.19)\\ s_{z,\mathrm{j}} &= 1 - \frac{{}^\mathrm{c}z_\mathrm{j}}{z_\mathrm{max}} && (2.20) \end{aligned}$$

representing each criterion. The number of points in each cluster j is $n_\mathrm{j}$, $({}^\mathrm{i}x_\mathrm{j}, {}^\mathrm{i}y_\mathrm{j})$ are the image coordinates of the cluster center, and $w_b$ and $h_b$ are the bounding box width and height. The maximum range of the Kinect is denoted by $z_\mathrm{max}$, and ${}^\mathrm{c}z_\mathrm{j}$ is the centroid depth of the cluster. All scores range between $0$ and $1$. We chose the ground truth centroid depth of the $k^\mathrm{th}$ training example as the centroid depth of the best cluster j*,

$$\begin{aligned} {}^\mathrm{c}z_\mathrm{p}^{(k)} &\leftarrow {}^\mathrm{c}z_{\mathrm{j}^*}^{(k)}, && (2.21)\\ \text{where } \mathrm{j}^{*(k)} &= \arg\max_\mathrm{j} \left( s_{n,\mathrm{j}}^{(k)} \cdot s_{xy,\mathrm{j}}^{(k)} \cdot s_{z,\mathrm{j}}^{(k)} \right). \end{aligned}$$

We evaluated our heuristic on $500$ manually labeled bounding boxes where an experimenter selected the most representative depth clusters. The heuristic selected the same cluster in $97.8\,\%$ of the cases. For the remaining $2.2\,\%$, the mean absolute depth error between the clusters was $1.78\,\mathrm{m}$. This means that our heuristic is very reliable, but outliers are possible. After labeling the 3D centroid depths for the DepthJet dataset, we transfered them to the RGB bounding boxes. For labels that are present in RGB but not in DepthJet, e.g., due to the limited depth range of the camera, we deactivated the depth label by setting $\hat{p}_z = 0$, see Eq. (2.3).

### 2.4.2 Object Detection Performance

This section evaluates and compares the object detection performance of the extended Faster R-CNN with the centroid depth regression output separately, without the tracking module. We compared our detection module to our initial mobility aids work [147], which was also designed and trained for detecting people according to their mobility aids. In [147], we used the predecessor of Faster R-CNN: the Fast R-CNN framework. Fast R-CNN relies on external object proposals for people detection, which we generated in [147] by clustering the 3D point clouds obtained from the depth images. We determined the centroid depths as the mean depths of all proposal cluster points.

Our initial work always required depth images for people detection and could not work on RGB images only because it used the depth images for the object proposal generation. Unfortunately, people who were only visible in the RGB images and not in the depth images because of the limited range of the depth sensor could not be detected. In contrast, our improved method uses Faster R-CNN, which generates object proposals from the input images and can work on either RGB or depth images. Instead of clustering point clouds for centroid depth prediction, we regress the 3D centroid depth in the network.

We compared our initial Fast R-CNN-based work [147] to our Faster-RCNN detection framework with different backbone architectures: ResNet-50 [54] (in the following denoted by R-50), VGG-CNN-M [22] (VGG-M) and GoogLeNet-xxs [98] (G-xxs), which we also used in [147]. We used the standard average precision (AP) metric for evaluation. We followed the Pascal VOC 2010 evaluation [37] to calculate the AP and paired each detection to the ground truth example with the highest 2D bounding box intersection over union above a threshold of $0.5$. The detections were paired with decreasing confidence, and each label could only be paired to one detection.

To include the 3D centroid regression performance in the metric, we additionally only counted detections for which the absolute difference between the estimated and true centroid depth was below a threshold and varied this threshold during evaluation. Detections with depth errors above the threshold were regarded as false negatives. Since we cannot determine the depth errors for detections and labels that could not be paired, they were regarded as false examples irrespective of the depth error threshold. Note that some exam-

**Table 2.2:** Object detection performance in terms of average precision (AP, in $\%$), detection only, on the Mobility Aids dataset. We show the 2D image APs and the APs at a detection distance threshold of $0.5\,\mathrm{m}$.

| | | DepthJet | | | | RGB | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Fast R-CNN | Faster R-CNN centroid (ours) | | | Fast R-CNN | Faster R-NN centroid (ours) | | |
| | | G-xxs | R-50 | VGG-M | G-xxs | G-xxs | R-50 | VGG-M | G-xxs |
| 🚶 | $0.5\,\mathrm{m}$ | 75.8 | 80.6 | 77.3 | 77.0 | 68.1 | 84.3 | 73.1 | 74.6 |
| | 2D | 81.2 | 84.5 | 81.5 | 82.8 | 74.9 | 93.5 | 87.2 | 89.0 |
| ♿ | $0.5\,\mathrm{m}$ | 94.8 | 92.5 | 93.5 | 93.6 | 91.0 | 95.5 | 92.4 | 91.9 |
| | 2D | 96.0 | 96.6 | 95.4 | 94.8 | 92.0 | 98.8 | 98.4 | 98.1 |
| 🦽 | $0.5\,\mathrm{m}$ | 49.6 | 77.4 | 78.8 | 67.3 | 76.1 | 93.9 | 94.8 | 90.4 |
| | 2D | 51.8 | 84.9 | 81.0 | 74.7 | 72.4 | 95.0 | 98.3 | 96.3 |
| 🩼 | $0.5\,\mathrm{m}$ | 76.3 | 85.6 | 80.8 | 84.3 | 71.5 | 95.0 | 88.2 | 88.4 |
| | 2D | 76.6 | 88.3 | 81.5 | 84.7 | 74.2 | 96.8 | 91.3 | 93.4 |
| 🚶 | $0.5\,\mathrm{m}$ | 63.7 | 67.8 | 66.5 | 67.9 | 80.1 | 89.4 | 88.2 | 94.7 |
| | 2D | 64.8 | 68.6 | 67.7 | 69.2 | 82.2 | 98.5 | 98.3 | 97.3 |
| mAP | $0.5\,\mathrm{m}$ | 72.04 | **80.78** | **79.38** | **78.01** | 77.37 | **91.62** | **87.35** | **87.98** |
| | 2D | 74.10 | **84.61** | **81.41** | **81.23** | 79.16 | **96.52** | **94.71** | **94.84** |

ples do not have depth labels (see 2.4.1). Detections paired to those labels were ignored and did not contribute to the metric. However, unpaired labels without depth information still counted as false negatives. Due to the ignored detections, the 2D image mAP is different from the depth error mAP, even as the threshold approaches infinity.

Table 2.2 compares the depth error APs for the Mobility Aids dataset at a threshold of $0.5\,\mathrm{m}$ and the 2D image APs. Note that we also updated the results from our initial work presented in [147] for GoogLeNet-xxs with Fast R-CNN by applying minor modifications during test time (we allowed multiple detections of different classes per segment, which improves mAP). Consistent with the results in [119], Faster R-CNN clearly outperforms Fast R-CNN for the 2D image mAP metric. In addition, the depth error mAPs at $0.5\,\mathrm{m}$ for our approach are higher than for Fast R-CNN for all backbone architectures.

Figure 2.6 shows the evolution of the mAP as a function of the depth error threshold for the different methods. Our approach surpasses the performance of our initial work for all backbones architectures after a threshold of $0.3\,\mathrm{m}$. Our initial work can precisely estimate the centroid depths, as illustrated by the steep ascent of the mAP curve at low distance thresholds. However, the mAP ends at a much lower level than our approach,

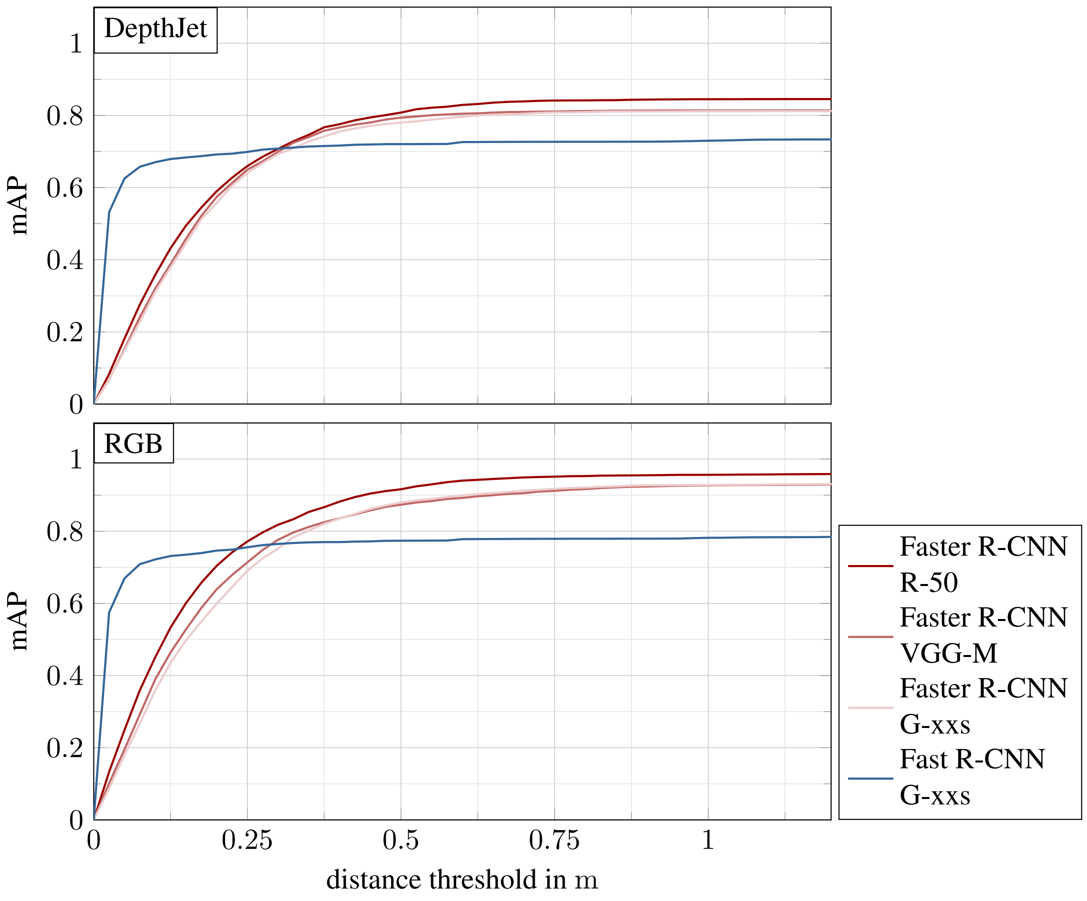

**Figure 2.6:** Mean average precision (mAP) comparison of Fast R-CNN and Faster R-CNN with different backbone architectures as a function of the detection distance threshold, for DepthJet and RGB.

which shows that it misses many examples that our method can detect. The differences are especially prominent for the RGB detection results, which is explained by the limited depth range of the camera.

Figure 2.7 compares the centroid regression performance in more detail. It shows the absolute distance between the true and estimated centroid depths at different depth intervals. Note that we can only evaluate the depth regression error for positive detection-label pairs. It is again visible that our initial work generally yields lower centroid depth errors. However, it also produces a few strong outliers, likely because some proposals were generated from depth information not belonging to the detected persons. Our approach produces fewer strong outliers, but the depth regression errors are overall higher. We can also see that our method yields better centroid depth regression performance for the DepthJet than the RGB dataset, which is expected since the DepthJet images include the color-encoded depth information. For both image modalities, the medium range between $1\,\mathrm{m}$ and $5\,\mathrm{m}$ has the lowest depth regression errors. This range corresponds to possible

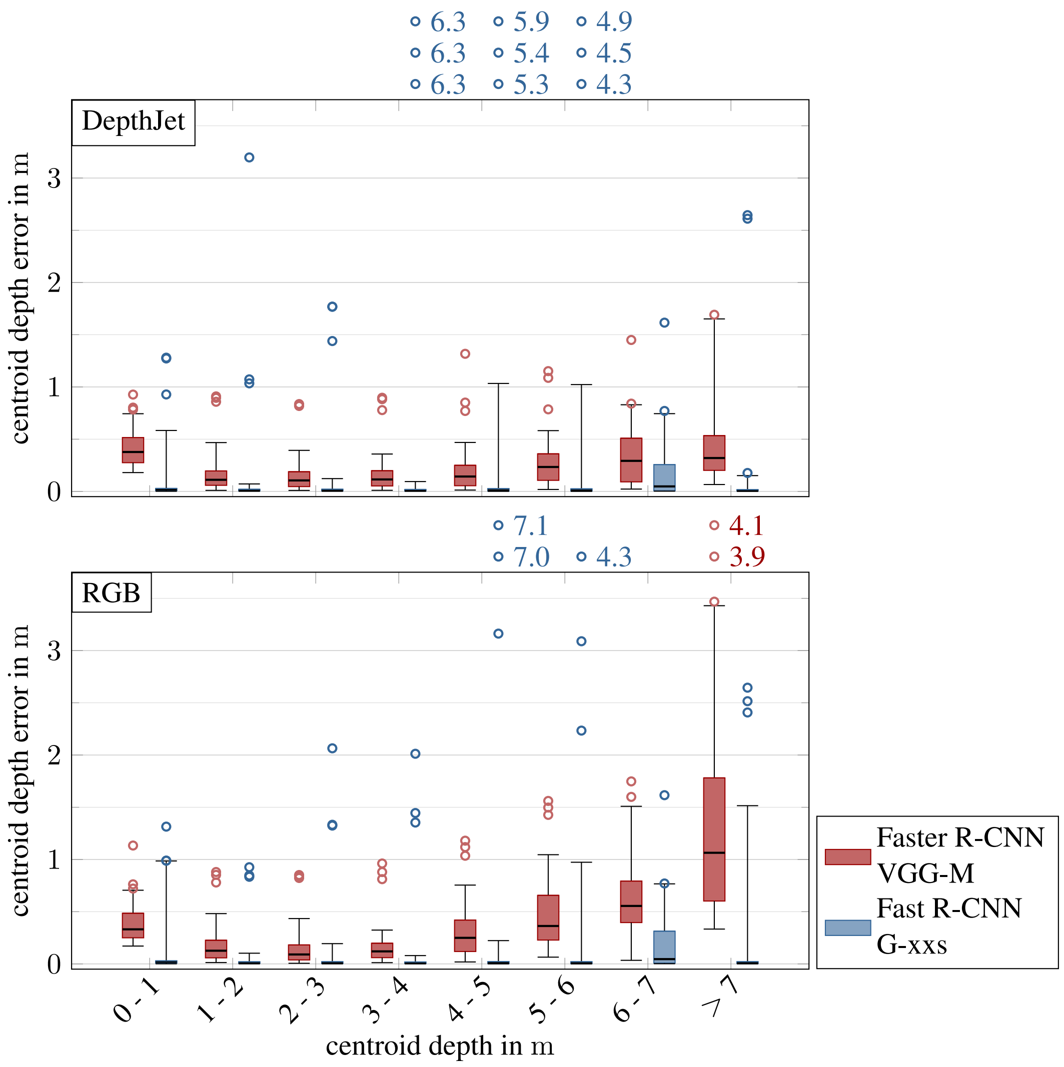

**Figure 2.7:** Variation of the centroid depth error with respect to the distance of people to the camera. The centroid depth error is the absolute distance between true and regressed centroid depth. The boxes show the interquartile range with the median, while the whiskers mark the 5th and 95th percentile. For visibility, we only mark the three greatest outliers at each interval with circles and put them above the plot area with the error value next to them if they exceed the range of the plot.

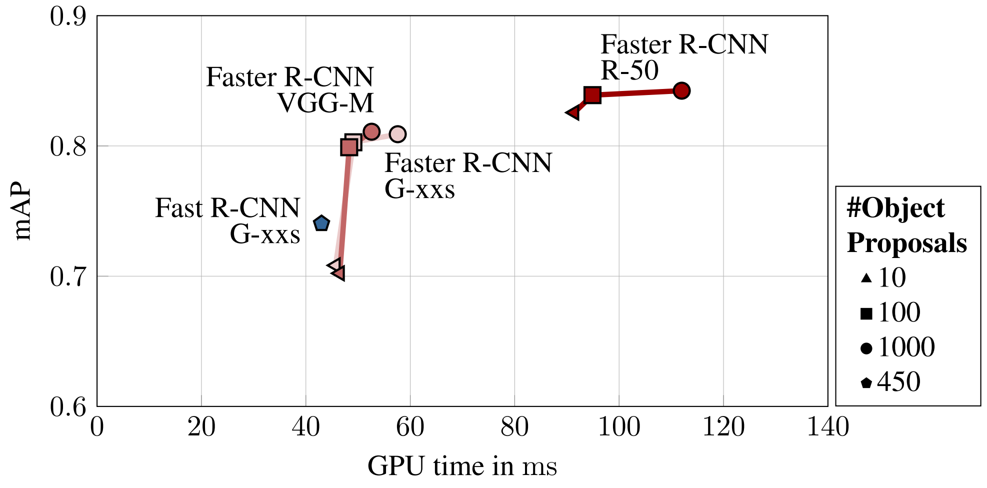

**Figure 2.8:** Runtime vs. mean average precision comparison for Fast R-CNN and Faster R-CNN with different backbone architectures for the Mobility Aids dataset with the DepthJet modality.

human-robot interactions within a short time window. For the greater distances, the error increases. Especially on RGB images we need to expect larger depth errors with increasing distance. Future work should investigate how the 3D object detection performance for the close and far range can be improved.

The results confirm that our detection framework can reliably detect people and distinguish them according to their mobility aids. Our new framework presents a substantial improvement to our initial work. It can perceive people that were previously missed entirely. Applications that do not require centimeter accuracy can profit greatly from our approach. Figure 2.12 on page 34 shows qualitative detection results for the RGB and the DepthJet detectors.

We further compared the runtime and mAP performance of the methods in Figure 2.8. To this end, we varied the number of top-scoring proposals evaluated by Faster R-CNN, between $10$, $100$, and $1000$ proposals per image. With Fast R-CNN, we used $450$ proposals per image on average [147]. Faster R-CNN with the VGG-M network architecture provides the best tradeoff between runtime and performance. With $100$ proposals for each image, the forward pass takes $48\,\mathrm{ms}$, and the mAP is $0.80$. Measurements were obtained using a computer with a 12-Core CPU and a GeForce GTX TITAN X with 12GB of memory.

### 2.4.3 Framework Detection Performance

We evaluated our complete framework on our test set 2 to assess the contribution of the different framework modules to the overall detection performance. We compared the performance in terms of precision, recall, and F1 score rather than mAP because the tracker

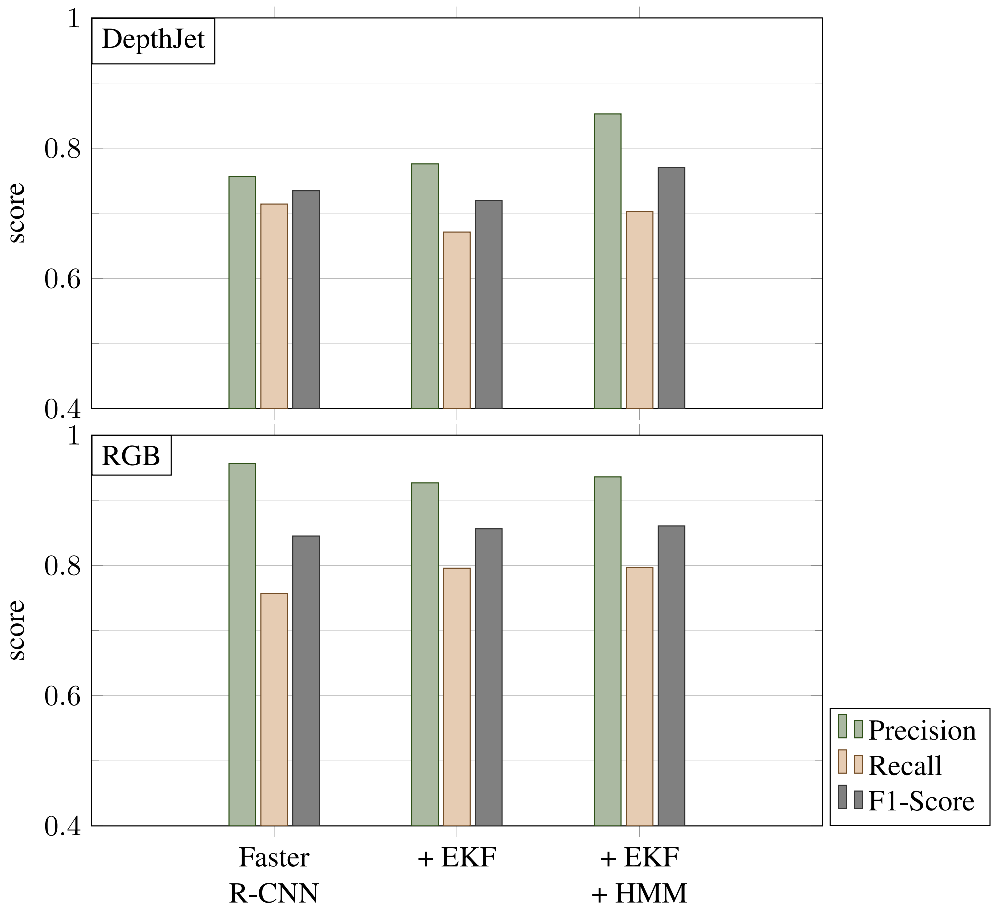

**Figure 2.9:** Object detection performance evolution compared to stand-alone Faster R-CNN with the VGG-M backbone.

processes thresholded detections. We set the detection thresholds based on test set 1 as the class confidences where the precision equals the average precision to be comparable to Table 2.2. Figure 2.9 shows the 2D image precision, recall, and F1 scores of the framework stages for RGB and DepthJet. Test set 2 is particularly challenging because of occlusions, which explains the decrease in performance compared to the test set 1 evaluation. In the DepthJet case, the EKF stage decreases the recall, while the precision only improves slightly. The addition of the HMM finally boosts the precision by ten percentage points compared to detection only and recovers the recall back to the detection level. For the RGB case, the tracking stages improve recall by almost four percentage points, while the precision is slightly decreased by two percentage points. The detector is already very strong, so filtering over time can resolve occlusions and thus improve recall, but filter effects like errors in the assumptions of constant velocity or wrong data associations impact the precision. The F1 score, which is the harmonic mean of the precision and recall, improves with the addition of the tracking modules for both modalities.

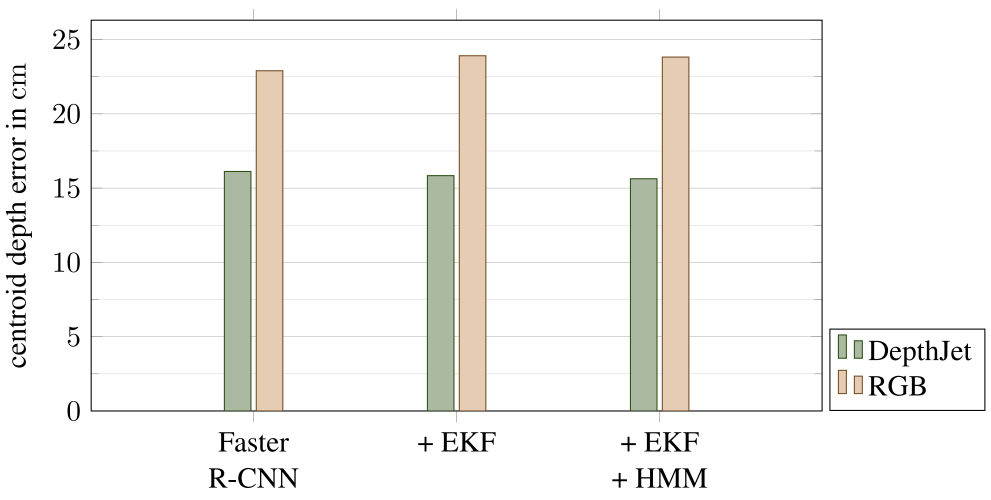

**Figure 2.10:** Depth regression performance evolution compared to stand-alone Faster R-CNN with the VGG-M backbone.

In addition to the object detection scores, we evaluated the mean absolute distance between the predicted and true centroid depths. Figure 2.10 summarizes the centroid depth regression error for the framework stages. The tracking module does not influence the centroid depth regression performance of our framework visibly. All in all, the experiments confirm that the tracking module has positive effects on the detection performance, even for the already strong RGB detector.

### 2.4.4 Person Guidance Scenario

We tested our system in a person guidance scenario to show the applicability of our framework to a real-world service robot task. The task of the robot was to guide visitors to the professor's office in our lab, building 80 at the Faculty of Engineering of the University of Freiburg. The professor's office is located on the first floor opposite the staircase at the main entrance. An elevator is available on the other side of the corridor. Depending on the perceived mobility aids category of the visitor, the robot should guide her or him either to the elevator or to the stairs. Furthermore, to ensure that the person can follow the robot and not get lost, the robot adapted its velocity to the person while driving to its destination.

Our robot used a laptop computer with an 8-Core CPU and a GeForce GTX 1080 with 8 GB of memory. We used our RGB network with the probabilistic position, velocity, and class estimation module to process the color images of the Kinect 2 camera mounted on the robot at approx. 15 frames per second. We did not use the depth data of the Kinect 2 for this experiment. To ensure that the followers remain in the field of view of the robot while the robot is navigating, we pointed the camera to the back of the robot. For the navigation parts, we employed the ROS navigation stack [96] with a laser rangefinder

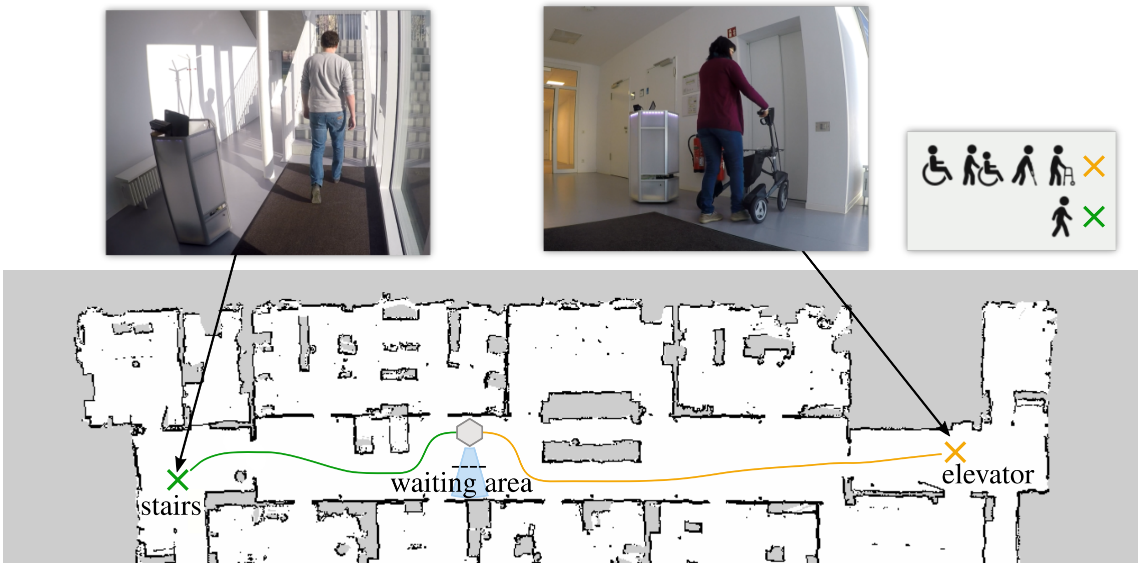

**Figure 2.11:** We used our framework to provide assistance in a person guidance experiment. The task of the robot was to guide all pedestrians to the nearest staircase (left image) and all people with mobility aids to the elevator (right image).

pointing to the front of the robot for localization and obstacle avoidance. We used the global_planner package from the ROS navigation stack to generate the global navigation paths. For the local path planning, we adopted the omni_path_follower ROS package [71], which generates velocities for omnidirectional robots to closely follow a navigation path.

We selected the initial waiting pose of the robot in the hallway of the ground floor and two goal poses, see Figure 2.11. At the waiting pose, the robot was facing the wall while the camera was pointing backward into the hallway. The robot observed an area of interest $3\,\mathrm{m}$ in front of the camera and within $\pm 20°$ from its center. Once it detected a person in this area with an absolute velocity of less than $0.25\,\mathrm{m\,s^{-1}}$, the robot started to navigate to one of the two goals. For pedestrians without perceived motion impairments, it navigated to the goal by the stairs; people with mobility aids were guided to the elevator. The robot used predefined speech commands to ask the visitors to follow it and inform them how to proceed to the professor's office once it reached the navigation goal. Upon reaching the destination, the robot returned to the waiting pose and waited for the next visitor.

To ensure that the visitors can follow the robot, it kept track of them during navigation. To this end, the robot retained the track associated with the follower until it reached its goal. During navigation, the robot always turned the camera towards its follower by rotating the base with a rotational velocity of

$$\omega = k_\omega \arctan\left(\frac{{}^{\mathrm{r}}y_{\mathrm{p}}}{-{}^{\mathrm{r}}x_{\mathrm{p}}}\right), \tag{2.22}$$

where ${}^{\mathrm{r}}x_{\mathrm{p}}$ and ${}^{\mathrm{r}}y_{\mathrm{p}}$ denote the person's position in the robot coordinate frame and $k_\omega$ is a proportional gain. The robot coordinate system is at ground level, with the $x$-axis pointing forward and the $y$-axis pointing to the left. Note that since our robot is omnidirectional, it can follow a path at arbitrary orientations. This enables it to perform a base rotation for keeping the follower in view while navigating to the goal.

The perceived mobility aids indicate the follower's preferred velocity since people with mobility aids tend to move slower than people without motion impairments [100]. Therefore, our robot chose an initial guidance velocity of $v_0 = 0.2\,\mathrm{m\,s^{-1}}$ when guiding a person with walking aids, compared to $v_0 = 0.4\,\mathrm{m\,s^{-1}}$ when guiding a pedestrian. However, the motion capabilities likely vary from person to person. Therefore, our robot adjusted its guidance velocity to the person at each time step $t$ to

$$v_t = v_{t-1} + k_v \left(d_{\mathrm{p}} - |{}^{\mathrm{r}}x_{\mathrm{p}}|\right). \tag{2.23}$$

Here, $d_{\mathrm{p}}$ is a fixed target distance between the robot and the person, and $k_v$ is a proportional gain. We additionally restrict the guidance velocity to $0\,\mathrm{m\,s^{-1}} \leq v_t \leq 0.5\,\mathrm{m\,s^{-1}}$. The target distance $d_{\mathrm{p}}$ is determined as the distance at which the person approached the robot in the waiting area, plus a small offset of $+0.2\,\mathrm{m}$ to account for the robot driving ahead. If the track of the follower was removed because the background class was dominant or the position uncertainty was too high, the robot considered a person as lost. When it lost the follower, the robot turned towards the path and traveled to its destination with the initial guidance velocity $v_0$.

We tested twenty guidance runs with different people from our lab, four for each of the mobility aids categories: pedestrian, person with crutches, person in a wheelchair, person with walking frame, and person pushing another person in a wheelchair. We marked the waiting area between $1.5\,\mathrm{m}$ and $3\,\mathrm{m}$ in front of the camera with tape on the floor and asked the participants to approach the robot at the waiting area and follow it once it gives the speech command. Furthermore, we asked the test subjects to keep a distance to the robot roughly as indicated by the tape during the entire experiment to make sure they stay in the field of view of the camera. Additionally, we told the participants that the robot would adjust its velocity to them, so they could walk as fast as they like.

In all of the runs, the robot perceived the correct mobility aids category and successfully navigated the follower to the right destination. Furthermore, the robot successfully kept track of its follower until it reached the goal in seventeen runs. It lost track of the person in one run with a pedestrian and in two runs where a person was pushing another person in a wheelchair. In these runs, the people came too close to the camera and were therefore not detected for multiple frames. A different camera with a wider field of view could be used to solve this problem. Furthermore, the robot could try to find the follower again, maybe taking visual features into account. This is, however, out of the scope of this work and remains for future research.

The tracking module estimated the correct class of the follower in $92.9\,\%$ percent of all frames, over all runs. The mean guidance velocity of the robot was $0.38\,\mathrm{m\,s^{-1}}$ with a standard deviation of $0.09\,\mathrm{m\,s^{-1}}$. Here, the robot moved at an average speed of $0.36\,\mathrm{m\,s^{-1}}$ when guiding people with walking aids and at $0.42\,\mathrm{m\,s^{-1}}$ when guiding pedestrians. However, these velocities are not very meaningful since all test subjects were young and healthy. They were physically able to keep up with the robot and likely co-adapted to its velocity. Many of the test subjects, however, tried different velocities during the experiment and also stopped to test the robot behavior. Some test subjects reported that the robot took too long to adapt its velocity and only started moving again after a stop when they came very close. More sophisticated methods for generating the guidance velocity based on the people detections could be used to yield a more natural and prompt guidance behavior, but they exceed the scope of this work.

The experiment demonstrates that our approach can be successfully applied on a moving robot in an authentic environment. Further, it showcases how our approach can give appropriate, individual assistance to people, according to their needs.

## 2.5 Conclusion

This chapter introduced a novel perception system to detect and distinguish people according to the mobility aids they use, based on a deep neural network and supported by tracking and class estimation modules. Our method shows a significant increase in object detection performance, compared to a Fast R-CNN baseline with depth-based proposal generation. We added a 3D centroid regression output to our network to estimate the 3D centroids of people from image data only without additional geometric information. Our person guidance experiment showed that our detection pipeline enables robots to provide individual assistance to people with advanced needs.

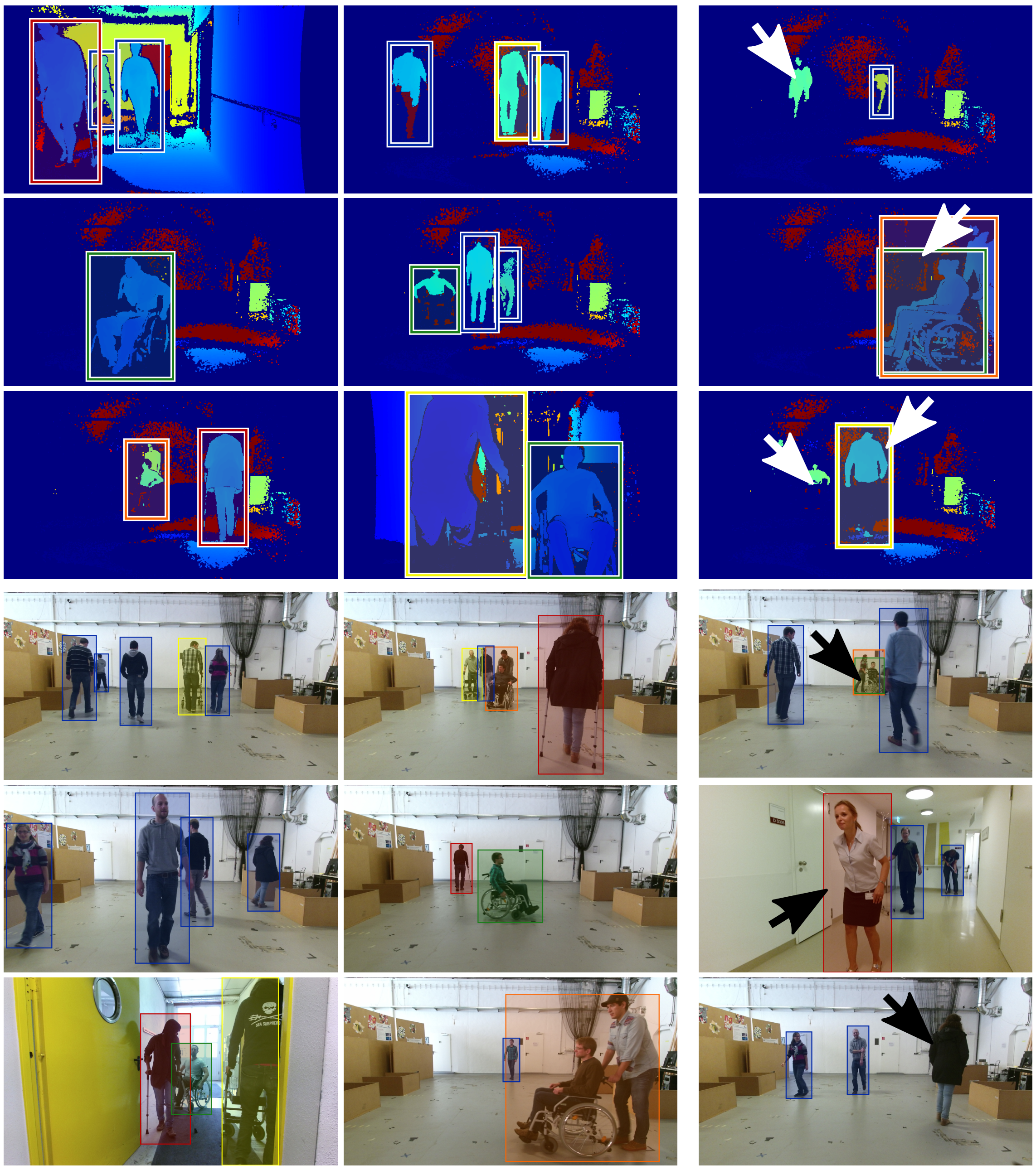

**Figure 2.12:** Qualitative object detection results obtained using the Faster R-CNN network with VGG-M backbone, for DepthJet and RGB. Left: positive examples. Right: cases of failure with missed or multiple detections and wrong classifications.

# Chapter 3

# Perception of Interaction Forces

**Most conventional approaches to mobile robot navigation try to avoid unintended physical contact with the environment or with humans. As distance sensors typically have a limited – and often only two-dimensional – field of view, collisions with the environment or contacts with humans cannot be entirely avoided in practical mobile robot applications. On the other hand, direct physical contact allows for intuitive communication between a robot and humans. This chapter presents a novel whole-body sensory concept to perceive physical interaction between a mobile robot and humans. Our design is the first to include a model-free force filter based on a neural network that distinguishes between contact and disturbance forces introduced by the motion of the robot and oscillations. Thus, our novel design enables the perception of interaction forces during autonomous motion. In extensive experiments with our robot Canny, we demonstrate the effectiveness and advantages of the neural network approach, which clearly outperforms a classical model-based filter.**

## 3.1 Introduction

Robots are envisioned to increasingly share their workspaces with humans. This development poses novel challenges for robotics and AI. First, populated environments are typically cluttered and may change over time. Second, mobile robots need to consider and react to the people they encounter, requiring accurate people perception and reasoning about the people's navigation intents. Still, collisions with the environment or accidental contacts with people may happen due to incomplete sensor information, imperfect people detections, or the highly dynamic nature of the environment, as depicted on the left in Figure 3.1. Robots operating in populated environments must react appropriately to these contacts to avoid injuries and damage. At the same time, the

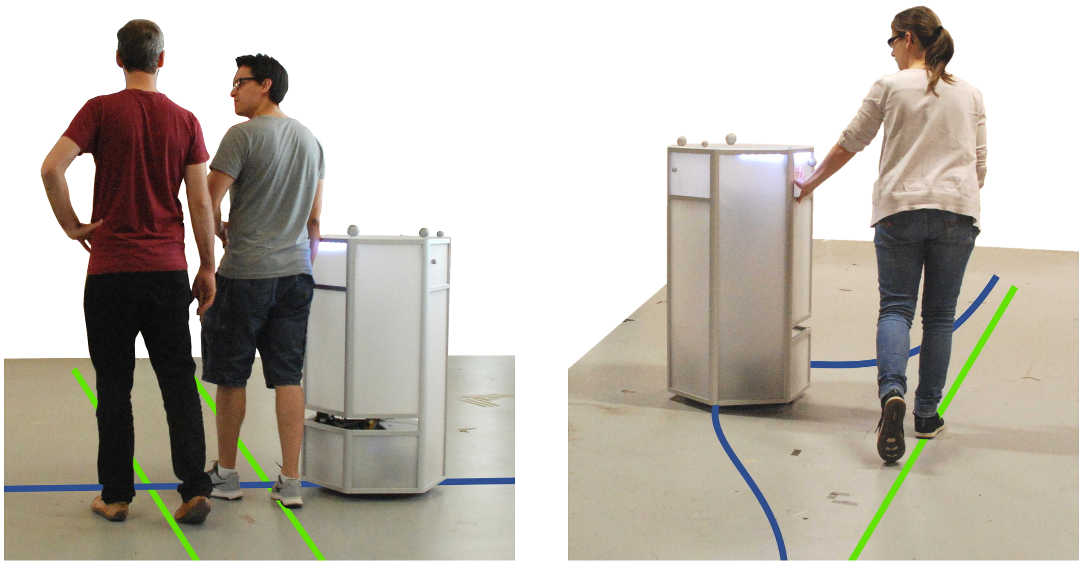

**Figure 3.1:** The perception of interaction forces can be advantageous for mobile robots in various interaction scenarios. The left image shows an accidental collision, which a force-sensitive robot could mitigate, and on the right, a person pushes the robot away from her desired path.

close proximity to people allows for intuitive and efficient physical human-robot interaction. One possible interaction scenario is shown on the right in Figure 3.1, where a person pushes the robot out of her desired path. Here, the robot reactively responds to the interaction force. Chapters 4 and 5 introduce two more elaborate applications, where the robot learns improved environment models and adapts its navigation behavior around people, based on the perceived collisions and interaction forces. This chapter provides the basis for these applications by enabling the robot to perceive interaction forces during autonomous motion. Substantial parts of the ideas, results, figures, tables, and text presented in this chapter have been previously published in [73], ©2018 IEEE, `https://doi.org/10.1109/ICRA.2018.8460510`. Section 1.4 outlines the author's contribution to this work.

We present a novel whole-body sensory concept based on a six degrees of freedom (DoF) force-torque sensor mounted stiffly on an omnidirectional mobile base. A solid outer shell attached to the other side of the sensor absorbs and directs external forces to the sensor. Our setup enables the robot to perceive the extend, direction, and location of interaction forces on the robot shell. Even though the motion capabilities of wheeled mobile robots are limited to the 2D plane, knowledge of the 3D direction and impact of forces on the shell can be valuable to distinguish different types of interactions. An intended contact is most likely occurring in reach of a person's hands, while a collision can occur anywhere on the robot. A person leaning against the robot likely causes a force with a sideways and downwards component. The appropriate robot reaction depends on the type of contact, and the force characteristics can help distinguish them. For multiple contacts, our setup can only process the resultant force but cannot discriminate between them. We accept this limitation since the resultant force already provides valuable information for the envisioned contact scenarios in populated spaces.

Force-sensitive mobile robots have been presented in related works, e.g., for corporative object transport [56, 65] or as walking helper systems [57, 122, 130]. These platforms always commanded a velocity proportional to the force exerted by the user. In contrast, we aim for a mobile robot that can perceive and react to interaction forces during autonomous motion. Unfortunately, many sensory designs are sensitive to force stimuli caused by the robot motion and oscillations, especially without a person touching the robot and thus damping the impacts. The induced internal disturbance forces superimpose the true external interaction forces, thus the force perception becomes unreliable as soon as the robot drives autonomously. To address this limitation, we present two novel filtering techniques to distinguish disturbance forces from external forces. The first, probabilistic approach uses a physical model of the base response to external and inertial force stimuli. It serves as a reference for our main method, which employs end-to-end learning using a neural network with a multi-task output. The sensory concept combined with the filtering technique enables us to precisely estimate the magnitude, direction, and location of external forces, even when the robot is in motion. In extensive experiments, we evaluate the performance of our system. To this end, we present a novel experimental setup where varying external forces of known magnitudes, directions, and impact points can be applied to any point of the robot shell.

Force-torque sensors like the one we use for our robot Canny are standard components widely applied in robotics. Thus, our design is easily applicable to other robot platforms. We believe that our design is an important contribution along the way to safe, interactive mobile robots that operate in close proximity to humans.

## 3.2 Related Work

Autonomous robot navigation among humans is a challenging problem since the shared environments are typically cluttered, highly dynamic, and hardly predictable. Especially in crowded environments, it is necessary to assume that all agents cooperate to progress towards a navigation goal [78, 142]. Prassler *et al.* [114] presented a robot wheelchair that used the Velocity Obstacle approach [39] for navigation in crowded environments. The Minerva museum tour-guide robot [141] reacted with angry voices and expressions on its actuated face to eke out a path. Our work seeks to extend such existing navigation approaches by enabling the robot to perceive interaction forces, allowing it to reason about and react to intentional physical contact and accidental collisions.

Early work on force-controlled robots dates back to the 1990s when Khatib [65] introduced the robotic assistant for cooperative object manipulation between humans and mobile manipulators. Later, Haddadin *et al.* [50] presented a concept for human safety in shared workspaces for robot manipulators. Hirata *et al.* [56] proposed an approach to cooperative object transportation where a human pushed the robot along a pre-planned path, indicating the desired robot velocity. Similar to us, they used a 6-DoF force-torque sensor between the robot base and the shell to measure interaction forces. However, they did not evaluate the impact locations of the forces.

Walking helper systems for the elderly have been presented that react to a force exerted by the user, measured, e.g., by strain gauges [122] or 6-DoF force-torque sensors [57, 130]. The design of the devices is similar to ours in that a force sensor adds sensitivity to a stiff mounted structure. However, unlike our robot shell, not all exposed parts of the walking devices were force-sensitive because the mounted structure did not cover the entire devices. Furthermore, the motion of the walking helpers was always a reaction to an external force. Instead, we strive to combine autonomous motion with a compliant response to interaction forces.

Manuelli and Tedrake [95] used a particle filter to estimate the contact points of external forces on rigid body humanoid robots from the joint torques. Their approach was able to accurately assess the locations of multiple impact points for a simulated humanoid robot. Kim *et al.* [67] also estimated the external forces based on the joint torques of their mobile base using torque sensors in the drive trains of the three omnidirectional wheels. Their robot could react to interaction forces and detect collisions with the entire robot body, at rest and in motion. However, they only estimated the position and direction of the external force in the horizontal plane due to the limited degrees of freedom of the robot. Furthermore, they did not evaluate the location, direction, and magnitude of external forces when the robot was in motion. Frémy *et al.* [43] also used the torques on the caster wheel joints of their mobile robot to estimate external forces. However, their platform was not fully omnidirectional, which limited the possible reactions to interaction forces.

Our work uses the approach of Bicchi *et al.* [13] to determine the impact point of an applied force. We extend their work by filtering out disturbance forces caused by the motion of the robot and oscillations of its shell. Thus, we render their approach to real mobile robot applicability. We first introduce a model-based filtering technique that serves as a reference for our main, neural network-based filter. Signal filtering with neural networks has been explored in various areas. Schirrmeister *et al.* [123] demonstrated the potential of deep convolutional neural networks for processing EEG data, which typically suffer from substantial signal-to-noise ratios. Our work is the first to investigate signal filtering with neural networks for force-sensitive mobile robots.

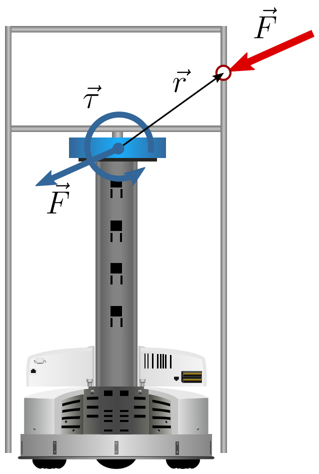

**Figure 3.2:** Our sensory concept is based on a force-torque sensor (blue rectangle) mounted between the omnidirectional base and the solid robot shell and estimates the magnitude, direction, and impact point of an external force (red arrow).

## 3.3 Whole-Body Sensory Concept

Our robot Canny is based on the omnidirectional research platform Robotino. The Robotino is shipped with a mounting tower that attaches tightly to the base. We mounted a high-precision 6-DoF force-torque sensor to the tower, as depicted in Figure 3.2. The other side of the force-torque sensor is attached to a solid robot shell made of aluminum profiles and semi-transparent acrylic glass. An inertial measurement unit (IMU) at the robot base measures the base accelerations in the six DoFs. Furthermore, colored LEDs at the rim of the robot shell display user feedback, such as the magnitude and impact point of a perceived interaction force.

We calculate the impact point of an external force from the measured forces and torques, following Bicchi *et al.* [13]. We restrict ourselves to the model of a *point contact with friction*, which means that the force is applied on a single point instead of an area. Accordingly, we neglect local torques during the impact and assume that the magnitude and direction of the external force is directly measured by the sensor.

For an impact force $\vec{F}$ on the robot shell at a lever arm $\vec{r}$, the force-torque sensor measures the force $\vec{F}$ and a corresponding torque $\vec{\tau}$ (see Figure 3.2), where

$$\vec{\tau} = \vec{r} \times \vec{F}. \tag{3.1}$$

We can restructure the above relation to

$$\begin{aligned}\vec{\tau} &= \bar{\mathbf{F}}\vec{r}, \\ \text{with } \bar{\mathbf{F}} &= \begin{pmatrix} 0 & F_z & -F_y \\ -F_z & 0 & F_x \\ F_y & -F_x & 0 \end{pmatrix}.\end{aligned} \tag{3.2}$$

To find the impact point of the force on the robot shell, we need to calculate $\vec{r}$ from the measured $\vec{F}$ and $\vec{\tau}$. However, $\bar{\mathbf{F}}$ is singular; therefore, it is not possible to directly solve Eq. (3.2) for $\vec{r}$. Instead, all possible solutions for $\vec{r}$ lie on a straight line in the 3D space, given by

$$\vec{r}_\lambda = \vec{r}_0 + \lambda\vec{F}, \tag{3.3}$$

where $\lambda$ is the line coordinate parameter, and

$$\vec{r}_0 = \frac{\vec{F} \times \vec{\tau}}{||\vec{F}||_2^2}. \tag{3.4}$$

Intersecting this line with the shell geometry gives all potential points of impact on the shell.

For convex shell geometries, we can calculate a unique point of impact $\vec{r}$, assuming that the impact force results from pushing on the robot shell instead of pulling. We approximate Canny's hexagonal shell by a cylinder with radius $R$ and axis $\vec{z} \in S^2$ and obtain $\vec{r}$, following Bicchi *et al.* [13], by

$$\begin{aligned}\vec{r} &= \vec{r}_0 + \lambda_r\vec{F}, \\ \text{where } \lambda_r &= \frac{-\vec{F}^\perp \cdot \vec{r}_0^\perp - \sqrt{(\vec{F}^\perp \cdot \vec{r}_0^\perp)^2 - ||\vec{F}^\perp||_2^2(||\vec{r}_0^\perp||_2^2 - R^2)}}{||\vec{F}^\perp||_2^2}.\end{aligned} \tag{3.5}$$

Thereby, $\vec{v}^\perp = \vec{v} - (\vec{v} \cdot \vec{z})\,\vec{z}$ denotes the part of a vector $\vec{v} \in \mathbb{R}^3$ which is orthogonal to $\vec{z}$.

## 3.4 External Force Filtering

We present two methods to separate the external impact forces from disturbance forces caused by robot motion and oscillations. The first approach presented in Section 3.4.1 is probabilistic and model-based. It serves as a reference for our main method presented in Section 3.4.2, which is model-free and based on a neural network.

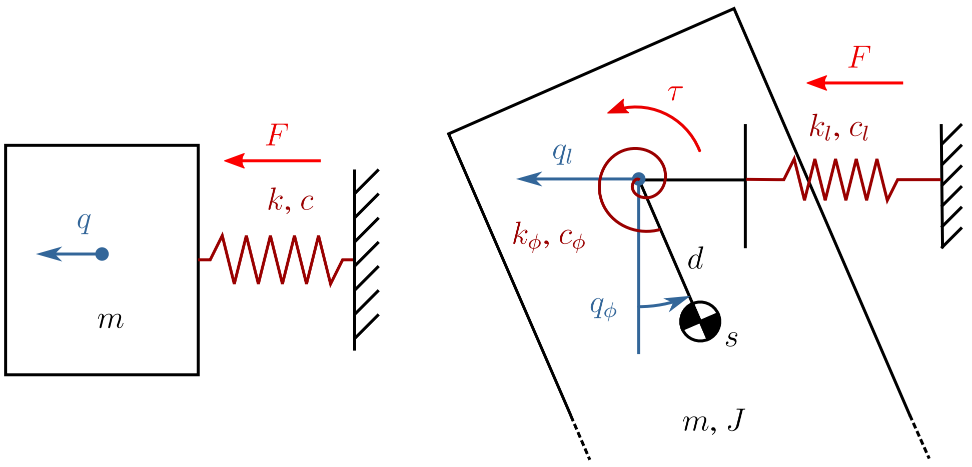

**Figure 3.3:** One-dimensional linear spring-mass system (left) with spring constant $k$ and damping constant $c$. On the right, our extended 2D case with displaced center of mass, resulting in coupled linear and angular oscillations. The walls in the drawings correspond to the robot base, which itself is movable, resulting in additional accelerations of the reference frame.

### 3.4.1 Model-Based External Force Estimation

For our model-based force filter, we model our system as a stiff body suspended on a system of springs and dampers. As a simplification, we assume that the robot structure, the shell, and the mounting tower are perfectly stiff, while the force-torque sensor is the sole source of elasticity. We use the coordinate system of the sensor, where the $z$-axis points vertically up, and model the kinematics of the system in six degrees of freedom.

To derive the equations of motion, we first consider the 1D case of an oscillatory system, as depicted in Figure 3.3 (left). The mass $m$ is connected to a spring $k$ and a damper $c$. The deflection of the spring is denoted by $q$. The equation of motion for this linear 1D system is given by

$$m\ddot{q} + c\dot{q} + kq = F. \tag{3.6}$$

The force acting on the mass

$$F = F_\mathrm{e} - F_a \tag{3.7}$$

comprises an external force $F_\mathrm{e}$ and a force $F_a = -ma$ caused by the acceleration $a$ of the inertial frame, i.e., the acceleration of the robot. Thus, we can rewrite the equation of motion to

$$m\ddot{q} + c\dot{q} + kq = F_\mathrm{e} - ma. \tag{3.8}$$

If the robot shell is mounted at its center of mass, the degrees of freedom are decoupled and can be described by Eq. (3.8). However, this does not hold in our case, as the mounting of the shell is displaced vertically along the $z$-axis. Hence, some degrees of freedom are coupled, as depicted in Figure 3.3 (right), where we consider the 2D case of coupled linear oscillation in $q_l$ and angular oscillation in $q_\phi$. The stiff robot shell with mass $m$ and moment of inertia $J$ is mounted at the distance $d$ from its center of mass $s$. Linear and angular springs $k_l$ and $k_\phi$ and dampers $c_l$ and $c_\phi$, representing the elasticity of the force-torque sensor, are attached to the shell. Again, a force $F = F_\mathrm{e} - ma_l$ and a torque $\tau = \tau_\mathrm{e} - Ja_\phi$ act on the body.

We calculate the coupled equation of motion with the Euler-Lagrange equations as

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_{l/\phi}}\right) - \frac{\partial L}{\partial q_{l/\phi}} = Q_{l/\phi}, \tag{3.9}$$

where $Q_{l/\phi} = F/\tau - c_{l/\phi}\dot{q}_{l/\phi}$ are the two generalized forces and $L = T - V$ is the Lagrangian. The kinetic energy $T$ and the potential energy $V$ for our 2D system are calculated as

$$T = \frac{1}{2}m\left((\cos(q_\phi)\dot{q}_\phi d + \dot{q}_l)^2 + (\sin(q_\phi)\dot{q}_\phi d)^2\right) + \frac{1}{2}J\dot{q}_\phi^2, \tag{3.10}$$

$$V = \frac{1}{2}k_l q_l^2 + \frac{1}{2}k_\phi q_\phi^2 - mg\cos(q_\phi)d, \tag{3.11}$$

where $g$ is the gravitational constant. Solving Eq. (3.9) results in two coupled equations of motion:

$$\ddot{q}_l = f_l\left(q_l, q_\phi, \dot{q}_l, \dot{q}_\phi, F, \tau\right) \tag{3.12}$$

$$\ddot{q}_\phi = f_\phi\left(q_l, q_\phi, \dot{q}_l, \dot{q}_\phi, F, \tau\right) \tag{3.13}$$

In the following, we will denote the linear deflections along the axes of the sensor as $x_l$, $y_l$ and $z_l$ and the angular deflections as $x_\phi$, $y_\phi$ and $z_\phi$. For our sensory system, we model $x_l$ and $y_\phi$ and also $y_l$ and $x_\phi$ as coupled degrees of freedom with Eq. (3.12) and Eq. (3.13). The origins of the $x$ and $y$ axes coincide with the center of mass of the shell. Hence $z_\phi$ is not coupled with any other degree of freedom. Furthermore, we neglect coupling effects between $z_l$, $x_\phi$, and $y_\phi$, as the angular deflections are expected to be very small, and the induced deflections in the $z$-direction are even smaller due to the second-order dependency. Hence, $z_l$ and $z_\phi$ can each be described by Eq. (3.8). We end up with a non-linear, second-order differential equation that describes our sensory setup in 6 DoFs of the form

$$\begin{aligned} \mathbf{M}(\ddot{\mathbf{q}}) + \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{F}) &= \mathbf{0}, \\ \text{with } \mathbf{q} &= \begin{pmatrix} x_l & y_l & z_l & x_\phi & y_\phi & z_\phi \end{pmatrix}^T \\ \text{and } \mathbf{F} &= \begin{pmatrix} F_x & F_y & F_z & \tau_x & \tau_y & \tau_z \end{pmatrix}^T. \end{aligned} \tag{3.14}$$

As the next step, we linearize Eq. (3.14) around $(\mathbf{q}^T, \dot{\mathbf{q}}^T, \mathbf{F}^T) = \mathbf{0}$, using the first two terms of the Taylor expansion and the linear dependency on $\ddot{\mathbf{q}}$, and we obtain

$$\bar{\mathbf{M}}\ddot{\mathbf{q}} + \bar{\mathbf{C}}\dot{\mathbf{q}} + \bar{\mathbf{K}}\mathbf{q} = \bar{\mathbf{F}}. \tag{3.15}$$

We can formulate the above relation as the first-order differential equation

$$\dot{\mathbf{x}}_{\mathbf{q}} = \underbrace{\begin{pmatrix} \mathbf{0} & \mathbf{I} \\ \bar{\mathbf{M}}^{-1}\bar{\mathbf{K}} & \bar{\mathbf{M}}^{-1}\bar{\mathbf{C}} \end{pmatrix}}_{\mathbf{A}} \mathbf{x}_{\mathbf{q}} + \underbrace{\begin{pmatrix} \mathbf{0} \\ \bar{\mathbf{M}}^{-1} \end{pmatrix}}_{\mathbf{B}} \bar{\mathbf{F}}, \tag{3.16}$$

where $\mathbf{I}$ denotes the identity matrix and $\mathbf{x}_{\mathbf{q}} = \begin{pmatrix} \mathbf{q} & \dot{\mathbf{q}} \end{pmatrix}^T$ represents the system state with system matrix $\mathbf{A}$ and input matrix $\mathbf{B}$.

We now discretize Eq. (3.16) with the time interval $\Delta t$ to

$$\begin{aligned} \mathbf{x}_{\mathbf{q}}(k+1) &= \mathbf{A}_{\mathbf{k}}\mathbf{x}_{\mathbf{q}}(k) + \mathbf{B}_{\mathbf{k}}\mathbf{F}(k), \\ \text{with } \mathbf{A}_{\mathbf{k}} &= \exp(\mathbf{A}\Delta t) \\ \text{and } \mathbf{B}_{\mathbf{k}} &= \mathbf{A}^{-1}(\mathbf{A}_{\mathbf{k}} - \mathbf{I})\,\mathbf{B}. \end{aligned} \tag{3.17}$$

$\mathbf{F}(k)$ is a partially unknown input to our system. While we have (noisy) measurements for the acceleration from our IMU, we cannot measure the external force and torque components in $\mathbf{F}_{\mathbf{e}}$ directly.

In the following, we neglect the force contribution of the robot acceleration and assume

$$\mathbf{F} \approx \mathbf{F}_{\mathbf{e}}. \tag{3.18}$$

We further assume that the force $\mathbf{F}_{\mathbf{e}}$ does not change drastically from one time step $k$ to the next

$$\mathbf{F}_{\mathbf{e}}(k+1) \approx \mathbf{F}_{\mathbf{e}}(k). \tag{3.19}$$

Both assumtions restructure Eq. (3.17) to

$$\begin{aligned} \mathbf{x}(k+1) &\approx \underbrace{\begin{pmatrix} \mathbf{A}_{\mathbf{k}} & \mathbf{B}_{\mathbf{k}} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}}_{\hat{\mathbf{A}}} \mathbf{x}(k), \\ \text{with } \mathbf{x} &= \begin{pmatrix} \mathbf{x}_{\mathbf{q}} \\ \mathbf{F}_{\mathbf{e}} \end{pmatrix} \end{aligned} \tag{3.20}$$

as the new state vector.

The force-torque sensor measures the forces and torques with strain gauges. Therefore, we assume that the measured force $\mathbf{F}_{\mathbf{m}}$ is proportional to the deflection $\mathbf{q}$. We hence formulate the measurement equation as

$$\tilde{\mathbf{F}}_{\mathbf{m}} = \begin{pmatrix} \mathbf{F}_{\mathbf{m}} \\ \mathbf{0} \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{K} & \mathbf{0} \end{pmatrix}}_{\mathbf{C}} \mathbf{x}(k), \tag{3.21}$$

where $\mathbf{K}$ is a diagonal matrix, consisting of the six linear and angular spring constants $k_{l,x/y/z}$ and $k_{\phi,x/y/z}$.

The external force and torque matrix $\mathbf{F}_{\mathbf{e}}$ is now part of our state space, and we use a standard Kalman filter to estimate it, together with the spring deflections and their derivatives, using $\hat{\mathbf{A}}$ as the state transition matrix and $\mathbf{C}$ as the measurement matrix. The state vector has a dimension of 18. We assume Gaussian noise in the prediction and update step,

$$\mathbf{x}(k+1) = \hat{\mathbf{A}}\mathbf{x}(k) + \nu_{\mathbf{R}}, \tag{3.22}$$

$$\tilde{\mathbf{F}}_{\mathbf{m}} = \mathbf{C}\mathbf{x}(k) + \nu_{\mathbf{Q}}, \tag{3.23}$$

with $\nu_{\mathbf{R}} \sim \mathcal{N}(0, \mathbf{R})$ and $\nu_{\mathbf{Q}} \sim \mathcal{N}(0, \mathbf{Q})$. For the measured force $\mathbf{F}_{\mathbf{m}}$, we model the measurement noise covariance matrix

$$\mathbf{Q} = \mathrm{diag}(\sigma_{\mathbf{m}}) \tag{3.24}$$

as a diagonal matrix with standard deviations $\sigma_{\mathbf{m}}$ for the individual components. For the prediction noise covariance, we assume that the error of the assumption in Eq. (3.19) introduces the dominant contribution and use the control matrix $\mathbf{B}_{\mathbf{k}}$ to estimate how the noise $\nu_{\mathbf{F}}$ in $\mathbf{F}_{\mathbf{e}}(k)$ propagates into the state space. Hence, we model the prediction noise by

$$\nu_{\mathbf{R}} = \underbrace{\begin{pmatrix} \mathbf{B}_{\mathbf{k}} \\ \mathbf{I} \end{pmatrix}}_{\mathbf{\Gamma}} \nu_{\mathbf{F}}. \tag{3.25}$$

The covariance of the prediction noise yields

$$\mathbf{R} = \mathrm{E}(\nu_{\mathbf{R}}\nu_{\mathbf{R}}{}^{T}) = \mathrm{E}(\mathbf{\Gamma}\nu_{\mathbf{F}}\nu_{\mathbf{F}}{}^{T}\mathbf{\Gamma}^{T}) = \mathbf{\Gamma}\sigma_{\mathbf{w},\mathbf{F}}\mathbf{\Gamma}^{T}. \tag{3.26}$$

We obtain the noise standard deviations by optimizing for $\sigma_{\mathbf{m}}$ and $\sigma_{\mathbf{w},\mathbf{F}}$, minimizing $\sum_{k=0}^{N} \|\hat{\mathbf{F}}_{\mathbf{e}}^{(k)} - \mathbf{F}_{\mathbf{e}}^{(k)}\|_1$, where $\hat{\mathbf{F}}_{\mathbf{e}}$ denotes the ground truth external force and torque values and the superscript $(k)$ refers to the $k^{\text{th}}$ training example, for all $N$ training examples in the *vinyl-motion* training set introduced in Section 3.5.1.

At last, we need to identify the model parameters of the mass-spring-damper system:

- the shell mass $m$,

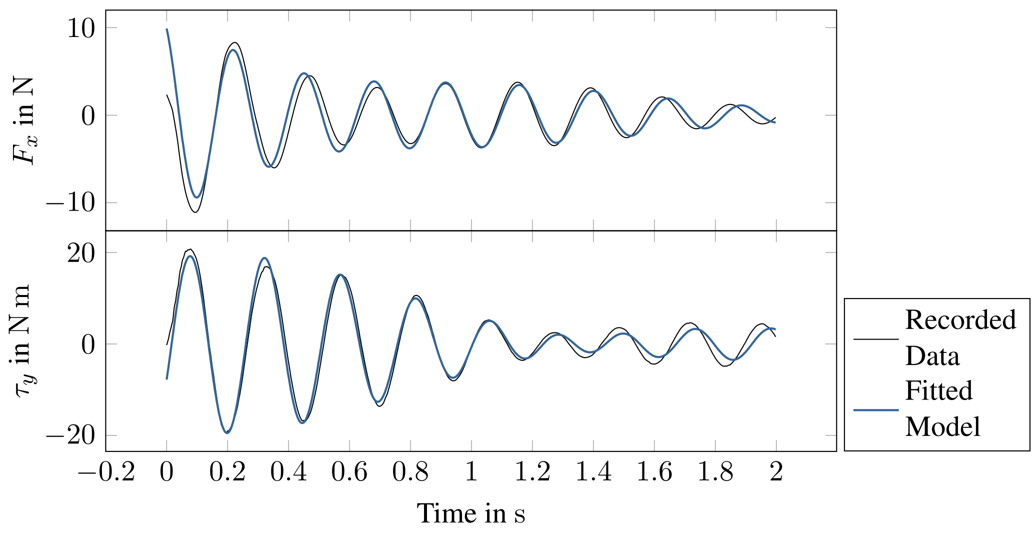

**Figure 3.4:** The recorded system response in $x_l$ and $y_\phi$, in black, for an external force released at $t = 0$ shows the coupling effects between both degrees of freedom. The model response from the fitted model parameters is shown in blue.

- moments of inertia $J_{x/y}$ and $J_z$,
- the distance $d$ between mount and center of gravity,
- and the spring and damping constants $k_{l/\phi}$ and $c_{l/\phi}$.

We can measure the mass directly with the force sensor when the robot is at rest and no external force is present. We calculate the moments of inertia by approximating the shell as a cylindrical barrel with a negligible wall thickness of radius $R$ and height $L$ as

$$J_{x/y} = \frac{1}{2}mR^2 + \frac{1}{12}mL^2, \tag{3.27}$$

$$J_z = mR^2. \tag{3.28}$$

The spring and damping constants and the distance $d$ are estimated from the step response of the system. To this end, we exerted constant forces by pushing on the robot shell and then introduced a step in external force by letting go of the shell. We fit the recorded system response to the harmonic solution of Eq. (3.14), which we solve numerically using the classical Runge-Kutta method. Figure 3.4 shows the identification results for the coupled force $F_x$ and torque $\tau_y$.

## 3.4.2 Neural Network for External Force Estimation

As a second external force filter, we use a model-free approach based on a time-delay neural network (TDNN) [150]. Instead of modeling the system equations explicitly, the

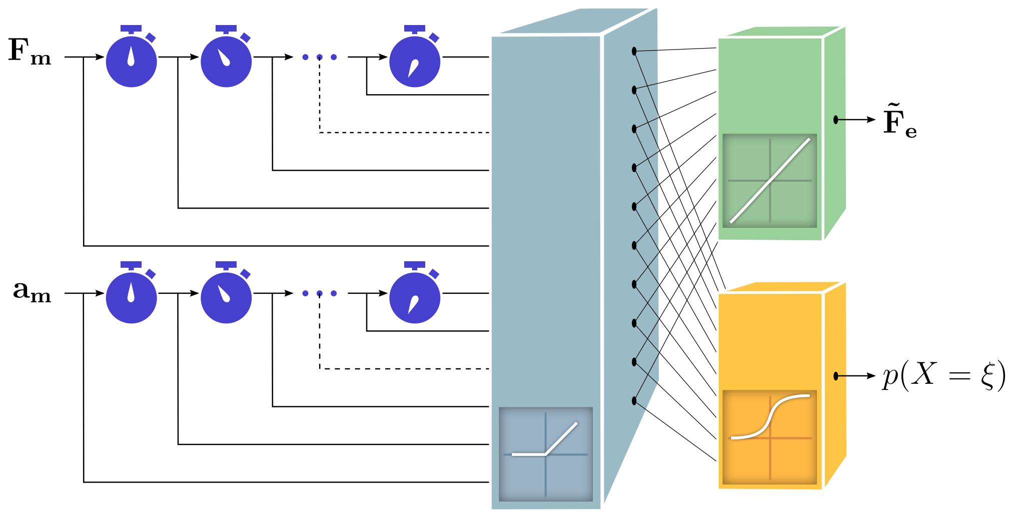

**Figure 3.5:** Time-delay neural network. The inputs are delayed before entering the network, which splits up into one regression and one classification part.

TDNN learns the model from data. TDNNs capture temporal information by showing the network multiple consecutive data points of a time series at one instance. To this end, the discrete input time series is delayed and buffered before entering the network. The TDNN has a feedforward structure and can be trained using standard backpropagation.

We split our problem into a regression and a classification part to leverage the superior performance of neural networks for classification tasks. Thus, our network has two network outputs, as depicted in Figure 3.5. The classification output $p(X = \xi)$ estimates the probability that there is currently an external force acting on the robot shell. The random variable $X$ that specifies whether an external force is present is binary and can take on two values: $X = \{\xi, \bar{\xi}\} = \{\text{force present}, \text{no force present}\}$. The regression output $\tilde{\mathbf{F}}_\mathbf{e}$ estimates the values of the six external force and torque components. We combine both network outputs and calculate the expectancy of $\tilde{\mathbf{F}}_\mathbf{e}$, given the estimated $p(X = \xi)$, as our final prediction

$$\begin{aligned} \mathbf{F_e} &= \mathrm{E}_{p(X)}(\tilde{\mathbf{F}}_\mathbf{e}) \\ &= \tilde{\mathbf{F}}_\mathbf{e} \cdot p(X = \xi) + \bar{\tilde{\mathbf{F}}}_\mathbf{e} \cdot p(X = \bar{\xi}), \end{aligned} \tag{3.29}$$

where the external force without force impact is $\bar{\tilde{\mathbf{F}}}_\mathbf{e} \equiv 0$.

The network input $\mathbf{x}^{(k)} = \left(\mathbf{F_m}^{(k-n:k)} \quad \mathbf{a_m}^{(k-n:k)}\right)^T$ consists of the forces and torques $\mathbf{F_m}$ measured by the force-torque sensor and the accelerations $\mathbf{a_m}$ measured by the IMU, delayed and buffered by $n$ time steps. The input vector passes to the shared part of the network with two fully connected layers of $256$ neurons with non-linear rectified-linear (ReLU) activations each. The network then splits up into separated regression and classification parts of one layer with $128$ neurons each. The regression part consists of neurons with linear activations, while the classification layer uses non-linear sigmoid activations.

The network parameters $\phi$ are optimized for all $N$ training examples using stochastic gradient descent with momentum according to

$$\phi^* = \arg\min_{\phi} \sum_{k=1}^{N} \mathcal{L}\left(\hat{\mathbf{y}}^{(k)}, \mathbf{y}^{(k)}\right), \tag{3.30}$$

where $\hat{\mathbf{y}}^{(k)}$ denotes the ground truth value, or label, of the $k^{\text{th}}$ training example. Our loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ is a multi-task loss comprising the squared Euclidean norm loss of the regression output $\tilde{\mathbf{F}}_{\mathbf{e}}$ and the cross-entropy loss of the classification output $p(\xi)$,

$$\begin{aligned} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) &= w_r \|\tilde{\mathbf{F}}_{\mathbf{e}} - \hat{\mathbf{F}}_{\mathrm{e}}\|_2^2 \\ &- w_c \left[p(\xi)\log(\hat{p}(\xi)) + (1 - p(\xi))\log(1 - \hat{p}(\xi))\right]. \end{aligned} \tag{3.31}$$

The weight factors $w_r$ and $w_c$ are hyperparameters that can be adjusted for the desired regression and classification performance. We found that $w_r = 4w_c$ yields good overall performance.

## 3.5 Experiments

We performed a series of experiments to evaluate the performance of our sensory concept and the force filtering approaches. We conducted experiments with the robot at rest and in motion and assessed the methods in terms of

- the mean absolute error between the estimated force magnitude $\|\vec{F}_{\mathrm{e}}\|_2$ and the ground truth force magnitude $\|\hat{\vec{F}}_{\mathrm{e}}\|_2$,

$$e_{\|F\|} = \frac{1}{N} \sum_{k=1}^{N} |(\|\vec{F}_{\mathrm{e}}^{(k)}\|_2 - \|\hat{\vec{F}}_{\mathrm{e}}^{(k)}\|_2)|, \tag{3.32}$$

- the mean absolute distance between the estimated impact point $\vec{r}$ and the true impact point $\hat{\vec{r}}$ of the external force,

$$e_r = \frac{1}{N} \sum_{k=1}^{N} \|\vec{r}^{(k)} - \hat{\vec{r}}^{(k)}\|_2, \tag{3.33}$$

- and the mean angle between the estimated external force vector $\vec{F}_{\mathrm{e}}$ and the true

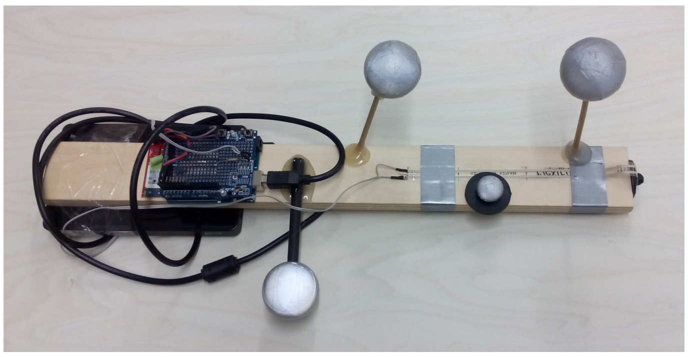

**Figure 3.6:** Force stick used to collect ground truth external force data. The optical markers are used to determine the 6D pose of the stick with respect to the robot.

external force vector $\hat{\vec{F}}_\mathrm{e}$,

$$e_{\angle F} = \frac{1}{N}\sum_{k=1}^{N} \angle(\vec{F}_\mathrm{e}^{(k)}, \hat{\vec{F}}_\mathrm{e}^{(k)}). \tag{3.34}$$

The number of time steps in a sequence is denoted by $N$. Note that we only evaluated the impact point distance $e_r$ and the angular error $e_{\angle F}$ for predicted force magnitudes $\|\vec{F}_\mathrm{e}\|_2 \geq 5\,\mathrm{N}$.

Our experimental setup and the data collection procedure are presented in Section 3.5.1. We compared our model-based and model-free approach to a lowpass filter baseline introduced in Section 3.5.2. Finally, Section 3.5.3 evaluates the performance of all methods when the robot is at rest to when the robot is in motion and compares them for different floor conditions.

## 3.5.1 Experimental Setup

We assembled a force stick depicted in Figure 3.6 to generate the ground truth external force magnitudes. It is based on a flexible 1D pressure sensor attached to a wooden stick. For the data collection, one experimenter repeatedly exerted forces of varying extents at different impact points by pressing the force stick against the robot shell, as depicted in Figure 3.7. A ceiling-mounted motion capture system provided the direction and impact point of the force, when available. We recorded data with the robot in motion and at rest. For the datasets with motion, a second experimenter teleoperated the robot to execute random translational and rotational motions at speeds of $|v| \leq 0.75\,\mathrm{m\,s^{-1}}$ and $|\omega| \leq 1.5\,\mathrm{rad\,s^{-1}}$, respectively. The exerted force magnitudes ranged between $0$ and $\approx 50\,\mathrm{N}$.

We collected one training, one validation, and two test datasets in the motion capture area of our robot hall, which has a vinyl flooring. The training, validation, and test sets *vinyl-motion* were collected with a moving robot, the test set *vinyl-rest* when the robot

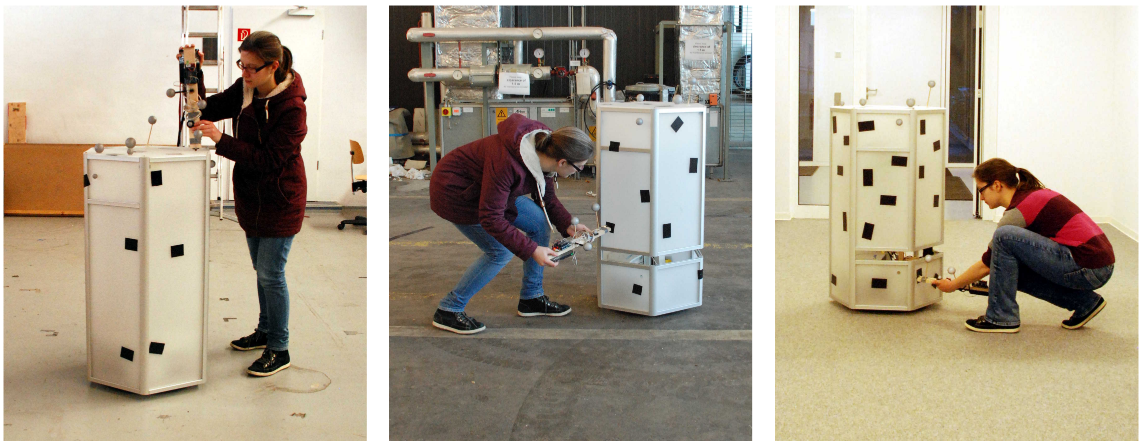

**Figure 3.7:** Data collection on different floor conditions: on vinyl (left), stone (middle), and carpet (right).

**Table 3.1:** Statistics of the collected datasets.

| | total frames | impact frames | impact sequences | $\mu(\lVert\vec{F}_\mathrm{e}\rVert_2)$ in $\mathrm{N}$ | $\mu(\lvert v\rvert)$ in $\mathrm{m\,s^{-1}}$ | $\mu(\lvert\omega\rvert)$ in $\mathrm{rad\,s^{-1}}$ |
|---|---|---|---|---|---|---|
| **training** | | | | | | |
| vinyl-motion | 42,180 | 21,889 | 251 | 17.54 | 0.34 | 0.13 |
| **validation** | | | | | | |
| vinyl-motion | 12,545 | 5774 | 67 | 15.66 | 0.40 | 0.12 |
| **testing** | | | | | | |
| vinyl-motion | 26,334 | 12,912 | 161 | 15.93 | 0.32 | 0.12 |
| vinyl-rest | 32,847 | 15,566 | 163 | 14.78 | 0.00 | 0.00 |
| stone-motion | 13,370 | 6475 | 95 | 16.43 | 0.35 | 0.13 |
| carpet-motion | 10,539 | 4422 | 42 | 9.78 | 0.31 | 0.23 |

was standing still. We further collected two test sets on different floor conditions, *stone-motion* and *carpet-motion*. Those two sets were recorded outside of the motion capture area and hence only provide labels for the external force magnitudes.

Table 3.1 summarizes the characteristic properties of the datasets: The total number of data points and the number of data points with force impact, the number of impact sequences (one sequence means exerting force at one impact location and letting go), the mean external force magnitude $\mu(\lVert\vec{F}_\mathrm{e}\rVert_2)$, and the mean translational and rotational velocities of the robot, $\mu(\lvert v\rvert)$ and $\mu(\lvert\omega\rvert)$. We recorded a total of $137{,}815$ data points at a sampling rate of $50\,\mathrm{Hz}$, corresponding to more than $45\,\mathrm{min}$ of operation.

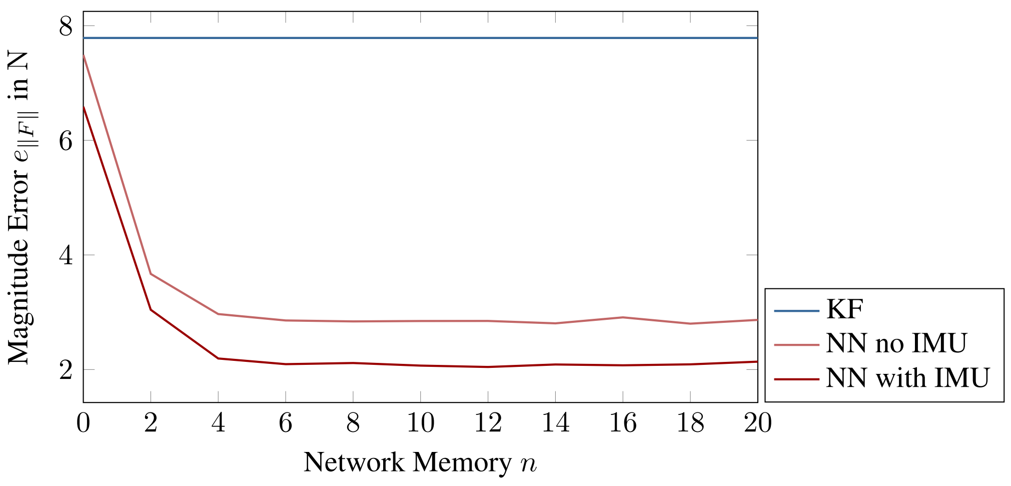

**Figure 3.8:** Mean force magnitude error $e_{\|F\|}$ for the Kalman filter, compared to the neural network filter for different network memories $n$.

### 3.5.2 Baseline Lowpass Filter

Lowpass filters are standard tools for filtering high-frequency sensor noise from measurement data. A finite impulse response (FIR) lowpass filter calculates the signal at time step $t$, $x(t)$, as the weighted mean of the past $M + 1$ measurements,

$$x(t) = \sum_{k=0}^{M} w_k x_m(t - k). \tag{3.35}$$

We designed an FIR lowpass filter by windowing [115] as a model-free force filtering baseline. The filter coefficients $w_k$ were calculated based on a least-squares approximation and a Hamming window [52] for smoothing the impulse response [97]. We filtered all force and torque components of $\mathbf{F_m}$ separately with an individual lowpass filter and kept the same filter order $M$ for all filters, so all force and torque components have the same filter delay. We optimized the cutoff frequencies and the filter order by a grid search on the *vinyl-motion* test set to ensure the best possible performance for the baseline. The cutoff frequencies for the force components were optimized together with the filter order, minimizing the mean absolute force magnitude error $e_{\|F\|}$. The cutoff frequency of each torque component was optimized individually for its mean absolute torque error $e_{\tau,x/y/z} = \frac{1}{N}\sum_{k=0}^{N} |\tau_{x/y/z}^{(k)} - \hat{\tau}_{x/y/z}^{(k)}|$. We compensated for the filter delay during optimization by shifting the signal back by $M/2$ time steps. Thus, the resulting filter is optimal within its capabilities, but the signal will be delayed.

## 3.5.3 Force and Impact Point Estimation

The first experiment compares our model-based Kalman filtering and the model-free neural network force filtering approach. The model-based Kalman filter estimates the combined force acting on the robot shell (see Eq. (3.7)) and does not use acceleration measurements. Therefore, we compared it to the neural network filter with and without IMU information. We used the *vinyl-motion* test set and evaluated the mean absolute force magnitude error $e_{\|F\|}$.

Figure 3.8 shows the performance of the Kalman filter compared to the neural network filter for different network memories $n$. The neural network filters with and without IMU information show a smaller force magnitude error, even for a network memory of $n = 0$ where only the current force and torque measurements are processed. While the network performance drops slightly without IMU information, it still clearly outperforms the Kalman filter. For the following experiments, we use the neural network filter with IMU information and a memory of $n = 10$.

The second experiment evaluates the estimation of the magnitude, direction, and location of external forces on the *vinyl-rest* and *vinyl-motion* test sets. Figure 3.9 shows an excerpt from the ground truth and estimated external force signals with and without force filtering, with and without robot motion. When the robot is at rest, all filters resemble the true external force closely. However, the lowpass filter visibly delays the signal. When the robot is moving, the raw sensor measurements vary significantly from the true external forces. The neural network filter still resembles the external force signal closely, while the performance drops visibly for both other filters. In particular, we can see that the neural network filter reliably estimates a force of $\approx 0\,\mathrm{N}$ when no external force is present between the force impacts, which is essential for stable, oscillation free motion, e.g., during autonomous navigation.

Figure 3.10 compares the performance of all approaches in terms of the mean absolute force magnitude error $e_{\|F\|}$, the mean impact point distance $e_r$, and the mean force vector angle $e_{\angle F}$. While all approaches give comparable results when the robot is at rest, the performance drops drastically when the robot is in motion for the Kalman filter and the lowpass filter baseline. Only the neural network filter can maintain a good filtering performance.

We can only evaluate the impact point distance and angular error for frames with a ground truth impact. Furthermore, we only calculate them for predicted force magnitudes above a threshold of $5\,\mathrm{N}$. All approaches meet this criterion for around $90\,\%$ of frames with ground truth impact, except the lowpass filter, which only selects $\approx 80\,\%$ of frames. Additionally, a calculated impact point is only valid if it lies inside the shell geometry. For a fair comparison, we evaluated the impact point distance and force angle error only on frames that meet the $>5\,\mathrm{N}$ criterion and are valid for all approaches. Consequently, $65.7\,\%$ of frames with ground truth impact at rest and $55.5\,\%$ in motion are evaluated for

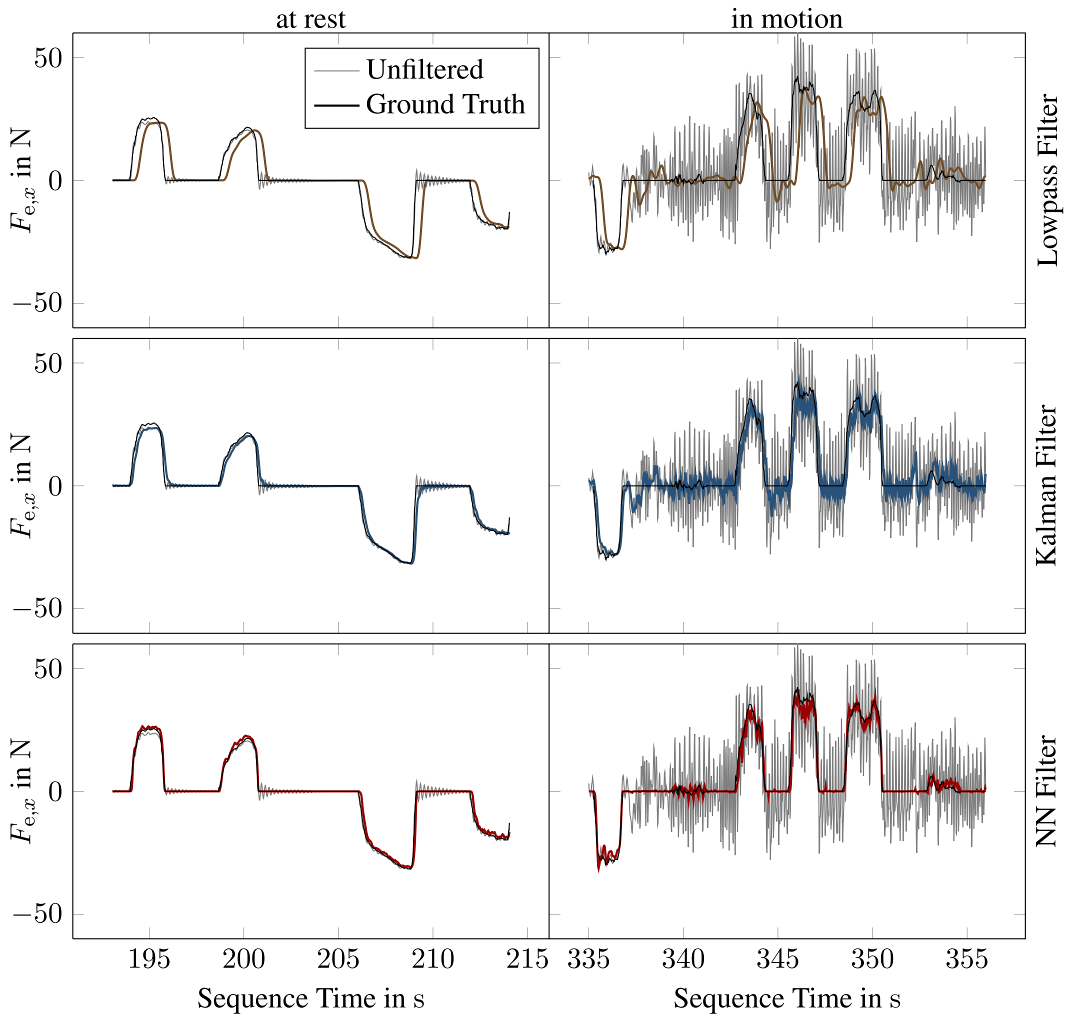

**Figure 3.9:** Unfiltered, estimated, and ground truth external force $F_{e,x}$ along the $x$-axis of the robot (pointing forward), at rest and in motion, for the lowpass filter baseline, the Kalman filter, and the neural network filter.

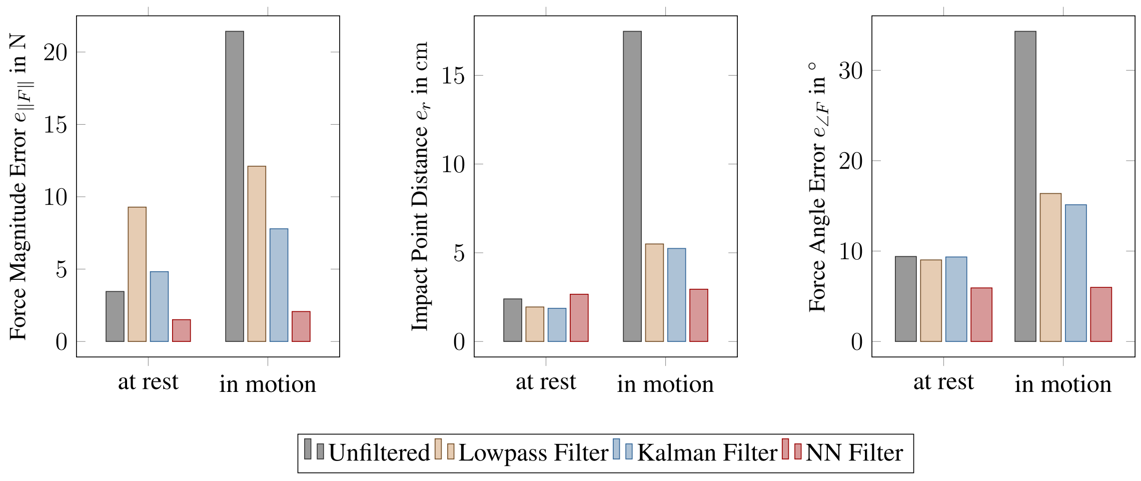

**Figure 3.10:** Mean absolute magnitude error $e_{\|F\|}$, mean impact point distance $e_r$ and mean force angle error $e_{\angle F}$ for the unfiltered case and for the lowpass filter, Kalman filter, and our neural network approach. The left columns show the errors for when the robot is at rest, the right columns for when the robot is in motion.

**Table 3.2:** Amount of predicted impact points that are invalid because they lie outside of the shell geometry, in $\%$.

| | Unfiltered | Lowpass Filter | Kalman Filter | NN Filter |
|---|---|---|---|---|
| rest | 8.74 | 9.21 | 9.05 | **3.42** |
| motion | 21.87 | 11.91 | 13.22 | **3.02** |

the impact point distance. For the force angle error, $76.1\,\%$ of the frames are evaluated at rest and $77.3\,\%$ in motion. Table 3.2 summarizes the percentage of invalid impact points for all approaches. The neural filter outperforms all other methods in its ability to predict a valid impact point.

The last experiment evaluates the sensitivity of the force filters to varying floor conditions. The floor condition likely influences the base excitation from the motion of the robot, which is the main cause of force measurement errors with our setup. In addition to the vinyl flooring of our motion capture area, we collected and evaluated data on a stone floor and carpet, as depicted in Figure 3.7. Table 3.3 presents the mean absolute force magnitude error $e_{\|F\|}$ for the *vinyl-motion*, *stone-motion* and *carpet-motion* test set. Since we collected the stone and carpet datasets outside the motion capture area, we do not have ground truth force directions and impact points and cannot compare the impact point distance $e_r$ and the force angle error $e_{\angle F}$. The stone and carpet datasets were not used for training the approaches.

Interestingly, the lowpass filter and our model-based Kalman filter perform better on carpet than on the vinyl floor for which the approaches have been adjusted. Only our

**Table 3.3:** Mean absolute force magnitude error $e_{\|F\|}$ for different floor conditions, in N.

| | Vinyl | Stone | Carpet |
|---|---|---|---|
| Unfiltered | 21.24 | 30.04 | 20.00 |
| Lowpass Filter | 12.52 | 15.93 | 8.29 |
| Kalman Filter | 8.02 | 10.12 | 7.02 |
| NN Filter | **2.06** | **2.74** | **2.66** |

neural network filter performs slightly worse on carpet than on vinyl, suggesting that it implicitly learns the floor characteristics. Nevertheless, it outperforms all baselines on all tested floor conditions. The results show that our neural network filter generalizes well to previously unseen floor conditions, confirming that our sensory concept can successfully be employed outside our robot hall.

## 3.6 Conclusion

In this chapter, we presented a sensory concept for perceiving interaction forces exerted onto the shell of a mobile robot. Further, we introduced a neural network-based filtering technique and a model-based baseline to distinguish external forces from those stemming from oscillations or force stimuli introduced by robot motion. We carried out extensive experiments with our mobile robot Canny. They demonstrate that our sensor concept enables the robot to precisely estimate the magnitude, location, and direction of the impact force, even when the robot is in motion or deployed on different floor conditions. Our neural network filtering technique outperforms the model-based approach and a lowpass filter baseline; in particular, it can better handle the chaotic oscillation behavior of our robot in motion. The following two chapters discuss two applications where knowledge about contact or interaction forces enables a mobile robot to adapt to unstructured and populated environments.

# Chapter 4

# Learning Obstacle Footprints From Collisions

**Mobile robots in human-centered environments need to handle the unstructured and cluttered nature of their surroundings appropriately. At the same time, incomplete sensor information is characteristic of mobile indoor robots due to cost and computation limitations. Many mobile indoor robots use horizontally scanning 2D laser rangefinders for localization because of their high accuracy and the compactness of the resulting 2D maps. As the scanners in this configuration only provide information about one slice of the environment, the measurements typically do not capture the full extent of a large variety of obstacles, including chairs or tables. Accordingly, obstacle avoidance based on planar laser rangefinders can fail. In this chapter, we propose a novel learning-based method to predict collisions in 2D occupancy maps. Our method uses a convolutional neural network trained on 2D occupancy map patches and collision events recorded while the robot is navigating in its environment. As the network operates on local structures only, it can generalize to new environments. In addition, the robot can collect and integrate new collision examples after an initial training phase. Extensive experiments in simulation and in a realistic real-world environment confirm that our approach allows robots to learn from collision events to avoid collisions in the future.**

## 4.1 Introduction

Robots are increasingly employed in spaces designed for and populated by humans. These environments are typically cluttered and may change over time. At the same time, we demand rising autonomy levels and decreasing costs from personal robots and

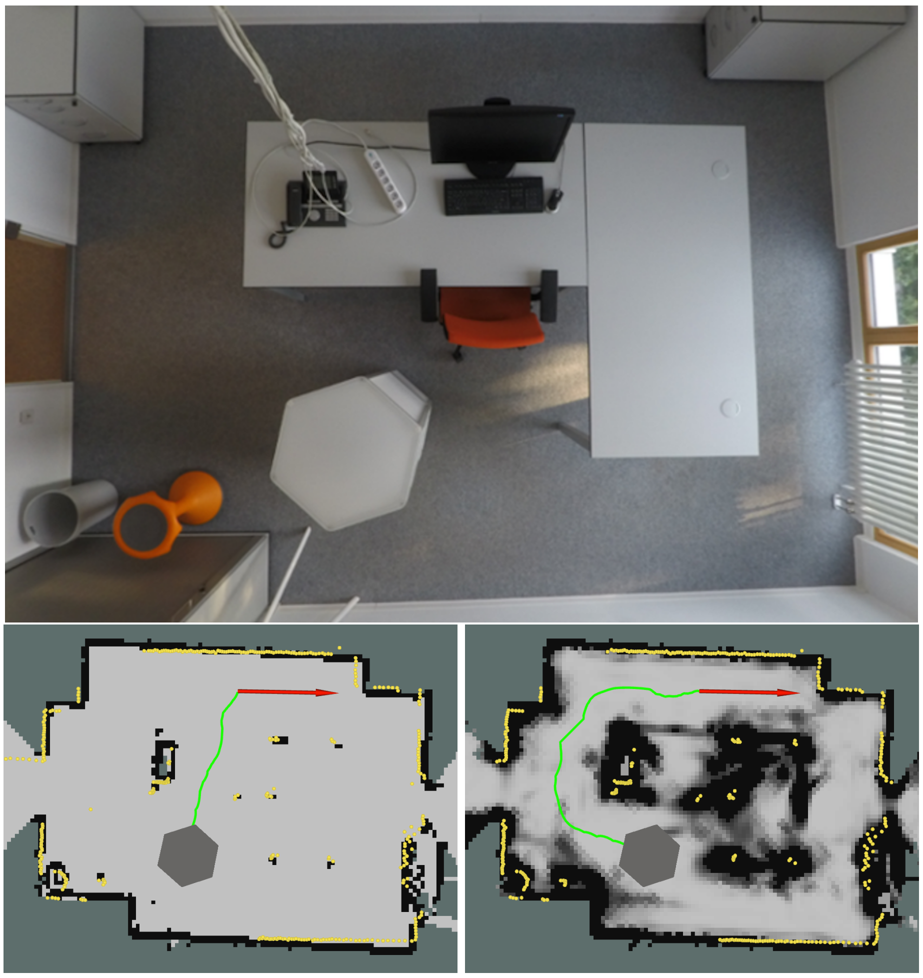

**Figure 4.1:** The top image shows a typical indoor office environment with a large desk. The images at the bottom display the occupancy map, the laser rangefinder data in yellow, the navigation goal as a red arrow, and the planned path in green. The 2D occupancy map from laser data is shown on the bottom left image. The planned path leads to a collision with a part of the table not visible to the laser rangefinder. The right image shows the collision map learned with our approach with a planned path successfully avoiding the table.

robots employed in public spaces. This chapter presents a novel approach to predicting collisions with parts of objects that cannot be perceived by a mobile robot due to the specific mounting of its sensors. Substantial parts of the ideas, results, figures, tables, and text presented in this chapter have been previously published in [75], ©2020 IEEE, `https://doi.org/10.1109/ICRA40945.2020.9197474`. Section 1.4 outlines the author's contribution to this work.

Throughout this chapter, we focus on horizontally scanning 2D laser rangefinders, which are popular for autonomous indoor navigation. Horizontal laser rangefinders only perceive one slice of the environment and, therefore, often miss large parts of obstacles.

Accordingly, the resulting 2D occupancy maps typically do not correctly resemble the free space in the environment. One example is depicted in Figure 4.1. As large parts of the tables in the room are not visible to the laser rangefinder, many planned navigation paths based on the 2D occupancy map lead to collisions.

One possible solution is to use additional sensors, such as RGB-D cameras or 3D Lidar sensors. However, additional sensors increase the computational cost and power consumption, and most cannot reliably perceive all obstacles either, such as glass surfaces. Collision sensors, such as bumpers, provide a reliable method for sensing collisions outside the planar perceptive field of laser rangefinders. While they are cheap and have very low computational and power requirements, they cannot prevent collisions; the robot can only react to them. Also, inferring and maintaining occupancy from bumpers is not straightforward if the environment changes over time since occupied space is never updated if the robot avoids it.

Chapter 3 introduced our novel sensory concept allowing mobile robots to perceive interaction forces and contacts with the environment during autonomous navigation. In this chapter, we show how the perception of collisions with the environment can enable mobile robots to learn improved environment models. To this end, we propose a novel learning method to predict collisions in 2D occupancy maps. Our approach is trained on collision events recorded with a collision sensor, in our case, with the force-sensitive robot shell presented in Chapter 3. We employ a convolutional neural network for predicting map areas that will likely result in a collision (see Figure 4.1). With our approach, the robot can anticipate collision areas and plan paths to avoid them. We introduce a collision dataset recorded with a simulated robot in diverse simulated indoor scenes, which we used to train an initial collision prediction model. We further demonstrate how the robot can combine simulated data with collision examples from real-world scenarios to train models for new environments.

Previous approaches have been proposed for predicting object parts outside the sensor's field of view in the context of 3D scene completion [129], 3D object reconstruction [145], occlusions[107], or for predicting laser images from obstacle footprints [93]. These approaches all relied on additional sensors for training that were removed after the training phase. Thus, the learned models remained fixed when employed on a robotic system and could not adapt to environments and object arrangements insufficiently represented in the training data. Our approach is the first that allows robots to integrate new training examples after an initial training phase since they can collect collision examples with their on-board sensors in a self-supervised fashion. To this end, we propose a novel network structure that allows training on single collision events and fast, efficient, high-resolution processing of full occupancy maps. With our method, mobile robots can adapt to new environments by learning better models over time to represent them. Our dataset and the code of our method are available at `https://github.com/marinaKollmitz/learn-collisions`.

## 4.2 Related Work

Robotic systems rely on adequate models of the environments they interact with to operate safely and efficiently. However, sensor data is noisy and typically incomplete. Therefore, much research is concerned with anticipating and predicting missing sensor information, e.g., 3D semantic scene completion [129] and 3D object reconstruction [145]. Another example is the work of Ondrúška and Posner [107], who predicted occluded objects from laser rangefinder data.

In the context of mobile robot navigation, Burgard *et al.* [19] presented an approach for laser rangefinder-based obstacle avoidance in the presence of partially invisible obstacles. The robot avoided collisions with parts of obstacles that the sensors could not perceive by using an adapted version of the dynamic window approach [41], $\mu$DWA [42]. However, a full model of the environment, including all obstacles, had to be provided. Axelrod *et al.* [5] considered robot navigation in the presence of obstacle uncertainty, but their approach is not suitable when large parts of the obstacles are invisible to the sensor.

Thrun [140] used a fully connected neural network to estimate the occupancy in maps using sonar sensor readings as inputs. They also used simulated data to train their model, but the approach was also not designed for cases where large parts of the obstacles are invisible. Guizilini and Ramos [49] learned to reconstruct 2D and 3D structures for occupancy mapping. They trained a convolutional variational autoencoder to recover data gaps, but they did not consider partially detectable objects.

Various works used additional sensors for obstacle avoidance, such as RGB-D sensors [94, 106]. Baltzakis *et al.* [9] fused laser and visual data to perceive parts of objects which were not visible in the laser data alone. The approaches rely on additional sensors with high computational and power demand while we learn from collision events. Plagemann *et al.* [111] used Gaussian process classification and regression techniques to detect collisions with unseen obstacles. While their approach can perceive collisions when they occur, it cannot anticipate them beforehand.

The work by Lundell *et al.* [93] is the one most closely related to ours. The authors used a fully convolutional autoencoder to predict laser rangefinder readings with true obstacle distances from 2D laser scans. Their later work [148] integrated the processed laser scans into occupancy maps with uncertainty estimation. The ground truth obstacle distances for training were collected with a 3D camera. The authors showed that their approach could avoid collisions with obstacles in realistic navigation scenarios. However, it relies on an additional sensor for generating the training examples that is removed after the training phase. In contrast, our method allows us to integrate further training examples after the initial training phase and adapt to the environment over time.

Our approach relies on image segmentation techniques to process 2D occupancy maps efficiently. Image segmentation, especially semantic segmentation, is an active research field in which convolutional neural networks have caused large performance gains over

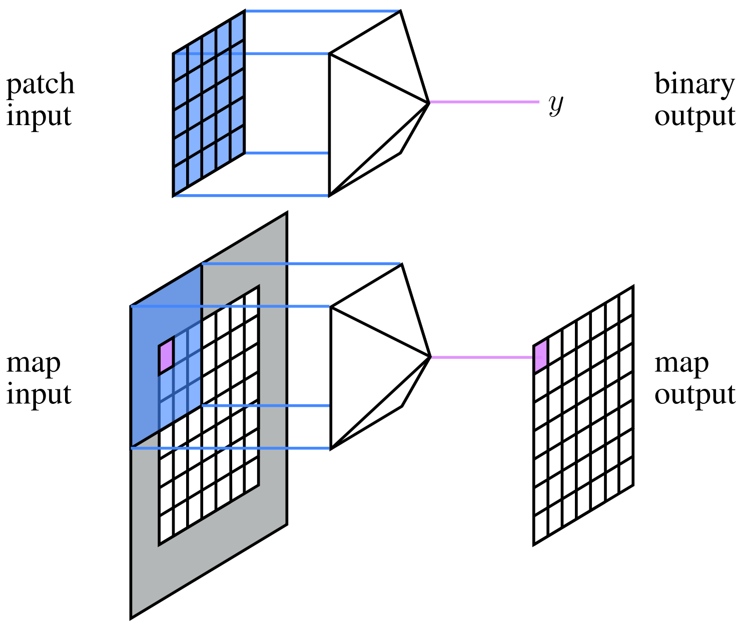

**Figure 4.2:** Our neural network structure can be used for binary classification on image patches (top). During inference on map inputs, the network performs image segmentation and outputs full resolution maps (bottom).

the last years [6, 23, 89, 110, 112, 153, 157]. We predict collisions based on 2D occupancy maps, which differ from usual image data because of their small size and resolution. Because of the already small spatial resolution, we want to avoid further downsampling of the occupancy maps. Various approaches aim to maintain a large output resolution [110, 112]. Dilated convolutions [153] have been used in semantic segmentation to reduce downsampling in segmentation networks [23, 153], like in our work. We base our approach on the Fully Convolutional Network (FCN) model [89] and use dilated convolutions to eliminate output downsampling. The FCN model uses convolutionalized versions of CNNs originally designed for image classification. The FCN paradigm is well suited for our approach because we can train our model as a classification network on binary collision events and afterward apply it efficiently for segmentation, as described in the next section.

## 4.3 Learning Collision Maps

Predicting collisions in 2D occupancy maps amounts to segmenting them into collision space and free space. We aim at achieving this by learning from binary collision events recorded by the robot in a self-supervised fashion. Our method is based on a neural network suited for binary classification and image segmentation, as depicted in Figure 4.2. When applied to image patches, the network output is scalar and binary. We can hence

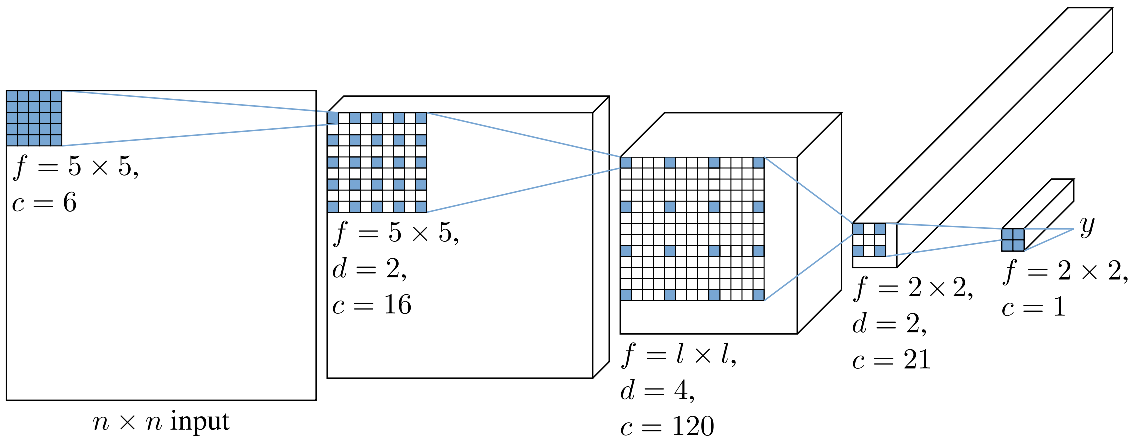

**Figure 4.3:** Dilated convolution layers replace convolution and pooling in our network structure that we adapted from LeNet. The kernel size $f$, dilation factor $d$, and the number of filters $c$ are specified for each layer. The dilated layers enlarge the receptive field and reduce the number of parameters.

train the network on binary collision events, using occupancy map patches as input. When applied to full occupancy maps, the network efficiently slides over the input and outputs segmented collision maps.

To achieve a network that can perform both classification and segmentation, we follow the fully convolutional network (FCN) paradigm presented by Long *et al.* [89]. They proposed to convolutionalize the fully connected layers of CNNs initially designed for classification. To this end, the fully connected layers are replaced by equivalent convolution layers with kernels that span over all input neurons of the layer. As fully convolutional structures, FCNs can efficiently convolve arbitrary-sized inputs and thus segment them into class heat maps, as visualized in Figure 4.2 (bottom). For input patches that exactly match the network input shape, FCNs output class predictions like the original classification CNNs (Figure 4.2 top). Image segmentation with FCNs is equivalent to patch-wise processing of the input image, but it is much more efficient because computations are shared over overlapping patches.

FCNs typically downsample the input image due to striding in the network, e.g., by pooling layers. However, such a decrease in resolution is problematic for our use-case because we already operate on low-resolution occupancy maps. In the original FCN paper, Long *et al.* [89] proposed to add deconvolution layers with skip connections for upsampling the class heatmaps. However, the resulting structure cannot be trained as a binary classifier anymore. Instead, we replace the pooling layers with dilated convolutions [153] to maintain full map resolution during inference. Dilated convolutions have spaces between the kernel neurons. The dilation factor determines the number of skipped neurons. They span larger input regions than conventional convolution filters with the same number of neurons. Thus, they increase the receptive field in the same way as pooling layers, but they do not downsample the input.

Our network architecture is visualized in Figure 4.3. We based our architecture on the LeNet model [82] because it is fast and showed superior performance on the MNIST handwriting dataset [82], which is similar to our type of data. The architecture is a fully convolutional version of LeNet with dilated convolutions instead of pooling. Furthermore, we use sparse connectivity throughout the network to keep the number of parameters similar to the original LeNet.

The first convolutional layer is equal to the original LeNet; the second uses a dilation of 2. Without the pooling layers, the feature map size after the convolutional part is larger than in the original LeNet. Therefore, we use a convolution filter with a dilation of $4$ after the second layer. The filter size of the third layer $l \times l$ depends on the size of the network input $n \times n$ and is calculated by $l = {}^{n}\!/_{4} - 3$. Note that the feature map of the original LeNet after the convolutional part is also $l \times l$. Thus, the dilated layer has the same number of parameters. The last two layers of our network are independent of the input shape. Note that we add dilations and reduce the number of convolution channels compared to the original LeNet to keep the number of parameters similar.

Our architecture can classify collisions and segment maps at the full input resolution. In the following, we will explain in more detail how it can be trained on collision events and how it segments full occupancy maps.

### 4.3.1 Learning From Collision Events

We train our network to classify patches of the occupancy map, as visualized in Figure 4.4. Each input patch is centered around a map pose $\mathbf{S} = (x, y, \theta)^{\mathrm{T}}$, and the network outputs the probability that the map pose $\mathbf{S}$ is in collision as $m_{\mathbf{S}} = p(\mathbf{S}) \in \{0, 1\}$.

The parameters $\phi$ of the network are optimized using stochastic gradient descent according to

$$\phi^* = \arg\min_{\phi} \sum_{k=1}^{N} \mathcal{L}\left(\hat{m}_{\mathbf{S}}^{(k)}, m_{\mathbf{S}}^{(k)}\right). \tag{4.1}$$

The number of training examples is $N$, and $\hat{m}_{\mathbf{S}}$ denotes the collision label that specifies whether the robot perceived a collision at pose $\mathbf{S}$ from which the input patch was generated. The loss function $\mathcal{L}$ is a weighted cross-entropy loss,

$$\mathcal{L}\left(\hat{m}_{\mathbf{S}}, m_{\mathbf{S}}\right) = w \cdot \hat{m}_{\mathbf{S}} \log m_{\mathbf{S}} + (1 - \hat{m}_{\mathbf{S}}) \log(1 - m_{\mathbf{S}}). \tag{4.2}$$

To increase the impact of the less frequent positive collision examples, we used a weight of $w = 2$. We implemented the collision networks in PyTorch and trained them using stochastic gradient descent with an initial learning rate of $0.025$, reduced by half after each epoch. We trained for $10$ epochs on minibatches with $32$ examples.

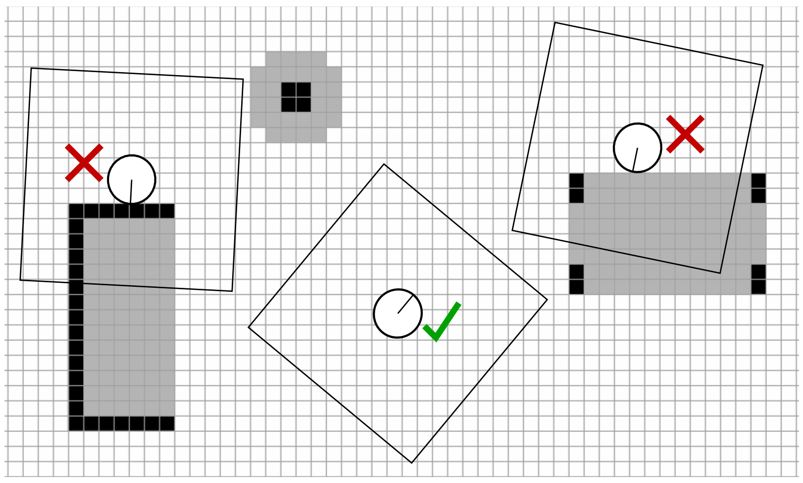

**Figure 4.4:** Our collision classifier takes patches of the 2D occupancy map to classify map poses into collision (red cross) and free space (green checkmark). The gray areas indicate parts of the obstacle footprints not visible in the 2D occupancy map that are to be predicted by our method.

### 4.3.2 Segmenting 2D Occupancy Maps

Due to the fully convolutional structure, the trained collision classification network can segment full 2D occupancy maps without further modifications. When applied for segmentation, we first need to pad the input image as visualized by the gray border in Figure 4.2 to account for the shrinkage at the image borders caused by the network size itself. Since the collision poses for training included orientation information, we process eight equidistant orientations for each occupancy map. To this end, we rotate the map by $\theta_j = j \cdot 45°, j = \{0, ..., 7\}$, for predicting the collision map $m_{\theta,j}$. We then take the maximum predicted collision probability for each cell as the final map value

$$m = \max_j m_{\theta,j}. \tag{4.3}$$

## 4.4 Experiments

We devised a set of experiments to evaluate the ability of our network to predict collisions in 2D occupancy maps. Section 4.4.1 introduces our simulated collision dataset. The classification and segmentation capabilities are reviewed in Section 4.4.2 and Section 4.4.3, respectively. Finally, Section 4.4.4 shows a real-world scenario in which a robot combines simulated data with new collision events to train updated environment models.

## 4.4.1 Simulated Collision Dataset

We used the SceneNet [53] synthetic indoor scenes dataset to train and test our approach. The SceneNet dataset consists of 3D models of 59 indoor scenes from 5 room categories with different furniture items and room layouts. We removed a total of 7 rooms from our set because of missing furniture parts or different model scales and divided the remaining into 9 rooms for testing, 9 for validation, and 34 for training.

We used a simulated version of our robot Canny to explore the simulated environments and collect collision examples. Like the real version, the simulated Canny can detect collisions with a force-sensitive shell, as presented in Chapter 3. To collect the collision examples, we teleported the robot to random poses in free areas of the simulated environments and drove it forward until it encountered a collision. We then saved the pose of each perceived collision as a positive collision example. During collision-free motion, we saved non-collision examples at regular time intervals of $1\,\mathrm{s}$. Here, we selected the front corner of the robot and sampled one configuration inside the robot footprint. We collected a total of $206{,}532$ examples from the train and validation rooms, which corresponds to almost $33$ hours of exploration.

We generated ground truth collision maps by marking whether the cells intersect with the simulated environment models. Intersections were checked with the simulation physics engine. Note that collisions only occur at the object borders. Therefore, we performed a wavefront exploration from the free space inside each environment to find and mark the object borders as in collision. Areas inside objects or outside the simulated environment models do not have a valid label and are ignored for evaluation. We further generated 2D occupancy maps for each simulated scene. To create realistic occupancy maps, we simulated noisy laser rangefinder beam arrays and integrated all measurements using the counting model [51] to produce maximum-likelihood grid maps. Example simulated environments, together with the generated occupancy maps and ground truth collision maps, are depicted in Figure 4.5.

## 4.4.2 Evaluation of Collision Classification

The first experiment evaluates the performance of our binary classifier on our simulated collision dataset. We performed 3-fold cross-validation on the training set. The leave-out set is not part of the training, but it is composed of collision examples collected in the same environments used for training. We will refer to the leave-out set as the *known env* set. The final model is trained on the entire training set and tested on the validation set. The validation set examples stem from unseen environments, referred to as *unknown env*.

To analyze the influence of the network receptive field on the classification performance, we varied the network input sizes during training and testing between $20 \times 20$ and $100 \times 100$ pixels, corresponding to $1 \times 1\ \mathrm{m}^2$ and $5 \times 5\ \mathrm{m}^2$ at $0.05\,\mathrm{m}$ map resolution.

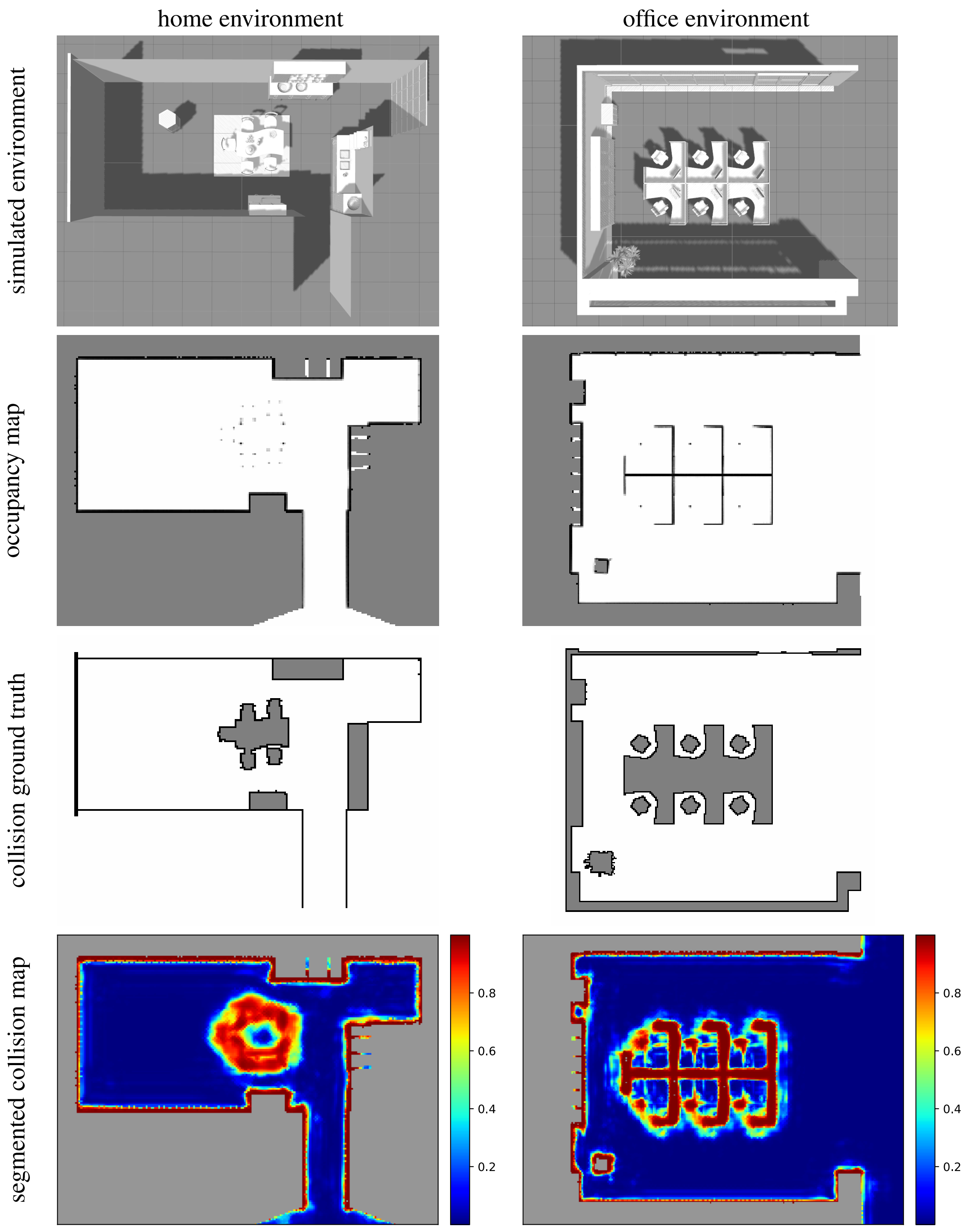

**Figure 4.5:** Example simulation environments used for data generation with occupancy and ground truth collision maps. The bottom row shows the resulting collision maps segmented by our approach, where the colors indicate the predicted collision probability.

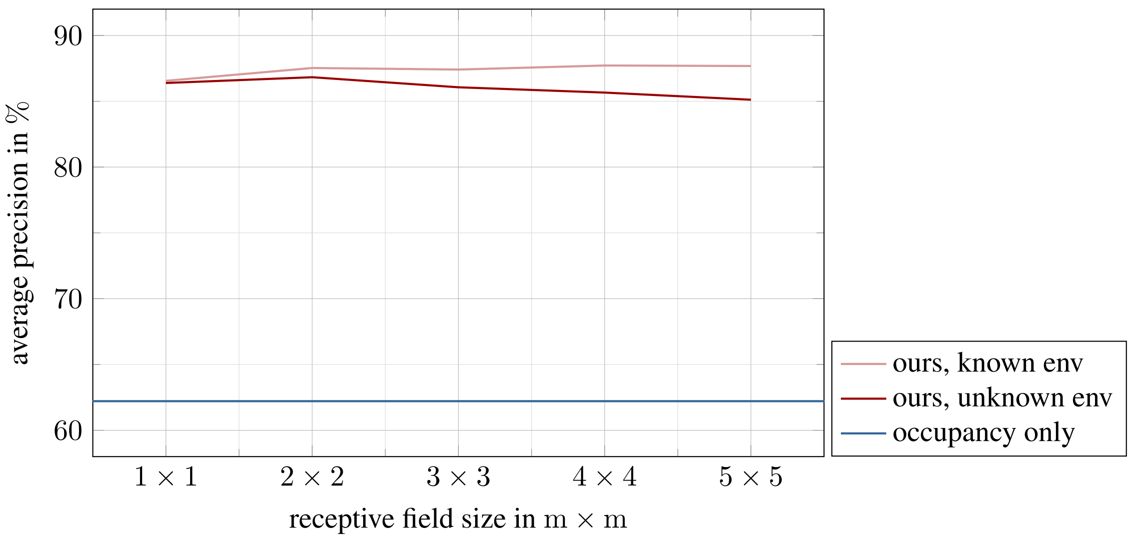

**Figure 4.6:** Collision classification performance in terms of average precision for varying receptive field sizes for our approach, compared to using the occupancy map information alone.

Note that the receptive field size does not influence the prediction resolution since the network always estimates only the center pixel (Figure 4.2). We compare our network performance to the naive occupancy-only baseline, where we classify a pose as collision if the corresponding cell in the occupancy map is occupied and as free otherwise.

Figure 4.6 shows the performance of our approach and the occupancy-only baseline in terms of average precision for varying network input sizes. Even for small receptive fields, our method outperforms the baseline by a large margin. This confirms that our approach can learn to predict collisions in 2D occupancy maps from collision events. For the *unknown env* set, the performance slightly drops for receptive fields larger than $2 \times 2$ $\mathrm{m}^2$. The drop could indicate that collision examples that rely on local features captured by small receptive fields are similar between the known and unknown environments. However, more complex examples, captured only by larger receptive fields, differ between the sets. Therefore, a larger receptive field can cause a loss in performance by overfitting to specific environments. The gain in performance with larger receptive fields on the *known env* set, where the network can exploit more information from known arrangements of objects, also supports this hypothesis.

The receptive field choice depends on the environment complexity and is always a trade-off between performance and inference speed. In the following, we will show results for the $3 \times 3$ $\mathrm{m}^2$ receptive field network only.

**Table 4.1:** Collision map segmentation performance in terms of precision, recall, average precision, and map processing time.

| | prec. | recall | AP | GPU time in s |
|---|---|---|---|---|
| occupancy only | 87.90 | 53.27 | 49.31 | - |
| LeNet + patch classification | 34.47 | 82.22 | 51.70 | 181.07 |
| FCN LeNet + bilinear upsample | 31.06 | 81.40 | 35.50 | 0.0825 |
| dilated FCN LeNet (ours) | 33.32 | 82.33 | 55.66 | 0.3118 |

### 4.4.3 Evaluation of Occupancy Map Segmentation

The second experiment evaluates the ability of our approach to segment occupancy maps. We test on the occupancy maps in the *test* split, which was not part of the training set. To evaluate the collision map reconstruction performance, we calculate the pixel-wise precision and recall and regard a cell as in collision if the predicted collision probability is larger than $0.5$. We further compare the average precision score and the runtime on an Nvidia Titan Black GPU, normalized for a $100 \times 100$ pixel input. The ground truth collision maps were generated from simulation, as described in Section 4.4.1.

As before, we compare our approach to the occupancy-only baseline, where the collision probability of each cell corresponds to its occupancy value. Furthermore, we evaluate the map processing performance of other network variants to test the impact of our network adaptations from the original LeNet. We first compare our approach to the original LeNet with patch processing. Instead of convolving the network over the input image, all cells are processed individually by sampling image patches. The second variant is a fully convolutional version of the original LeNet, including the pooling layers. The pooling layers downsample the input image by a factor of $4$. To compensate for the downsampling, we performed bilinear upsampling to yield the full map resolution. Note that we did not consider larger networks like AlexNet [77], VGG [127], or ResNet [54] because their large input dimensions make them unsuitable for our low-resolution map data. Furthermore, they strongly downsample the input due to the many pooling layers.

Table 4.1 shows the map segmentation performance. All architecture variants show a drop in precision but a notable improvement in recall compared to the occupancy-only baseline. The occupancy-only baseline represents the pixels that are definitely in collision, resulting in a high precision score. However, the low recall shows that it misses large parts of the actual collision space. The networks tend to overestimate the collision space, hence the drop in precision, but identify many previously missed collision areas, as shown by the higher recall. The LeNet variant with patch classification performs comparably to our approach, but the runtime is orders of magnitude higher. In contrast, the fully convolutional LeNet yields very fast processing speeds. However, the down- and upsampling causes a notable segmentation performance drop. Our approach combines both fast

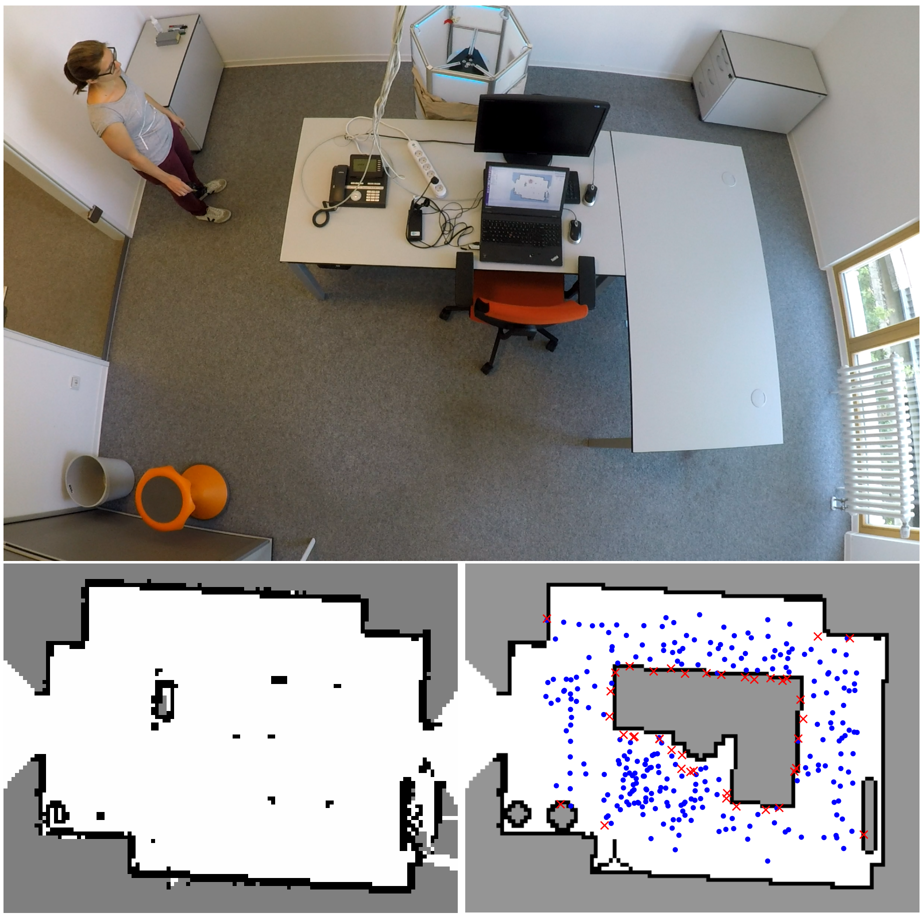

**Figure 4.7:** Top: Office environment for the real-robot experiment. Our force-sensitive robot Canny is collecting new collision examples for improving the environment model. Bottom: 2D Occupancy map (left) and ground truth collision map (right) with recorded collision examples as red crosses and free space examples as blue dots.

processing speeds with high segmentation scores and even outperforms patch classification in terms of average precision. Figure 4.5 (bottom) shows two example collision maps generated by our method for two simulated environments.

This experiment confirms that our approach can process occupancy maps fast and that it successfully identifies areas in the map where collisions may occur. However, we can see that our method is sometimes too conservative and classifies areas as in collision that are actually in free space. For navigation, we argue that overcautious obstacle avoidance is preferable to risking collisions. However, overestimating occupancy can lead to incomplete path planning in tight spaces. Future work should investigate exploration/exploitation strategies so the robot can carefully test and confirm if map areas are actually occupied.

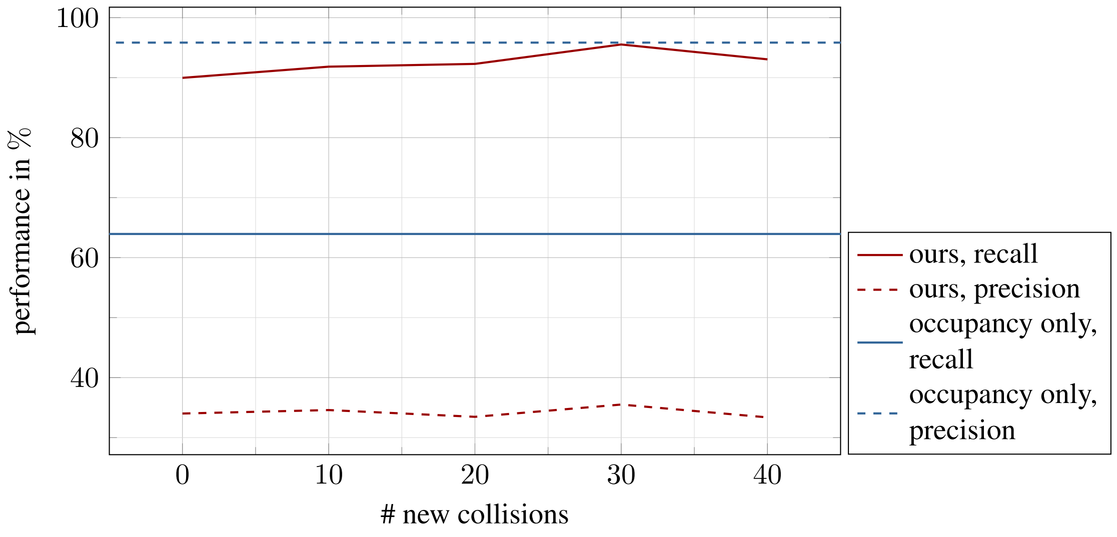

**Figure 4.8:** Map segmentation performance for the real office environment vs. number of integrated new collision examples.

### 4.4.4 Real Robot Experiment With Canny

Finally, we show the performance of our network for a real-robot scenario. Our force-sensitive robot Canny operated in the previously unseen office environment depicted in Figure 4.7. Also shown are the occupancy map and the manually annotated ground truth collision map. The setting is challenging because a large portion of the objects is not visible in the 2D occupancy map: e.g., only the table legs are visible. This experiment evaluates both the ability of our approach to generalize from simulation to the real robot and the retraining performance on new collision examples. For retraining, we collected 40 new collision examples by colliding the robot with the room furniture. We first trained our approach on the training set from simulation and iteratively retrain with the new collision examples. We added the new collision examples to the training set and retrained the model from scratch, as described in Section 4.3.1. To give them more weight, we added 100 copies of the new collision examples to the training set.

Figure 4.8 shows the performance of our approach after retraining with varying numbers of collision examples, compared to the occupancy-only baseline. The collision maps segmented by our method after integrating 0, 10, and 40 collision examples are visualized in Figure 4.9. As in the previous experiment, we can notice a drop in precision for our approach compared to the baseline, but the recall improves by a large margin. Without the new collision examples, we already see a greatly improved recall, which means that the robot can anticipate a large fraction of the collisions in the new environment with the model trained in simulation. The recall further improves with a rising number of integrated collision examples, which means that the approach can improve its model with the new examples. Figure 4.9 shows qualitatively that the robot learns an improved model:

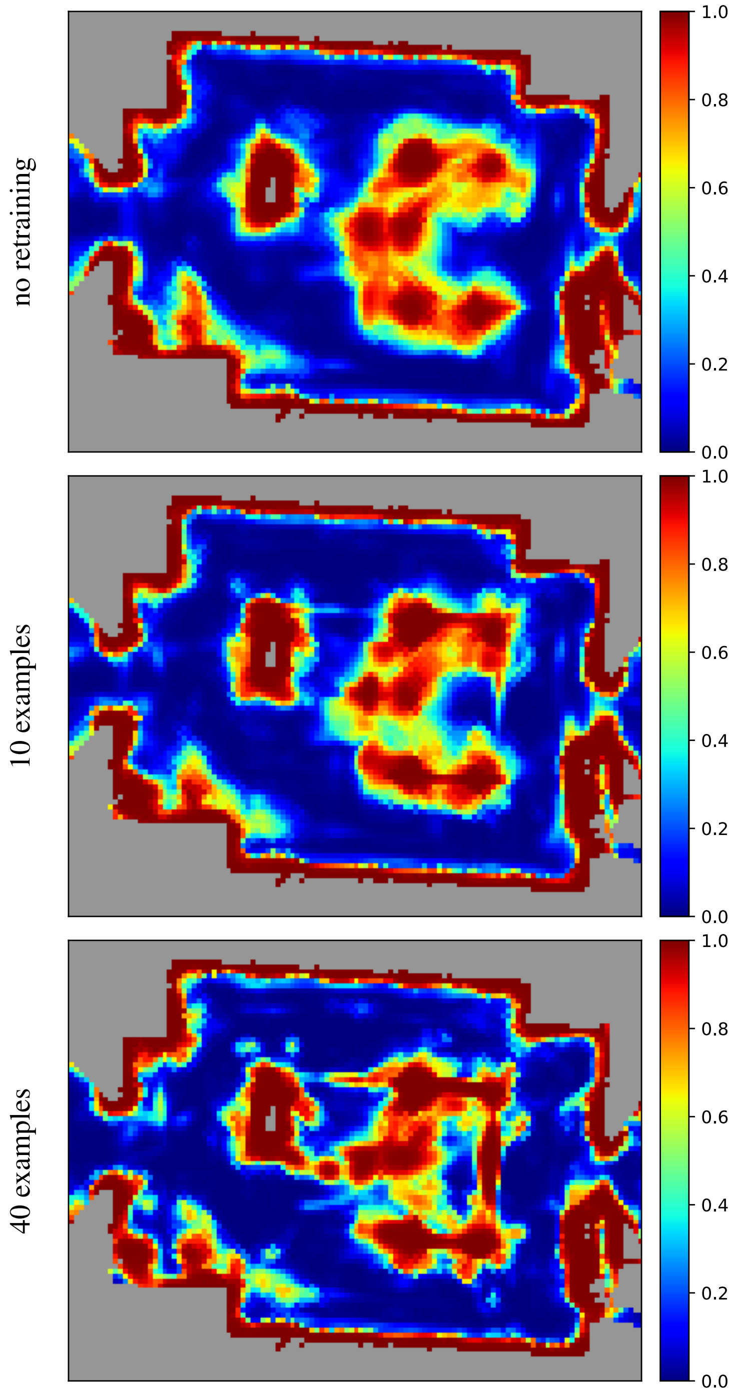

**Figure 4.9:** Processed collision maps before retraining and after integrating 10 and 40 collision examples from the new environment. The colors indicate the predicted collision probability.

The contours of the previously unseen objects are more and more visible with an increasing number of new collision examples. Figure 4.1 shows two navigation paths planned between two poses in the environment: one on the original 2D occupancy map and one with our approach. While the original path results in a collision, our method can prevent the collision and produces a safe navigation path.

## 4.5 Conclusion

We presented a novel approach for predicting collisions in 2D occupancy maps built with horizontally scanning 2D laser rangefinders. Our approach uses a convolutional neural network trained on collision events recorded with the force-sensitive shell of our mobile robot. The network structure enables training on binary collision events and fast, efficient segmentation of entire occupancy maps while maintaining the map resolution. Our experiments confirm that the model trained on a simulated collision dataset can reliably predict collisions in 2D occupancy maps. We also show that the performance of our approach can be further improved by integrating new collision examples collected during real-world operation. Our approach enables robots with standard indoor navigation setups to learn from collision events and reduce the number of collisions with objects only partially visible in the 2D occupancy map. The next chapter shows how mobile robots can adapt their social navigation behavior based on physical interactions with people.

# Chapter 5

# Learning Social Navigation From Physical Interaction

**Autonomous systems, such as delivery robots, are increasingly employed in shared environments where they carry out tasks alongside humans. This development poses the question of how robots can carry out their tasks while, at the same time, behaving in a socially compliant manner. Further, service robots in populated environments encounter people with little experience with robots. Inexperienced users need to be able to communicate their preferences in a simple and intuitive way, and robots should adapt their behavior accordingly. This chapter investigates force control as a natural way to interact with an autonomously navigating mobile robot by pushing it along desired trajectories. We employ Inverse Reinforcement Learning (IRL) to learn from human interaction and to adapt the robot behavior to its users' preferences, thereby eliminating the need to predefine and manually program the behavior by hand. We evaluate our approach in a real-world experiment, where test subjects interact with an autonomously navigating robot in close proximity. The results confirm that force control presents an intuitive means to interact with a mobile robot and that our robot can quickly adapt to the test subjects' personal preferences.**

## 5.1 Introduction

Robots that share their workspace with people need to pay attention to them to ensure their comfort and safety. In addition, robots operating in public spaces may encounter people with little experience with robots. The interaction between users and robots should thus be as intuitive as possible to ensure that inexperienced people can easily communicate with the robot. This chapter presents an innovative concept that enables robots to be-

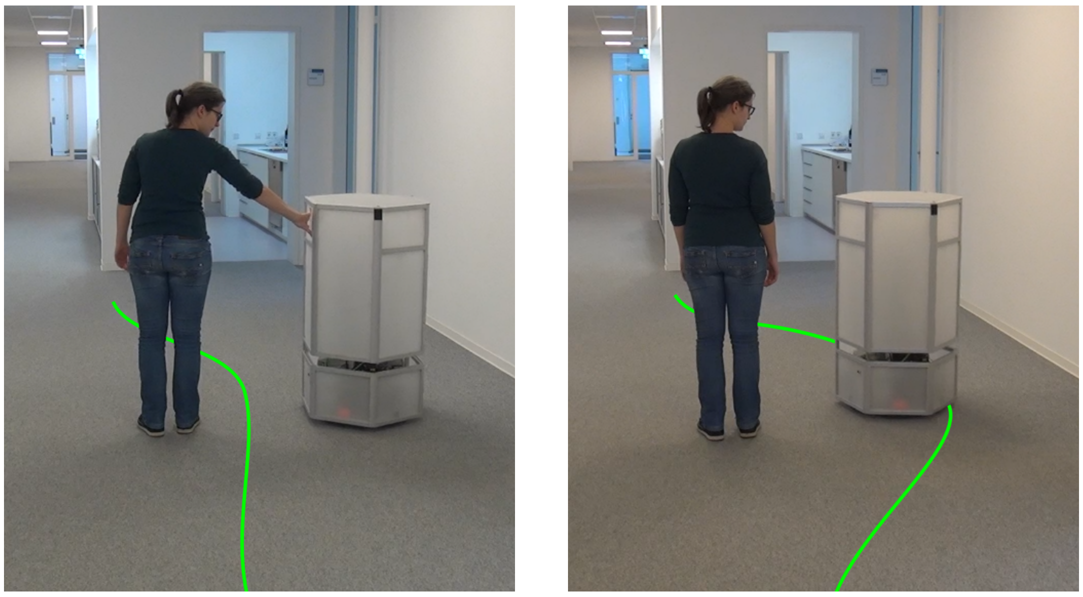
**Figure 5.1:** A person communicates her personal space preference by pushing the navigating robot away from her (left). Based on this interaction, the robot adapts its navigation behavior and learns to keep a greater distance to people (right).

have appropriately around people and adapt to their preferences. Substantial parts of the ideas, results, figures, tables, and text presented in this chapter have been previously published in [76], ©2020 IEEE, `https://doi.org/10.1109/IROS45743.2020.9340865`. Section 1.4 outlines the author's contribution to this work.

Imagine a delivery robot operating in a hospital. The robot has to ensure the safety and comfort of patients, hospital staff, and visitors when operating in their proximity. At the same time, the robot has a task-related objective of transporting items from one location to another as quickly as possible. Both objectives may contrast each other, and a reasonable trade-off is required to do both tasks well.

To balance navigation objectives, path planning is often formulated as an optimization problem, where the objectives are represented as costs. Traditionally, path planning aims to find the shortest or fastest path, but additional social costs have been formulated for keeping appropriate interaction distances [70, 90, 128], for avoiding to pass behind a person [128], or for preferring one side for passing [70]. Adjusting the cost function parameters for the desired robot behavior is not straightforward since it can depend on many factors. The preferred interaction distance, for example, is influenced by the task and role of the robot [68], the person's gender and familiarity with robots [138], and the appearance [20, 101] and speed [125] of the robot.

This chapter presents a novel approach for learning the parameters of the navigation cost function through physical human-robot interaction that eliminates the need for manual tuning of the parameters for every application and domain. To this end, we formulate a cost function that balances social space preferences and the desire to reach a goal location. During autonomous operation, people can correct the navigation behavior of the robot by pushing it along the trajectory they would prefer, as depicted in Figure 5.1. We regard the corrected trajectories as expert demonstrations of the desired robot behavior and use maximum entropy inverse reinforcement learning to adapt the cost function parameters accordingly. As a result, the robot learns to balance task objectives and social constraints, allowing it to travel on socially compliant paths.

The previous chapter illustrated how service robots with limited sensing capabilities can use force feedback to learn improved models of typical human-centered environments. In this chapter, we go one step further and investigate force feedback for directly interacting with people in the environment. While kinesthetic teaching – teaching via force feedback – has been primarily researched for robot manipulators [116], this is the first work that examines physical interaction for teaching socially compliant mobile robot navigation. Our approach does not presume any special skills or experience regarding robot control. Furthermore, since the interaction does not require an external control device, the robot can refine its navigation behavior over time and continuously adapt to the people it interacts with.

We conducted a user study with our mobile robot Canny interacting with human test subjects via its force-sensitive shell presented in Chapter 3. The experiments confirm that our approach can improve the navigation behavior of the robot based on force feedback and that this way of interaction is easy and intuitive. The code and experiment data are available at `https://github.com/marinaKollmitz/learning-nav-irl`.

## 5.2 Related Work

Obstacle avoidance techniques like the Dynamic Window Approach (DWA) [41] enable mobile robots to safely navigate in dynamic environments. However, the resulting paths often pass pedestrians very closely, and humans often perceive insufficient interaction distances as uncomfortable [109]. Various approaches have been proposed to represent human preferences during navigation. Pacchierotti *et al.* [109] developed a passing module that controls the signaling and lateral passing distance between the robot and people in a corridor passing setting. Other works model navigation preferences of people as navigation costs to cover a broader range of navigation scenarios. Kirby *et al.* [70] modeled social space requirements and a preference to pass a person on the right side as additional costs for path planning. Aspects of comfort, safety, and visibility are included as additional navigation constraints in the work of Sisbot *et al.* [128].

Tuning the cost function parameters for the desired robot behavior for different robots, environments, and tasks is not straightforward. Instead, various approaches proposed to learn robot navigation behavior via human demonstrations or observations. Trautman and Krause [142] and Luber *et al.* [92] learned human-like navigation from top-view pedestrian scenes to plan socially acceptable paths among humans. In addition, Kuderer *et al.* [78] optimized joint collision avoidance models, where all agents are expected to co-operate during navigation, via inverse reinforcement learning. To this end, they observed avoidance trajectories of people frequently passing each other in an open area. Ziebart *et al.* [159], as well as Bennewitz *et al.* [11], used observations from people walking inside office environments to learn human path prediction models and incorporate these models to plan hindrance-free robot paths.

The approaches presented above all learned navigation behavior from observing humans. However, robots are not necessarily expected to behave human-like around people [28]. Lichtenthäler *et al.* [84] proposed an "inverse Oz-of-Wizard" approach in which people teleoperated the robot in a path crossing scenario to demonstrate how they expected the robot to behave. Similarly, Kim and Pineau [66] collected demonstrations via teleoperation in crowded navigation scenarios to learn a cost function using inverse reinforcement learning. They used a cost function with binary features, while ours is continuous and explicitly models personal space. Herman *et al.* [55] investigated inverse reinforcement learning for navigation scenarios where both the cost function parameters and the state transition dynamics are unknown. In their work, humans provided demonstrations by controlling a simulated mobile robot in a populated hallway scenario. While teleoperation allows people to directly demonstrate how a robot should behave in a given situation, the demonstrators are not actively involved in the interaction. They have to anticipate the preferences of the people the robot interacts with. In our approach, people can directly demonstrate their own intent via physical interaction. Furthermore, since no external control device is required, the robot can continue to learn from demonstrations over time.

Physical interaction has been frequently employed for teaching via demonstration, often in the context of force-sensitive manipulators [1, 63]. Our work is inspired by Bajcsy *et al.* [7], who aim to correct the behavior of their robot during autonomous task execution. Similar to their work, we want to use physical interaction to enable humans to correct the current behavior of the robot according to their preferences. While they focus on tabletop tasks with a robot manipulator, our goal is to find an appropriate model for human preferences in the robot navigation domain.

So far, force-sensitive mobile robots have mostly reactively responded to physical interaction. Walking helper systems, like the ones presented by Sabatini *et al.* [122] and Spenko *et al.* [130], commanded a velocity according to the user input but did not actively navigate by themselves. Khatib [65] presented an approach for load sharing where people manipulated heavy objects in cooperation with mobile manipulators. Hirata *et al.* [56] also considered cooperative object transportation where users guided mobile robot helpers along a preplanned trajectory. Load sharing policies for cooperative transportation have been investigated by Lawitzky *et al.* [80]. In their work, the task completion time and the user effort could be reduced when robots proactively worked towards the goal instead of reacting passively to user input. Later, Lawitzky *et al.* [81] proposed to learn the motion paths for cooperative transport of heavy objects via physical human-robot interaction to provide proactive assistance. In this work, we also want the robot to adapt its behavior according to physical interaction instead of passively reacting to user inputs. The robot considers the user-adapted trajectories as demonstrations of its desired behavior and adjusts its navigation cost function to match the demonstrations.

## 5.3 Fundamentals of Markov Decision Processes

In this work, we use Inverse Reinforcement Learning (IRL) to learn preferred navigation behavior from human demonstrations. IRL builds upon Markov Decision Processes (MDPs), which we briefly introduce in the following. Note that while the term cost function is prevalent for path planning, we will, in the following, use the term reward function that is common in the MDP and IRL context.

MDPs model sequential, stochastic decision-making processes in which the agent only has partial control over the outcomes of its actions. An MDP $M = \{S, A, T, r\}$ is defined by a set of states $S$, a set of actions $A$, a transition function $T(s, a, s')$, and a reward function $r(s, a, s')$. A state $s \in S$ characterizes the agent's environment, and an action $a \in A$ describes how the agent can influence its current state. The outcomes of an action may not be deterministic, and the transition function $T(s, a, s') = p(s' \mid s, a)$ specifies the probability of reaching a successor state $s'$ from state $s$ by executing action $a$. The agent receives a reward signal $r(s, a, s') \rightarrow \mathbb{R}$ after executing an action, and the agent's goal is to find the sequence of actions that maximizes the expected cumulative rewards from the start to the goal state. The reward function thus decodes the objective of the agent. Finding the shortest navigation path in a grid world can, as an example, be formulated by a small negative reward for taking an action, often called a living reward, leading the agent to take as few actions as possible to reach the goal state.

The *Bellman equation* formulates the expected cumulative rewards from each state, also referred to as its value

$$V(s) = \sum_a \pi(a \mid s) \underbrace{\sum_{s'} T(s, a, s')[r(s, a, s') + \gamma V(s')]}_{Q(s,a)}, \tag{5.1}$$

where $\pi(a \mid s)$ is the agent's policy: the probability that the agent chooses to execute action $a$ in state $s$. The back part of Eq. (5.1) is often referred to as the Q-value of the state-action pair, $Q(s, a)$, and $\gamma \in [0, 1]$ can be used to discount future rewards. Maximizing the expected cumulative rewards translates to choosing the action with the best expected outcome in each state, resulting in the *Bellman optimality equation*

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[r(s,a,s') + \gamma V^*(s')]. \tag{5.2}$$

Multiple actions may maximize the expected cumulative rewards, and the set of optimal actions $\Pi^*(s)$ at state $s$ is given by

$$\Pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')[r(s,a,s') + \gamma V^*(s')]. \tag{5.3}$$

An optimal policy $\pi^*(a \mid s)$[1] is then calculated by

$$\pi^*(a \mid s) = \begin{cases} \frac{1}{|\Pi^*(s)|} & a \in \Pi^*(s) \\ 0 & \text{else.} \end{cases} \tag{5.4}$$

To find a sequence of actions that maximizes the expected cumulative reward, the agent can execute the optimal policy step by step, starting from the start state until it reaches the goal state.

The Bellman optimality equation represents a recursive formulation, and dynamic programming techniques can be used to solve it for finite horizon or discounted MDPs. One popular method is value iteration, which calculates the state values interactively,

$$V^*_{k+1}(s) = \max_a \sum_{s'} T(s,a,s')[r(s,a,s') + \gamma V^*_k(s')]. \tag{5.5}$$

Starting at $k = 0$ with an initial guess $V_0$, value iteration applies Eq. (5.5) until convergence. The optimal policy $\pi^*(a \mid s)$ can be calculated from the converged values with Eq. (5.3) and Eq. (5.4).

## 5.4 Learning From Physical Interaction

In this section, we present our approach to learning navigation behavior from physical interaction. Section 5.4.1 explains how people can demonstrate their desired robot behavior with our force-sensitive robot Canny. Section 5.4.2 gives an overview of the inverse reinforcement learning technique we employ to learn from the resulting demonstrations. Finally, Section 5.4.3 introduces our navigation reward function to model the navigation task.

[1]Note that the optimal policy is often denoted by a function $\pi^*(s) \to a$, retrieved directly from Eq. (5.3). We choose a representation here that explicitly accounts for the fact that multiple actions may maximize the expected reward at a given state.

### 5.4.1 Demonstrating Desired Robot Behavior

We focus on correcting the behavior of an autonomously navigating robot. The robot commands a navigation velocity $\vec{v}_\mathrm{n} = (v_{\mathrm{n},x}\ v_{\mathrm{n},y})^T$ to follow its current navigation path as closely as possible. In particular, we employ the omni_path_follower [71] package to generate navigation commands. The robot further overlays external command velocities $\vec{v}_\mathrm{c}$ to adapt its path according to user feedback. The executed robot velocity results in

$$\vec{v}_\mathrm{r} = \vec{v}_\mathrm{n} + \vec{v}_\mathrm{c}. \tag{5.6}$$

We use our force-sensitive robot Canny to learn from physical interactions. Canny deploys a 6-DoF force-torque sensor between its omnidirectional mobile base and its solid shell to perceive interaction forces, as introduced in Chapter 3. To integrate the force feedback, we translate the interaction force $\vec{F} = (F_x\ F_y)^T$ to a proportional velocity command,

$$\vec{v}_{\mathrm{c},F} = k \cdot \vec{F}, \tag{5.7}$$

where $k[\mathrm{s\,kg^{-1}}]$ is the proportionality factor. Other input modalities can be integrated to adapt the robot behavior, following Eq. (5.6).

### 5.4.2 Learning Human-Aware Navigation From Demonstrations

We see the user-adapted paths as demonstrations of the desired robot behavior and use Inverse Reinforcement Learning (IRL) to recover a reward function that describes the demonstrations. In contrast to Reinforcement Learning (RL), where an agent learns to solve a task by actively exploring its environment, the inverse RL problem involves finding an expert's objective by passively observing her/his behavior. The expert is assumed to act according to an unknown reward function, and the demonstrations represent noisy-optimal samples from it.

We model the robot navigation task as an MDP (see Section 5.3), where the discretized cells of the navigation map are the states $S$, and the actions $A$ are traversing into adjacent map cells in an eight-connected fashion. We assume that the state transitions are deterministic, i.e., executing action $a$ from state $s$ always results in the same next state $s'$. Thus, we can write the transition function as $s' = \hat{T}(s, a)$. Finally, we are given a reward function $r_\theta$ with unknown parameters $\theta$ and a set $D$ of demonstrations $\tilde{\tau}_i = (s_0, a_0, \ldots, a_{N_i-1}, s_{N_i})_i$, $i = 1, \ldots, |D|$ of varying lengths $N_i$. The goal is then to recover the parameters $\theta^*$ that best explain this set of given demonstrations.

We want to learn from human demonstrations, and we cannot assume that humans always act optimally when performing a task. Therefore, our approach follows the Maximum Entropy IRL method of Ziebart *et al.* [158], which processes the noisy-optimal demonstrations in a probabilistic fashion. Furthermore, it solves the inherent IRL ambiguity that many reward functions may be optimal for given demonstrations by choosing the distribution that remains maximally uncertain beyond matching the expert demonstrations.

Maximum Entropy IRL assumes that the probability of executing a trajectory $\tau$ grows exponentially with its utility. For a given reward function $r_\theta$, this results in a distribution over trajectories of

$$P(\tau \mid \theta) = \frac{1}{Z(\theta)} \exp(R_\theta(\tau)), \tag{5.8}$$

where $R_\theta(\tau)$ is the sum of rewards along the trajectory $\tau$ and the partition function

$$Z(\theta) = \sum_\tau \exp(R_\theta(\tau)) \tag{5.9}$$

normalizes the distribution. While the original Maximum Entropy IRL work [158] assumed a reward function that is linear in $\theta$, Wulfmeier *et al.* [152] showed that the formulation is also suitable for general non-linear, sufficiently smooth reward functions. Our reward function presented in Section 5.4.3 is feature-based but non-linear. Therefore, we adhere to the general formulation where the reward is parametrized by feature weights but may be non-linear.

To find the reward parameters $\theta^*$ that best explain the demonstrated behavior, Maximum Entropy IRL seeks to maximize the log-likelihood of the demonstrations,

$$\theta^* = \arg\max_\theta L(\theta) = \arg\max_\theta \sum_{\tilde{\tau} \in D} \log p\left(\tilde{\tau} \mid \theta\right), \tag{5.10}$$

which can be optimized using gradient-based techniques. The gradient with respect to the reward parameters,

$$\nabla_\theta L(\theta) = \sum_{\tilde{\tau} \in D} \frac{\partial R_\theta(\tilde{\tau})}{\partial \theta} - |D| \sum_\tau p(\tau) \frac{\partial R_\theta(\tau)}{\partial \theta}, \tag{5.11}$$

requires to sum over all trajectories $\tau$, which is intractable in large discrete or even continuous state spaces. Instead, we can unroll the trajectories to state-action pairs to obtain the tractable representation,

$$\begin{aligned} \nabla_\theta L(\theta) &= \sum_{\tilde{\tau} \in D} \sum_{(s,a) \in \tilde{\tau}} \frac{\partial r_\theta(s,a)}{\partial \theta} && (5.12) \\ &- |D| \sum_{t=0}^{\infty} \sum_{(s,a)} p_\theta(a \mid s) p_{\theta,t}(s) \frac{\partial r_\theta(s,a)}{\partial \theta}, && (5.13) \end{aligned}$$

with unknowns $p_\theta(a \mid s)$ and $p_{\theta,t}(s)$. In practice, we replace the infinite time horizon with a predefined or adaptively chosen finite horizon $T$.

The term $p_\theta(a \mid s)$ represents the stochastic policy inducing the distribution over trajectories in Eq. (5.8). Note that, since we cannot assume that people always act perfectly optimal, the stochastic policy is different from the optimal policy we would obtain by solving the MDP with the standard Bellman optimality equation, Eq. (5.2) - Eq. (5.4). Instead, it was shown in [159] that $p_\theta(a \mid s)$ can be obtained by solving a "softened" version of the Bellman equations, where the value of a state $V^{\sim}(s)$ is not the maximum of its Q-values, but the log-sum of the exponentiated Q-values $Q^{\sim}(s,a)$. That is, the corresponding Bellman update is given by

$$\begin{aligned} Q^{\sim}(s,a) &= r_\theta(s,a) + V^{\sim}(\hat{T}(s,a)), && (5.14)\\ V^{\sim}(s) &= \log \sum_a \exp(Q^{\sim}(s,a)), && (5.15) \end{aligned}$$

which can be solved using standard value iteration. The desired stochastic policy is then obtained via

$$p_\theta(a \mid s) = \exp(Q^{\sim}(s,a) - V^{\sim}(s)), \tag{5.16}$$

i.e., the probability of choosing an action is proportional to the expected exponentiated future rewards.

Given the stochastic policy $p_\theta(a \mid s)$, we can now obtain the distribution over states at a certain time step, $p_{\theta,t}(s),\ t = 0, \ldots, T$, by propagating the policy through the state space. Starting from an initial distribution over states $p_0(s)$, all future state distributions can be calculated by

$$p_{\theta,t+1}(s') = \sum_{(s,a)} \mathbb{1}_{\{s'=\hat{T}(s,a)\}} p_\theta(a \mid s) p_{\theta,t}(s), \ \ \forall s' \in S, \tag{5.17}$$

where $\mathbb{1}_{\{.\}}$ denotes the indicator function.

Once the gradient is calculated, we can use standard (stochastic) optimization techniques to solve for the reward function parameters. Since we have to solve equations Eq. (5.12) - Eq. (5.17) for every iteration of the optimization, calculating the gradient amounts to solving an MDP in every step. While this can become computationally infeasible very quickly, the state-action space is small enough in our work to iteratively update our guess of $\theta^*$ during our experiments as new demonstrations arrive.

### 5.4.3 A Reward Function for Socially Compliant Navigation

To learn social navigation from demonstrations, our robot requires a general parametric reward function $r_\theta(s,a)$ that explains socially compliant behavior. This function should reflect the desire to reach a predefined goal state $s_{\mathrm{g}}$ while avoiding static obstacles, and it should acknowledge the personal space of a person standing at a given state $s_{\mathrm{p}}$.

We consider static obstacles $S_\mathrm{o} \subset S$ as a subset of the state space that the robot must avoid. Since we assume that the state transitions are deterministic, we can simply assign a large penalty term for being in an obstacle state, i.e.,

$$r_\mathrm{o}(s) = -\infty \cdot \mathbb{1}_{\{s \in S_\mathrm{o}\}}, \tag{5.18}$$

without introducing conservatism. We further assign a step reward term

$$r_a(s, a) = c_a \cdot ||s - s'||_2, \tag{5.19}$$

where $s' = \hat{T}(s, a)$ is the successor state and $c_a$ is a weighting parameter. The step reward term leads the robot to favor short paths from the start to the goal.

To further encourage goal-directed and predictable robot behavior, we define a path deviation reward $r_\mathrm{d}(s)$. It penalizes the deviation from any nominal path between the start state $s_0$ and the goal state $s_\mathrm{g}$,

$$r_\mathrm{d}(s) = a_\mathrm{d} \cdot \min_{\bar{s} \in S_\mathrm{n}} ||s - \bar{s}||_2^2, \tag{5.20}$$

where $a_\mathrm{d}$ is a weighting parameter, and $S_\mathrm{n} \subset S$ denotes the set of states belonging to any nominal path. A nominal path is a minimum-length path between the start and the goal state that avoids the static obstacles. There may be multiple nominal paths since the robot environment is a discretized grid world. To find the states along the nominal paths, we first define an auxiliary reward function that considers static obstacles and minimizes the path length,

$$\bar{r}(s, a) = r_\mathrm{o}(s) + r_a(s, a). \tag{5.21}$$

We then solve the navigation MDP with $\bar{r}(s, a)$ by value iteration (Eq. (5.5)) and calculate the optimal policy $\pi^*_{\bar{r}}(a \mid s)$ with Eq. (5.4). To find the state distribution $p_{\bar{r}}(s)$, we propagate the policy through the state space. Starting from an initial state distribution $p_{\bar{r},0}(s) = \mathbb{1}_{\{s=s_0\}}$, the state distributions for the following time steps can be calculated by

$$p_{\bar{r},t+1}(s') = \sum_{(s,a)} \mathbb{1}_{\{s'=\hat{T}(s,a)\}} \pi^*_{\bar{r}}(a \mid s) p_{\bar{r},t}(s), \;\; \forall s' \in S, \tag{5.22}$$

and the overall distribution over states is given by

$$p_{\bar{r}}(s) = \sum_{t=0}^{\infty} p_{\bar{r},t}(s). \tag{5.23}$$

Eq. (5.22) is equivalent to calculating the distribution over states with the stochastic policy in Eq. (5.17). In practice, we do not need to compute the infinite sum above since the optimal policy reaches the goal state after the minimum possible number of actions. The state distribution $p_{\bar{r}}(s)$ specifies the probability that a state is visited by following the optimal policy. Accordingly, we can calculate the set of states along any nominal path by

$$S_{\mathrm{n}} = \{s \in S \mid p_{\bar{r}}(s) > 0\}. \tag{5.24}$$

The path deviation reward implicitly encodes a signaling behavior; it determines at what point the social space reward dominates, causing the robot to leave its original trajectory.

Finally, we use a squared-exponential reward function to reflect a person's personal space requirements,

$$r_{\mathrm{p}}(s) = a_{\mathrm{p}} \cdot \exp(-\sigma_{\mathrm{p}} ||s - s_{\mathrm{p}}||_2^2), \tag{5.25}$$

parametrized by a weight coefficient $a_{\mathrm{p}}$ and scaling coefficient $\sigma_{\mathrm{p}}$ to model the steepness and width of the personal space reward. This type of distance function is well-known to accurately describe a person's personal space in social navigation [70, 128].

In the terminal state $s_{\mathrm{g}}$, we set all rewards to zero, i.e.,

$$r_a(s_{\mathrm{g}}, a) = r_{\mathrm{d}}(s_{\mathrm{g}}) = r_{\mathrm{p}}(s_{\mathrm{g}}) = 0. \tag{5.26}$$

The overall reward function is given by the sum of the individual reward components,

$$r_\theta(s, a) = r_a(s, a) + r_{\mathrm{d}}(s) + r_{\mathrm{p}}(s) + r_{\mathrm{o}}(s), \tag{5.27}$$

and the parameter set is $\theta = \{c_a, a_{\mathrm{d}}, a_{\mathrm{p}}, \sigma_{\mathrm{p}}\}$.

## 5.5 Experiments

We conducted a user study with test persons interacting with a mobile robot to investigate whether the robot can learn preferred social navigation behavior with our approach. We further aimed to test if force feedback is an easy and intuitive way to interact with a mobile robot. To this end, we compared force feedback control to using a joystick controller to drive the robot. Section 5.5.1 explains the experiment design. In the first task of the user study, presented in Section 5.5.2, the test subjects controlled the robot along an obstacle course. The experiment tests force feedback for robot control without autonomous motion. Section 5.5.3 evaluates the second task, for which the test subjects interacted with the autonomously navigating robot. Task two tests force control for correcting an initial robot behavior and investigates whether the robot can adapt to the participants' preferences. In the final task, the path replay task presented in Section 5.5.4, the participants judged the learned navigation behavior of the robot, compared to their joystick-adapted paths and pure obstacle avoidance. Finally, Section 5.5.5 evaluates the generalization capabilities of the learned reward function for two unseen navigation scenarios.

### 5.5.1 Experiment Design

We recruited thirteen test subjects (5 female, 8 male, aged 21 - 63). Eight participants had a technical background (2 female, 6 male), and six had previous experiences with robots (2 female, 4 male). We further asked the participants' skill level with a joystick controller, e.g., from playing console or computer games. Seven participants reported their skill level as rather high or high (1 female, 6 male), and six participants as rather low or low (4 female, 2 male). None of the participants was involved in the research project or had prior knowledge of our research aim. After a general introduction to the experimental procedure, the test subjects gave written consent for participation. They further received 10€ as reimbursement.

The experiments took place in the hallway of building 80 at the Faculty of Engineering of the University of Freiburg with our force-sensitive mobile robot Canny. Using the force-sensitive sensor setup presented in Chapter 3, Canny perceived and reacted to interaction forces on its shell. Each experiment run lasted about $45\,\mathrm{min}$, during which the test subjects performed two sets of tasks with the robot. In one set, the participants controlled the robot via force feedback; they used a joystick in the other set. The joystick experiments serve as a baseline to compare the ease of use and intuitiveness of physical interaction against a more common control method. Furthermore, participants can influence the entire robot trajectory with the joystick, while in the physical interaction setting, the robot needs to be within their reach. We wanted to see the influence of this limitation on the quality of the learned navigation behavior.

We randomized the order of the task sets for each participant to balance familiarization effects. We explained the robot control for both control modalities and gave the test subjects some time to become familiar with it. Once the participants felt confident with the respective control modality, they performed an obstacle course task presented in Section 5.5.2. Afterward, they continued with the passing task presented in Section 5.5.3. Finally, the participants judged different robot trajectories in the path replay task, presented in Section 5.5.4.

### 5.5.2 Obstacle Course Task

To test how the participants handle controlling the robot without autonomous motion, we performed an obstacle course task. As depicted in Figure 5.2, the participants controlled the robot past two traffic cones from a start to a goal location in the corridor. For this task, the participants commanded all robot motion, and we limited the robot velocity to $\|\vec{v}_\mathrm{r}\|_2 \leq 0.4\,\mathrm{m\,s^{-1}}$. Each participant performed one obstacle course run per control modality. Figure 5.3 shows two participants performing the obstacle course task with a joystick and by force feedback, respectively. To compare the ease and intuitiveness of both control modalities, we timed how long the participants took to control the robot from

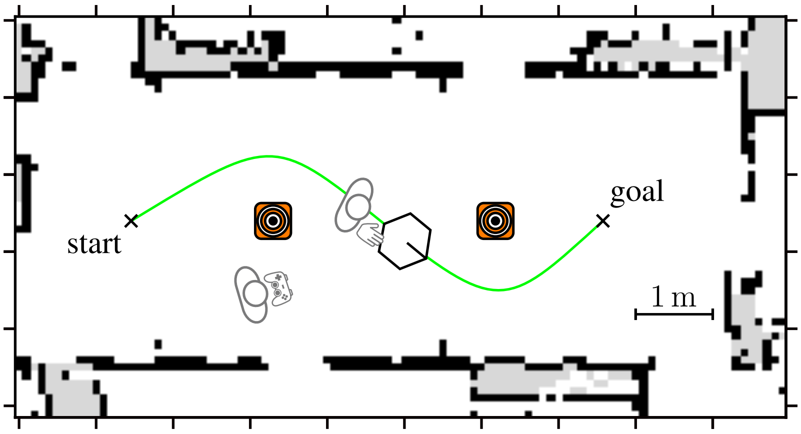

**Figure 5.2:** Experiment setup of the obstacle course task. For this task, the participants controlled the robot around two traffic cones.

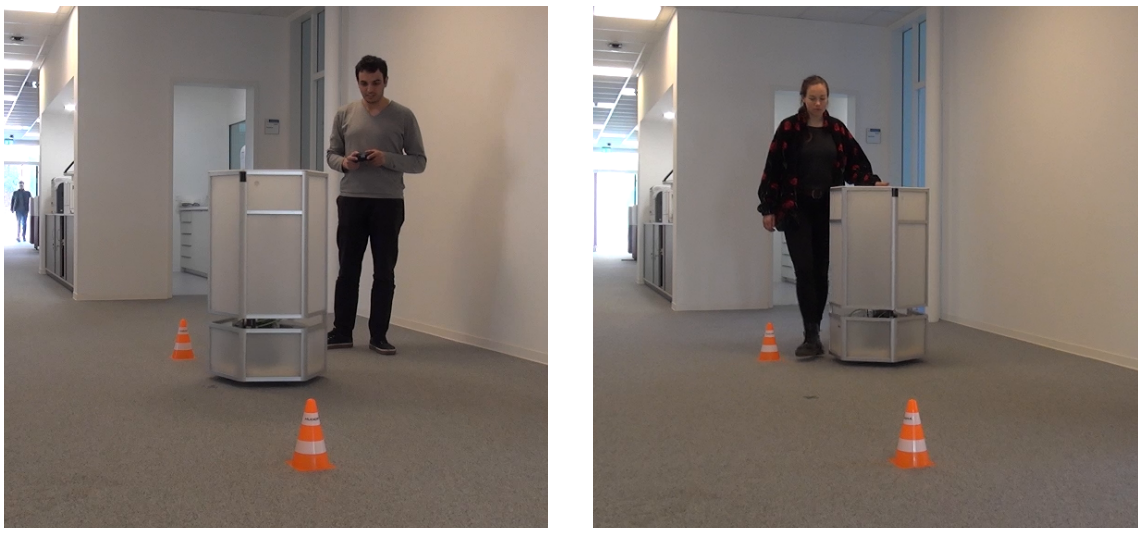

**Figure 5.3:** Participants control the robot through the obstacle course with the joystick (left) and force feedback (right).

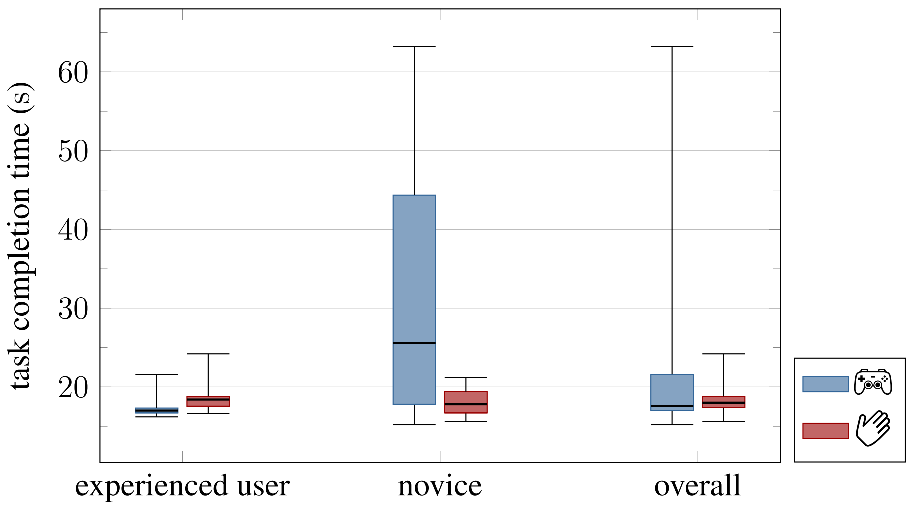

**Figure 5.4:** Task completion times of the obstacle course task for the joystick and force feedback modality for expert, novice users, and over all participants. The box plot shows the median and interquartile range, and the whiskers mark the minimum and maximum task completion times.

the start to the goal position. Note that we did not inform the participants that we timed their runs because we wanted them to behave naturally.

We evaluated the task completion times for experienced users and novice users separately. For the force feedback modality, we count people with previous experience with robots as experienced users and the other participants as novices. For the joystick modality, all participants who reported their skill level as rather high or high are in the experienced users group; the others are in the novices group.

Figure 5.4 displays the task completion times for experienced users, novice users, and over all participants. While the experienced and novice users performed comparatively with the force feedback modality, there are vast differences in the joystick modality. Especially the large variance in the novice users group for joystick control is striking. While some inexperienced participants quickly finished the obstacle course with the joystick, half of them took exceptionally long with over 30 s to complete the task. In contrast, no participant took longer than 25 s to finish the obstacle course task with force control. The experiment suggests that controlling the robot via force feedback is easy and intuitive for most people, while some have difficulties with the joystick. The results indicate that force feedback can facilitate the interaction with mobile robots in populated spaces, and especially non-expert users seem to benefit from it.

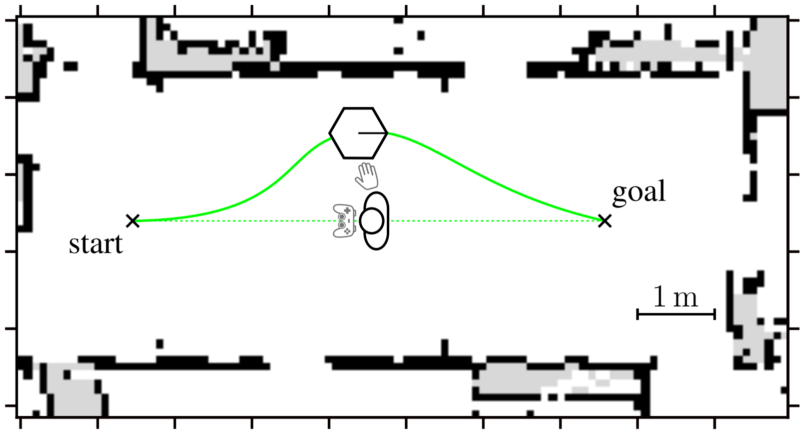

**Figure 5.5:** Experiment setup for the passing task. For this task, participants guided the autonomously navigating robot past themselves.

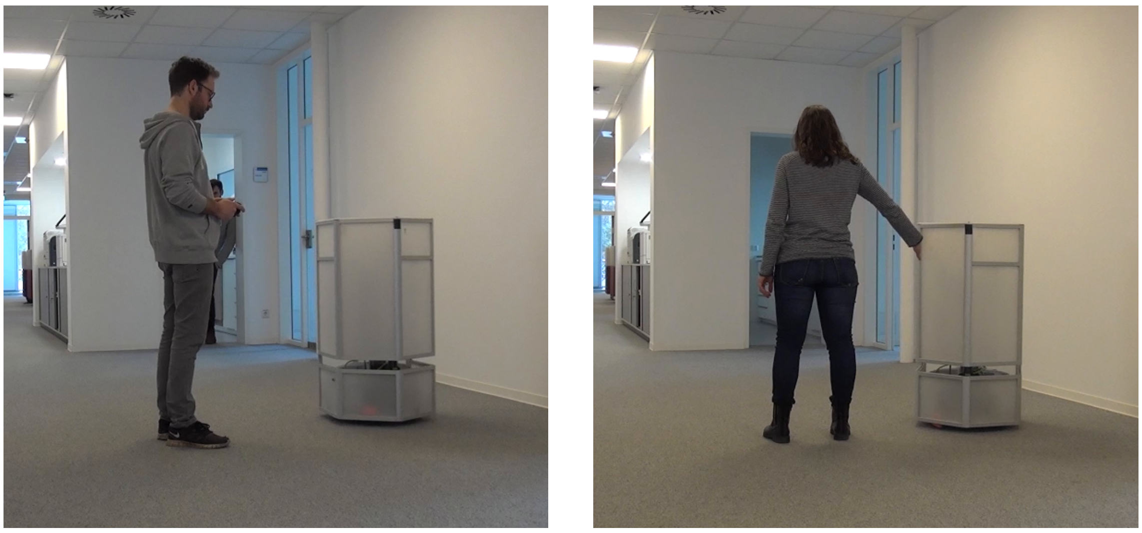

**Figure 5.6:** Participants guide the navigating robot past themselves with the joystick (left) and force feedback (right).

### 5.5.3 Passing Task With IRL

The passing task tests force and joystick feedback for interacting with an autonomously navigating robot. The participants stood in the center of the hallway and guided the robot past themselves, as sketched in Figure 5.5. The robot navigated autonomously from the start to the goal position, and the participants adapted its path either via joystick or by pushing on its shell. Figure 5.6 shows two participants performing the passing task with a joystick and by force feedback, respectively. We instructed the participants to adapt the robot path like they would prefer it to drive past them autonomously. The robot navigated with a velocity of $\vec{v}_\mathrm{n} = (v_{\mathrm{n},x}\ v_{\mathrm{n},y})^T$ and overlaid the user-commanded control velocity $\vec{v}_\mathrm{c}$, as described in Section 5.4.1. The forward motion $v_{\mathrm{n},x}$ was kept constant at $0.4\,\mathrm{m\,s^{-1}}$. The robot further aimed to follow its navigation path by commanding a sideways velocity $v_{\mathrm{n},y}$ proportional to the distance to the path but limited to $|v_{\mathrm{n},y}| \leq 0.2\,\mathrm{m\,s^{-1}}$. Before the actual passing task, the participants performed two practice tasks to familiarize themselves with controlling the autonomously navigating robot. Firstly, the robot moved in a straight line from start to goal, and the participants freely deviated it from its path to test the behavior. Secondly, we arranged four traffic cones as obstacles and asked the participants to guide the navigating robot around them to ensure they could control it along their desired trajectory.

For both control modalities, the robot initially navigated autonomously from the start to the goal without considering the person in the hallway. In the joystick set, the participants performed two passing runs with joystick feedback that serve as a reference for the true desired behavior without reachability constraints. The robot iteratively adapted its navigation behavior in the force feedback set by optimizing the navigation reward function presented in Section 5.4.3 via IRL. To this end, we performed two force feedback passing cycles with two passing runs each. For the first two runs, the robot started with a small offset of $0.4\,\mathrm{m}$ to the left and right, respectively, to facilitate the interaction. The robot then performed one online IRL run to optimize the reward function parameters according to the corrected trajectories. We then performed another cycle with two passing runs, where the robot paths were generated from the learned model by sampling from the stochastic policy (Eq. (5.16)). Again, the participants could correct the paths according to their preferences. We finally retrained the reward parameters with a second IRL run on the last two trajectories. Note that the robot did not learn the reward parameters from the joystick trajectories since we aimed to investigate learning from physical interaction. Furthermore, we used the participants' joystick trajectories as a baseline for their final path evaluation in Section 5.5.4, and we did not want to bias the baseline. Figure 5.7 visualizes the joystick and force feedback trajectories from all test subjects.

During the force feedback tasks, we used the Adam [69] optimizer with an L2 regularization term, as suggested by Wulfmeier *et al.* [152]. We set the learning rate to $0.1$ and ran the optimization until the log-likelihood of the demonstrations converged, which

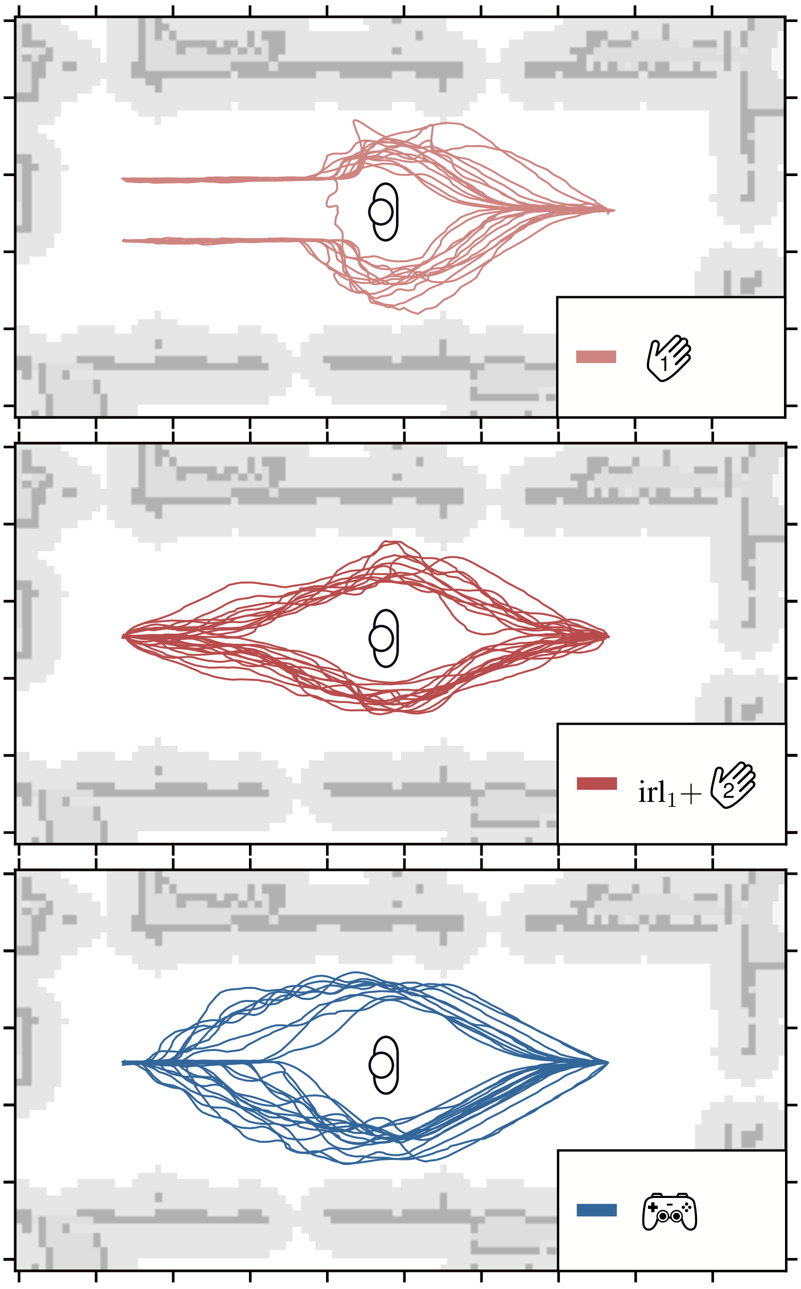

**Figure 5.7:** Robot paths adapted via user feedback in the passing task. Top: initial paths, adapted through force feedback. The robot started with a small offset to facilitate the interaction. Middle: paths from learned reward function after the first IRL run, adapted through force feedback. Bottom: initial paths, adapted through joystick feedback.

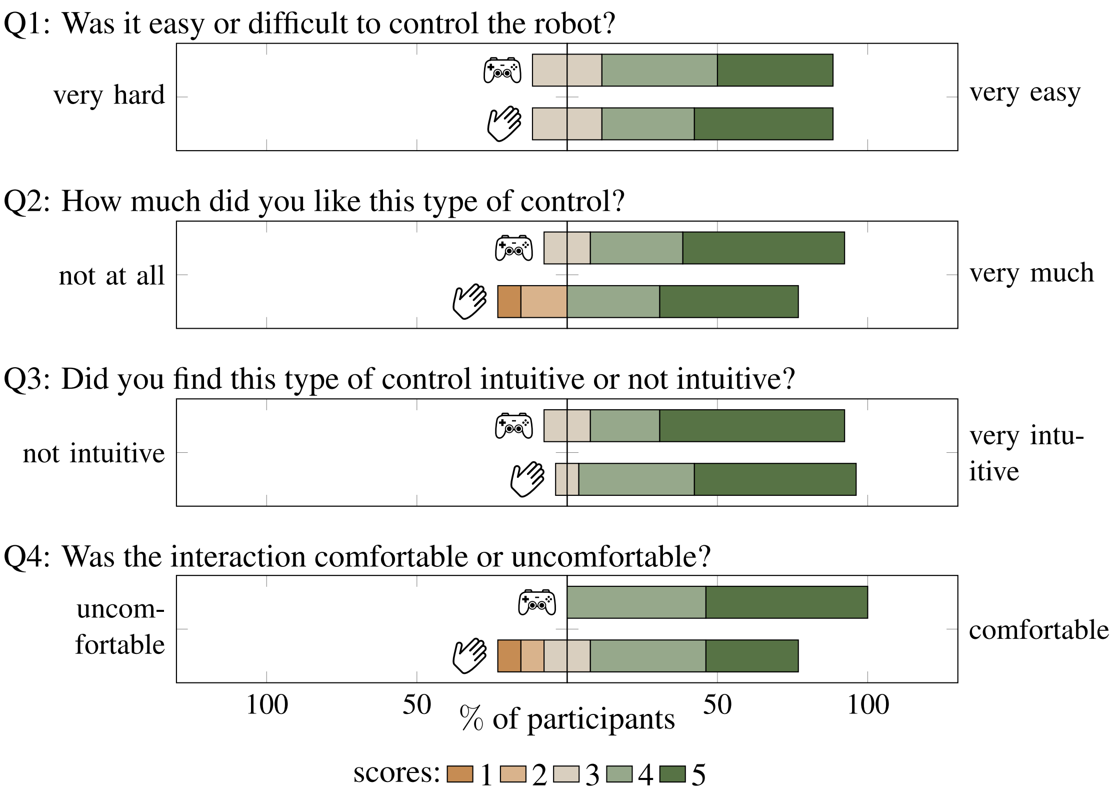

**Figure 5.8:** Questionnaire ratings after the passing task for force feedback and joystick interaction. Bars left of the center show negative, bars on the right show positive responses.

took under 2 min for 100 to 250 iterations on the experiment laptop. To speed up the training, we initialized the reward function with parameters we found during initial tests in simulation and refined the parameters with every IRL cycle.

Once the participants performed all tasks in a set, we asked them to rate the control modality on a 5-point Likert scale questionnaire. Figure 5.8 summarizes the questionnaire findings. Both the force control and the joystick modality received very positive ratings for the ease of controlling the robot and the intuitiveness. Most participants liked both the joystick and the force control and felt comfortable during the interaction. Three participants did not like the physical interaction (rating $\leq 2$), and two felt uncomfortable (rating $\leq 2$). In additional comments on the questionnaires, two participants stated that they did not like touching the robot and preferred to control it without physical contact. One participant wrote that she felt uneasy when the robot directly approached her, not knowing how it would react to her interaction. Fortunately, since the robot can learn from the interaction, it can correct such behavior after only a few interactions. One participant stated that she found the joystick control more fun than force feedback. Overall, the participants perceived the force feedback modality almost as positively as the more common joystick control. However, joystick control requires an external control device

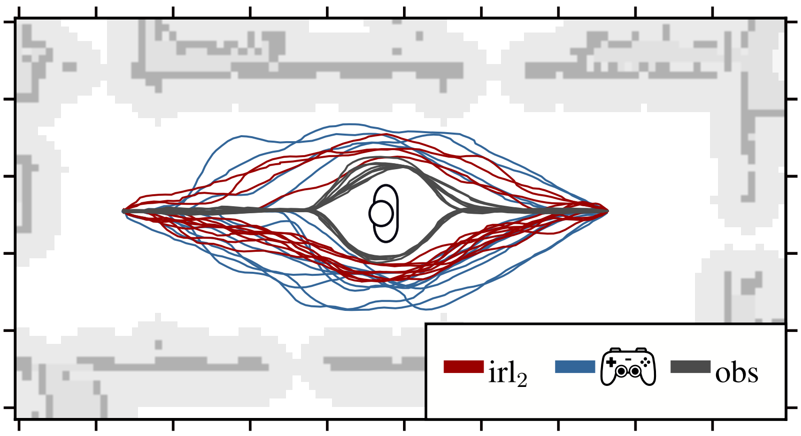

**Figure 5.9:** Final trajectories presented to the test subjects: paths from the learned model after the second IRL run, joystick path replays, and obstacle avoidance paths (obs).

and a designated teaching phase. With force feedback, the robot can continue to adapt to the people in the environment, and everyone in reach of the robot can communicate with it. Based on the mostly positive answers, we think that force control has great potential for interacting with a mobile robot in populated environments. However, it is a very novel concept, and some people might need more time to get used to interacting with a mobile robot, especially via touch.

### 5.5.4 Path Replay Task

To validate the learned navigation behavior, we finally presented three different paths to the participants. The robot navigated past them without their interaction on

- a path generated from the leaned reward function by sampling from the stochastic policy
- a replay of one of the trajectories the participant adapted with the joystick,
- and a path with obstacle avoidance only.

Figure 5.9 visualizes the resulting paths from all participants. We randomized the order of the three trajectories and did not tell the participants how the paths were generated. Afterward, we asked the participants to rate the three paths in a third questionnaire. The questionnaire results are presented in Figure 5.10. While most of the participants rated both the joystick and the learned force path comfort positively, the obstacle avoidance path received a very negative rating. All participants rated the obstacle avoidance path as too close, and many criticized on the questionnaire that the robot indicated much too late that it would avoid them.

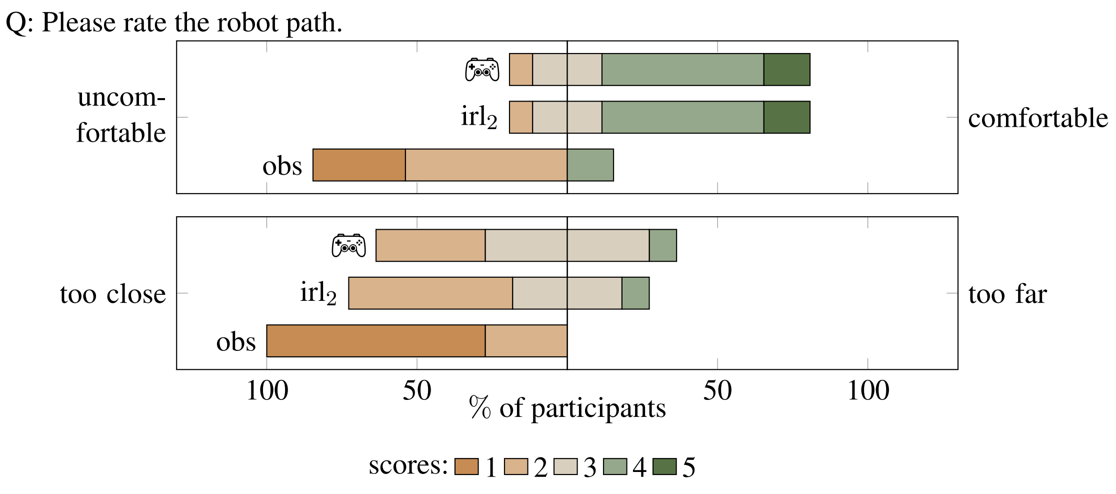

**Figure 5.10:** Participant ratings of the final trajectories in the path replay task, for the joystick path replay, the path generated from the learned reward function after the second IRL run, and performing obstacle avoidance only (obs).

Interestingly, despite the positive comfort ratings, many participants rated the joystick paths and the paths generated from the learned reward function as rather too close. This effect is visible for both the joystick and the force feedback paths but more prominent in the latter. We were surprised by this finding, especially since the participants commanded the joystick trajectories themselves, and we always confirmed that the participants agreed with the passing runs before saving them. One possible explanation is that participants might tolerate a smaller distance to the robot when they know that they can influence its behavior. Once they cannot interact, they prefer a larger distance.

To compare the final paths and confirm that the learned navigation behavior accurately follows the demonstrations, we empirically evaluated the paths. We compared the path lengths, the minimum passing distances, and the path areas, i.e., the areas between the straight paths from start to goal and the commanded trajectories. While the path length and minimum distance already capture important path features, the area under the path indicates how early the robot started the passing maneuver. Figure 5.11 presents the properties of the paths from the path replay experiment. We further included the force feedback paths from the second passing cycle on which the reward function was optimized. We can see that the force feedback paths from the final passing cycle exhibit very similar properties to the paths from the learned reward function. This indicates that the reward function can capture important path properties and that the robot can learn sensible reward function parameters to mimic the demonstrated behavior.

We can further see that both force feedback paths pass closer to the participants than the joystick paths. Reachability constraints might cause this discrepancy: the participants could not push the robot further away from where they were standing. However, we do not think this is the reason in this scenario because we instructed the participants to step

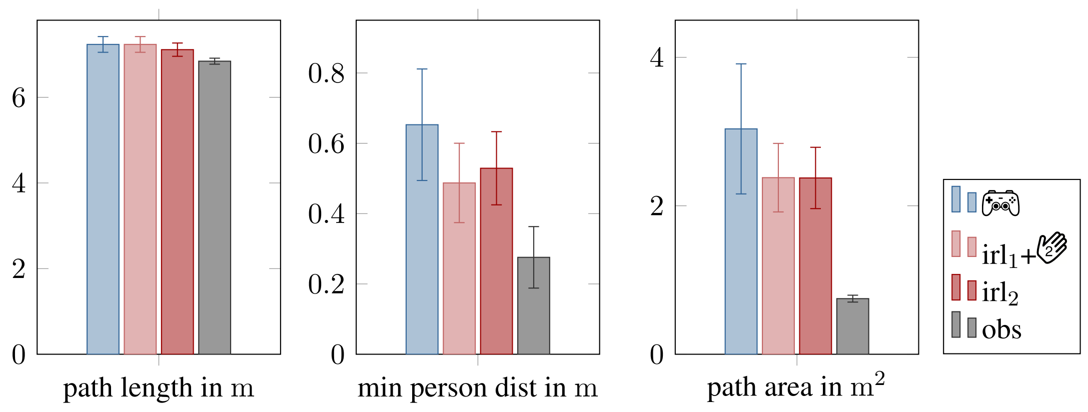

**Figure 5.11:** Properties of the final trajectories: path length, minimum distance between the robot shell and the center of the person, and area under the path. The bar plots show the mean property value and the standard deviation for the joystick feedback paths, the force-adapted paths in the second passing cycle, the paths after the second IRL run, and with obstacle avoidance (obs).

towards the robot if desired. From our observations, we instead think that the participants tolerated slightly unpleasant behavior before they intervened, even if they would have preferred the robot to behave differently. Some participants did not touch the robot at all in the second passing cycle but later reported the resulting behavior as too close. We do, however, acknowledge that the limited reach presents a drawback of our work. Future work could include other cues for adapting the robot behavior to address this limitation, such as a person's gaze or her/his own evasive maneuvers.

Finally, to visualize the learned navigation behavior, we trained the reward function parameters on the last two force feedback passing trajectories of all participants. Figure 5.12 shows the propagated policy and the learned state reward.

### 5.5.5 Generalization to New Environments

To verify that the learned reward function generalizes to new navigation situations, we employed it for two new navigation scenes depicted in Figure 5.13. In the first scene in Figure 5.13 (top), the robot navigates in the same hallway as before, but two people are present. As we can see from the propagated policy, the learned navigation function is able to consider both people. The interaction distances are similar to those in the learned combined model in Figure 5.12. In the second scenario, shown in Figure 5.13 (bottom), we tested the navigation in a different part of the hallway with a large obstacle. This time, the robot has to pass closer to the person because of the limited space. However, the propagated policy stays as far away as possible from the person. In this navigation situation, it might be more appropriate for the robot to wait next to the large obstacle until the person has passed the narrow passage. While it is generally possible to teach

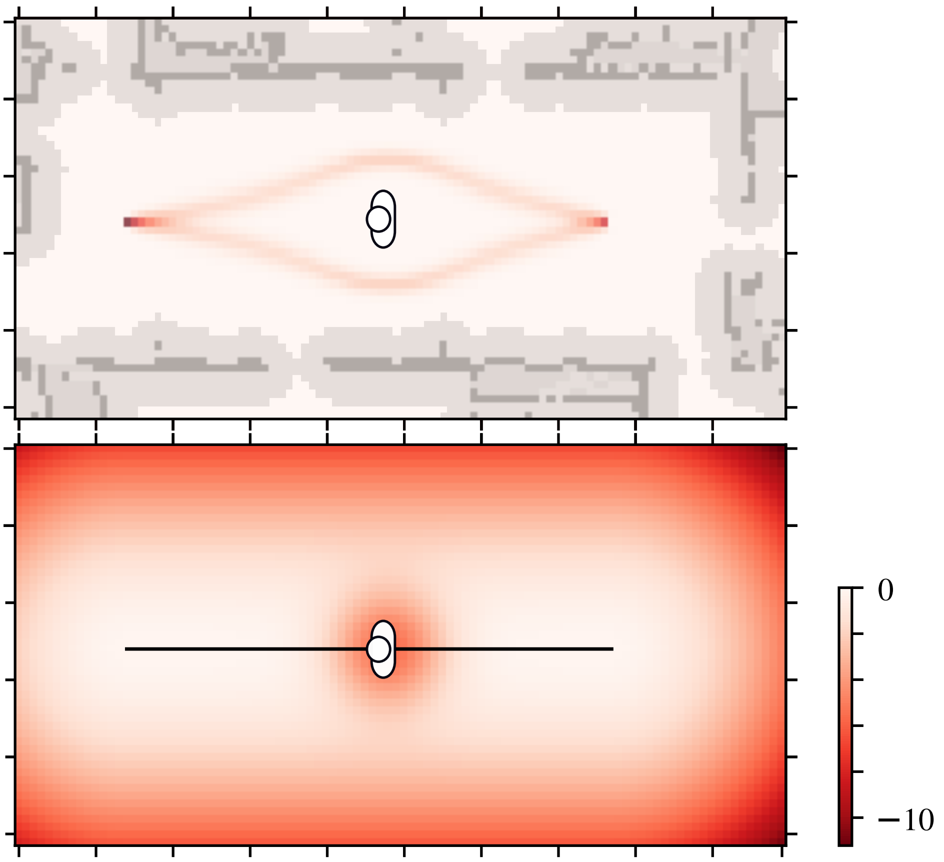

**Figure 5.12:** Propagated policy from the learned combined model (top) and state reward of the final model with the nominal direct path (bottom).

our robot Canny this type of behavior via force feedback, waiting would require the state space representation to include the time dimension. Reasoning about the time dimension would also be necessary for interaction with moving people, so this would be a sensible and necessary next step for this work. The experiment still suggests that the learned navigation function can represent and generalize to both new navigation scenes. However, a more natural and considerate robot behavior could be achieved by reasoning about the proximity to people with respect to time.

## 5.6 Conclusion

We introduced a novel approach for teaching robots social navigation behavior among humans via physical interaction and inverse reinforcement learning. We introduced a parametric, non-linear reward function to model the social navigation task. Through real-world experiments with human test subjects, we demonstrated that controlling a mobile robot via pushing is a viable means of communicating navigation preferences and that

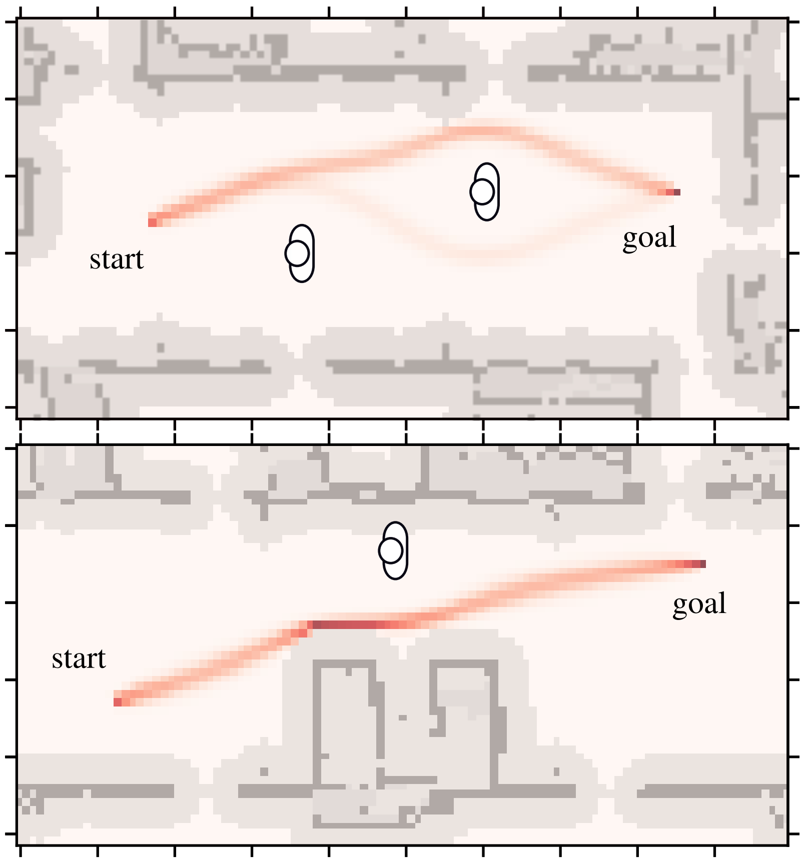

**Figure 5.13:** Propagated policy from the learned combined model, applied to two unknown navigation scenes. The top image shows a navigation scene with two people in the hallway, and the bottom image a scene with one person next to a large obstacle in a different part of the hallway.

the robot can learn to adjust its behavior accordingly. Our results further suggest that the method can generalize to more complex scenes with multiple people. Future work should investigate additional cues like gaze or human path deviations for demonstrating preferred trajectories for situations where the robot is not within a person's reach. Furthermore, future work should verify whether our current model is sufficient in new navigation contexts or if multiple models are required for different scenarios. Finally, extending our approach to handle dynamic situations with moving people would be worthwhile for more complex navigation scenarios in populated spaces.

# Chapter 6

# Discussion

This thesis presented advanced perception and learning methods addressing three main challenges that mobile robots face in human-centered environments. Firstly, robots in populated spaces encounter people with different needs and requirements. To address this challenge, we devised a novel people detection module that distinguishes people according to the mobility aids they use. The second challenge is that many people in populated spaces have very little experience with robots. We introduced force control as an intuitive way to interact with a mobile robot and teach individual navigation preferences. The unstructured, cluttered nature of populated environments is the third challenge we addressed by enabling mobile robots to learn better environment models based on perceived collisions with obstacles. We presented related works and thorough derivations for all our methods and evaluated them in diverse experiments with a special focus on their applicability in real-world scenarios.

We first presented a method for detecting people in images and classifying them based on their mobility aids. Our approach uses a recent deep learning-based object detection framework. In addition to detecting people in images, mobile robots require knowledge of their positions in a world reference frame to interact with them. To this end, we extended the object detection framework to predict the Cartesian positions of people. Previous approaches relied on depth data, e.g., from RGB-D cameras, to obtain people's positions in a world reference frame. In contrast, our approach can work on pure RGB data to overcome the limited sensing range of depth sensors. Furthermore, we introduced a probabilistic people tracker that estimates the position and velocity of a person and reasons about the perceived mobility aids category over time to account for occlusions and false detections. A set of experiments evaluated the performance of our detection and tracking modules. Furthermore, we presented a real-world experiment where a mobile robot guided participants either to the elevator or to the stairs, depending on the perceived mobility aids category. The experiment underlines how perceiving people's mobility aids can help mobile robots provide better assistance. Our novel mobility aids people detector enables robots to respond to the individual needs and requirements of the people that service robots will encounter on a daily basis.

Next, we investigated a novel sensory concept that allows mobile robots to perceive collisions and interaction forces. Due to the unstructured and dynamic nature of populated environments, mobile robots cannot entirely avoid collisions with obstacles or with people. Our sensory concept uses a 6-DoF force-torque sensor between the omnidirectional robot base and the solid shell to perceive the magnitude, impact point, and direction of a force acting on the shell. We further devised a learning-based filtering method to distinguish the impact forces from disturbances caused by the motion of the robot and oscillations. With our sensory concept, mobile robots can perceive interaction forces during autonomous motion. This sensing capability is useful for mobile robots in populated environments because they can appropriately react to contacts and collisions to avoid harm and damage. Furthermore, physical interaction can provide an intuitive way to communicate with a mobile robot.

The perception of impact forces during autonomous motion allows robots to react to collisions with obstacles when they occur. However, collisions are still undesirable events that the robot should try to avoid. Thus, we introduced a learning method that allows robots to not only passively react to collisions with obstacles but to learn from collision events so they can prevent them in the future. We focused on mobile robots with planar laser rangefinders popular for indoor localization. Unfortunately, robots with such sensor configurations typically miss parts of obstacles, e.g., they detect table and chair legs but miss their tops. To address this issue, we presented a novel method that predicts where collisions may occur in 2D occupancy maps from planar laser rangefinders. To this end, we turned the collisions perceived with the force-sensitive robot shell into training examples and trained a neural network to classify occupancy map patches. The network can be trained on single collision events and afterward efficiently segment full occupancy maps. Our experiments confirm that our approach can predict collisions that are not visible in the 2D occupancy map, enabling our robot to avoid them and navigate safely. Furthermore, our setup allows integrating further training examples after an initial training phase with the onboard sensor, which is essential in changing environments. Our novel approach enables robots with incomplete sensor information to learn improved environment models over time, leading to safer and more reliable autonomous navigation.

After we explored learning from collision events, we investigated how the perception of interaction forces can improve the interaction between humans and robots in populated spaces. Robots that navigate among people need to consider their safety and also care for their comfort by navigating in a socially compliant way. However, people's navigation preferences can vary depending on many factors like their culture, age, or the appearance of the robot, and manually tuning the behavior for every robot and context is an elaborate process. Instead, we presented a method where mobile robots learned socially compliant navigation via physical interaction. In our method, the robot navigated autonomously and people could teach the robot their preferred navigation behavior by pushing it along their preferred trajectory. Based on the interaction, the robot updated its navigation reward function via inverse reinforcement learning to adapt to a person's preference. We tested our method in a user study and found that the robot could improve its behavior based on force feedback. Most participants rated force feedback as easy and intuitive. As it does not require an external control device, such as a joystick, everyone in reach of the robot can interact with it. Also, the gesture of pushing the robot along is very intuitive, making it well suited for people with little experience with robots.

The approaches presented in this thesis are pieces of the puzzle of service robots that robustly and reliably operate in human-centered spaces. However, many additional steps are required to achieve this ambitious goal. This thesis showcases and argues for the importance of perceiving interaction forces and contacts with obstacles. While the 6-DoF force-torque sensor used for the force-sensitive shell is a widely-used robot component, it is likely too expensive for service robots for wide-area use. Furthermore, force-torque sensors with strain gauges are sensitive to overload, and robots that come in contact with people require very robust hardware. Especially children have been found to sometimes physically attack service robots [17]. Thus, we need further research and development of robust and cost-efficient force-sensitive mobile platforms.

In this thesis, the robot was able to learn from both accidental collisions with obstacles and intended physical interaction. There are many other areas of applications for force-sensitive mobile robots that are worthwhile to explore, such as a mobile robot platform that can support elderly or disabled people while walking. Furthermore, a force-sensitive mobile platform could be interesting for physiotherapy or rehabilitation.

In our work, the robot treated collisions and intended physical interaction differently. However, we did not address how the robot would distinguish both types of contacts. Other types of contacts, such as an unintended physical interaction where a person leans onto the robot, should not even evoke a robot reaction. Future work should investigate how different types of contact could be distinguished and define suitable reaction strategies for them. We expect the ability to perceive the 3D impact point and direction of interaction forces to play an important role here; Physical interaction likely happens in reach of a person's hand, while an accidental collision could impact anywhere on the shell.

This thesis also showed how mobile robots that perceive a person's mobility aids can provide better assistance. We introduced a dataset with people using different kinds of mobility aids. While the dataset includes color-encoded depth images recorded inside a hospital, most images were collected in our robot hall and featured our colleagues using props. The dataset is thus a good starting point, but more research is required to confirm that it sufficiently represents people of all ages, genders, and ethnicities. In a 2019 study, Obermeyer *et al.* [104] revealed a strong racial bias in algorithms for health care. Additionally, Buolamwini and Gebru [18] identified significant discrepancies in image classification performance, discriminating people from certain population groups. We need to make sure that a service robot can provide the same level of assistance and care to all people it encounters. Thus, we strongly advocate for more research in that direction and admit that diversification of our Mobility Aids dataset is likely necessary to achieve an unbiased detection performance.

In our mobility aids research, we presented a scenario where the robot guided visitors either to the stairs or to the elevator. People tracking further enabled the robot to adapt its velocity to its followers. Our example was predefined, and we specified the navigation goals and conditions by hand. An autonomous service robot will need a much richer world model or the ability to reason flexibly about the limitations associated with the use of mobility aids and the consequences for its navigation planning. Much more research is needed to enable such flexible, context-aware adaptation. Other factors still limit the use of elaborate perception techniques that require considerable GPU power, foremost the extensive battery consumption. We need to address those challenges to enable robust, flexible, inclusive, and truly autonomous operation of mobile robots in human-centered environments.

# List of Figures

# List of Tables

# Bibliography

[1] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 391–398, 2012.

[2] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari. The history began from AlexNet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

[3] R. C. Arkin and R. R. Murphy. Autonomous navigation in a manufacturing environment. *IEEE Transactions on Robotics and Automation*, 6(4):445–454, 1990.

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

[5] B. Axelrod, L. P. Kaelbling, and T. Lozano-Pérez. Provably safe robot navigation with obstacle uncertainty. *International Journal of Robotics Research (IJRR)*, 37 (13-14):1760–1774, 2018.

[6] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(12):2481–2495, 2017.

[7] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan. Learning robot objectives from physical human robot interaction. In *Conference on Robot Learning (CoRL)*, pages 217–226, 2017.

[8] M. Bajracharya, B. Moghaddam, A. Howard, S. Brennan, and L. H. Matthies. A fast stereo-based system for detecting and tracking pedestrians from a moving vehicle. *International Journal of Robotics Research (IJRR)*, 28(11-12):1466 – 1485, 2009.

[9] H. Baltzakis, A. Argyros, and P. Trahanias. Fusion of laser and visual data for robot motion planning and collision avoidance. *Machine Vision and Applications*, 15:92–100, 2003.

[10] J. Bačík, F. Ďurovský, M. Biroš, K. Kyslan, D. Perduková, and S. Padmanaban. Pathfinder – development of automated guided vehicle for hospital logistics. *IEEE Access*, 5:26892–26900, 2017.

[11] M. Bennewitz, W. Burgard, and S. Thrun. Adapting navigation strategies using motions patterns of people. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2000–2005, 2003.

[12] L. Beyer, A. Hermans, and B. Leibe. DROW: Real-time deep learning-based wheelchair detection in 2D range data. *IEEE Robotics and Automation Letters (RA-L)*, 2(2):585–592, 2017.

[13] A. Bicchi, J. K. Salisbury, and D. L. Brock. Contact sensing from force measurements. *International Journal of Robotics Research (IJRR)*, 12(3):249–262, 1993.

[14] Bosch. Robotic lawnmower - indego 350. `https://www.bosch-diy.com/gb/en/p/indego-350-06008b0000-v47626`. Accessed 10-11-2020.

[15] L. Bourdev, S. Maji, and J. Malik. Describing people: A poselet-based approach to attribute classification. In *International Conference on Computer Vision (ICCV)*, pages 1543–1550, 2011.

[16] A. Brunetti, D. Buongiorno, G. F. Trotta, and V. Bevilacqua. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, 300:17–33, 2018.

[17] D. Brščić, H. Kidokoro, Y. Suehiro, and T. Kanda. Escaping from children's abuse of social robots. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 59–66, 2015.

[18] J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, pages 77–91, 2018.

[19] W. Burgard, A. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *National Conference on Artificial Intelligence (AAAI)*, pages 11–18, 1998.

[20] J. T. Butler and A. Agah. Psychological effects of behavior patterns of a mobile personal robot. *Autonomous Robots*, 10(2):185–202, 2001.

[21] F. Capezio, F. Mastrogiovanni, A. Scalmato, A. Sgorbissa, P. Vernazza, T. Vernazza, and R. Zaccaria. Mobile robots in hospital environments: an installation case study. In *European Conference on Mobile Robots (ECMR)*, pages 61–68, 2011.

[22] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference (BMVC)*, 2014.

[23] L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[24] T. L. Chen and C. C. Kemp. Lead me by the hand: Evaluation of a direct physical interface for nursing assistant robots. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 367–374, 2010.

[25] M. Cortes. Predicting obstacle footprints from 2D occupancy maps by learning from physical interactions. Bachelor's thesis, University of Freiburg, Department of Computer Science, 2018.

[26] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 379–387, 2016.

[27] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. *International Journal of Computer Vision (IJCV)*, 37:175–185, 2000.

[28] K. Dautenhahn, S. Woods, C. Kaouri, M. L. Walters, K. L. Koay, and I. Werry. What is a robot companion - friend, assistant or butler? In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1192–1197, 2005.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[30] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 304–311, 2009.

[31] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(4):743–761, 2012.

[32] C. Dondrup, N. Bellotto, F. Jovan, and M. Hanheide. Real-time multisensor people tracking for human-robot spatial interaction. In *Workshop on Machine Learning for Social Robotics at IEEE International Conference on Robotics and Automation (ICRA)*, pages 26–31, 2015.

[33] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. Multimodal deep learning for robust RGB-D object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 681–687, 2015.

[34] J. Eriksson, M. J. Mataric, and C. J. Winstein. Hands-off assistive robotics for post-stroke arm rehabilitation. In *International Conference on Rehabilitation Robotics (ICORR)*, pages 21–24, 2005.

[35] A. Ess, B. Leibe, K. Schindler, and L. Van Gool. Robust multiperson tracking from a mobile platform. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(10):1831–1846, 2009.

[36] J. M. Evans. HelpMate: An autonomous mobile robot courier for hospitals. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1695–1700, 1994.

[37] M. Everingham, S. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes challenge: A retrospective. *International Journal of Computer Vision (IJCV)*, 111(1):98–136, 2015.

[38] J. B. Fernando, T. Tanigawa, E. Naito, K. Yamagami, and J. Ozawa. Collision avoidance path planning for hospital robot with consideration of disabled person's movement characteristic. In *IEEE Global Conference on Consumer Electronics (GCCE)*, pages 392–396, 2012.

[39] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research (IJRR)*, 17(7):760–772, 1998.

[40] M. E. Foster, R. Alami, O. Gestranius, O. Lemon, M. Niemelä, J.-M. Odobez, and A. K. Pandey. The MuMMER project: Engaging human-robot interaction in real-world public spaces. In *International Conference on Social Robotics (ICSR)*, pages 753–763, 2016.

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

[42] D. Fox, W. Burgard, S. Thrun, and A. B. Cremers. A hybrid collision avoidance method for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1238–1243, 1998.

[43] J. Frémy, F. Ferland, M. Lauria, and F. Michaud. Force-guidance of a compliant omnidirectional non-holonomic platform. *Robotics and Autonomous Systems*, 62 (4):579–590, 2014.

[44] R. Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[45] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[46] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron - GitHub. `https://github.com/facebookresearch/detectron`, Jan. 2018. Accessed: 17-12-2020.

[47] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[48] B. Graf, M. Hans, and R. D. Schraft. Care-O-bot II – development of a next generation robotic home assistant. *Autonomous Robots*, 16(2):193–205, 2004.

[49] V. Guizilini and F. Ramos. Learning to reconstruct 3D structures for occupancy mapping. In *Robotics: Science and Systems (RSS)*, 2017.

[50] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger. Requirements for safe robots: Measurements, analysis and new insights. *International Journal of Robotics Research (IJRR)*, 28(11–12):1507–1527, 2009.

[51] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1557–1563, 2003.

[52] R. W. Hamming. *Digital filters*. Dover Civil and Mechanical Engineering Series. Dover Publications, 1998.

[53] A. Handa, V. Pătrăucean, S. Stent, and R. Cipolla. SceneNet: An annotated model generator for indoor scene understanding. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5737–5743, 2016.

[54] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[55] M. Herman, T. Gindele, J. Wagner, F. Schmitt, C. Quignon, and W. Burgard. Learning high-level navigation strategies via inverse reinforcement learning: A comparative analysis. In *Advances in Artificial Intelligence*, pages 525–534, 2016.

[56] Y. Hirata, T. Takagi, K. Kosuge, H. Asama, H. Kaetsu, and K. Kawabata. Map-based control of distributed robot helpers for transporting an object in cooperation with a human. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3010–3015, 2001.

[57] Y. Hirata, T. Baba, and K. Kosuge. Motion control of omni-directional type walking support system "Walking Helper". In *IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN)*, pages 85–90, 2003.

[58] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3296–3297, 2017.

[59] IFR International Federation of Robotics. *World robotics 2019 industrial robots*. VDMA Verlag, 2019.

[60] IFR International Federation of Robotics. *World robotics 2019 service robots*. VDMA Verlag, 2019.

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

[62] J. L. Jones. Robots at the tipping point: The road to iRobot Roomba. *IEEE Robotics & Automation Magazine*, 13(1):76–78, 2006.

[63] M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal. Learning objective functions for manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[64] T. Kanda, M. Shiomi, Z. Miyashita, H. Ishiguro, and N. Hagita. A communication robot in a shopping mall. *IEEE Transactions on Robotics (T-RO)*, 26(5):897–913, 2010.

[65] O. Khatib. Mobile manipulation: The robotic assistant. *Robotics and Autonomous Systems*, 26(2–3):175–183, 1999.

[66] B. Kim and J. Pineau. Socially adaptive path planning in human environments using inverse reinforcement learning. *International Journal of Social Robotics*, 8: 51–66, 2016.

[67] K. S. Kim, T. Llado, and L. Sentis. Full-body collision detection and reaction with omnidirectional mobile platforms: A step towards safe human-robot interaction. *Autonomous Robots*, 40:325–341, 2016.

[68] Y. Kim and B. Mutlu. How social distance shapes human-robot interaction. *International Journal of Human-Computer Studies*, 72(12):783–795, 2014.

[69] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[70] R. Kirby, R. Simmons, and J. Forlizzi. COMPANION: A constraint optimizing method for person-acceptable navigation. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2009.

[71] M. Kollmitz. omni_path_follower - GitHub. `https://github.com/marinaKollmitz/omni_path_follower`, Nov. 2017. Accessed: 17-12-2020.

[72] M. Kollmitz, K. Hsiao, J. Gaa, and W. Burgard. Time dependent planning on a layered social cost map for human-aware robot navigation. In *European Conference on Mobile Robots (ECMR)*, pages 1–6, 2015.

[73] M. Kollmitz, D. Büscher, T. Schubert, and W. Burgard. Whole-body sensory concept for compliant mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5429–5435, 2018.

[74] M. Kollmitz, A. Eitel, A. Vasquez, and W. Burgard. Deep 3D perception of people and their mobility aids. *Robotics and Autonomous Systems*, 114:29–40, 2019.

[75] M. Kollmitz, D. Büscher, and W. Burgard. Predicting obstacle footprints from 2D occupancy maps by learning from physical interactions. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 10256–10262, 2020.

[76] M. Kollmitz, T. Koller, J. Boedecker, and W. Burgard. Learning human-aware robot navigation from physical interaction via inverse reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11025–11031, 2020.

[77] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[78] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: Science and Systems (RSS)*, 2012.

[79] R. R. Labbe. Kalman and bayesian filters in python - GitHub. `https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python`, Apr. 2014. Accessed: 17-12-2020.

[80] M. Lawitzky, A. Mortl, and S. Hirche. Load sharing in human-robot cooperative manipulation. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 185–191, 2010.

[81] M. Lawitzky, J. R. Medina, D. Lee, and S. Hirche. Feedback motion planning and learning from demonstration in physical robotic assistance: differences and synergies. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3646–3652, 2012.

[82] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.* Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[83] J. Li, X. Liang, S. Shen, T. Xu, J. Feng, and S. Yan. Scale-aware Fast R-CNN for pedestrian detection. *IEEE Transactions on Multimedia*, 20(4):985–996, 2018.

[84] C. Lichtenthäler, A. Peters, S. Griffiths, and A. Kirsch. Social navigation - identifying robot navigation patterns in a path crossing scenario. In *International Conference on Social Robotics (ICSR)*, pages 84–93, 2013.

[85] T. Linder, S. Wehner, and K. O. Arras. Real-time full-body human gender recognition in (RGB)-D data. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3039–3045, 2015.

[86] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5512–5519, 2016.

[87] T. Linder, K. Y. Pfeiffer, N. Vaskevicius, R. Schirmer, and K. O. Arras. Accurate detection and 3D localization of humans using a novel yolo-based rgb-d fusion approach and synthetic training data. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1000–1006, 2020.

[88] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

[89] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[90] D. V. Lu and W. D. Smart. Towards more efficient navigation for robots and humans. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1707–1713, 2013.

[91] M. Luber, L. Spinello, and K. O. Arras. People tracking in RGB-D data with on-line boosted target models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3844–3849, 2011.

[92] M. Luber, L. Spinello, J. Silva, and K. O. Arras. Socially-aware robot navigation: A learning approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 902–907, 2012.

[93] J. Lundell, F. Verdoja, and V. Kyrki. Hallucinating robots: Inferring obstacle distances from partial laser measurements. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4781–4787, 2018.

[94] D. Maier, A. Hornung, and M. Bennewitz. Real-time navigation in 3D environments based on depth camera data. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 692–697, 2012.

[95] L. Manuelli and R. Tedrake. Localizing external contact using proprioceptive sensors: The Contact Particle Filter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5062–5069, 2016.

[96] E. Marder-Eppstein. Navigation - ROS wiki. `http://wiki.ros.org/navigation`, July 2009. Accessed: 17-12-2020.

[97] MathWorks. designfilt: Design digital filters. `https://www.mathworks.com/help/releases/R2017a/signal/ref/designfilt.html`. Accessed: 07-09-2020.

[98] O. Mees, A. Eitel, and W. Burgard. Choosing smartly: Adaptive multimodal fusion for object detection in changing environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 151–156, 2016.

[99] D. Mitzel and B. Leibe. Close-range human detection for head-mounted cameras. In *British Machine Vision Conference (BMVC)*, 2012.

[100] M. Montemerlo, J. Pineau, N. Roy, S. Thrun, and V. Verma. Experiences with a mobile robotic guide for the elderly. In *National Conference on Artificial Intelligence (AAAI)*, pages 587–592, 2002.

[101] J. Mumm and B. Mutlu. Human-robot proxemics: Physical and psychological distancing in human-robot interaction. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 331–338, 2011.

[102] M. Munaro and E. Menegatti. Fast RGB-D people tracking for service robots. *Autonomous Robots*, 37:227–242, 2014.

[103] R. Muñoz-Salinas, E. Aguirre, and M. García-Silvente. People detection and tracking using stereo vision and color. *Image and Vision Computing*, 25(6):995–1007, 2007.

[104] Z. Obermeyer, B. Powers, C. Vogeli, and S. Mullainathan. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464): 447–453, 2019.

[105] G. L. Oliveira, A. Valada, C. Bollen, W. Burgard, and T. Brox. Deep learning for human part discovery in images. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1634–1641, 2016.

[106] A. Oliver, S. Kang, B. C. Wünsche, and B. MacDonald. Using the kinect as a navigation sensor for mobile robotics. In *Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 509–514, 2012.

[107] P. Ondrúška and I. Posner. Deep tracking: Seeing beyond seeing using recurrent neural networks. In *National Conference on Artificial Intelligence (AAAI)*, pages 3361–3367, 2016.

[108] A. Ošep, W. Mehner, M. Mathias, and B. Leibe. Combined image- and world-space tracking in traffic scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1988–1995, 2017.

[109] E. Pacchierotti, H. I. Christensen, and P. Jensfelt. Evaluation of passing distance for social robots. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 315–320, 2006.

[110] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. ENet: a deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[111] C. Plagemann, D. Fox, and W. Burgard. Efficient failure detection on mobile robots using particle filterswith gaussian process proposals. *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 2185–2190, 2007.

[112] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4151–4160, 2017.

[113] M. E. Pollack, L. Brown, D. Colbry, C. Orosz, B. Peintner, S. Ramkrishnan, S. Engberg, J. Matthews, J. Dunbar-Jacob, C. McCarthy, S. Thrun, M. Montemerlo, J. Pineau, and N. Roy. Pearl: A mobile robotic assistant for the elderly. In *AAAI Workshop on Automation as Eldercare*, 2002.

[114] E. Prassler, J. Scholz, and P. Fiorini. A robotics wheelchair for crowded public environment. *IEEE Robotics & Automation Magazine*, 8(1):38–45, 2001.

[115] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing*. Pearson Prentice Hall, 2007.

[116] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):297–330, 2020.

[117] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[118] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[119] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 91–99, 2015.

[120] Robotic Industries Association. UNIMATE // the first industrial robot. `https://www.robotics.org/joseph-engelberger/unimate.cfm`. Accessed: 16-04-2020.

[121] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115: 211–252, 2015.

[122] A. M. Sabatini, V. Genovese, and E. Pacchierotti. A mobility aid for the support to walking and object transportation of people with motor impairments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1349–1354, 2002.

[123] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball. Deep learning with convolutional neural networks for EEG decoding and visualization. *Human Brain Mapping (HMB)*, 38(11):5391–5420, 2017.

[124] G. Sharma and F. Jurie. Learning discriminative spatial representation for image classification. In *British Machine Vision Conference (BMVC)*, pages 6.1–6.11, 2011.

[125] D. Shi, E. Collins, A. Donate, X. Liu, B. Goldiez, and D. Dunlap. Human-aware robot motion planning with velocity constraints. In *International Symposium on Collaborative Technologies and Systems (CTS)*, pages 490–497, 2008.

[126] M. Y. Shieh, J. C. Hsieh, and C. P. Cheng. Design of an intelligent hospital service robot and its applications. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 4377–4382, 2004.

[127] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

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

[129] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 190–198, 2017.

[130] M. Spenko, H. Yu, and S. Dubowsky. Robotic personal aids for mobility and monitoring for the elderly. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(3):344–351, 2006.

[131] L. Spinello and K. O. Arras. People detection in RGB-D data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3838–3843, 2011.

[132] C. Sprunk, B. Lau, P. Pfaff, and W. Burgard. An accurate and efficient navigation system for omnidirectional robots in industrial environments. *Autonomous Robots*, pages 1–21, 2016.

[133] E. Steinfeld, S. Schroeder, and M. Bishop. Accessible buildings for people with walking and reaching limitations. Technical report, U.S. Department of Housing and Urban Development, April 1979.

[134] J. Stillig. The intelligent factory floor. `https://community.boschrexroth.com/t5/Rexroth-Blog/The-intelligent-factory-floor/ba-p/9171`, Sept. 2019. Accessed: 10-11-2020.

[135] P. Sudowe, H. Spitzer, and B. Leibe. Person attribute recognition with a jointly-trained holistic CNN model. In *IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 329–337, 2015.

[136] M. Svenstrup, T. Bak, and H. J. Andersen. Trajectory planning for robots in dynamic human environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4293–4298, 2010.

[137] M. Takahashi, T. Suzuki, H. Shitamoto, T. Moriguchi, and K. Yoshida. Developing a mobile robot for transport applications in the hospital domain. *Robotics and Autonomous Systems*, 58(7):889–899, 2010.

[138] L. Takayama and C. Pantofaru. Influences on proxemic behaviors in human-robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5495–5502, 2009.

[139] R. Tasaki, M. Kitazaki, J. Miura, and K. Terashima. Prototype design of medical round supporting robot "Terapio". In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 829–834, 2015.

[140] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[141] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and et al. MINERVA: A second-generation museum tour-guide robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1999–2005, 1999.

[142] P. Trautman and A. Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 797–803, 2010.

[143] R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers, V. Evers, M. Fiore, H. Hung, O. A. I. Ramírez, M. Joosse, H. Khambhaita, T. Kucner, B. Leibe, A. J. Lilienthal, T. Linder, M. Lohse, M. Magnusson, B. Okal, L. Palmieri, U. Rafi, M. van Rooij, and L. Zhang. SPENCER: A socially aware service robot for passenger guidance and help in busy airports. In *Field and Service Robotics*, pages 607–622, 2016.

[144] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International Journal of Computer Vision (IJCV)*, 104(2):154–171, 2013.

[145] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen. Shape completion enabled robotic grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2442–2447, 2017.

[146] A. Vasquez. People perception and classification for mobile robots in populated environments. Master's thesis, University of Freiburg, Department of Computer Science, 2017.

[147] A. Vasquez, M. Kollmitz, A. Eitel, and W. Burgard. Deep detection of people and their mobility aids for a hospital robot. In *European Conference on Mobile Robots (ECMR)*, pages 1–7, 2017.

[148] F. Verdoja, J. Lundell, and V. Kyrki. Deep network uncertainty maps for indoor navigation. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 112–119, 2018.

[149] K. Wada, T. Shibata, T. Musha, and S. Kimura. Robot therapy for elders affected by dementia. *IEEE Engineering in Medicine and Biology Magazine*, 27(4):53–60, 2008.

[150] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.

[151] C. Weinrich, T. Wengefeld, C. Schroeter, and H. Gross. People detection and distinction of their walking aids in 2D laser range data based on generic distance-invariant features. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 767–773, 2014.

[152] M. Wulfmeier, P. Ondruska, and I. Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

[153] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016.

[154] H. Zhang, C. Reardon, and L. E. Parker. Real-time multiple human perception with color-depth cameras on a mobile robot. *IEEE Transactions on Cybernetics*, 43(5): 1429–1441, 2013.

[155] L. Zhang, L. Lin, X. Liang, and K. He. Is Faster R-CNN doing well for pedestrian detection? In *European Conference on Computer Vision (ECCV)*, pages 443–457, 2016.

[156] S. Zhang, R. Benenson, and B. Schiele. Citypersons: A diverse dataset for pedestrian detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3221, 2017.

[157] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6230–6239, 2017.

[158] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *National Conference on Artificial Intelligence (AAAI)*, pages 1433–1438, 2008.

[159] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa. Planning-based prediction for pedestrians. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3931–3936, 2009.