Camera-Based Humanoid Robot Navigation

Daniel Maier

Technische Fakultät
Albert-Ludwigs-Universität Freiburg im Breisgau

UNI FREIBURG

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Betreuer: Prof. Dr. Maren Bennewitz

Camera-Based Humanoid Robot Navigation

Daniel Maier

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

Dekan:	Prof. Dr. Georg Lausen
Erstgutachter:	Prof. Dr. Maren Bennewitz
Zweitgutachter:	Prof. Dr. Andrea Cherubini
Tag der Disputation:	10. April 2015

Abstract

Humanoid robots possess unique locomotive and manipulation capabilities which makes them predestined as assistants in households or even in disaster scenarios. Their legs allow them to walk across rough terrain and clutter, climb elevations, or pass narrow passages. At the same time, with their arms, they could deliver objects, remove debris, or even use power tools to cut through walls. However, to enable this kind of behavior, novel navigation techniques are required that exploit the special capabilities of humanoids.

One of the great challenges in navigation is that a robot always acts under uncertainty. It possesses only imperfect knowledge about itself and its environment, yet this knowledge is fundamental. Motions and observations are affected by noise and need to be handled appropriately. Data has to be associated in the presence of ambiguities to obtain consistent representations of the environment. For humanoid robots, the problem aggravates as the kinematic complexity that needs to be handled is higher compared to wheeled robots. The shaking motion of the humanoids adds further errors to the sensor data, making it harder to interpret. Additional constraints like balance and payload need to be considered.

In this thesis, we present novel methods that contribute to the development of autonomous humanoid robots. Hereby, we focus on cameras as primary sensor. First, we describe a method to self-calibration of the robot's kinematic model. Hereby, our approach automatically selects appropriate calibration postures. Further, we present a method to identify safe areas for the robot to step onto based on self-supervised classification of camera images. Additionally, we describe an integrated navigation system for robots equipped with depth cameras. The approach estimates the robot's 6D pose within a map, constructs a volumetric representation of the unknown parts of the environment and plans collision-free paths to a target location. We introduce extensions that allow navigation in challenging, cluttered scenarios based on anytime footstep planning. Thereby we enable the robot to step over or onto obstacles and traverse narrow passages. Finally, we demonstrate a method that enables accurate manipulation by tracking the pose of objects in the camera images.

All of our techniques are implemented and thoroughly evaluated on a Nao humanoid. Our contributions advance the state-of-the-art in humanoid robot navigation and enable autonomous navigation capabilities even for affordable humanoids.

Zusammenfassung

Im letzten Jahrzehnt stieg die Anzahl der Roboter, die im Haushalt eingesetzt werden, deutlich an. Dabei dominieren zwei Arten von Robotern, nämlich solche die Staubsaugen und solche, die Rasen mähen. Dass gerade diese beiden Typen so verbreitet sind, mag daran liegen, dass diese Aufgaben anhand relativ simpler Durchführungsstrategien umgesetzt werden können. Ein Roboter etwa, der mit konstanter Geschwindigkeit durch einen Raum fährt und dabei zufällig seine Richtung ändert, hat irgendwann jede erreichbare Fläche desselben überquert. Natürlich gibt es mittlerweile auch Roboter, die ihre Aufgabe zielgerichteter erledigen, zum Beispiel indem sie ihre Position innerhalb eines Raumes bestimmen und so den Bewegungsablauf optimieren. Dennoch sind solche Roboter in ihrer Funktionsvielfalt sehr eingeschränkt und weit von dem entfernt, was man aus der Fiktion als Service-Roboter kennt.

Wir hätten gerne einen Roboter, der vielfältige Aufgaben im Haushalt übernimmt – etwa Wäsche aufhängen, Geschirr waschen oder eben auch den Boden saugen. Er soll uns im Alltag zur Hand gehen und dabei auch anspruchsvolle Aufgaben erledigen. Vor dem Hintergrund von Katastrophen wie der des havarierten Atomkraftwerks in Fukushima stellt sich zudem die Frage, ob Roboter nicht auch in solchen Szenarien eingesetzt werden könnten, um Schlimmeres zu verhindern und Menschenleben zu schützen. Denn ein Roboter lässt sich ersetzen, ein Mensch dagegen nicht. So könnten Roboter in Gefahrensituationen eingesetzt werden und beispielsweise wichtige Informationen über den Zustand beschädigter Gebäude liefern, Wege freiräumen, Feuer löschen oder Notstrom-Aggregate in Stand setzen. Angesichts dieser Möglichkeiten ist es sinnvoll, sich mit der Weiterentwicklung von autonomen Robotern zu beschäftigen. Hierzu leistet diese Arbeit einen wichtigen Beitrag.

Nach der Katastrophe von Fukushima wurde die Robotics Challenge von der US-amerikanischen DARPA (Defense Advanced Research Projects Agency) ins Leben gerufen. Der Wettkampf soll die Entwicklung von Robotern für den Einsatz in Katastrophenszenarien fördern. Dabei richten sich die Anforderungen an die Roboter an Tätigkeiten aus, die in solchen Szenarien von Bedeutung sind. Dazu gehört es etwa, unwegsames Gelände zu überschreiten, Trümmerteile wegzuräumen, Leitern zu erklimmen, oder gar Wände mit entsprechendem Werkzeug einzureißen. Die hohe Komplexität der genannten Aufgaben

hat das Interesse einer Reihe von internationalen Forschungsgruppen geweckt und zur Teilnahme am Wettbewerb animiert. Interessant dabei ist, dass sich beinahe alle Teilnehmer mit der Entwicklung von menschenähnlichen, also *humanoider* Robotern, befasst haben. Dies könnte daran liegen, dass die Roboter aufgrund ihrer Physis in besonderem Maße für solche Aufgaben geeignet erscheinen.

Eines der Hauptprobleme bei der Entwicklung von autonomen Robotern ist die Navigation. Um sich in seiner Umgebung zurechtfinden zu können, muss ein Roboter seine eigene Position bestimmen, Hindernisse erkennen und entsprechen darauf reagieren sowie eine interne Karte seiner Umgebung anlegen können. Im Hinblick auf die genannten Aufgaben muss er zudem dazu fähig sein, Objekte wahrzunehmen und zu manipulieren, etwa um Trümmerteile wegzuräumen, die den Eingang zu einem Gebäude versperren. Die vielfältigen Teilprobleme der Navigation müssen gelöst werden, damit Roboter autonom agieren können. Die Schwierigkeit liegt dabei darin, dass sämtliches Handeln und Wahrnehmen des Roboters von Unsicherheit betroffen ist. Das betrifft sowohl das Ausführen von Aktionen als auch die Wahrnehmung oder Perzeption der Umgebung. Zum Beispiel sind Sensordaten oft fehlerhaft und müssen dennoch korrekt interpretiert werden. Verschiedene Daten sind hierbei miteinander zu assoziieren, wobei Ambiguitäten auftreten können. Das Ausführen von Bewegungen ist ebenso fehleranfällig, zum Beispiel wenn der Roboter aufgrund der Materialbeschaffenheit seiner Fußsohlen auf dem Boden rutscht. Des Weiteren liegen dem Roboter keine perfekten Modelle über sich selbst und die Gegenstände in seiner Umgebung vor. Das kann an Fertigungstoleranzen bei der Herstellung des Roboters liegen oder weil nicht alle physikalischen Zusammenhänge modelliert oder bekannt sind. Speziell bei humanoiden Robotern muss mit weiteren Schwierigkeiten umgegangen werden. Dazu gehören zum Beispiel die hohe Komplexität, d.h. die Anzahl der Freiheitsgrade, die zu kontrollieren sind. Die Interpretation der Daten wird zusätzlich dadurch erschwert, dass die Sensormessungen stark vom Laufverhalten des Roboters beeinflusst werden. Außerdem müssen weitere Bedingungen, etwa bezüglich der Stabilität oder Balance des Roboters, erfüllt sein. Grundsätzlich müssen also folgende Fragen beantwortet werden, um eine zuverlässige und robuste Navigation für humanoide Roboter zu ermöglichen:

- Wie kann der Roboter sein Wissen über sich selbst verbessern?
- Woher weiß der Roboter, welche Bereiche er sicher betreten kann?
- Wie kann der Roboter wissen, wo er sich in seiner Umgebung befindet?
- Wie kann er seine Umgebung überhaupt wahrnehmen und geeignet intern repräsentieren?

- Wie kann der Roboter Hindernisse auf seinem Weg erkennen und angemessen darauf reagieren?
- Wie kann ein humanoider Roboter seine inhärenten Bewegungsfähigkeiten nutzen um schwierige Passagen zu meistern?
- Wie kann der Roboter Objekte wahrnehmen und manipulieren?

Ziel dieser Arbeit ist es, diese Fragen zu beantworten. Zuerst stellen wir ein Verfahren zur Selbstkalibrierung für humanoide Roboter vor. Hierbei beobachtet der Roboter mithilfe seiner Kamera Markierungen an seinen Endeffektoren. Unser Ansatz minimiert anschließend den Fehler zwischen Modellvorhersage und Beobachtungen um die Kalibrierungsparameter zu bestimmen. Dazu gehören die Referenzierung der Gelenk-Inkrementalgeber sowie die intrinsischen und extrinsischen Parameter der Kamera. Darüber hinaus präsentieren wir eine Methode, die Roboterkonfigurationen erzeugt und geschickt auswählt, sodass nur wenige Konfigurationen notwendig sind, um die Parameter zu bestimmen. Das Wissen, welches der Roboter dadurch über sich selbst gewinnt, verbessert seine Fähigkeit, verschiedene Beobachtungen zueinander in Relation zu setzen. Diese Fähigkeit bildet die Grundlage für alle Navigationsfähigkeiten.

Des Weiteren stellen wir eine Methode zur Klassifikation der Traversierbarkeit anhand von Kamerabildern vor. Wir klassifizieren das gesamte Kamerabild mittels erscheinungsbasierten Klassifikatoren, welche selbständig, d.h. ohne manuelles Training, lernen. Dazu erkennt unser Ansatz anhand eines geometrischen Modells Merkmale in den Kamerabildern des Roboters, welche planare Bereiche auf dem Boden repräsentieren. Darüber hinaus wird anhand der klassifizierten Kamerabilder eine Belegtheitskarte der Umgebung erstellt. Bei diesem Ansatz werden lediglich Daten einer monokularen Kamera und der Odometrie verwendet, wodurch er auf vielen Robotern einsetzbar ist.

Für Roboter, die mit einer Tiefenkamera ausgestattet sind, stellen wir ein integriertes Navigationssystem vor. Dieses besteht aus der 6D-Lokalisierung des Roboters in einer gegebenen Karte, einem volumetrischen Kartierungsverfahren für unbekannte oder nicht-statische Bereiche der Umgebung, sowie einem Pfadplanungsansatz zur Vermeidung von Kollisionen. Das System ist insbesondere für die Navigation in komplexen Szenarien mit mehreren Stockwerken ausgelegt.

Darauf aufbauend stellen wir Erweiterungen vor, die das Navigieren in Umgebungen ermöglicht, in denen dies zum Beispiel durch viele, verstreute Hindernisse oder enge Passagen erschwert ist. Zu diesen Techniken gehört eine Komponente zur Planung der Fußschritte des Roboters, die durch eine Methode ergänzt wird, die die Umgebung effizient auf mögliche Kollisionen testet und dabei den ganzen Körper des Roboters berücksichtigt und nicht nur etwa die Fußspuren. Dadurch wird der Roboter befähigt, auch Hindernisse

sicher zu übersteigen oder zu besteigen. Darüber hinaus reduziert unser Ansatz anhand von Messungen der Tiefenkamera den Drift der Odometrie, um konsistente Karten von unbekannten Umgebungen erstellen zu können.

Schließlich beschreibt die Arbeit ein Verfahren zur akkuraten Manipulation von Objekten. Konkret wird dieses am Beispiel eines Metallophon-spielenden Roboters implementiert. Dazu schätzt der Roboter die Pose (Position und Orientierung) sowohl der Klöppel in seinen Greifern als auch die des Instruments. Mittels inverser Kinematik (IK) werden Schlagkonfigurationen für die Arme des Roboters berechnet und ausgeführt. Der Roboter prüft selbständig anhand visueller und akustischer Signale, ob das Schlagen der Klangtaste erfolgreich war. Für jede Taste werden Konfigurationen gespeichert, so dass der Roboter anschließend ganze Lieder selbständig spielen kann.

Alle vorgestellten Methoden wurden praktisch implementiert und sorgfältig anhand eines Nao-Roboters evaluiert. Die gezeigten Ergebnisse bestätigen, dass diese Arbeit einen wichtigen Beitrag zur Navigation von humanoiden Robotern darstellt und deren Autonomie verbessert, selbst für günstige Roboter mit beschränkten Ressourcen und Hardware.

Acknowledgements

Writing a PhD thesis requires a lot of effort. Not only from the author but also from the people directly or indirectly involved in this process. In the following, I would like to express my gratitude to those people.

First of all, I would like to thank Maren Bennewitz for her valuable advice and guidance of my research. I learned and am still learning a lot from her experience. Her ideas and thinking had a tremendous influence on the content of this thesis. I am grateful that she made it possible for me to visit many interesting conferences and that I could learn so much in my time as a PhD student. I would also like to thank Andrea Cherubini for agreeing to review this thesis.

Furthermore, I would like to thank my office mates Armin Hornung and Felix Burget for the great atmosphere and many interesting discussions. This also holds for all previous and current members of the Humanoid Robot Lab – it was a pleasure to working with you. Many thanks also go to Wolfram Burgard and all members of the Autonomous Intelligent Systems group for kindly hosting our lab in the beginning. I would like to thank the members of the Social Robotics Lab for the great mood in our building and regularly having lunch together. Together, it was a lot easier to bear the infamous Mensa food.

Many thanks to Felix Burget, Caroline Lais, Armin Hornung, Luigi Palmieri, Christoph Spunk, and Bastian Steder for proof-reading and providing valuable feedback on earlier versions of this document. These people are consequently responsible for all remaining errors in this document – just kidding.

For their assistance with technical and administrative issues, I would further like to thank Susanne Bourjaillat, Michael Keser, Manuela Kniß, and Dagmar Sonntag.

Finally, I would like to give thanks to my friends and family for their support. Most of all, I would like to thank Caroline for her consideration, her support, and the great time we spend together.

The work on this thesis has been generously supported by the German Research Foundation (DFG) within the Research Training Group 1103 *Embedded Microsystems*, the *BrainLinks-BrainTools* Cluster of Excellence (grant number EXC 1086), and within the SFB/TR-8 *Spatial Cognition*. Their support is gratefully acknowledged.

Contents

Chapter 1

Introduction

Nowadays, robots find their way into private homes more and more frequently. One of the most widely used robots is the *Roomba* vacuuming robot by the company iRobot, who sold more than 10 million units up to today. Vacuuming robots are offered by dozens of companies in different price ranges. The same holds for autonomous lawn mowers. Aldebaran Robotics just released *Pepper*, a robot advertised as a companion to live with humans and socially interact with them.

Further possible applications for robots in private households bestride tasks like home care, shopping assistance, or personal service robots that can perform a multitude of tasks like cleaning dishes, doing the laundry, or vacuuming. Unfortunately, for these types of applications, there are no solutions available down to the present day. The problem here is that these tasks are very complex and require various capabilities from a robot. Vacuuming or lawn mowing robots, on the other hand, are very specialized, yet primitive robots. They are specifically designed for the task at hand and solve it in a simple fashion. For instance, lawn mowing robots typically require a border wire around the lawn to define their limits and perform partly random maneuvers within these limits.

Designing new robots for each task individually requires a lot of resources and it is questionable whether humans would be willing to buy and hold a robot for each application. From a scientific as well as from an economic point of view, it thus makes sense to ask whether one can do better. Can we not build robots, that are, like humans, capable of performing multiple, complex tasks? Can we not create robots, that adapt to their environment and cope with it, instead of making the environment compatible with the robots?

Recently, there has been some progress in answering these questions. One of the most well known robots is the *ASIMO* study by Honda (shown in Figure 1.1). The robot has been in development for more than two decades with the goal to duplicate complex human motions and ultimately, assist humans in the household or in scenarios too dangerous for

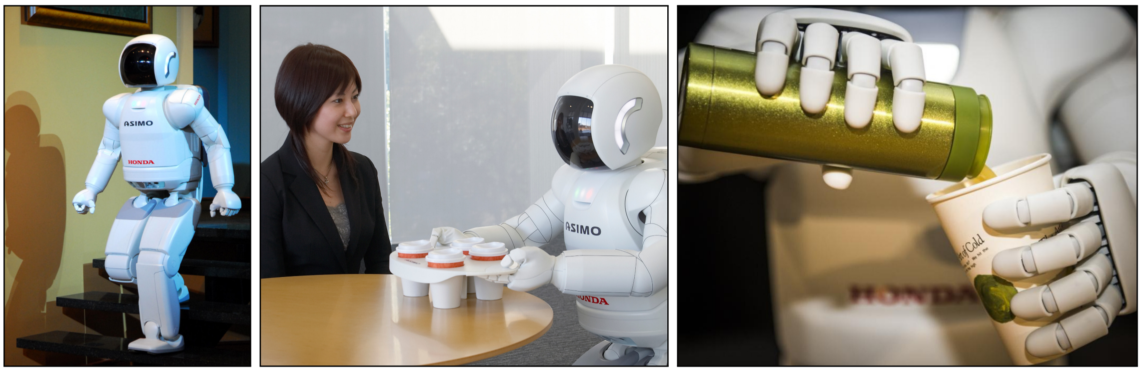


Figure 1.1: The ASIMO multi-functional robot is a study of how robots could assist humans in households in the future (Source: Honda).

them. One day, ASIMO could live with humans and provide services like preparing food, cleaning the house or care for the elderly.

After the catastrophe of Fukushima and the accompanying debate about the capabilities of current robots, the US Defense Advanced Research Projects Agency (DARPA) installed the DARPA Robotics Challenge (DRC). The competition aims at fostering the development of robots competent of assisting humans in disaster response. The competing robots have to perform challenging tasks, ranging from walking across rough terrain, over ladder climbing, up to driving a utility vehicle and cutting through walls (see Figure 1.2 for an illustration). Leading research organizations from around the world are participating in this event, looking at a two million dollar prize.

It is interesting to note that most participants in the DRC are developing human-shaped or *humanoid* robots like Honda's ASIMO. There seems to be a concentration on humanoid robots when complex tasks like disaster response or assistance in households are involved. And there are many reasons for that. Humanoids seem to be best suited to adapt to a man-made environment. Their legs allow them to walk over rough terrain and climb ladders or stairs – which is impossible for wheeled robots. With their arms, they are able to carry and manipulate objects or even use tools.

Independent of the type of robot, one of the most fundamental problems that need to be solved in order to accomplish such sophisticated behavior is that of navigation. Simply speaking, navigation is the problem of determining how to get from a one place to another. Without this capability, robots need to remain stationary or rely on a safe, robot-friendly environment as the lawn mower robots do. While navigating is an easy task for most humans, it is very challenging for robots, as it involves a multitude of subproblems that need to be tackled. These include the perception of the surroundings from sensor data, maintenance of an appropriate environment representation, planning of collision-free motions as well as self-localization. In the context of carrying out sophisticated

Figure 1.2: The DARPA Robotics Challenge (artist's illustration) sets humanoid robots difficult tasks resembling disaster scenarios (Source: DARPA).

assignments like disaster response, the robot further needs to be able to manipulate its environment. For example it needs to remove debris to reach a location or open doors prior to entering a building. Consequently, manipulation tasks can be considered part of the navigation problem as well.

This broad range of subtasks is also the main reason why navigation is so difficult in practice. In controlled scenarios, i.e., in simulation, impressive approaches have been presented enabling robots to perform actions that are difficult even for humans, such as parcouring or rock climbing. However, these approaches simplify the task by neglecting some subproblems that are important for real world operation, especially perception. For navigating in the real world, a robot needs to operate with imperfect assumptions about itself and the environment. It needs to be robust to improper execution of planned actions. To interact with the environment, a robot needs to perceive it through sensor data and interpret it geometrically as well as semantically. The robot thereby needs to handle errors or noise in the data and establish correspondences between multiple observations in presence of ambiguities. Consequently, everything a robot does and knows is affected by uncertainty. For humanoid robots, additional aspects aggravate the situation. For instance, the kinematic complexity that needs to be considered is higher compared to wheeled robots. The shaking motions of humanoids add further errors to the sensor data, making it harder to interpret. Other constraints like balance and payload need to be respected. Finally, affordable robots are typically equipped with less accurate sensors and actuators, thereby increasing the challenge. Given the limited resources, efficiency becomes more important.

Subsequently, for enabling humanoid robots to navigate in complex scenarios and perform challenging operations, one needs to answer at least the following questions:

- How can a robot obtain knowledge about its own parameters?
- How does a robot know which areas are safe to set foot in?
- How can a robot perceive its environment and maintain an internal representation of it?
- How can a robot locate itself with respect to that representation?
- How can a robot avoid obstacles on its way?
- How can a robot utilize its locomotive capabilities to overcome challenging environments?
- How can a robot manipulate objects in its environment to achieve a goal?

In this thesis, we describe techniques that enable autonomous navigation capabilities for humanoid robots which seem to be most suitable to perform complex tasks and become valuable assets to mankind in the future. Common to the approaches we present is that they rely on cameras for perceiving the environment. Our main motivation for working with cameras is that we know from our everyday life how powerful visual perception is. Eventually, it is the primary fashion humans use for navigation. Our methods do not assume state of the art hardware but are also compatible with affordable robot platforms and consumer level sensors.

Specifically, in Chapter 2 we introduce a self-calibration method for humanoid robots. Hereby, the robot estimates its calibration parameters including the camera's intrinsics and extrinsics, as well as joint offsets. Knowledge about these parameters is crucial for all aspects of navigation. Our system for self-calibration is based on graph-optimization and includes a method to generate and select appropriate robot postures yielding a good trade-off between accuracy and the required number of observations.

In Chapter 3, we present a method to classify the traversability of the floor. This allows the robot to safely decide where to step onto and which areas to avoid. The approach relies only on monocular vision data and odometry. We achieve a dense labeling of the image by applying appearance-based classifiers that are trained online in a self-supervised fashion. Our system constructs an occupancy map of the environment that can be used for collision-free navigation. The approach is not limited to humanoids but can also be applied to different kinds of robots.

Chapter 4 describes an integrated framework for robot navigation that relies on depth cameras. We present techniques to localize a robot in a given environment map for accurate pose estimation. The system enables robots to maintain a three-dimensional volumetric map of the non-static parts in their vicinity and plan collision-free paths around obstacles. This approach allows for robust navigation even in complex scenarios with multiple stories.

Based on these results, in Chapter 5 we present valuable extensions for navigation in *unknown*, *cluttered* scenarios. The techniques include the construction of a high resolution map for precise footstep planning. The approach does not require prior knowledge about the environment for pose estimation but compensates for the drift of the odometry by aligning consecutive camera observations. By planning footsteps, the robot is able to step over obstacles and climb onto or from them. Our approach considers the whole body of the humanoid during collision checking. We employ an anytime planner that generates initial solutions for the footstep placement quickly and then improves the initial result. Our system allows the robot to traverse challenging passages.

Finally, we introduce techniques suitable for accurate manipulation tasks in Chapter 6. We present a model-based object pose estimation framework and solve the inverse kinematics (IK) problem to approach the object with the robot's arms. We precompute arm configurations to accelerate the IK and compensate for local minima. As a concrete application, we implement these techniques for a humanoid playing the metallophone. Our system estimates the pose of the instrument and the beaters in the robot's grippers. From these, the system computes beating configurations for the arms of the robot via IK. The robot automatically validates these configurations via auditory and visual feedback. By enabling the robot to play complete songs, we demonstrate the achieved high accuracy. The proposed techniques can also be applied to different manipulation tasks.

Finally, in Chapter 7 we summarize the presented work and discuss remaining challenges and further research directions. The most essential mathematical background is summarized in Appendix A. We thoroughly evaluated all of our approaches on an affordable Nao humanoid. The robot is described in Appendix B.

1.1 Main Contributions

This thesis introduces several valuable techniques that enhance humanoid robot navigation and advance the state of the art in the corresponding scientific fields. We tackle problems like self-calibration, traversability classification, pose estimation, mapping, footstep planning, and manipulation. To summarize, our key contributions are:

- an efficient, accurate method for self-calibration, which is a prerequisite to all kind of navigation tasks (Chapter 2),
- a self-learning technique for traversability classification from monocular camera data to determine areas to safely step onto (Chapter 3),

- an integrated approach to navigation within a partly known environment based on depth camera observations, enabling robust navigation in complex scenarios with multiple levels (Chapter 4),
- further extensions that allow navigation in completely unknown environments, including challenging scenarios containing clutter, narrow passages, and traversable objects via efficient, anytime footstep planning (Chapter 5),
- a framework for accurate manipulation tasks based on model-based object localization and IK computation (Chapter 6).

1.2 Publications

Parts of this thesis have been published earlier in refereed journals and conferences, or in workshop proceedings. Further work emerged during my time as a PhD-student, which is not covered in this thesis.

Refereed Publications

- D. Maier, S. Wrobel, and M. Bennewitz. Whole-body self-calibration via graph-optimization and automatic configuration selection. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015. To appear
- D. Maier, R. Zohouri, and M. Bennewitz. Using visual and auditory feedback for instrument-playing humanoids. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2014
- A. Hornung, S. Osswald, D. Maier, and M. Bennewitz. Monte Carlo localization for humanoid robot navigation in complex indoor environments. *Int. Journal of Humanoid Robots (IJHR)*, 11(2), 2014b
- D. Maier, C. Lutz, and M. Bennewitz. Integrated perception, mapping, and footstep planning for humanoid navigation among 3d obstacles. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013a
- D. Maier, A. Hornung, and M. Bennewitz. Real-time navigation in 3D environments based on depth camera data. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2012

- D. Maier and M. Bennewitz. Appearance-based traversability classification in monocular images using iterative ground plane estimation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012

Workshop Publications

- D. Maier, C. Lutz, and M. Bennewitz. Autonomous biped navigation through clutter. In *Proc. of the RSS Workshop on Robots in Clutter: Preparing Robots for the Real World*, 2013b
- A. Hornung, D. Maier, and M. Bennewitz. Search-based footstep planning. In *Proc. of the ICRA Workshop on Progress and Open Problems in Motion Planning and Navigation for Humanoids*, 2013a

Publications Not Covered in This Thesis

- D. Maier, C. Stachniss, and M. Bennewitz. Vision-based humanoid navigation using self-supervised obstacle detection. *International Journal of Humanoid Robots*, 10(2), 2013c
- M. Bennewitz, D. Maier, A. Hornung, and C. Stachniss. Integrated perception and navigation in complex indoor environments. In *Proc. of the Humanoids 2011 Workshop on Humanoid Service Robot Navigation in Crowded and Dynamic Environments*, 2011

1.3 Collaborations

Parts of the work presented in this thesis were developed in cooperation with other authors. The integrated navigation approach shown in Chapter 4 resulted from a cooperation with Armin Hornung. During my time as PhD-student I supervised multiple student projects, including two Master's theses. The latter lead to joint research and publications after submission. In this manner, with his Master's thesis, Stefan Wrobel lay the foundation for the self-calibration system for humanoid robots presented in Chapter 2. The 3D footstep planning framework described in Chapter 5 was originally developed by Christian Lutz in his Master's thesis. Furthermore, while working as student assistant, Ramin Zohouri explored the possibility of the instrument-playing humanoids introduced in Chapter 6.

1.4 Notation

In this thesis, the following notation is being used if not indicated otherwise.

Symbol	Meaning
$\alpha, \beta, \dots, a, b, \dots$	scalar
$A, B, C, \dots$	matrix
A_{ij}	entry of matrix A at row i and column j
$A^\top$	transpose of matrix A
$\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$	column vector, point or pose
$\mathbf{a}^\top$	transpose of vector **a** (row vector)
$\mathbf{a}_i$	i$^{\text{th}}$ entry of vector **a**
$\tilde{\mathbf{p}}$	point **p** in homogeneous coordinates
$\lvert\lvert\mathbf{x}\rvert\rvert$	ℓ^2-norm (length) of vector **x**
$\mathcal{Q}, \{\omega_1, \dots, \omega_n\}$	sets
$\lvert\mathcal{Q}\rvert$	cardinality of set $\mathcal{Q}$
$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	Normal distribution with mean $\boldsymbol{\mu}$ and variance Σ
$p(X)$	probability distribution for a random variable X
$p(X\mid Y)$	conditional probability for X given Y
$f(\mathbf{p}), \mathsf{fun}(\mathbf{p}), \dots$	function of **p**
$\oplus, \ominus$	motion composition operator and its inverse (Smith et al., 1987)
$\mathcal{T}(\mathbf{x})$	the homogeneous transformation matrix corresponding to the pose **x**
$\mathcal{F}_E^B(\boldsymbol{\theta}, \mathbf{q})$	the forward kinematics function (see Appendix A.3)
$\boldsymbol{\theta}$	the calibration parameters of the robot (see Chapter 2)

Chapter 2

Whole-Body Self-Calibration

Exact knowledge about a robot's kinematic model is essential for all tasks involving navigation and manipulation. Therefore, in this chapter, we present a novel approach to accurately calibrate the kinematic model of a humanoid based on observations of its monocular camera. Our technique estimates the parameters of the robot model, consisting of the joint angle offsets of the whole body including the legs, as well as the camera's extrinsic and intrinsic parameters. We formulate the parameter estimation as a least-squares optimization problem. In the error function, we consider the residuals between camera observations of end-effector markers and their projections into the image based on the estimate of the calibration parameters. Furthermore, we developed an approach to automatically select a subset of configurations for the calibration process that yields a good trade-off between time and accuracy. As the experiments with a Nao humanoid show, we achieve an accurate calibration for this low-cost platform. Moreover, our approach to automatic configuration selection yields substantially better optimization results compared to randomly chosen viable configurations. Hence, our system only requires a reduced number of configurations to achieve accurate calibration results.

Knowledge about the parameters of a robot's kinematic model is essential for all tasks involving navigation and manipulation. To map unknown environments or to estimate the positions of obstacles, the transformations between the exteroceptive sensors and the robot's internal reference frame need to be known, in order to bring the observations in relation to the robot. Only then, the robot can correctly interpret its sensor measurements, avoid collisions with objects, and plan paths to target locations.

For manipulation, an accurate kinematic model is necessary to solve the inverse kinematics and control a manipulator to reach grasping targets, as well as for collision checking of arm trajectories with the environment. While techniques such as image-based visual servoing are robust to a certain degree of errors in the model (Espiau, 1993), accurate knowledge of the calibration parameters is necessary if the manipulator is to follow a given trajectory precisely. Further, an accurate model of the robot allows for faster control and easier implementation, including the possibility for open-loop control.

The kinematic structure of a robot is usually known from the mechanical design. However, errors can occur, e.g., as a result of imperfect manufacturing, wear, or repair. To compensate for such errors and to avoid time-consuming manual tuning of the true parameters of the models, we present in this chapter an automatic self-calibration technique for humanoid robots. Our approach calibrates the robot's kinematic model, given its structure, based only on observations from the robot's internal monocular camera. We estimate the parameters of the kinematic model of the whole body in form of joint angle offsets as well as the camera's extrinsic and intrinsic parameters. We formulate the calibration as an error minimization problem and apply the g^{2}o optimization framework by Kümmerle et al. (2011). Consequently, we minimize the residuals between observed positions of end-effector markers in the camera image and their expected locations given the estimated parameters. Additionally, we present a method to automatically generate robot configurations for the calibration and to determine a minimal subset of these configurations that lead to accurate calibration results, thereby trading off time and accuracy.

We applied our approach to calibrate the parameters of a Nao humanoid. Hereby, the optimization using our proposed algorithm to automatic configuration selection substantially outperforms the optimization based on sets of randomly chosen viable configurations. We found that using the configuration selection algorithm, the system requires only few configurations to achieve an accurate calibration of the robot. Figure 2.1 shows a Nao performing the self-calibration (left), as well as the expected marker location and a 3D visualization of the kinematic model overlaid to the onboard camera image, before (top right) and after (bottom right) the calibration.

To the best of our knowledge, this is the first approach to calibrate the complete kinematic model of a legged humanoid. Our implementation provides a one-click calibration routine for the Nao robot, which is the most commonly used humanoid. Adaptations for different humanoids can easily be obtained by adjusting configuration files.

2.1 Graphed-Based Optimization of the Calibration Parameters

Poor calibration of a robot leads to a discrepancy between self-observation, e.g., from a camera, and expected observation according to underlying models and their estimated parameters. Our calibration framework hence measures this discrepancy and adjusts the

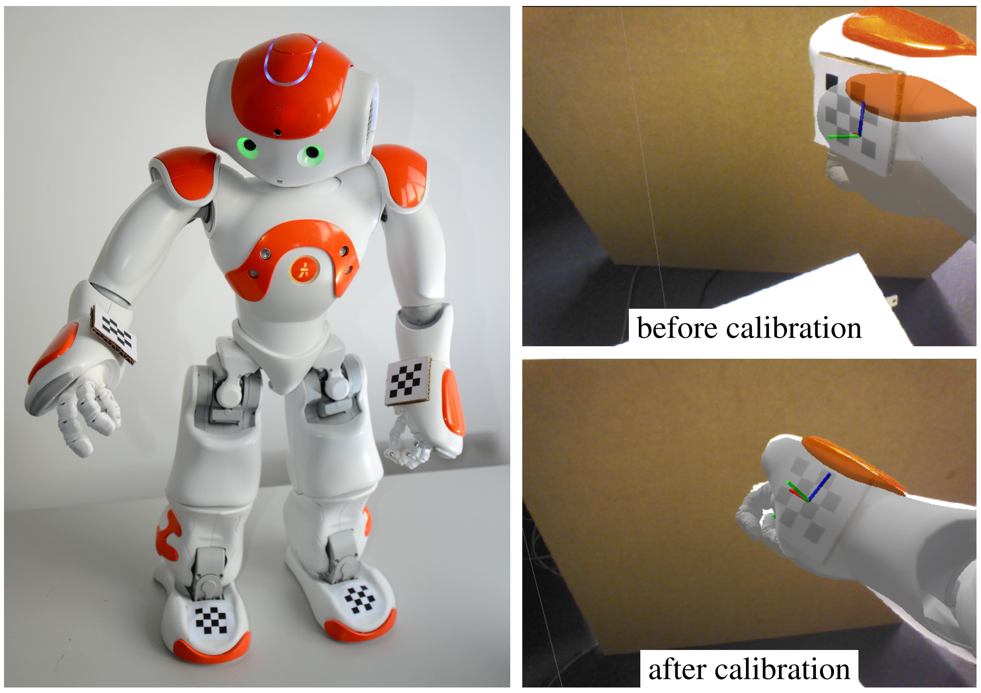


Figure 2.1: Left: Nao humanoid with markers attached to the end-effectors (EEF) performing automatic self-calibration. Right: Marker on the right EEF observed by the onboard-camera before and after the calibration, with the robot model (semi-transparent) and the expected marker's center (coordinate frame) overlaid.

parameters so that the error is minimized. We hereby rely on camera observations of point markers attached to the robot's end-effectors. We determine their expected locations in the camera image and compute the error when compared to the actual observations.

2.1.1 Measurement Model and Parameters

One of the most obvious reasons for a misalignment between a true observation and the expected one is that the camera's intrinsic and extrinsic parameters are not known accurately. Our system uses the standard pinhole model to compute the projection of a point in the camera image and also considers the radial distortion of the lens, which is typically the most dominant reason for deviations. The model is further described in Appendix A.2. In summary, let $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^\intercal$ be a 3D point in the camera frame, then its corresponding distorted image coordinates are given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \mathsf{proj}(\mathbf{p}). \tag{2.1}$$

The function proj is defined in (A.11) and depends on the camera's focal lengths f_x and f_y, its principle point $\begin{bmatrix} k_x & k_y \end{bmatrix}^\mathsf{T}$, and a scalar κ that models the strength of the radial distortion.

Also, the calibration of the camera's extrinsics parameters is of great importance, i.e., its placement relative to the reference frame that the camera is attached to, i.e., typically the robot's neck joint. Consequently, we seek the orientation and center of the camera (i.e. its pose) relative to the reference frame. These parameters are typically described by a rotation matrix R and a camera center $\mathbf{c} = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^\mathsf{T}$ (see Appendix A.2). However, for the optimization, a minimal representation is required. Such a representation can be obtained in form of the corresponding Euler angles of R and the center point $\mathbf{c}$, i.e.,

$$\begin{bmatrix} x_c & y_c & z_c & \varphi_c & \theta_c & \psi_c \end{bmatrix}^\mathsf{T}. \tag{2.2}$$

Determining these values is often referred to as *extrinsic calibration* of the camera.

Another important reason for poor calibration is that the rotary joint encoders, which contribute to the kinematic state of the robot, are affected by systematic offsets due to lack of precision in the manufacturing process or wear. Hence, we model the true position of the joints at time i as the sum of the offsets $\mathbf{q}^{\text{off}}$ and the encoder readings $\hat{\mathbf{q}}_i$

$$\mathbf{q}_i = \hat{\mathbf{q}}_i + \mathbf{q}^{\text{off}}. \tag{2.3}$$

Finally, we need to consider the artificial markers attached to the end-effectors (EEF) as we rely on them as observations. While their approximate positions can be obtained from manual measuring, positional errors directly affect the automatic evaluation of the state of calibration of the robot. Hence, we include the position for the markers M_{EEF} on each end-effector in the estimation and model them as three-dimensional vectors relative to the EEF-frame by

$$\mathbf{m}_{\text{EEF}} = \begin{bmatrix} x & y & z \end{bmatrix}^\mathsf{T}. \tag{2.4}$$

To summarize, the parameters considered in the error minimization are

- the camera's intrinsics and distortion fx, fy, k_x, k_y, κ,
- the camera's extrinsics $x_c, y_c, z_c, \varphi_c, \theta_c, \psi_c$,
- one marker location $\mathbf{m}_{\text{EEF}}$ per end-effector, and
- the joint offsets $\mathbf{q}^{\text{off}}$.

For mathematical simplicity, in the remainder of this chapter, we stack all these parameters in a vector $\boldsymbol{\theta} \in \mathbb{R}^L$, where L is the number of parameters.

2.1.2 Formulation as Least-Squares Optimization

Thus, our calibration procedure aims at estimating the parameters $\boldsymbol{\theta} \in \mathbb{R}^L$ that minimize the error between the estimated poses of the markers attached to the end-effectors and their observations in the camera image. To obtain a robust estimate for $\boldsymbol{\theta}$, we consider the accumulated error for a set $\mathcal{S}$ consisting of n different robot configurations, where each configuration is defined by its joint encoder readings $\hat{\mathbf{q}}_i \in \mathcal{S}$. We formulate the parameter estimation as a least-squares optimization. Hence, we minimize

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} F_{\mathcal{S}}(\boldsymbol{\theta}), \tag{2.5}$$

where $F_{\mathcal{S}} : \mathbb{R}^L \longrightarrow \mathbb{R}$ is the accumulated error given by

$$F_{\mathcal{S}}(\boldsymbol{\theta}) := \sum_{i=1\ldots n} \mathbf{e}_i(\boldsymbol{\theta}, \hat{\mathbf{z}}_i, \hat{\mathbf{q}}_i)^\intercal \mathbf{e}_i(\boldsymbol{\theta}, \hat{\mathbf{z}}_i, \hat{\mathbf{q}}_i), \tag{2.6}$$

and $\mathbf{e}_i(\boldsymbol{\theta}, \hat{\mathbf{z}}_i, \hat{\mathbf{q}}_i)$ is the error residual for the robot configuration $\hat{\mathbf{q}}_i \in \mathcal{S}$ and the image measurement $\hat{\mathbf{z}}_i$. For the employed point markers, the error function is given by

$$\mathbf{e}_i(\boldsymbol{\theta}, \hat{\mathbf{z}}_i, \hat{\mathbf{q}}_i) := \hat{\mathbf{z}}_i - \mathsf{predict}_{M_{\mathrm{EEF}}}(\boldsymbol{\theta}, \hat{\mathbf{q}}_i). \tag{2.7}$$

Here, $\hat{\mathbf{z}}_i \in \mathbb{R}^2$ is the observed marker location in the image, from which we subtract the estimate for the projection of the marker, $\mathsf{predict}_{M_{\mathrm{EEF}}}(\boldsymbol{\theta}, \hat{\mathbf{q}}_i)$, given the robot's kinematic structure, the calibration parameters $\boldsymbol{\theta}$, and the joint readings $\hat{\mathbf{q}}_i$. The predicted location of a marker M_{EEF} is given by

$$\mathsf{predict}_{M_{\mathrm{EEF}}}(\boldsymbol{\theta}, \hat{\mathbf{q}}_i) := \mathsf{proj}_{\boldsymbol{\theta}}\left(\begin{bmatrix} R & -R\,\mathbf{c} \end{bmatrix} \mathcal{F}^{N}_{\mathrm{EEF}}(\boldsymbol{\theta}, \hat{\mathbf{q}}_i)\, \tilde{\mathbf{m}}_{\mathrm{EEF}}\right), \tag{2.8}$$

where $\mathsf{proj}_{\boldsymbol{\theta}}$ is as defined in (A.11) using the intrinsic parameters from the current calibration $\boldsymbol{\theta}$. The function $\mathcal{F}^{N}_{\mathrm{EEF}}(\boldsymbol{\theta}, \hat{\mathbf{q}}_i)$ describes the forward kinematics of the end-effector with respect to the neck frame evaluated from the joint readings $\hat{\mathbf{q}}_i$ and the joint offsets contained in $\boldsymbol{\theta}$. R and $\mathbf{c}$ are extracted from the calibration parameters $\boldsymbol{\theta}$ as well. Consequently, the argument to $\mathsf{proj}_{\boldsymbol{\theta}}$ yields the location of the marker M_{EEF} relative to the camera frame, given the joint readings $\hat{\mathbf{q}}_i$ and the calibration parameters $\boldsymbol{\theta}$. The forward kinematic function $\mathcal{F}$ is described in Appendix A.3.

To solve the least-squares problem defined in (2.5) to (2.8), we apply a linear approximation. With the definition

$$\mathbf{f}(\boldsymbol{\theta}) := \begin{bmatrix} \mathbf{e}_1(\boldsymbol{\theta}, \hat{\mathbf{z}}_1, \hat{\mathbf{q}}_1)^\intercal & \ldots & \mathbf{e}_n(\boldsymbol{\theta}, \hat{\mathbf{z}}_n, \hat{\mathbf{q}}_n)^\intercal \end{bmatrix}^\intercal, \tag{2.9}$$

we can rewrite (2.6) as

$$F_{\mathcal{S}}(\theta) = \mathbf{f}(\boldsymbol{\theta})^\top \mathbf{f}(\boldsymbol{\theta}), \tag{2.10}$$

We apply a linear first-order approximation of **f** by defining

$$\boldsymbol{\theta} := \breve{\boldsymbol{\theta}} + \Delta\boldsymbol{\theta}, \tag{2.11}$$

and consequently approximate

$$\mathbf{f}(\boldsymbol{\theta}) = \mathbf{f}(\breve{\boldsymbol{\theta}} + \Delta\boldsymbol{\theta}) \approx \mathbf{f}(\breve{\boldsymbol{\theta}}) + J\,\Delta\boldsymbol{\theta}, \tag{2.12}$$

$$\text{where} \quad J := \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}(\breve{\boldsymbol{\theta}}) \tag{2.13}$$

is the Jacobian of **f** evaluated at the linearization point $\breve{\boldsymbol{\theta}}$. By plugging (2.12) into (2.10), a linear approximation for (2.6) is given by

$$F_{\mathcal{S}}(\boldsymbol{\theta}) \approx (\mathbf{f}(\breve{\boldsymbol{\theta}}) + J\Delta\boldsymbol{\theta})^\top (\mathbf{f}(\breve{\boldsymbol{\theta}}) + J\Delta\boldsymbol{\theta}) \tag{2.14}$$

$$= \underbrace{\mathbf{f}^\top(\breve{\boldsymbol{\theta}})\mathbf{f}(\breve{\boldsymbol{\theta}})}_{=:c} + 2\underbrace{\mathbf{f}^\top(\breve{\boldsymbol{\theta}})J}_{=:b^\top}\Delta\boldsymbol{\theta} + \Delta\boldsymbol{\theta}^\top \underbrace{J^\top J}_{=:H}\Delta\boldsymbol{\theta}. \tag{2.15}$$

Thus, we obtain an approximate error function that does not depend on $\boldsymbol{\theta}$ anymore but on $\Delta\boldsymbol{\theta}$. We call this error function $\breve{F}_{\mathcal{S}}$, with

$$\breve{F}_{\mathcal{S}}(\Delta\boldsymbol{\theta}) := c + 2b^\top\Delta\boldsymbol{\theta} + \Delta\boldsymbol{\theta}^\top H \Delta\boldsymbol{\theta}. \tag{2.16}$$

To minimize (2.16), we set its derivative to zero and solve the linear system,

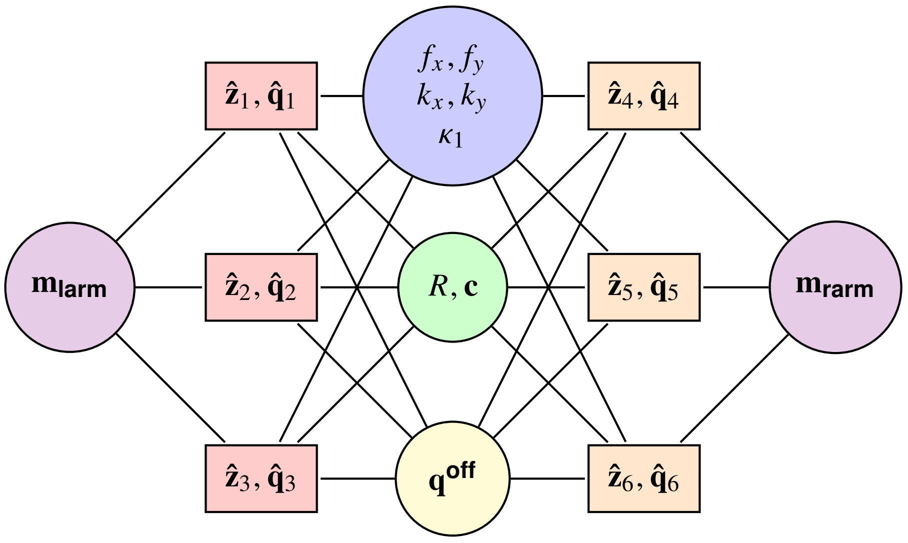
$$\frac{\partial \breve{F}_{\mathcal{S}}}{\partial \Delta\boldsymbol{\theta}} = 0 \iff 2b + 2H\Delta\boldsymbol{\theta} = 0 \tag{2.17}$$

$$H\Delta\boldsymbol{\theta} = -b \tag{2.18}$$

$$\Delta\boldsymbol{\theta} = -(J^\top J)^{-1} J^\top \mathbf{f}(\breve{\boldsymbol{\theta}}). \tag{2.19}$$

An estimate for $\boldsymbol{\theta}$ can then be obtained via the definition in (2.11). To minimize the error, we use the Levenberg-Marquardt (LM) algorithm, which repeats the three steps, i.e., the linearization in (2.12), the minimization in (2.19), and the increment in (2.11), until convergence or a maximum number of iterations is reached.

To estimate $\boldsymbol{\theta} \in \mathbb{R}^L$, the system of equations may not be under-determined. As each observation yields two constraints, i.e., x- and y-location of the marker in the camera



f_x, f_y
k_x, k_y
κ_1
$\hat{\mathbf{z}}_1, \hat{\mathbf{q}}_1$
$\hat{\mathbf{z}}_2, \hat{\mathbf{q}}_2$
$\hat{\mathbf{z}}_3, \hat{\mathbf{q}}_3$
$\hat{\mathbf{z}}_4, \hat{\mathbf{q}}_4$
$\hat{\mathbf{z}}_5, \hat{\mathbf{q}}_5$
$\hat{\mathbf{z}}_6, \hat{\mathbf{q}}_6$
$\mathbf{m}_{\text{larm}}$
$R, \mathbf{c}$
$\mathbf{m}_{\text{rarm}}$
$\mathbf{q}^{\text{off}}$

Figure 2.2: Example g^{2}o graph for calibration of a humanoid's upper body. The graph includes nodes for the camera's intrinsics, camera's extrinsics, joint offsets, and the markers of the two arms. Three measurements per chain are indicated.

image, it must hold that $2n = 2|\mathcal{S}| \geq L$. Consequently, at least $n = \lceil L/2 \rceil$ observations are required to solve the system.

2.1.3 Implementation

We implemented our calibration system using the g^{2}o library by Kümmerle et al. (2011). It solves problems of the form given in (2.5) and (2.6) by optimizing a hyper-graph. The graph consists of a set of vertices that encode the sub-components of the parameter vector $\boldsymbol{\theta}$ to be optimized and the measurements $(\hat{\mathbf{z}}_i, \hat{\mathbf{q}}_i)$. The edges represent the error functions from (2.7) between corresponding vertices. Figure 2.2 shows an example for such a graph with six measurements and two markers. As can be seen, the edges for one measurement connect multiple vertices.

The employed framework g^{2}o automatically takes care of singularities when estimating 6D poses such as the camera's extrinsics. Further, the framework allows using a robust kernel such as the Huber loss function in (2.6). Thus, it optimizes

$$F'_{\mathcal{S}}(\boldsymbol{\theta}) = \sum_{i=1\ldots n} \rho_H\left(\sqrt{\mathbf{e}_i(\boldsymbol{\theta}, \hat{\mathbf{z}}_i, \hat{\mathbf{q}}_i)^\top \mathbf{e}_i(\boldsymbol{\theta}, \hat{\mathbf{z}}_i, \hat{\mathbf{q}}_i)}\right) \tag{2.20}$$

$$\text{where} \quad \rho_H(x) = \begin{cases} x^2, & \text{if} \|x\| < b \\ 2b\|x\| - b^2, & \text{else} \end{cases}, \tag{2.21}$$

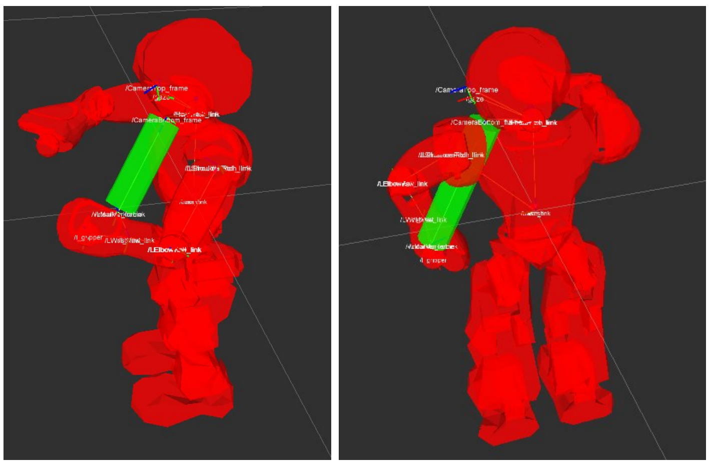


Figure 2.3: Self-collision and marker visibility check using mesh-model. Left: an acceptable configuration, right: the left shoulder occludes the sight of the marker, thus the configuration is invalid.

and b is a scaling parameter. This means, $\rho_H(x_i)$ is quadratic for small errors x_i, and linear for larger ones. In case of $b \to \infty$, (2.20) is identical to (2.6). By using a robust kernel, the effect of outliers on the optimization is reduced, which might occur due to measurement noise or false associations. In Section 2.3.2 we compare the effect of outliers on the optimization with and without using a robust kernel.

2.2 Automatic Selection of Robot Configurations

The result of the optimization described in Section 2.1.2 depends on the initial linearization point and the set of configurations with their measurements for which the error minimization is carried out. A higher number of configurations can lead to a more accurate parameter estimation but, obviously, they consume more time. A small set of configurations on the other hand might lead to overfitting on the calibration data. In order to reduce the number of required configurations, while still enabling robust estimates, we developed an approach to automatically select a set of close-to-optimal configurations. The goal is to achieve a good trade-off between the time needed to carry out the calibration and the accuracy of the result.

2.2.1 Generating a Pool of Configurations

Our system initially samples a large pool Q of configurations from which subsequently a subset of configurations that optimizes a certain criterion is selected. The configurations are uniformly sampled in joint space and checked for self-collisions as well as marker observability. A configuration $\mathbf{q}_i$ is a vector of admissible values for all the joints that are included in the calibration.

For each sampled configuration $\mathbf{q}_i$, we first project the marker location into the camera image, given the initial estimate of $\boldsymbol{\theta}$ and the kinematic structure of the robot using the predict-function as defined in (2.8). The system only accepts $\mathbf{q}_i$, if $\mathbf{z_i} = \mathsf{predict}_{M_{\text{EEF}}}(\boldsymbol{\theta}, \mathbf{q}_i)$ is within the boundaries of the camera image minus a safety margin to account for errors in the initial estimate. Next, the algorithm tests whether the robot is in self-collision using a mesh-model and the Flexible Collision Library (FCL) by Pan et al. (2012). Finally, using the same technique, our approach checks whether the marker is not occluded by any part of the body in this configuration and consequently visible by the camera. To this end, we insert an additional collision object in shape of a cylinder between marker and camera. We use a cylinder here instead of a line to represent the spatial extend of the marker and to account for errors in the initial calibration. Figure 2.3 shows an example for the performed mesh-model checks. The left image displays an acceptable configuration for observing the marker on the left arm, while the right image shows an invalid configuration, as the left shoulder occludes the sight from the camera to the marker.

2.2.2 Selecting a Locally Optimal Subset of Configurations

To automatically generate a set of near optimal configurations for calibration, we examine the Jacobian defined in (2.13). The Jacobian is decomposed using Singular Value Decomposition (SVD) into

$$J = U \Sigma V^{\top}, \tag{2.22}$$

where U and V are orthogonal matrices and Σ is the diagonal matrix of the singular values $\sigma_1, \ldots, \sigma_L$ of J, where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_L$. Here, we assume that there are $L = |\theta|$ columns, corresponding to the number of parameters, and $2n = 2\,|S| \geq L$ rows in J, otherwise the system would be under-determined. On this decomposition, we apply a so-called *observability index*, which is based on the σ_i and generally yields higher values for larger σ_i and lower values for smaller σ_i. The rationale behind maximizing the σ_i is that, given the linearization in (2.12), under some assumptions including the invertibility of $J^{\top} J$, the covariance of $\boldsymbol{\theta}$ can be approximated (Seber and Wild, 2003, Chapter 2) by a matrix $C \in \mathbb{R}^{L \times L}$ with

$$C := (J^{\top} J)^{-1} \propto \operatorname{cov}(\boldsymbol{\theta}). \tag{2.23}$$

Thus, maximizing the singular values σ_i of J is equivalent to minimizing the Eigenvalues λ_i of C and consequently of $\operatorname{cov}(\boldsymbol{\theta})$, as $\sigma_i = \lambda_i^{-2}$. Hence, we seek to reduce the variance or uncertainty of the parameters. To this end, we consider the following observability indices (Carrillo et al., 2013; Nahvi and Hollerbach, 1996):

$$O_{\mathrm{D}} = (\sigma_1 \dots \sigma_L)^{1/L} L^{-1/2} \tag{2.24}$$

$$O_{\mathrm{A}} = (\sigma_1^{-1} + \dots + \sigma_L^{-1})^{-1} \tag{2.25}$$

$$O_{\mathrm{NAI}} = \sigma_L^2 / \sigma_1 \tag{2.26}$$

$$O_{\mathrm{E}} = \sigma_L . \tag{2.27}$$

O_{D} corresponds to the determinant of C, O_{A} to its trace, and O_{E} to the largest Eigenvalue of C, while O_{NAI} has no direct relation with the covariance, but can be seen as the product of the Jacobian's inverse condition number and its smallest singular value. In Section 2.3.1, we compare the different observability indices with respect to their performance in selecting configurations for robot calibration.

For accurate robot calibration, we consequently choose the configurations and thereby the rows of J, such that the chosen observability index is maximized. How to select a set of N^* configurations is explained exemplary for the index O_D in the following and in Algorithms 1 and 2. Here, the function $\texttt{compute}O_{\mathrm{D}}(\mathcal{S}, \boldsymbol{\theta})$ computes O_D from a set of configurations $\mathcal{S} \subseteq \mathcal{Q}$ and $\boldsymbol{\theta}$. Therefore, the Jacobian J is computed around the current estimate for $\boldsymbol{\theta}$ from the partial derivatives of the error functions (2.7), corresponding to the set of configurations $\mathcal{S}$. Algorithm 1 initially samples a set of $\lceil L/2 \rceil$ configurations from the pool $\mathcal{Q}$, which is the minimum number of configurations needed to solve the optimization problem. The configurations are uniformly sampled, proportional to the degrees of freedom (DOF) of the corresponding kinematic chains. This is to ensure that all the parameters can be determined. Each sampled set is optimized by exchanging its configurations (see Algorithm 2), such that the observability index is maximized. Algorithm 1 keeps the best optimized set of configurations $\mathcal{S}^*$ and solves (2.5) for $\boldsymbol{\theta}$ as described in Section 2.1.2. Finally, the algorithm iterates between greedily adding configurations to $\mathcal{S}^*$ and solving for the calibration parameters given the current $\mathcal{S}^*$.

It has to be noted that the observability indices relate singular values whose units depend on the parameters that are used to calculate the corresponding Jacobian. This can lead to a bad choice of configurations if the units are differently scaled. Hence, our approach compensates for this problem by scaling the columns of the Jacobian (Hollerbach and Wampler, 1996). Therefore, prior to computing $\texttt{compute}O_{\mathrm{D}}$, the algorithm right-multiplies J by a scaling matrix $S = \mathsf{diag}(s_1, \dots, s_L)$, where

$$s_i = \begin{cases} \|\mathbf{j_i}\|^{-1} & \text{if} \quad \|\mathbf{j_i}\| \neq 0 \\ 1 & \text{else} \end{cases} , \tag{2.28}$$

Algorithm 1: O_D-`select`: Selects N^* configurations from a pool Q for calibration and optimizes $\boldsymbol{\theta}$

```
Input:  Pool of configurations $Q$
        Initial calibration parameters $\boldsymbol{\theta}$, with $|\boldsymbol{\theta}| = L$
        Desired number of configurations $N^*$
        Max. number of restarts $T$ for initializing the configuration set
Output: Optimized configurations $\mathcal{S}^*$
        Optimized calibration parameters $\boldsymbol{\theta}^*$
1  $\mathcal{S}^* \longleftarrow \emptyset$
2  for $t = 1$ to $T$ do
3      $\mathcal{S} \longleftarrow$ sample $\lceil L/2 \rceil$ configurations from $Q$
4      $\mathcal{S} \longleftarrow$ exchange($\mathcal{S}$,$Q$)
5      if computeO_D($\mathcal{S}$, $\boldsymbol{\theta}$) > computeO_D($\mathcal{S}^*$, $\boldsymbol{\theta}$) then
6          $\mathcal{S}^* \longleftarrow \mathcal{S}$
7  $\boldsymbol{\theta}^* \longleftarrow \arg\min_{\boldsymbol{\theta}} F_{\mathcal{S}^*}(\boldsymbol{\theta})$
8  while $|\mathcal{S}^*| < N^*$ do
9      $q^+ \longleftarrow \arg\max_{q \in Q \setminus \mathcal{S}^*}$ computeO_D($\mathcal{S}^* \cup \{q\}$,$\boldsymbol{\theta}^*$)
10     $\mathcal{S}^* \longleftarrow \mathcal{S}^* \cup \{q^+\}$
11     $\boldsymbol{\theta}^* \longleftarrow \arg\min_{\boldsymbol{\theta}} F_{\mathcal{S}^*}(\boldsymbol{\theta})$
```

and $\mathbf{j_i}$ indicates the ith column of J. The advantage of this approach is that the user does not need to provide a range for the parameters, which can sometimes be hard to determine (e.g., for camera intrinsics or translations).

2.3 Experiments

In this section, we present an extensive evaluation of our self-calibration method using a Nao humanoid robot. The robot is described in details in Appendix B. In addition, we installed one checkerboard marker per end-effector, which we detect in the image using the corresponding OpenCV function. For calibration, we only consider the center point of the marker. The Nao robot has a total of 23 joints where one of them is a mimic joint. Our system respects mimic joints by only considering one of the joints in the configuration selection and optimization. Five of the joints are redundant with respect to the camera pose and the markers, i.e., the head pitch joint and the last joints for each EEF. The head pitch joint is redundant because for each arbitrary joint offset, the camera pose can be adjusted accordingly to yield the same consistent observations. Similarly, by adjusting the transform to the point marker, each offset for the joint just before the marker can be compensated to represent the same physical system. Consequently, as these

Algorithm 2: `exchange`: Optimizes a set $\mathcal{S}$ of N configurations for calibration from a pool Q

```
          Pool of configurations $Q$
Input:    Initial configurations $\mathcal{S} = \{q_1, \ldots, q_N\}$
          Calibration parameters $\boldsymbol{\theta}$
Output:   Optimized set of configurations $\mathcal{S}^*$ of size $N$
1 repeat
2     $q^+ \longleftarrow \arg\max_{q \in Q \setminus \mathcal{S}^*}$ compute$O_\mathrm{D}(\mathcal{S}^* \cup \{q\}, \boldsymbol{\theta})$
3     $\mathcal{S}^* \longleftarrow \mathcal{S}^* \cup \{q^+\}$
4     $q^- \longleftarrow \arg\max_{q \in Q \setminus \mathcal{S}^*}$ compute$O_\mathrm{D}(\mathcal{S}^* \setminus \{q\}, \boldsymbol{\theta})$
5     $\mathcal{S}^* \longleftarrow \mathcal{S}^* \setminus \{q^-\}$
6 until $q^+ = q^-$
```

offsets cannot be determined, they are ignored in the calibration process. In addition to the joint offsets, we estimate the four marker positions (3 DOF each), the camera pose (6 DOF), its intrinsics (4 DOF) and one radial distortion parameter. Hence, there is a total of 41 parameters to be estimated and at least 21 measurements are needed for a full calibration. Our algorithm sets the initial state for the calibration according to the manufacturer specifications (i.e., zero) for the joint offsets, while we quickly calibrated the camera with the standard OpenCV calibration routine. Furthermore, we manually measured the placement of the markers relative to the end-effectors.

To allow a quantitative evaluation of our system, we generated a large database of 3000 configurations, observing each of the robot's four end-effectors 750 times, using the sampling method described in Section 2.2.1 and then recorded the camera observations and joint encoder readings. We performed a 5-fold cross-validation on the data set to evaluate the system, including the automatic configuration selection.

2.3.1 Pose Selection

To evaluate our approach to automatic configuration selection as described in Section 2.2, we compared the results from the O_D-`select` algorithm with a strategy that randomly selects configurations from the pool Q. For the random strategy, we repeated the complete 5-fold cross-validation three times and accumulated the results to obtain more representative values. The results are shown in Figure 2.4 (top) as root mean square error (RMSE) and its standard deviation over the folds (and repetitions in case of the random strategy). As can be seen, our O_D-`select` algorithm leads to accurate results with an RMSE of 7.2 ± 1.1 px requiring only 25 configurations, which is close to the theoretic minimum of 21. The random strategy yields a substantially higher RMSE of 19.7 px with a large standard deviation of 9.1 px. Even for 50 configurations, O_D-`select` performs better with an RMSE of 5.5 ± 0.8 px compared to 6.1 ± 0.8 px.

We also compared the different observability indices introduced in Section 2.2. The results are shown in Figure 2.4 (bottom). As can be seen, the O_D-`select` performs better than the other algorithms for few measurements, with O_A-`select` performing similar for more than 32 configurations. Both algorithms generally perform better than the remaining two, even for many measurements.

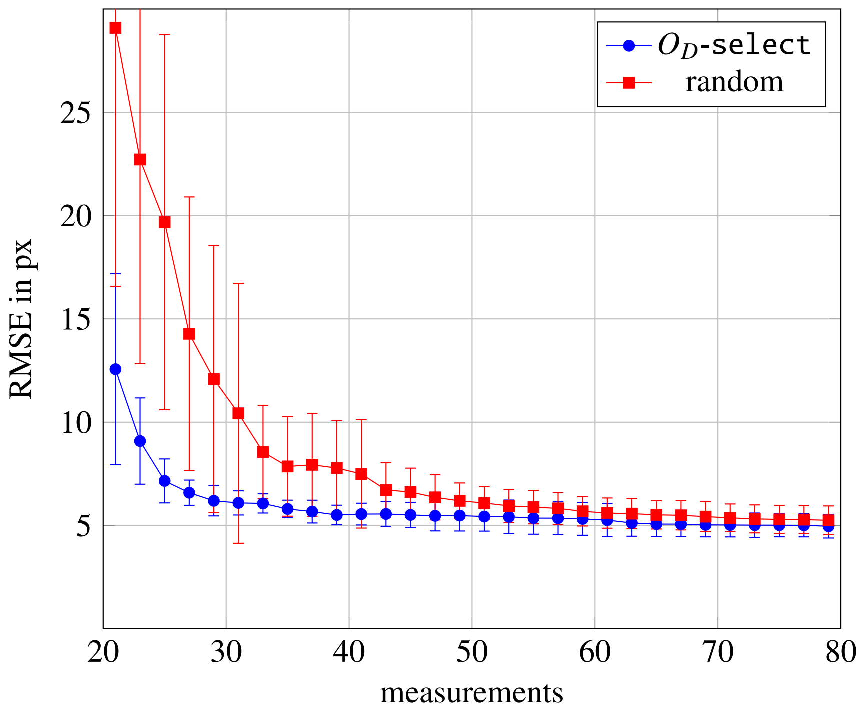
Finally, we compared the performance for arms and legs separately, as some users might want to calibrate only sub-parts of the humanoids body. The results are shown in Figure 2.5. As can be seen, in each case, it is advisable to use a selection algorithm. For the arms, we obtained slightly better results compared to the legs, which is due to the better observability of the former, as the legs typically need to be in front of the robot to be seen by the camera, which means that the configurations do not cover the whole parameter space.

2.3.2 Effect of Measurement Noise

Finally, we tested the effect of noisy measurements on the optimization. Therefore, we used 60 configurations selected by O_D-`select` and repeated the optimization, but this time we randomized the detected marker location over the entire image for different percentages of measurements. This is to simulate the effect of false marker observations, which may occur in presence of clutter and uncontrolled lighting. We optimized using the noisy data with and without using a robust kernel and performed a 5-fold cross-validation as previously. The results are shown in Figure 2.6. As can be seen, using a robust kernel yields substantially better results compared to a quadratic kernel. For up to 10 noisy measurements out of 60 total, the robust kernel performs similar to using completely noise-free measurements, whereas without robust kernel, the RMSE rises up to 45.1 px (versus 6.6 px for the robust kernel). Concluding, the robust kernel handles up to 16% of noisy measurements without strong impacts on the performance and clearly outperforms the standard kernel.

2.3.3 Resulting Calibration Compared to the Initial State

Figure 2.7 shows the residuals in x- and y-direction (in px) in image coordinates for each of the four chains. The top plot shows the results for the uncalibrated state according to the manufacturer specifications with a good initial guess for the markers, and the lower plot shows the residuals after the calibration using 60 configurations selected by O_D-`select`.



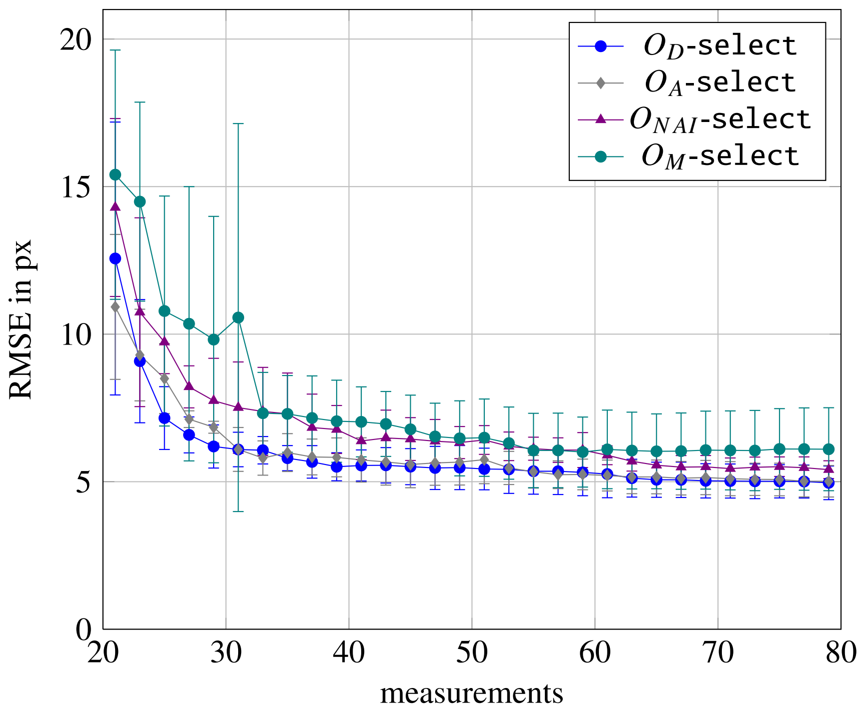


Figure 2.4: RMS-Error on validation data. Top: comparison of O_D-`select` with a random selection strategy. Bottom: comparison of different observability criteria.

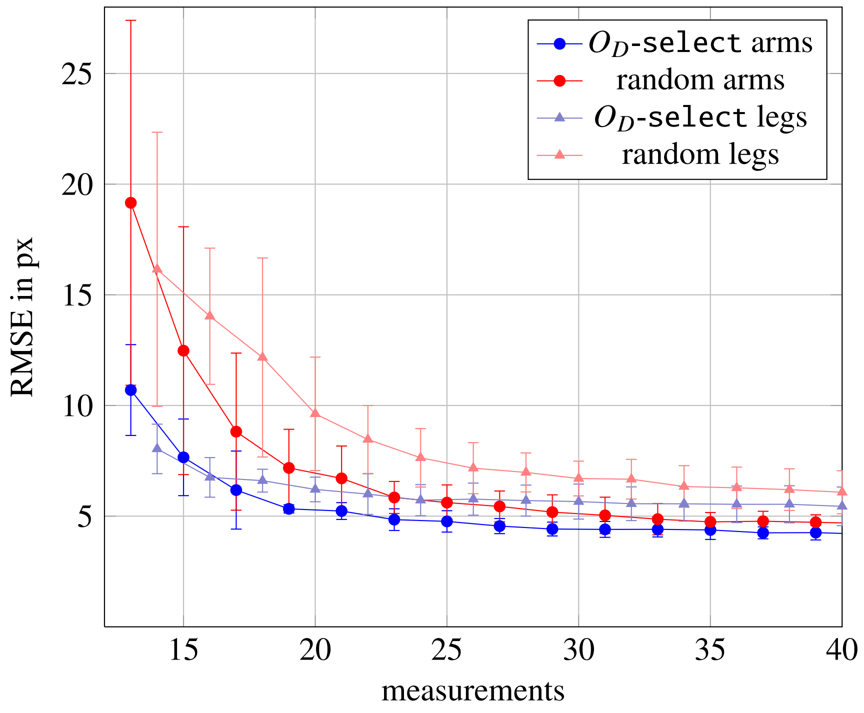


Figure 2.5: RMS-Errors on validation data. Comparison of O_D-`select` with a random selection strategy, for arms and legs separately.

For the lower plot, we overlaid the results from each of the 5-folds of the data set. The initial error was 43.4 px and after calibration we achieved an RMSE of only 5.2 px.

2.4 Related Work

The calibration problem for humanoids is composed of different sub-problems, including calibration of exteroceptive sensors (typically cameras) and the calibration of the kinematic structure. The latter is strongly related to the calibration of manipulators, which has been studied for at least two decades.

One of the most well-known *camera* calibration procedures is described by Zhang (1999). It uses a closed-form maximum likelihood (ML) estimation to solve for the intrinsic and the extrinsic parameters of the camera. In the system described in this chapter, we also compute a ML estimation for these parameters. Agrawal and Davis (2003) use spheres as calibration objects to estimate the camera's intrinsics and extrinsics via semi-definite programming. The spheres of known size are identified in the camera images by fitting ellipses to the corresponding edge images. Asfour et al. (2008) developed an approach to calibrate a stereo camera relative to a humanoid's head by observing a marker and using a least-squares formulation of the problem. Hence, they perform a

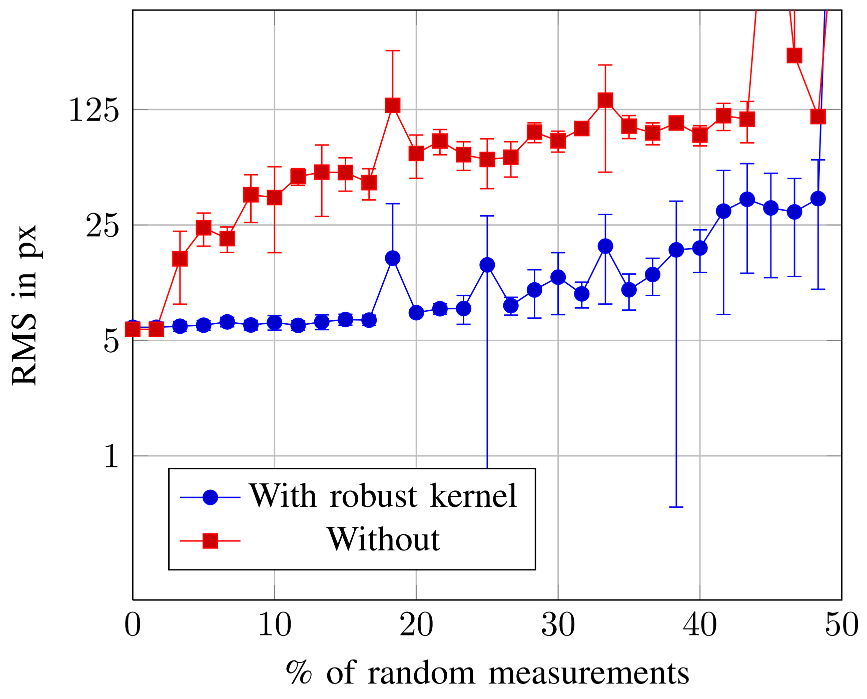


Figure 2.6: Effect of noise on the optimization. Different percentages of camera observations were randomized in the image and optimized with and without a robust kernel. The plot shows the results of a 5-fold cross validation.

classical head-eye calibration. Basso et al. (2014) showed a calibration framework for a depth and RGB sensor couple. They calibrate the intrinsic parameters of both cameras as well as the transform between them by observations of a checkerboard. They further compute an undistortion map to compensate for systematic errors in the depth sensor. While the latter would also be possible within our framework by extension, it is probably faster to compute such a map within a specialized system that considers the image as a whole instead of individual marker locations. However, our system is not restricted to one monocular camera but can easily be used to calibrate multiple cameras. This allows estimating the alignment between depth and RGB sensor and further, estimating both camera's intrinsic parameters.

To calibrate a *manipulator*, a multitude of approaches have been presented. Bennett and Hollerbach (1991) fix the end-effector of the manipulator to the ground. The manipulator is then moved into different constraints. The measurements consist of the joint sensor readings only. Knowing that the endpoint is always in a fixed position, the kinematic parameters can be estimated using forward kinematics.

Yamane (2011) proposed to calibrate the joint offsets of a humanoid's legs using measurements of a chest-mounted Inertial Measurement Unit (IMU) and the constraint that the feet of the robot are fixed on a plane, while manually moving the robot. However,

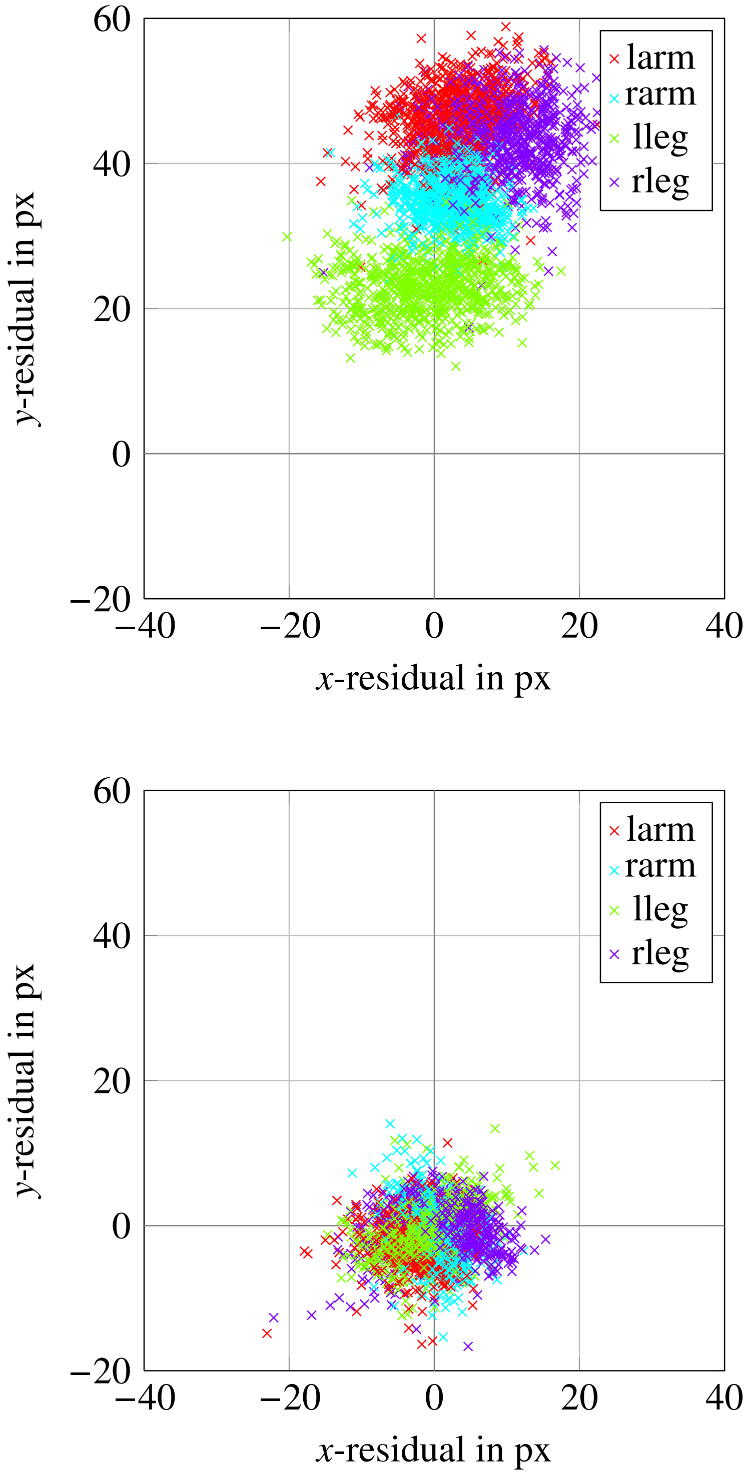


Figure 2.7: Comparison of measurement residual before (top) and after (bottom) calibration. The bottom plot show the union of the residuals for a 5-fold cross-validation.

the upper body cannot be calibrated with such a method as the corresponding joints do not affect the IMU measurements. Our method instead uses artificial markers on the end-effectors, allowing calibration of the whole body.

Park et al. (2012) optimize the Denavit-Hartenberg parameters of a manipulator from observations of a laser attached to the end-effector and an externally placed camera within a Kalman-Filter framework. While this allows for precise measurements, it also requires a substantial modification of the hardware.

Traslosheros et al. (2010) introduced a calibration scheme for a robot with two parallel manipulators and a camera attached to the EEF. As a marker, the approach uses a ball of known dimensions that is fixed within the environment. The ball's projection into the image plane is used as measurement to constrain the system of equations arising from multiple configurations. In their approach, the authors assume that the camera's intrinsics are already well-known from a separate calibration.

Pradeep et al. (2010) presented a bundle adjustment approach for calibrating a robot with multiple sensors and two arms. The main idea here is that multiple sensors can observe the same points in the world and hence, the parameters can be jointly optimized.

Birbach et al. (2012) proposed an approach to calibrate the intrinsic and extrinsic parameters of a stereo camera, the joint elasticities and the joint angle offsets of an upper body of a humanoid with two arms, and the head mounted IMU. As in the work by Pradeep et al. (2010), the authors jointly optimize the parameters from manually determined robot configurations. The authors use a single point marker located on each of the arms' wrists. Similarly, Hubert et al. (2012) applied a maximum a posteriori approach (MAP) for estimating the parameters of hand-eye kinematics of a humanoid robot. The prior is assumed to be normal distributed with an empirically determined variance. If the prior is well chosen, the MAP approach requires fewer measurements and thus is faster than a maximum likelihood formulation. Our approach, on the other hand, automatically selects configurations for calibration such that the number of needed measurements is reduced.

Selection and generation of measurement configurations has been studied previously, especially in the context of manipulator calibration, for instance as described by Borm and Menq (1989). Their goal is to minimize the effect of noise in measurement. Using an observability measure based on the Jacobian of the error function, the authors show that the selection of the measurement configurations is more important than the actual number of measurements. Our system follows that idea by evaluating different observability indices. Also, Li et al. (2011a,b) used observability indices to choose configurations for calibration of a manipulator. The authors showed that using the optimized configurations yields better results than using configurations sampled from the boundary of the working space, which is otherwise a reasonable heuristic. Similar results were shown by Daney (2002) for a hexapod.

Sun and Hollerbach (2008) presented a theoretical survey on five different observability indices for configuration selection and their relationship to the paradigm of optimal experimental design. Most similar to our approach, Carrillo et al. (2013) recently experimentally compared different criteria for the selection of measurement configurations for the calibration of a humanoid's upper body. The authors conclude that for greedy-selection algorithms, determinant-based criteria should be preferred. However, the authors only showed that the determinant-based criteria maximize the other criteria as well, but not whether it minimizes the actual calibration error. In this chapter, we presented extensive cross-validation results for four different selection criteria.

2.5 Conclusions

Accurate calibration is a necessity for all types of navigation and manipulation, including the approaches presented in the following chapters of this thesis. In this chapter, we therefore introduced new techniques for the automatic calibration of the complete kinematic model for legged humanoids. Our system relies on least-square minimization of the sum of residuals between camera observations of markers and their expected locations in the image. Our approach automatically chooses robot configurations that enable an accurate, robust calibration by minimizing the variance of the parameters to be estimated. The calibration framework is general and can be used for other robots with a different kinematic model and with different types and numbers of cameras, including RGB-D cameras by estimating the camera poses of the RGB and the infrared camera. In extensive experiments with a real Nao humanoid, we showed that our framework requires only few observations to obtain accurate calibration results. In the next chapter, we describe an approach to obstacle detection that uses odometry information to predict the feature flow between consecutive images. Consequently, this approach requires and profits from an accurate calibration as provided by the work described in this chapter.

Chapter 3

Traversability-Estimation with Monocular Cameras

In this chapter, we present an approach to traversability classification solely based on monocular images and odometry estimates. We iteratively estimate the ground plane by detecting and matching features. Since the features are only sparse in the images and do not lead to dense information about traversability, we propose a technique to train appearance-based floor detectors. In this way, we achieve a dense classification of the image data. Our approach trains the classifiers online in a self-supervised fashion from the ground plane estimation. During robot navigation, the classifiers are automatically updated and applied to the image stream to decide which areas are traversable. From this information, the robot computes a two-dimensional occupancy grid map of the environment which can be used to plan collision-free paths. As we illustrate in thorough experiments with a real humanoid, the classification results of our approach are highly accurate and the resulting occupancy map enables the robot to reliably avoid obstacles during navigation. Our appearance-based classifiers can also be used to augment stereo or RGB-D data in close ranges where these sensors cannot provide any depth information.

Collision-free navigation is one of the most essential capabilities of autonomous robots. Typically, they use distance data, e.g., from laser range finders or stereo cameras to detect obstacles. Recently, depth cameras based on structured light, such as the Microsoft Kinect or the Asus Xtion, have become available on the consumer market and have been used for navigation with mobile robots (Biswas and Veloso, 2011; Bylow et al., 2013; Huang et al., 2011). Also in Chapter 4 and Chapter 5 of this thesis, we describe navigation approaches using these sensors. However, due to size and weight constraints, monocular cameras

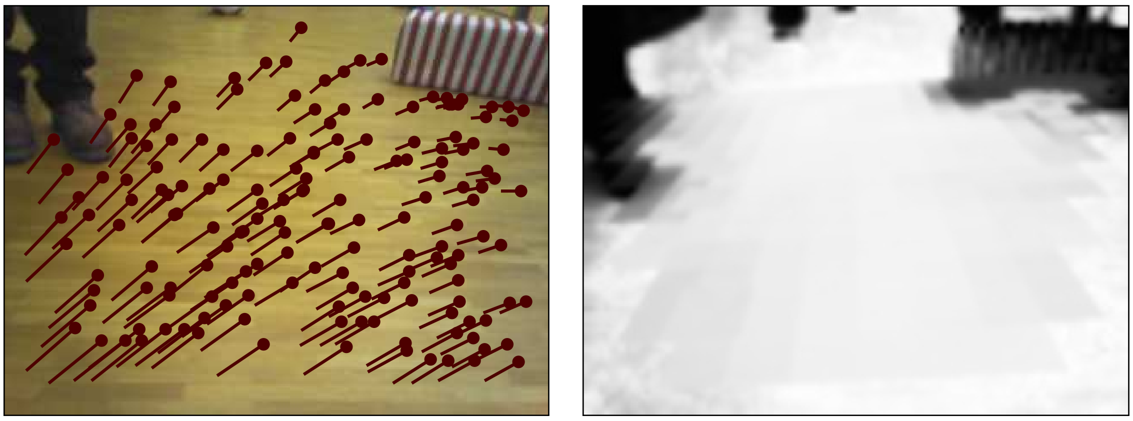


Figure 3.1: Left: Image captured by the camera of a walking humanoid robot with flow vectors of detected features on the estimated ground plane. Given the floor features, we train visual classifiers to densely label patches of subsequent images as being traversable or not. Right: Resulting traversability probabilities for the successive camera image as estimated by our approach (the brighter the color, the higher the probability of being traversable).

are still the typical sensor light-weight robots, including humanoids, are equipped with. Furthermore, depth cameras typically provide no information in the crucial near range, as opposed to monocular cameras.

In this chapter, we therefore present an extension of our previous work on obstacle detection (Maier et al., 2013c) that operated on monocular images and sparse laser data. In the previous approach, the robot trained visual classifiers for obstacles and the floor based on classified distance data, which lead to highly accurate image labeling and traversability classification results. Since not all humanoid robots possess a laser range finder, whose data can be used to acquire the training data, we developed a novel approach that relies solely on monocular images and odometry data. The approach does not require any distance data. Instead, our method iteratively finds sparse features on the floor plane from geometric information and deploys these features to train appearance-based classifiers. To obtain geometric information, the approach considers the flow of the sparse image features and uses odometry from forward kinematics to constrain the search for correspondences. The trained classifiers are subsequently applied to densely label the camera images during walking. From the labeled pixels, we construct a traversability map of the robot's surroundings in form of a 2D occupancy grid map in which a collision-free path can be computed. The learned classifiers are constantly updated during navigation. In this way, the robot can deal with changing ground appearance and reliably identify the traversable floor.

The left image of Figure 3.1 shows an image acquired from the onboard camera of a walking robot. The dots show the location of detected features on the floor plane. Due

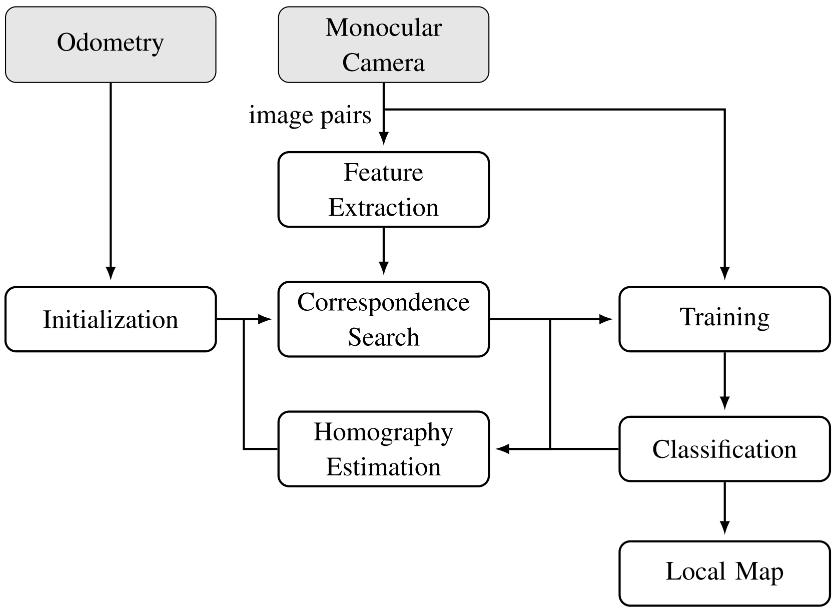


Figure 3.2: Overview of the proposed system. We compute corresponding features in image pairs by iteratively estimating the ground plane homography and compatible feature pairs. The system is bootstrapped from the odometry. Features identified as belonging to the floor are then used to train appearance-based visual classifiers. These classifiers are applied to estimate dense traversable areas in the images and to guide the homography estimation in subsequent image pairs.

to the motion of the camera, the locations of the features in the image change from one image to another as indicated by the lines. The displacements are often referred to as flow vectors. Our approach iteratively estimates the ground plane based on the detected flow. From the identified features lying on the floor, we then train visual classifiers to densely label the traversable area in successive camera images (right image of Figure 3.1).

Several approaches for estimating traversability from a moving monocular camera have been proposed (Braillon et al., 2006; Einhorn et al., 2009; Kim and Kim, 2004; Low and Wyeth, 2005). Most of them try to recover structural information about the environment from the camera motion. These methods often rely on simplifying assumptions such as accurate knowledge of the camera pose or smooth motions that do not hold for walking humanoid robots. Further popular approaches relying only on feature correspondences (Klein and Murray, 2007) are unable to cope with rotations when the translation is small and hence, are only of limited use for humanoid navigation. In contrast to that, our technique provides a dense labeling of the images from which the traversable areas can be inferred, while the robot is walking, standing, or turning on the spot.

As the experiments with a Nao humanoid demonstrate, our approach achieves high classification rates and leads to an accurate, dense labeling of the camera images. The resulting occupancy grid map allows collision-free navigation of the robot.

3.1 Outline of the Approach

Figure 3.2 gives an overview of our system. At its core, the system extracts features in consecutive camera images and finds their correspondences. To constrain the search and find only features on the ground plane, we make use of the *homography* between two images. A homography is a projective mapping between two images and explained in Section 3.2. It can be estimated from a set of feature correspondences. Consequently, our approach iteratively matches features within an image pair, guided by the homography, and refines the estimate for the latter from the correspondences. To bootstrap and constrain the system, the homography is initialized from the odometry obtained via forward kinematics. Once the correspondences on the floor plane are established, we interpret the sparse features as training examples for the appearance of the floor and train visual classifiers. Subsequently, the classifiers are applied to assign *dense* traversability labels to successive images, which we then use to construct a local map of the scene around the robot for navigation. These steps are repeated continuously.

3.2 Homographies

To relate the location of two feature points between consecutive images, our approach relies on homographies. A homography is an invertible, projective mapping between two planes. Here, we consider the special case that corresponding points $\tilde{\mathbf{x}} \leftrightarrow \tilde{\mathbf{x}}$ in two images of a plane $\boldsymbol{\pi}$ in 3D are mapped to each other by a homography $H \in \mathbb{R}^{3\times3}$. This means, that $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}$ represent the same point on the plane $\boldsymbol{\pi}$ in 3D but in different camera views. We call H the homography induced by $\boldsymbol{\pi}$ and all corresponding points on $\boldsymbol{\pi}$ fulfill

$$\tilde{\mathbf{x}} = H\,\tilde{\mathbf{x}}, \tag{3.1}$$

where $\tilde{\mathbf{x}}$ indicates that the image point is expressed in homogeneous coordinates as described in Appendix A.1 of this thesis.

3.3 Geometric Floor Estimation

For identifying points on the floor, which we seek to train the visual classifiers as described in Section 3.4, we match image features that are compatible with the floor homography of image pairs captured by the onboard camera while the robot is moving. One of the biggest problems with extracting and matching features between images from a moving camera is the occurrence of motion blur. Especially humanoid robots often move

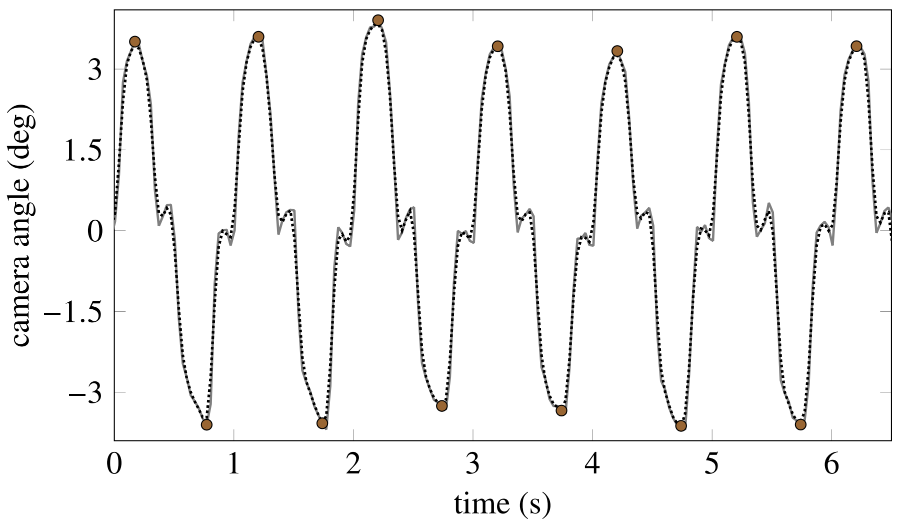


Figure 3.3: The plot shows the camera's lateral inclination while the robot moves. The gray line is the lateral inclination angle of the robot's camera relative to gravity. The black line shows the smoothed trajectory and the dots indicate the detected steady phases for image extraction.

jerkily which induces blur effects in the camera image. Hence, we first introduce our technique to acquire steady images that further have the maximum possible baseline.

3.3.1 Identification of Steady Images While Walking

To identify phases in which the camera is steady, we exploit the specific movement of humanoid robots, which often sway sideways during walking. When the direction of the swaying changes, the body has to undergo a short phase with zero velocity before it sways the other way. Our algorithm tries to detect these phases and extracts the camera images at that moment. A further advantage of this approach is that the recorded camera images have the maximum possible baseline during two foot steps which makes computing the planar homography numerically more stable.

To detect the corresponding phases, we compute the lateral inclination angle of the camera relative to gravity using an integrated inertial measurement unit (IMU). We use a delayed running average filter with a window length of 0.1 s and group delay compensation on the angular measurements to smooth the data. Then, we identify the steady points as the extrema of the smoothed function that exceed an experimentally determined threshold of 3°. Figure 3.3 shows a plot of typically obtained angles, the smoothed trajectory and extracted steady points. As a result, we are able to identify camera images with little motion blur and a relatively large baseline that we use for geometric floor estimation as described next.

3.3.2 Feature Extraction and Association for Floor Estimation

To identify points on the floor plane, we follow an iterative approach. We consider steady image pairs in which we detect and match features that are compatible with the homography H induced by the ground plane. The actual initialization of H is described further below. From such a homography, we search for correspondences $\tilde{\mathbf{x}}_j \leftrightarrow \tilde{\mathbf{x}}_{j+1}$, with $\tilde{\mathbf{x}}_j \in I_j$ and $\tilde{\mathbf{x}}_{j+1} \in I_{j+1}$, where I_j and I_{j+1} are images from the robot's onboard camera via

$$\left\| \tilde{\mathbf{x}}_{j+1} - H \, \tilde{\mathbf{x}}_j \right\| < \xi, \tag{3.2}$$

for a threshold ξ. In practice, the threshold is larger than 0 due to imperfect camera calibration and models. From the feature pairs corresponding to points on the floor, the homography H can in turn be re-estimated via least-squares fitting.

Specifically, we consider successive steady images I_j and I_{j+1} as identified in Section 3.3.1 in which we extract Speeded Up Robust Features (SURF, Bay et al., 2006). Let the set of detected features in image I_j be $\mathcal{B}_j$ and the features from I_{j+1} be $\mathcal{B}_{j+1}$. We seek all the correspondences $\{(\tilde{\mathbf{x}}_j, \tilde{\mathbf{x}}_{j+1})\}$, where $\tilde{\mathbf{x}}_j \in \mathcal{B}_j$ and $\tilde{\mathbf{x}}_{j+1} \in \mathcal{B}_{j+1}$, that represent the same point on the floor in the images I_j and I_{j+1}, respectively. Consequently, we seek the set $\mathcal{S}$, where

$$\mathcal{S} = \left\{ (\tilde{\mathbf{x}}_j, \tilde{\mathbf{x}}_{j+1}) \;\mid \tilde{\mathbf{x}}_j \in \mathcal{B}_j, \tilde{\mathbf{x}}_{j+1} \in \mathcal{B}_{j+1}, \left\| \tilde{\mathbf{x}}_{j+1} - H \, \tilde{\mathbf{x}}_j \right\| < \xi \right\}. \tag{3.3}$$

As neither $\mathcal{S}$ nor H can be estimated directly, our system iterates the estimation of both, given a rough estimate for H from the odometry (see below). We denote by $\mathcal{S}^{(k)}$ and $H^{(k)}$ the corresponding estimates at iteration k. At each iteration k of our algorithm, we update the estimate $H^{(k-1)}$ for the floor homography from the current set of correspondences $\mathcal{S}^{(k-1)}$ to yield $H^{(k)}$ and then refine the correspondences $\mathcal{S}^{(k-1)}$ from $H^{(k)}$ to yield $\mathcal{S}^{(k)}$. The two iterating steps, searching for correspondences and estimating the homography, as well as the initialization from the odometry, are explained in the following.

Correspondence Search

To find the set of correspondences $\mathcal{S}^{(k)}$, we consider the homography $H^{(k)}$ and the 64-dimensional SURF descriptor. We define the set of candidates of potentially matching features C_i for a feature $\tilde{\mathbf{x}}_j \in \mathcal{B}_j$ according to the homography $H^{(k)}$ as

$$C_i = \{ \tilde{\mathbf{y}} \in \mathcal{B}_{j+1} \mid \left\| \tilde{\mathbf{y}} - H^{(k)} \, \tilde{\mathbf{x}}_j \right\| < \xi^k \} \tag{3.4}$$

where ξ^k is the current search radius. As actual correspondence to $\tilde{\mathbf{x}}_j$, we select the most visually similar candidate feature $\tilde{\mathbf{x}}_{j+1} \in C_i$ according to the SURF descriptor $d_{\mathrm{SURF}}(\cdot)$, i.e.,

$$\tilde{\mathbf{x}}_{j+1} = \underset{\tilde{\mathbf{y}} \in C_i}{\arg\min} \left\| d_{\mathrm{SURF}}(\tilde{\mathbf{y}}) - d_{\mathrm{SURF}}(\tilde{\mathbf{x}}_j) \right\|. \tag{3.5}$$

We also apply the common constraint that the ratio between the descriptor distance of the best match $\tilde{\mathbf{x}}_{j+1}$ to the descriptor distance of the next best match exceeds a predefined threshold to find robust correspondences. Further, we perform the mutual consistency check, i.e., we demand that $\tilde{\mathbf{x}}_j$ is also within the most similar correspondences for $\tilde{\mathbf{x}}_{j+1}$ in terms of the descriptor distance, for the same reason. $\mathcal{S}^{(k)}$ is then the set of all pairs $(\tilde{\mathbf{x}}_j, \tilde{\mathbf{x}}_{j+1})$ fulfilling these constraints. In each iteration k, we reduce the search radius ξ^k. In this way, we reduce the ambiguities in the feature matching process because the number of candidates $|C_i|$ shrinks with ξ^k.

Homography Estimation

To estimate $H^{(k)}$, we apply a standard non-linear optimization from $\mathcal{S}^{(k-1)}$. Thus, we find $H^{(k)}$ via a least-squares solution for $\tilde{\mathbf{x}}_{j+1} = H^{(k)}\,\tilde{\mathbf{x}}_j$, i.e., the algorithm minimizes

$$\sum_{i=1}^{n} \left(\tilde{\mathbf{x}}_{j+1} - H^{(k)}\,\tilde{\mathbf{x}}_j\right)^{\mathsf{T}} \left(\tilde{\mathbf{x}}_{j+1} - H^{(k)}\,\tilde{\mathbf{x}}_j\right), \tag{3.6}$$

where $n = \left|\mathcal{S}^{(k-1)}\right|$. For the minimization, we use the corresponding OpenCV (Bradski and Kaehler, 2008) function.

Initialization

To initialize the algorithm, we consider the homography $H^{(0)}$ computed from the odometry and the IMU as

$$H^{(0)} = K \begin{bmatrix} \mathbf{e}_1^{j+1} & \mathbf{e}_2^{j+1} & \mathbf{e}_4^{j+1} \end{bmatrix} \begin{bmatrix} \mathbf{e}_1^{j} & \mathbf{e}_2^{j} & \mathbf{e}_4^{j} \end{bmatrix} K^{-1} \tag{3.7}$$

where K is the camera's intrinsic matrix and $\mathbf{e}_i^j$ is the ith column of the camera's extrinsic matrix corresponding to image I_j, i.e., its pose according to the odometry and forward kinematics at the time of capturing image I_j (see Appendix A.2). The $\mathbf{e}_3^j$ have been dropped due to the projective nature of homographies. A derivation for (3.7) can be found in Bradski and Kaehler (2008, Chapter 11).

We then initialize $\mathcal{S}^{(0)}$ from $H^{(0)}$ as well as the detected features $\mathcal{B}_j$ and $\mathcal{B}_{j+1}$ as described before, using an appropriate search radius ξ^0, which depends on the noise in the odometry estimate and the error in the robot calibration (see Chapter 2). To include prior knowledge, we further demand that the features $\tilde{\mathbf{x}}_j \in \mathcal{B}_j$ of the first image are from parts labeled as floor by the appearance-based classifiers presented in the next section, if available. This constraint further improves the initialization but is waived for all subsequent iterations, in order to be able to adapt to changing floor appearances. From the remaining correspondences in $\mathcal{S}^{(0)}$, our system robustly estimates $H^{(1)}$ using the random sample consensus (RANSAC) algorithm introduced by Fischler and Bolles (1981). The advantage of using RANSAC for estimating the homography is that few outliers have no effect on the estimate (given a sufficiently high number of samples), while each outlier will impact the estimate in a least-squares optimization, even if a robust kernel (see Section 2.1.3) is used. Outliers can still occur due to ambiguities in the matching process and the uncertainty of the initial homography. RANSAC is only used to estimate $H^{(1)}$. Afterwards, we proceed with the iterative refinement using least-squares optimization, as a robust homography is established.

Iterative Refinement

We iterate the two steps of *correspondence search* and *homography estimation* until the number of correspondences $\left|\mathcal{S}^{(k)}\right|$ has converged or a maximum number of iterations is reached. Typically, five to ten iterations are sufficient. In this way, the final $\mathcal{S}^{(k)}$ consists of a set of corresponding features on the floor plane, obtained from two subsequent images I_j and I_{j+1}, We denote by $\mathcal{S}_{\text{floor}}$ the final $\mathcal{S}^{(k)}$ and its elements are the pairs $(\mathbf{x}_{\text{floor}}, \mathbf{x}'_{\text{floor}}) \in \mathcal{S}_{\text{floor}}$ where all the $\mathbf{x}_{\text{floor}}$ are extracted from I_j and the $\mathbf{x}'_{\text{floor}}$ from I_{j+1}. These features are consequently used to train visual classifiers as described in the following section. Figure 3.1 illustrates one result of our algorithm. The left image shows the image I_j along with the detected floor points $\mathbf{x}_{\text{floor}}$ and the corresponding flow vectors, i.e., the displacements $\mathbf{x}'_{\text{floor}} - \mathbf{x}_{\text{floor}}$ for all $(\mathbf{x}_{\text{floor}}, \mathbf{x}'_{\text{floor}}) \in \mathcal{S}_{\text{floor}}$. As can be seen, our algorithm extracts features from a challenging floor and the flow is consistent with the camera's motion, which is a right-forward movement with a slight roll component.

3.4 Appearance-based Traversability Estimation

The detected points on the floor (i.e., $\{\mathbf{x}_{\text{floor}}\}$ and $\{\mathbf{x}'_{\text{floor}}\}$) are typically sparse in the images as illustrated in Figure 3.1. However, for collision-free navigation, dense information is required. We therefore developed an appearance-based approach to estimate traversability from images. Here, we consider a one-class classification problem since we have accurate information about features on the floor but no reliable information about obstacles.

Consequently, we train a texture-based and a color-based classifier using the detected floor points $\mathbf{x}_{\text{floor}}$ from the image I_j, as explained in the next section. These classifiers are then used to label all successive images, including I_{j+1}, until new training data is available. Note that all images can be labeled, not only the steady images used for ground plane estimation (see Section 3.3.1). This is necessary to be able to update the map while the robot is standing or turning.

3.4.1 Texture-Based Classification

The texture classifier that we designed labels image patches according to their representation in the frequency-domain. In particular, we employ the discrete cosine transformation (DCT) to extract texture information (Ahmed et al., 1974). For an input image patch, the DCT computes a set of coefficients which can be regarded as weights of a set of two-dimensional basis functions corresponding to different frequencies. Hence, the coefficients represent the amount of presence of certain frequencies in the image.

Since the DCT is not invariant to perspective, we do not operate on the camera image directly but project it to a virtual downwards-looking camera. This projection is computed by applying a homography induced by the floor plane between the camera and a virtual camera looking perpendicular at the floor at a fixed height of 2 m and 1 m in front of the real camera. The homography is computed from the odometry and IMU estimate for the camera, similar to (3.7). This allows to compensate for potentially different camera pitch and roll angles between training and test data. The left image in Figure 3.4 shows an example of such a top-down view. Note that this step works also with inaccurate odometry information: Planar translational errors have no impact on the projection, the height is typically fixed, and we observed the classifier to be robust to rotational (yaw) errors up to circa 30°.

From the projected image, we extract rectangular patches of size 16×16 pixels. To these patches, we apply the DCT and keep only the 13 coefficients corresponding to the lowest frequencies in the image patch for dimensionality reduction and generalization. The extraction of these features is described in more detail in our previous publication (Maier et al., 2013c). For training, we extract the patches from the projected camera image at the location of the features $\{\mathbf{x}_{\text{floor}}\}$, compute the DCT-based descriptor and store it in a k-d tree (k-dimensional tree, Bentley, 1975).

For classification, we project an image to the top-down view and thereby align the orientation of the top-down view with the orientation of the top-down view of the training image. In this way, we compensate for the rotational invariance of the DCT-based descriptor. Subsequently, we extract 16×16 patches at a fixed distance of 8 pixels, compute for each patch its DCT-based descriptor $d_{\text{DCT}}(\cdot)$ and determine the traversability

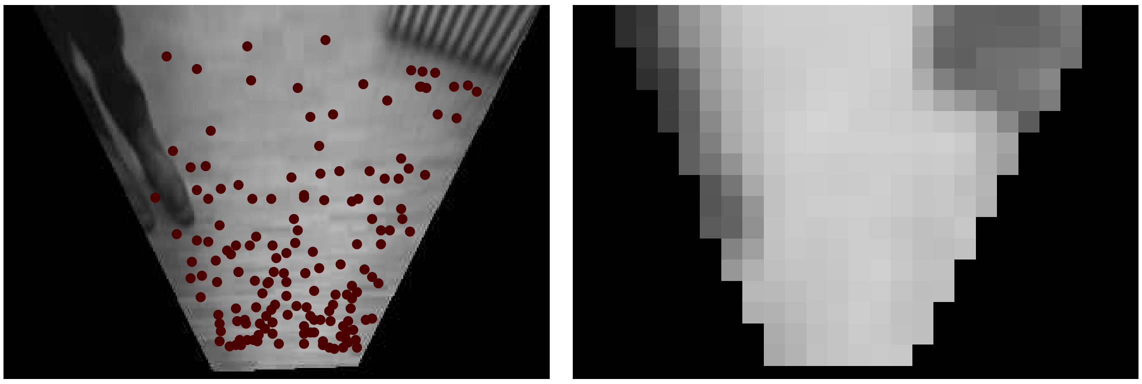


Figure 3.4: Virtual downwards-looking camera as described in Section 3.4.1. The images correspond to the situation depicted in Figure 3.1. In the left image, the dots indicate the detected floor points $\mathbf{x}_{\text{floor}}$. The right image shows the traversability estimation based on texture (the brighter the color, the higher the probability of being traversable).

probability from the similarity to the training data. Here, we use the distance of the descriptor $d_{\text{DCT}}(h)$ for a patch h to its nearest neighbor h' in the k-d tree according to $d_{\text{DCT}}(\cdot)$ and define the probability of h being traversable as

$$p(h) = \exp\left(-\frac{\|d_{\text{DCT}}(h) - d_{\text{DCT}}(h')\|}{\sigma_{\text{DCT}}}\right). \tag{3.8}$$

Hence, we apply the exponential probability density function with scale σ_{DCT}, which controls the steepness of the function. We average the probabilities for individual pixels from overlapping patches. The right image in Figure 3.4 shows an example classification result in terms of traversability probabilities.

3.4.2 Color-Based Classification

Due to the projection to the virtual downwards-looking camera, some parts of the original image cannot be labeled. This is demonstrated in Figure 3.4 where a part of the background of the original camera image shown in Figure 3.1 is missing in the projection. Also, the regions close to the border of the visible area are not suitable for texture-based classification, as the patches might include parts outside the visible area. Therefore, we train a second classifier based on color to achieve a classification of the whole image.

We compute the average HSV (hue, saturation, value) color representation in a 4×4 pixels neighborhood around the floor points $\mathbf{x}_{\text{floor}}$ in the original image and store them in a k-d tree. The descriptor $d_{\text{HSV}}(\cdot)$ is a vector of the three HSV components. For classification, we again divide the image into 4×4 patches and assign traversability probabilities according to the distance to the nearest neighbor using an exponential probability density function as in (3.8).

3.4.3 Probabilistic Relaxation Labeling

For the final classification, we combine the prediction from both classifiers via a weighted average, where we assign a higher weight to the texture classifier. For considering dependencies between nearby areas and smoothing, we apply probabilistic relaxation labeling (Maier et al., 2013c; Rosenfeld et al., 1976). Here, we assume that each node v_i, corresponding to a pixel in a down-sampled version of the combined classification image, stores a binary probability distribution about its traversability label. More concrete, $p_i(t)$ is the probability that node v_i is labeled $t \in \mathcal{T}$, where we denote the set of possible labels by $\mathcal{T} = \{\text{traversable}, \text{non-traversable}\}$. The neighborhood relation between two nodes is represented by two coefficients: c_{ij} representing the *influence* between two nodes v_i and v_j as well as r_{ij} describing the *compatibility* between the labels of the two nodes.

In details, $\mathcal{C} = \{c_{ij} \mid v_j \in \mathcal{N}(v_i)\}$ is the set of weights indicating the influence of node v_j on node v_i. Here, $\mathcal{N}(v_i)$ describes the 8-neighborhood of v_i. In our case, we set the weights c_{ij} to $|\mathcal{N}(v_i)|^{-1}$, thereby assigning equal influence to each neighbor. The compatibility coefficients are the set $\mathcal{R} = \{r_{ij}(t,t') \mid v_j \in \mathcal{N}(v_i)\}$, where $r_{ij}(t,t')$ with $t,t' \in \mathcal{T}$ defines the compatibility between the label t of node v_i and the label t' of node v_j by a value between -1 and 1. A value $r_{ij}(t,t')$ close to -1 indicates that the label t' is unlikely at node v_j given that node v_i has label t. Values close to 1 indicate the opposite. For our application, we set $r_{ij}(t,t')$ to a constant $r \in \mathbb{R}$, if $t = t'$ and to $-r$ otherwise, where we determined r is empirically.

Given an initial estimation for the probability distribution $p_i^{(0)}(t)$ over the traversability label for each node v_i, as obtained via the weighted average over the texture and color classification, the probabilistic relaxation method iteratively computes estimates $p_i^{(k)}(t)$, $k = 1,2,\ldots$ for the labels via

$$p_i^{(k+1)}(t) = \frac{p_i^{(k)}(t)\left(1 + q_i^{(k)}(t)\right)}{\sum_{t' \in \mathcal{T}} p_i^{(k)}(t')\left(1 + q_i^{(k)}(t')\right)}, \tag{3.9}$$

where

$$q_i^{(k)}(t) = \sum_{v_j \in \mathcal{N}(v_i)} c_{ij} \left(\sum_{t' \in \mathcal{T}} r_{ij}(t,t') p_j^{(k)}(t') \right). \tag{3.10}$$

In the equations above, the term $q_i^{(k)}$ represents the change of the probability distribution $p_i^{(k)}$ for node v_i in iteration k based on the current distribution for its neighbors, the compatibility coefficients and the weights c_{ij}. We stop the method after a fixed number of iterations or if the change in the label probabilities falls below a threshold. The final $p_i^{(k)}$ yield the traversability classification of the image. An example for the final classification after relaxation labeling is depicted in the right image of Figure 3.1.

3.4.4 Using Traversability Information for Navigation

For planning the motion of the robot, we maintain a 2D occupancy grid map (Moravec and Elfes, 1985) to integrate the traversability information from the classified camera data over time. Each cell of the map represents the probability of being traversable. To integrate the traversability information from the camera, we project the labeled camera image to a virtual top-down looking camera as described in Section 3.4.1. By using bilinear interpolation, the traversability information from the camera image is mapped to the coordinate frame of the occupancy map. We update each cell using standard occupancy grid map updating. An example of such a map is shown in Figure 3.6 (right image). Afterwards, the map is used for planning collision-free motions by applying a variant of the A* algorithm (Hart et al., 1968). More details on occupancy grid maps can also be found in Section 4.3 of this thesis.

3.5 Experiments

We conducted a series of experiments on a Nao V4 humanoid robot to demonstrate that our approach is able to reliably learn about traversability using only odometry information and monocular images. In the experiments, we used the top camera in the robot's head. The robot is described in detail in Appendix B. For all experiments, we weighted the prediction of the texture-based classifier with 0.9 and the color-based classifier with 0.1. The values were experimentally determined. We further set the radius ξ, in which candidates for feature correspondences are searched, proportional to the length of the expected flow according to the homography. This way we allow smaller deviations for features in the background, which move only little between two images, and allow larger deviation for features in the foreground.

3.5.1 Qualitative Results on Traversability Estimation

In the first experiment, we placed various obstacles on the floor in our lab. The robot walked through the scene to a designated goal location. Figure 3.5 shows classifica-

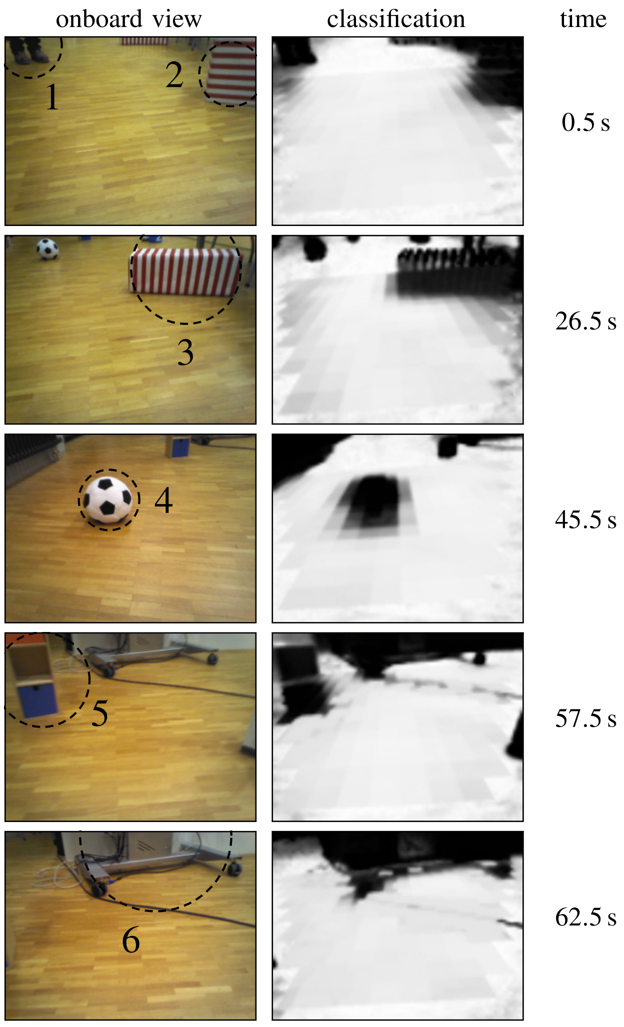


Figure 3.5: Qualitative evaluation of traversability estimates. The rows show the robot's onboard camera images (left) and the corresponding traversability estimates (middle) using our approach along with the timestamps of the images (right). Note that the varying coarseness within the labeled images stems from combining two different classifiers with different resolutions. The numbers in the first column indicate correspondences of objects with Figure 3.6.

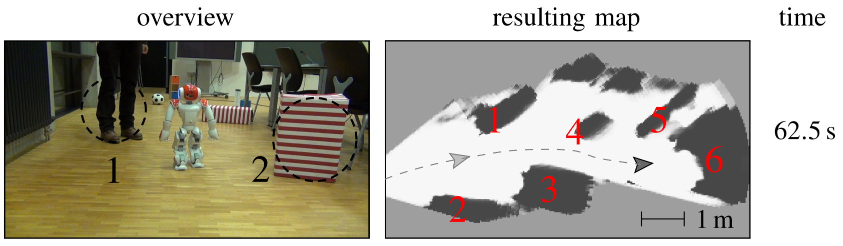


Figure 3.6: Overview of the scene (left) and the learned occupancy grid map (right). In both pictures, the numbers correspond to the obstacles marked in Figure 3.5. In the map, obstacles appear dark, traversable area white, and unknown gray. The light arrow indicates the robot's pose corresponding to the left picture and the dashed line illustrates the robot's trajectory. The map is shown at the time the robot reached its goal location (dark arrow).

tion results obtained during this experiment. The rows show the camera image and the traversability estimates, ordered chronologically from top to bottom. Figure 3.6 shows an overview picture of the scene as well as the state of the learned occupancy grid map at the time when the robot reached the goal location. As one can see, our approach results in correctly labeled images and the constructed grid map contains all the encountered obstacles, as well as appropriate free-space for collision-free navigation. Note that some obstacles might appear slightly larger in the map than their actual footprint due to the applied perspective transformation. This issue resolves as soon as the robot approaches the object and observes the free space behind it.

3.5.2 Adapting to Changing Ground Appearance

The following experiment is designed to illustrate that our approach can automatically adapt to new environments, specifically to changing ground appearance. To illustrate this, the robot started to navigate on a parquet floor. Distant, we placed a PVC floor coating that was visually dissimilar from the parquet. Figure 3.7 shows in the columns the onboard camera view, the resulting classification and the detected feature points with corresponding flow vectors, respectively. The rows represent different time stamps in ascending order. As can be seen, the robot immediately labeled the parquet as traversable (first row) but considered the PVC floor as obstacle due to its dissimilarity from the parquet and only few feature matches for the PVC floor in the background. As soon as the robot got closer (second row) and hence, could detect features on the PVC floor for training the classifiers, the PVC as well as the parquet were labeled as traversable (third row). The resulting grid map, shown in Figure 3.8, correctly represents the area corresponding to the PVC floor as traversable.

3.5.3 Classification Accuracy

To quantitatively evaluate our approach, we compared the traversability estimates for the camera images assigned by our algorithm to manually assigned ground truth labels for the experiments described in Section 3.5.1 (experiment 1) and Section 3.5.2 (experiment 2), respectively. For economic reasons, we only evaluated every fifth steady image as explained in Section 3.3.1, yielding 14 images for each experiment. To compare the binary manual labels of the images with the traversability probabilities provided by our approach, we considered each pixel in the latter as representing floor area, if its traversability probability was above 0.5 and as obstacle otherwise. Then, we counted the number of pixels coinciding with the manual labels. Table 3.1 presents the classification accuracy. As can be seen, the classification rates are highly accurate. The probability that

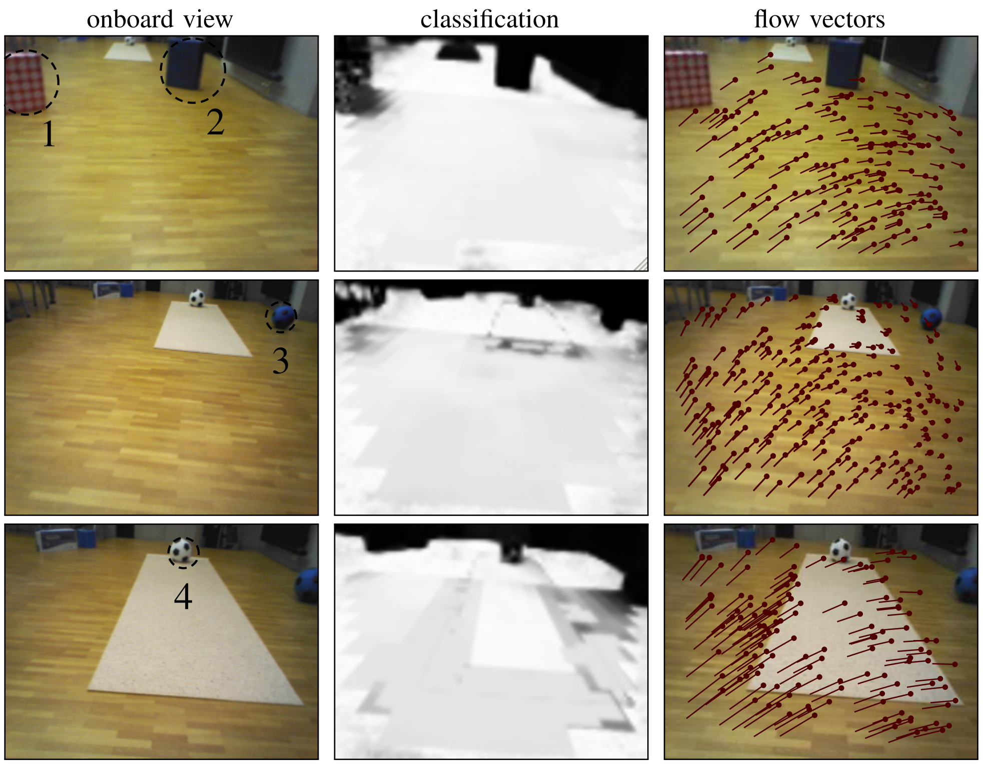


Figure 3.7: Our approach automatically adapts when approaching a floor with different appearance. The rows show the robot's onboard camera images (first column), the traversability estimates (second column), and the detected feature locations with flow vectors (third column) at different times. Both floors are correctly identified as traversable. Correspondences with Figure 3.8 are numbered.

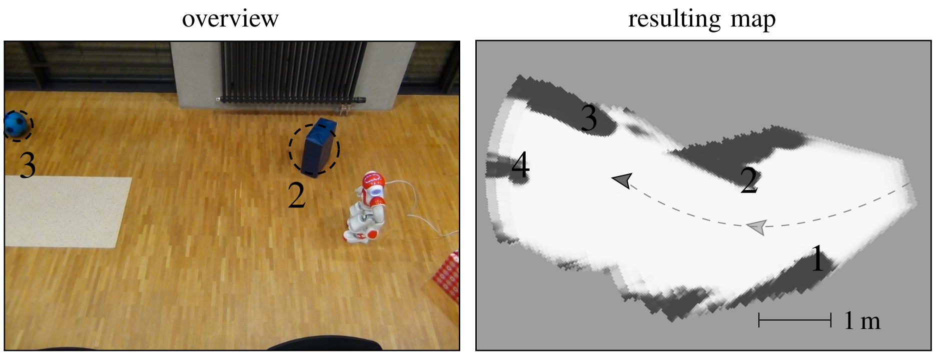


Figure 3.8: External overview picture (left) and the constructed grid map (right). The numbers correspond to the obstacles marked in Figure 3.7. The bright arrow in the map indicates the robot's pose in the overview. The map is shown at the time the robot reached its goal location (dark arrow).

Table 3.1: Traversability classification accuracy.

	experiment 1		experiment 2	
	estimated as		estimated as	
true class	obstacle	floor	obstacle	floor
obstacle	0.88	0.12	0.94	0.06
floor	0.03	0.97	0.06	0.94

a pixel corresponding to an obstacle was classified as traversable space lies between 6% and 12%. Note that in practice, the number of potentially dangerous false classification is even lower because most false classifications occurred in the background of the scene. In these regions, the texture-based classifier has no information due to the projection of the image into a top-down view (see Section 3.4.1) and thus, only color information can be used. As soon as the robot gets closer to these obstacles, they appear in the projected image and the texture classifier can be used as well.

3.5.4 Turning on the Spot

In the next experiment, we demonstrate that our technique is able to classify camera images even if the robot is turning on the spot. Many structure-from-motion based approaches, e.g., *PTAM* (Klein and Murray, 2007) fail in these situations since it is not possible to compute structural information for rotations around the optical center of the camera. In contrast to that, our approach yields correct classification results also for rotations by using the previously learned visual classifiers. After turning, the robot then automatically re-trains the classifiers when it starts to move. Figure 3.9 shows results from the traversability estimation while the robot turned counter-clockwise by circa 100°. As can be seen, the classification results allow the robot to detect obstacles in the scene and will improve even further once the robot starts navigating and updating the classifiers.

3.5.5 Dealing with Moving Obstacles

In a final experiment, we illustrate the ability of our approach to deal with dynamics in the environment. While the robot was walking, we rolled a soccer ball several times into the robot's field of view. Figure 3.10 depicts results for the traversability estimation as well as the corresponding camera images. As can be seen, the camera images are accurately labeled even in the presence of moving obstacles.

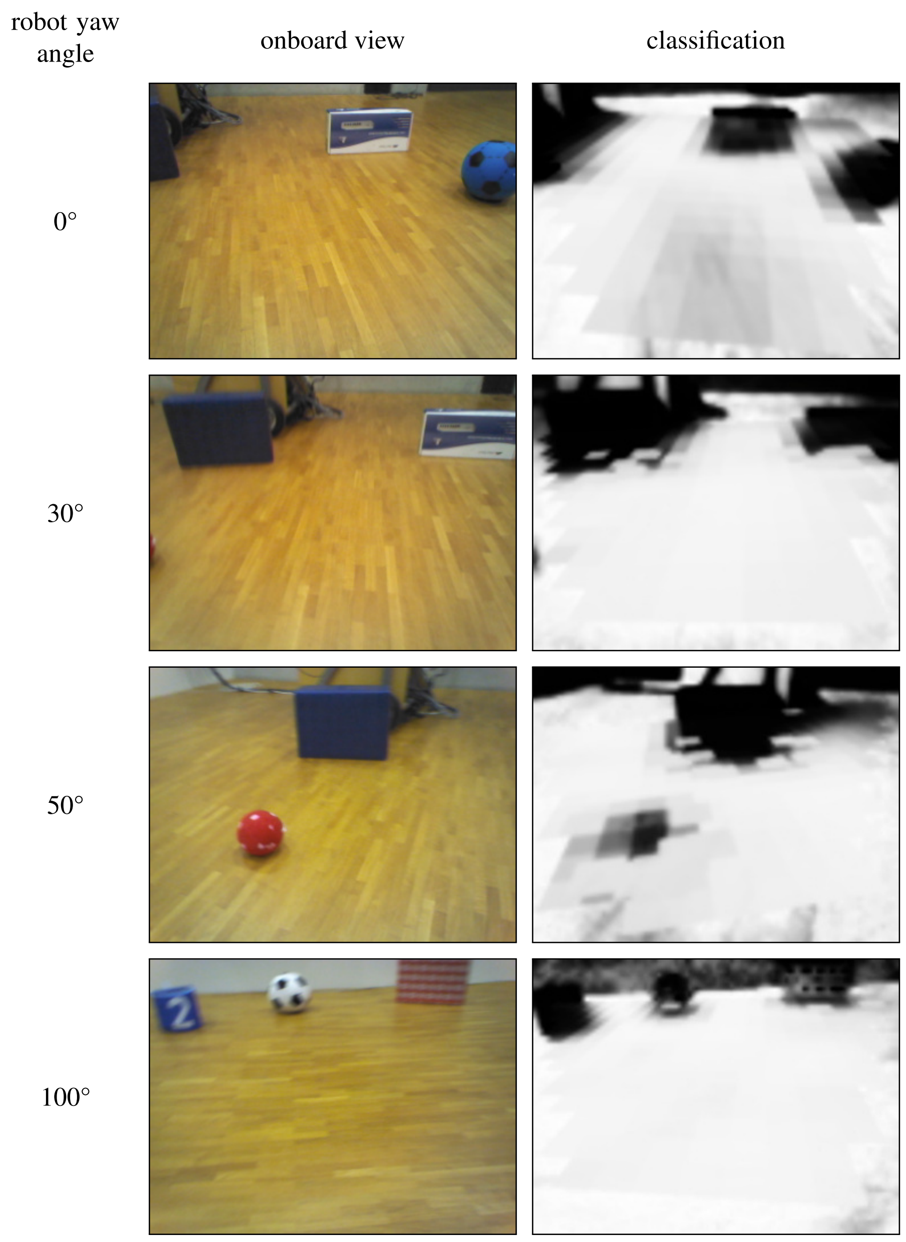


Figure 3.9: Obtained classification results while the robot turns on the spot. The rows show the robot's onboard camera images and the traversability estimates for different rotation angles.

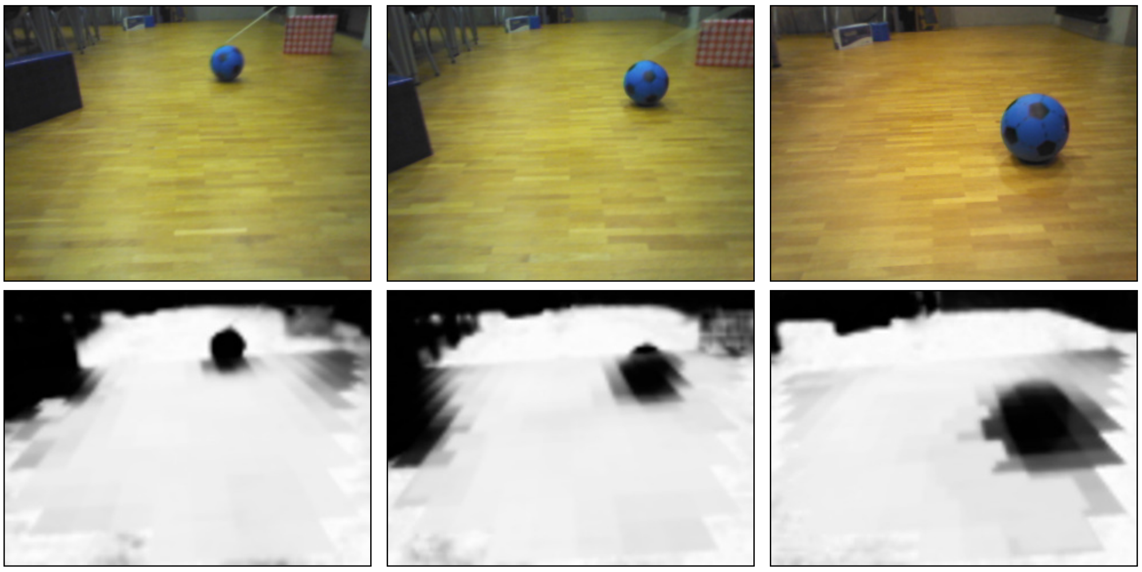


Figure 3.10: We achieve reliable traversability estimates also in presence of moving obstacles. We rolled a soccer ball multiple times in front of the robot's field of view while navigating. The top row shows the camera images and the bottom row the corresponding traversability estimates.

3.5.6 Remarks

If obstacles look (in terms of the texture features and color) very similar to the floor, appearance-based classification will fail. The same may hold for a textureless floor, since it will not generate any interest points. Furthermore, in our approach, we assume that the floor plane is dominant (with respect to the number of features) in the first image pair. Afterwards, the floor can be identified even in the presence of larger obstacles, by also considering the appearance model to establish feature correspondences and estimating the homography. Concerning the computation time, we found that labeling images takes 0.1 s on a single core of a 3 GHz i7 PC. Training the classifiers takes 2.0 s, of which the majority is required for the iterative feature matching. For real-time navigation, we thus cannot re-train from every image pair, but instead can use the existing appearance-based classifiers while the training is in process.

3.6 Related Work

Several techniques have been presented for obstacle detection based on monocular image data. Broadly, these approaches can be categorized according to whether they operate directly on the appearance level or try to recover structural information from a moving camera. Our approach can be considered as a hybrid technique as it acquires training data from structural information but reaches its traversability decision based on appearance. We first discuss related work focusing on structure from motion techniques and then discuss appearance-based approaches.

Einhorn et al. (2009) reconstruct the scene's geometry by tracking features in consecutive images from the robot's camera. Scene reconstructing requires accurate pose estimates of the camera which the authors obtain from odometry and filtering. Thus, this approach is mainly applicable for wheeled robots, as humanoids typically provide only rough odometry information. Scene reconstruction was also tackled by Pretto et al. (2009). They presented a visual odometry system based on specifically designed feature detectors that can cope with effects of motion blur typically occurring during humanoid walking. The detected features are sparse and hence not sufficient for reliable obstacle detection.

A different approach was followed by Wang et al. (2005) and Braillon et al. (2006). They propose to segment a camera image into different regions and apply a given model of floor motion to these regions. A region is considered as traversable if it can be matched in the consecutive image at the computed location based on the assumed floor motion. These methods also rely on accurate odometry data to estimate the floor model. Similarly, Zhou and Li (2006) presented a homography-based approach to detect the ground plane for obstacle detection with a wheeled robot. Hence, they imposed restrictions on the motion of the camera which simplified the computation of the homography. However, these assumptions do not hold for a humanoid, and the approach provides no dense labeling.

Kim and Kim (2004) use dense optical flow to compute the plane normals for small image regions. Traversability is estimated from the normals' consistencies with the estimated normal of the ground plane. However, the authors assumed a wheeled robot in their approach. Obtaining accurate dense optical flow for images from a walking humanoid's onboard camera is difficult to obtain even with a state-of-the-art approach (Sundaram et al., 2010) as we found out in preliminary experiments. The robot's quick motion induces erratic motion blur and the camera's rolling shutter distorts the image further according to the robot's motion. Further approaches based on optical flow to detect obstacles make certain assumptions about the environment that are not generally applicable. For example, Low and Wyeth (2005) presented a method that relies on the assumption that optical flow only occurs at obstacles. Hence, the approach is unsuitable for environments with a textured floor.

Newcombe et al. (2011b) presented *DTAM*, a system for monocular camera-based tracking and dense reconstruction and showed impressive results. The system reconstructs a scene by minimizing a photometric cost function, computed from multiple images. The approach relies on a static world assumption and on brightness consistency. Further, it requires a special initialization stage to construct an initial environment model. This is because an accurate model is needed for tracking and subsequent mapping. It is questionable how well the system performs in case of rapid exploration, i.e., quick movements that typically occur during humanoid navigation.

Recently, Alvarez et al. (2014) showed a collision avoidance system for micro-aerial vehicles (MAV). It computes a dense depth map from a set of images which is then rendered into a top-down view to generate waypoints for the MAV. Similar to our approach, they exploit the motion of the robot (while hovering in case of the MAV) to obtain the pose variation that is needed to extract structural information from a monocular camera. To estimate the pose, the system relies on *PTAM* (Klein and Murray, 2007), hence, the system will fail for pure rotations. Also, prior to navigating, the system requires a hovering phase of a couple of seconds to initialize the depth map for each scene.

Common to all these approaches is that they rely on information from a *moving* camera. Appearance-based approaches, on the other hand, typically classify each camera image independently. For instance, Ulrich and Nourbakhsh (2000) proposed to use color histograms to model traversable areas. They learn the model by retrospectively updating histograms over the color of the floor from a robot's camera while manually steering the robot through the free space. Cupec et al. (2005) expect obstacles at contrast edges in the camera image of a humanoid robot. From the detected edges, they employ a scene analysis that assumes rectangular objects to estimate their poses. This approach requires a controlled environment, especially with respect to shadows. Li et al. (2009) introduced a vision-based obstacle avoidance approach for the RoboCup domain. The authors assume known shapes of the obstacle, i.e., the other field players. They use gradient features learned from training images and, also, apply a color-based classifier and data from ultrasound sensors to determine obstacle positions. Their approach relies on a specific color coding further simplifying the problem.

Further approaches are concerned with estimating depth from a single monocular image. For instance, Michels et al. (2005) and Plagemann et al. (2010) apply regression techniques to learn a mapping from the feature space to distance. The drawback is that the approach requires a prior training period whereas our approach carries out automatic learning while navigating. Similarly, Ross et al. (2013) learn a mapping from feature space to control commands for autonomous MAV flying through a forest. The features are chosen such that they typically correlate with depth, such as those employed by Michels et al. previously. After training, the user can provide a forward velocity, while the controller reactively sets the left-right velocity. Hence, unlike ours, the approach can not be used for mapping and planning. Again, such approaches require an extensive learning phase.

Other authors proposed to divide the camera image into small rectangular patches and to compute feature vectors for each patch (Kim et al., 2006; Ott and Ramos, 2012).

Consequently, they learn models for the traversability of a patch by clustering in feature-space. Both approaches obtain the training data from the robot's interaction with the environment, for instance by driving over or into different terrain. Such strategies are not applicable for humanoids as they could easily lose balance.

Dahlkamp et al. (2006) use vision for extending the perception range of the autonomous car *Stanley* to allow for faster and more forward-looking driving. The authors apply laser data for learning a color-based obstacle classifier. Previously, we presented an approach where a humanoid repetitively adopts a scanning position and tilts its head to acquire 3D range data from an integrated laser scanner (Maier et al., 2013c). Afterwards the robot trains visual classifiers for obstacles and the floor based on classified laser data projected to the camera images. While this technique leads to highly reliable classification results, a drawback is that collecting the 3D data is time-consuming and not all robots can be equipped with depth or range sensors. Therefore, we extended that approach such that it only requires monocular camera data and odometry information. Consequently, the approach is suitable for obstacle detection with most humanoid robots. To the best of our knowledge, this is the first approach that combines geometric traversability analysis with appearance-based techniques to obtain a dense classification of the camera images.

3.7 Conclusions

In this chapter, we presented an approach to estimate traversable areas in the surroundings of a humanoid robot solely based on monocular images and odometry information. We apply an iterative approach to estimate the floor homography and detect feature correspondences compatible with the homography. To obtain dense information about traversability given the sparse floor features, we developed a technique to train appearance-based classifiers based on color and texture. Using the classifiers, which are automatically learned online, the robot labels its onboard camera images and updates a map of the environment while walking, standing, and turning. As we showed in practical experiments with a Nao humanoid, the achieved classification is highly accurate. The labeled images enable safe navigation by maintaining a probabilistic grid map and the robot is constantly updating the classifiers to adapt to changing ground appearance. The proposed appearance-based classifiers can also be used in combination with navigation systems relying on RGB-D sensors, such as the system presented in the next chapter. Depth data is typically not available from these sensors very close to objects. In this case, the RGB image can be analyzed for traversability using our classifiers.

Chapter 4

Robust Navigation Using Depth Cameras

In this chapter, we present an integrated approach for robot localization, obstacle mapping, and path planning in 3D environments based on data of an onboard consumer-level depth camera. We rely on state-of-the-art techniques for environment modeling and localization, which we extend for depth camera data. We thoroughly evaluated our system with a Nao humanoid equipped with an Asus Xtion Pro Live depth camera and present navigation experiments in a multi-level environment containing static and non-static obstacles. Our approach performs in real-time, maintains a 3D environment representation, and estimates the robot's 6D pose. As our results demonstrate, the depth camera-based approach is well-suited for robust localization and reliable obstacle avoidance in complex indoor environments.

A robot can only act autonomously, if it is equipped with *onboard sensors*, that allow pose estimation and obstacle detection independent of the environment. This is especially of importance if the robot is to assist humans with tasks such as home-care, where the environment cannot be adapted to the robot, but vice versa. Such high-level tasks require that the robot is able to localize itself in the environment, detect obstacles, and avoid collisions with them by keeping track of their locations and planning collision-free paths around them. Numerous sensors have been proposed for this purpose, including ultrasound sensors, laser range finders, as well as monocular and stereo cameras. All of these sensors suffer from shortcomings such as inaccuracy in case of the ultrasound sensors or sparseness of the data in case of lasers. Additionally, laser range finders are typically expensive and also heavy which is incompatible with the limited payload of

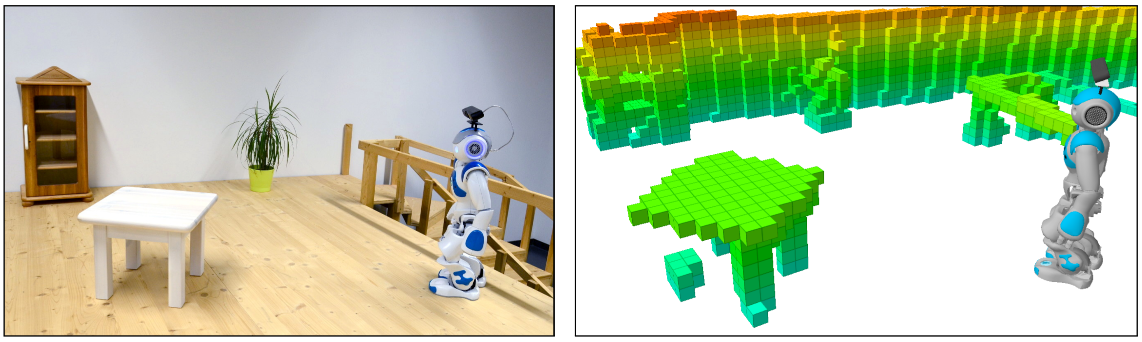


Figure 4.1: Left: Nao humanoid robot with a depth camera on its head and part of the multi-level environment. Right: 3D representation of the scene used for collision avoidance. The map was constructed in real-time by turning on the spot for about 60°, thereby integrating 28 depth images.

humanoids. Furthermore, interpreting stereo or monocular camera data involves highly complex algorithms and does not necessarily result in dense data.

In the previous chapter, we introduced an approach that identifies traversable areas in images from an integrated monocular camera and constructs an occupancy map from the classified images. We demonstrated highly accurate results for the obtained classification and grid maps that enable collision-free navigation. However, the approach relies on certain assumptions such as the availability of sufficiently many features on the floor and visual dissimilarity between traversable floor and obstacles, which cannot always be guaranteed.

To ease these assumptions, in this chapter we introduce a navigation system based on depth camera data, such as from the Microsoft Kinect or Asus Xtion series, which have become available to the consumer market recently. These cameras operate with projected infrared patterns and provide relatively accurate, dense, three-dimensional information directly on the hardware. To the best of our knowledge, the method presented in this chapter describes the first integrated navigation system based on depth camera data, consisting of localization, obstacle mapping, and collision avoidance for humanoid robots.

For enabling a humanoid robot to navigate in complex indoor environments containing multiple levels, our approach relies on a given 3D environment model in form of an octree (Hornung et al., 2013b; Wurm et al., 2010) that contains the static parts of the environment. In this representation, the robot estimates its pose using Monte Carlo localization based on acquired depth camera data. Given the estimated 6D pose of the humanoid and the depth camera data, our system continuously updates a second 3D representation of the environment, containing also non-static obstacles. Both octree-based representations are then used for planning collision-free paths to a target location.

Figure 4.1 shows a motivating example of our system. The left image depicts a Nao humanoid navigating on the top level of a two-story environment. The right image shows a

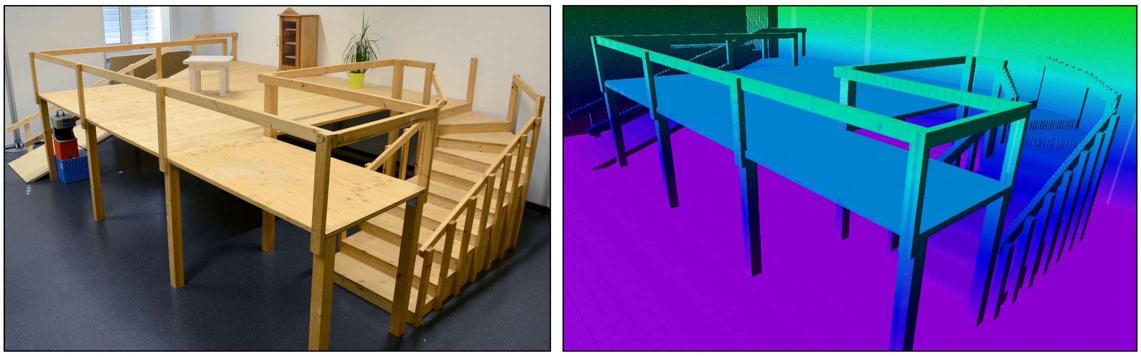


Figure 4.2: Photograph of the environment in which we carried out the experiments and the corresponding map constructed with a CAD software. The map contains only the static parts of the environment.

representation of the robot's pose estimate and the local environment model, both obtained from depth camera measurements. In the environment model, one can clearly identify individual objects such as the table, the cabinet, the plant, or parts of the railing. The map was constructed in real-time from 28 depth images by turning on the spot about 60°.

After presenting the basic techniques and our extensions for depth cameras, we illustrate the performance of our system for a Nao humanoid equipped with an Asus Xtion Pro Live sensor on top of the head. During the experiments, the robot navigated in a 3D environment consisting of multiple levels and containing several static and non-static obstacles. We thoroughly evaluated our approach and show results that demonstrate that our system leads to robust localization and reliable obstacle avoidance in real-time. We conclude that consumer-level depth cameras are well-suited for reliable humanoid robot navigation in complex indoor environments.

In the following, we describe our 3D navigation framework in details. First, we present the underlying map representation and how it is probabilistically updated from sensor readings. Then, we outline the localization technique for estimating the humanoid's pose, and finally, a method to plan collision-free paths given the map representations.

4.1 Volumetric Environment Representation

To enable modeling multi-level environments containing obstacles of various shapes, we use the octree-based mapping framework *OctoMap* (Hornung et al., 2013b; Wurm et al., 2010). It partitions the space into free and occupied voxels where each voxel is associated with an occupancy probability. Unknown space is implicitly modeled by missing information in the tree. As opposed to a fixed size voxel grid map, this tree-based approach allows the map to grow dynamically and is compact in memory as it only allocates memory as needed. Bounded occupancy values enable to appropriately react to changes over time and enable a compression by pruning the tree, particularly in the large free areas.

We use two different 3D maps. First, we consider a static map of the environment for localization and as prior knowledge for path planning. Figure 4.2 depicts an example map. Localization of the robot using this representation is described in Section 4.2. Secondly, we maintain an additional map containing *local* obstacles, which is continuously updated based on the depth data acquired by the robot while walking as described in Section 4.3. This representation is then used for path planning around non-static obstacles following the description in Section 4.4. For this process, we maintain a projected 2D map for efficient collision checks as presented by Hornung et al. (2012b). Each 3D map update of the local obstacle map also updates the 2D projection. To allow the robot to pass below underpasses and traverse the upper level of the environment, only obstacles within the vertical extent of the robot are hereby projected into the 2D obstacle map. Thus, if the robot walks on different floor levels, the projected map is adjusted accordingly from the 3D map. From both maps, we filter out voxels corresponding to the floor to avoid that small errors in the robot's height estimate lead to false collisions with the floor. In the following section, we describe how our approach updates the local map from depth sensor data.

4.2 Monte Carlo Localization via Particle Filter

For localization in the 3D model, we extend the Monte Carlo localization (MCL) framework by Hornung et al. (2010), which was originally developed for data from 2D laser range finders, to depth camera data. Hereby, the humanoid's 6D pose is tracked in the 3D world model using the observations from the depth camera, odometry, and an integrated inertial measurement unit (IMU).

In our approach, the humanoid's torso serves as its reference frame, i.e., we represent by $\mathbf{x}_t$ the pose of the torso at time t and identify it with the humanoid's location, as all other links can be estimated relative to the torso by forward kinematics (see Appendix A.3). A pose $\mathbf{x}_t$ consists of a 3D position $\begin{bmatrix} x_t & y_t & z_t \end{bmatrix}^\top$ and a rotation matrix R. However, for simplicity and readability, in this chapter we employ the corresponding roll, pitch, and yaw angles $(\varphi_t, \theta_t, \psi_t)$ instead of R to express rotations. These Euler angles can directly be obtained from R and represent the rotations about the x-, y-, and z-axis of the fixed map frame, respectively. Hence, we consider the pose representation $\mathbf{x}_t = \begin{bmatrix} x_t & y_t & z_t & \varphi_t & \theta_t & \psi_t \end{bmatrix}^\top$.

In MCL, a posterior probability density function over the state space (i.e., over all possible values for $\mathbf{x}_t$) is estimated by

$$\overbrace{p(\mathbf{x}_t \mid m, \hat{\mathbf{o}}_{1:t}, \mathbf{u}_{1:t})}^{\text{belief at } t} = \overbrace{\eta}^{\text{normalizer}} \overbrace{p(\hat{\mathbf{o}}_t \mid m_t, \mathbf{x}_t)}^{\text{observation model}}$$
$$\int_{\mathbf{x}_{t-1}} \underbrace{p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)}_{\text{motion model}} \underbrace{p(\mathbf{x}_{t-1} \mid m, \hat{\mathbf{o}}_{1:t-1}, \mathbf{u}_{1:t-1})}_{\text{belief at } t-1} \, d\mathbf{x}_{t-1}. \tag{4.1}$$

The density function $p(\mathbf{x}_t \mid m, \hat{\mathbf{o}}_{1:t}, \mathbf{u}_{1:t})$ is called the *belief* about the pose of the robot at time t. It is recursively estimated from the belief about all possible previous poses $\mathbf{x}_{t-1}$ of the robot and an observation model $p(\hat{\mathbf{o}}_t \mid m_t, \mathbf{x}_t)$ and an motion model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$. To evaluate the observation model, the sensor measurements $\hat{\mathbf{o}}_t$ and a map m_t are considered, while the motion model depends on the odometry readings $\mathbf{u}_t$. These models are crucial for the computation of the belief and will be explained later in this section. For tractability, in MCL, the belief at time t is represented by a discrete set of M weighted particles

$$\mathcal{X}_t := \left\{(\mathbf{x}_t^1, w_t^1), \cdots, (\mathbf{x}_t^M, w_t^M)\right\}. \tag{4.2}$$

Each sample $\mathbf{x}_t^i$ represents a hypothesis about the pose of the robot at time t, along with a weight w_t^i that is proportional to the likelihood of that hypothesis. Therefore, only a finite number of previous poses $\mathbf{x}_{t-1}$ has to be considered when updating the belief according to (4.1). One particular algorithm that implements this update is the *particle filter*. It consists of three main steps, which are summarized in the following.

Prediction Starting with an initial distribution $\mathcal{X}_0$ of the particles, the filter propagates each particle according to the odometry estimate $\mathbf{u}_t$ and the motion model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$. For humanoids, an odometry estimate can be computed incrementally from the measured joint angles by applying forward kinematics from the current stance foot to the torso reference frame (see Appendix B.1). We denote by $\hat{\mathbf{x}}_t$ the odometry estimate for the torso at time t, and define $\mathbf{u}_t$ by the motion displacement between $\hat{\mathbf{x}}_{t-1}$ and $\hat{\mathbf{x}}_t$, i.e.

$$\mathbf{u}_t = \hat{\mathbf{x}}_t \ominus \hat{\mathbf{x}}_{t-1}. \tag{4.3}$$

In the motion model, we assume that the robot's motion is only affected by Gaussian noise that correlates with $\mathbf{u}_t$. Thus, we propagate each particle $(\mathbf{x}_{t-1}^i, w_{t-1}^i) \in \mathcal{X}_t$ according to

$$\mathbf{x}_t^i = \mathbf{x}_{t-1}^i \oplus \mathbf{u}_t \oplus \varepsilon_{\mathbf{u}_t}, \tag{4.4}$$
$$\text{where} \quad \varepsilon_{\mathbf{u}_t} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{u}_t}). \tag{4.5}$$

In this equation, $\Sigma_{\mathbf{u}_t}$ is a covariance matrix depending on $\mathbf{u}_t$ which we scale with the magnitude of $\mathbf{u}_t$. Consequently, we add Gaussian noise to the particle after propagating it according to the odometry displacement.

Correction The propagated particles are now distributed according $p(\mathbf{x}_t \mid \hat{\mathbf{o}}_{1:t-1}, \mathbf{u}_{1:t})$. To obtain the belief at time t, the filter adjusts the particle weights according to the observation model $p(\hat{\mathbf{o}}_t \mid m_t, \mathbf{x}_t)$ and the data of the humanoid's sensors $\hat{\mathbf{o}}_t$. The measurements include the depth image from the RGB-D camera, which we represent as a set of rays $\mathbf{r}_t$. Further, we consider the height $\hat{z}_t$ of the humanoid's torso above the ground plane as obtained from the joint encoders using forward kinematics. Finally, we integrate the IMU estimates for the torso's roll and pitch angles $\hat{\varphi}_t$ and $\hat{\theta}_t$.

We assume that all these measurements are independent and combine them into one unified observation model to compute the likelihood of an observation $\hat{\mathbf{o}}_t$:

$$\begin{aligned} p(\hat{\mathbf{o}}_t \mid m_t, \mathbf{x}_t) &= p(\mathbf{r}_t, \hat{z}_t, \hat{\varphi}_t, \hat{\theta}_t \mid m_t, \mathbf{x}_t) \\ &= p(\mathbf{r}_t \mid m_t, \mathbf{x}_t) \cdot p(\hat{z}_t \mid m, \mathbf{x}_t) \cdot p(\hat{\varphi}_t \mid \mathbf{x}_t) \cdot p(\hat{\theta}_t \mid \mathbf{x}_t). \end{aligned} \tag{4.6}$$

Here, we omitted the map m from the likelihood terms for the pitch and roll angles, as the likelihoods do not dependent on the map. The likelihoods for the torso height $\hat{z}_t$, the roll angle $\hat{\varphi}_t$ and pitch angle $\hat{\theta}_t$ are modeled by a Gaussian distribution

$$\phi(d, \sigma) = \lambda \left(\sigma \sqrt{2\pi}\right)^{-1} \exp\left(-\frac{d^2}{2\sigma^2}\right), \tag{4.7}$$

over the difference d between measured values and the predicted ones for each particle $(\mathbf{x}_t^i, w_t^i)$. Here, λ is a normalizer that guarantees that the integral over $\phi(d, \sigma)$ evaluates to one, even if the range of admissible values is limited to a certain range, e.g. $-\pi$ to π for angles. With this definition, we can write the following likelihoods as

$$p(\hat{z}_t \mid m, \mathbf{x}_t) = \phi(z_{t,\text{ground}} - \hat{z}_t, \sigma_z) \tag{4.8}$$

$$p(\hat{\varphi}_t \mid \mathbf{x}_t) = \phi(\varphi_t - \hat{\varphi}_t, \sigma_\varphi), \tag{4.9}$$

$$p(\hat{\theta}_t \mid \mathbf{x}_t) = \phi(\theta_t - \hat{\theta}_t, \sigma_\theta), \tag{4.10}$$

where $z_{t,\text{ground}}$ is the height difference between z_t and the nearest floor level in the map. Further, $\sigma_z, \sigma_\varphi, \sigma_\theta$ are variances depending on the individual sensors and their noise characteristics. For integrating the depth camera observations, we consider a sparse set of rays $\mathbf{r}_t$ extracted from the depth image and evaluate the range sensing likelihood $p(\mathbf{r}_t \mid \mathbf{x}_t)$.

Our system classifies the rays into two groups: rays hitting the ground plane and rays hitting vertical objects. Rays which did not hit any object within a maximum distance are ignored. For identifying the rays hitting the ground plane we consider their end points. We first pre-filter them based on the height (z-coordinate) in the map frame to obtain consistency with the expected height of the floor. Then the approach identifies the inliers of the dominant plane obtained via RANSAC (Fischler and Bolles, 1981) with the constraint that the plane's normal vector points upwards (in z-direction) in the map frame. The remaining rays are considered to hit vertical objects. Our system uniformly samples half the rays from ground parts and the other half from vertical objects. Thereby, we compensate for the fact that the camera faces primarily the floor area for better obstacle avoidance. Rays hitting the floor, however, can provide no information about the translation in the horizontal plane, i.e., x, y, and ψ, which is typically more important than to estimate height or pitch and roll. The left image in Figure 4.3 illustrates this process. Here, eleven rays are sampled from the depth image (shown as a point cloud representation, i.e. the end points of all rays). Five of which hit the ground and six walls or poles. The figure used few rays for illustration only. In practice, we sample approximately 100 rays.

For evaluating the likelihood, we follow the suggestions by Thrun et al. (2005, Chapter 6). We assume that the sampled rays $r_{t,k}$ are conditionally independent. Thus, we multiply the likelihoods of the individual rays $p(r_{t,k} \mid \mathbf{x}_t)$ to obtain the joint likelihood

$$p(\hat{\mathbf{r}}_t \mid m, \mathbf{x}_t) = \prod_k p(r_{t,k} \mid m, \mathbf{x}_t). \tag{4.11}$$

The $p(r_{t,k} \mid \mathbf{x}_t)$ are determined via ray casting in the volumetric 3D environment representation. Thus, we extract for each ray $r_{t,k}$ the expected distance $d(r_{t,k})$ to the closest obstacle in the map along the ray direction, given the camera pose, which is computed via forward kinematics from the robot pose $\mathbf{x}_t$. This distance is then compared to the length of the ray, $d(\hat{r}_{t,k})$. The likelihood for the measurement is finally modeled as a mixture of a Gaussian distribution and a uniform distribution that represents random measurements,

$$p(r_{t,k} \mid m, \mathbf{x}_t) = w\phi\left(d(r_{t,k}) - d(\hat{r}_{t,k}), \sigma_r\right) + (1 - w)\frac{1}{z_{\max}}, \tag{4.12}$$

for $d(\hat{r}_{t,k}) \in [0, z_{\max}]$, and 0 otherwise. Here, $w \in [0, 1]$ is a weighting factor, $z_{\max}$ is the maximum range of the depth sensor, and σ_r is scaled proportionally with $d(\hat{r}_{t,k})^2$ to represent the noise characteristics of the sensor (Khoshelham and Oude Elberink, 2012).

From this, (4.6) can be evaluated and we set the importance weights w_t^i for each particle accordingly. The right image of Figure 4.3 shows a small set of particles, indicated by arrows. The particle which best matches the measurements, is assigned the highest weight and is highlighted in saturated colors in the figure. This particle is commonly called the *best particle*.

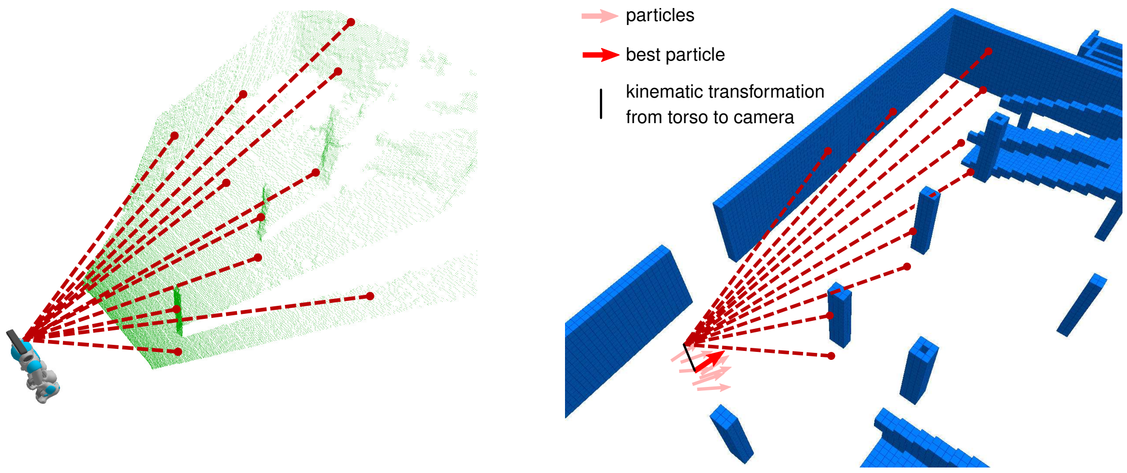


Figure 4.3: Likelihood computation for depth camera measurements. Left: Rays (red dashed) sampled from the depth image and its point cloud representation (green dots). Right: Particle cloud (red arrows) with a map rendering (blue) along with rays sampled for the particle with the highest likelihood (dark red arrow). The likelihood is evaluated via raycasting for each particle.

Resampling In a final step, the particle filter draws a new set $\mathcal{X}_{t+1}$ of M particles from the current set $\mathcal{X}_t$, proportionally to the importance weights $\{w_t^i\}$ and with replacement. Each new particle is assigned the same weight $w_{t+1}^i = 1/M$. This step is crucial, as it ensures that the pose estimate converges towards a hypothesis with a high likelihood. For obtaining the robot pose at time t, we consider all M particles $(\mathbf{x}_t^i, w_t^i) \in \mathcal{X}_t$ and compute the weighted average over their poses $\mathbf{x}_t^i$ according to their weights w_t^i.

4.3 Probabilistic 3D Map Update

We maintain a 3D octree-based map to integrate multiple depth camera observations probabilistically. Therefore, we assume that the robot's pose is known from the localization described in Section 4.2. In this map, the space is decomposed into a discrete set of voxels, where each voxel's probability of being occupied is estimated via a binary Bayes filter. The probability $p(n \mid \hat{\mathbf{o}}_{1:t})$ that voxel n is occupied is recursively computed given all sensor measurements $\hat{\mathbf{o}}_{1:t}$ according to the standard occupancy grid mapping formula (Moravec, 1988; Wurm et al., 2010)

$$p(n \mid \hat{\mathbf{o}}_{1:t}) = 1 - \left(1 + \frac{p(n \mid \hat{\mathbf{o}}_t)}{1 - p(n \mid \hat{\mathbf{o}}_t)} \underbrace{\frac{p(n \mid \hat{\mathbf{o}}_{1:t-1})}{1 - p(n \mid \hat{\mathbf{o}}_{1:t-1})}}_{\text{recursive term}} \frac{1 - p(n)}{p(n)} \right)^{-1}, \tag{4.13}$$

where $\hat{\mathbf{o}}_t$ is the measurement from the depth camera at time t, $p(n)$ is the prior probability for n being occupied, and $p(n \mid \hat{\mathbf{o}}_{1:t-1})$ is the previous estimate. The term $p(n \mid \hat{\mathbf{o}}_t)$ denotes the likelihood of voxel n being occupied given the measurement $\hat{\mathbf{o}}_t$. For efficiency, we use the *log odds* formulation to update the map. Thus, by assuming $p(n) = 0.5$, i.e., an equal probability of n being free or occupied, we can express (4.13) as

$$\begin{aligned} l(n \mid \hat{\mathbf{o}}_{1:t}) &= l(n \mid \hat{\mathbf{o}}_{1:t-1}) + l(n \mid \hat{\mathbf{o}}_t), \\ \text{where} \quad l(x) &:= \log \frac{p(x)}{1 - p(x)} \end{aligned} \tag{4.14}$$

is the log odds of $p(x)$. The likelihoods $p(n \mid \hat{\mathbf{o}}_t)$ are estimated by a beam-based inverse sensor model that assumes that each measurement is generated by observing the surface of obstacles and that the line of sight between the sensor origin and the endpoints does not contain any other obstacle. Thus, we update the last voxel on the beam as occupied, and all the others up to the last one as free by setting the corresponding log odds $l(n \mid \hat{\mathbf{o}}_t)$ for $p(n \mid \hat{\mathbf{o}}_t)$, i.e.,

$$l(n \mid \hat{\mathbf{o}}_t) = \begin{cases} l_{\text{occ}}, & \text{if the endpoint lies within } n \\ l_{\text{free}}, & \text{if } n \text{ is traversed by the beam} \end{cases}, \tag{4.15}$$

where l_{occ} and l_{free} are experimentally chosen according to the sensor characteristics and $l_{\text{occ}} > l_{\text{free}}$. Ray-casting determines which voxels lie on the beam. We only update a voxel as occupied, if the end point does not coincide with the ground plane as we do not want to represent the ground as occupied space. However, our system considers obstacles *on* the ground as obstacles appropriately. To identify the ground plane, we use the same technique as described in Section 4.2.

4.4 Path Planning and Collision Avoidance

For planning collision-free paths, we consider the static map of the environment as prior knowledge, as well as the locally constructed map based on depth camera data, which contains also non-static obstacles. By using the static map as well, we can avoid an exploration step that would force the robot to visit unobserved areas to decide whether they are occupied or not. Instead, we simply use the information from the static map, where no local information is available yet.

For the sake of real-time performance and robustness, our approach relies on a projection of the 3D maps to the floor level. Here, each cell of the aligned 2D map is assigned the maximum occupancy probability value of the corresponding voxels in the 3D map.

To be able to traverse underpasses, we restrict the considered area in the 3D map to the vertical extent of the robot. This is the area where collisions are potentially hazardous for the robot. Everything below and above can be safely ignored. Note that this is not the same as maintaining a simple 2D map. When the robot's z-coordinate changes, the projection is updated accordingly. This is only possible because we maintain the underlying 3D structure, hence enabling navigation in multi-level scenarios.

For collision checks in the projected map, we assume a circular robot model. This assumption prevents the robot from traversing very narrow passages but allows performing collision checks in constant time by employing a distance transform of the 2D obstacle map, which can be generated in real-time. With the distance transform, a collision check is reduced to a simple lookup operation. In Chapter 5, we lift the circular robot assumption and enable navigation in narrow passages.

To compute a collision-free path to the goal location, our system uses the A* algorithm (Hart et al., 1968). In case of a map update, it checks whether the previous plan is still valid and replans the path only if necessary. This way, we avoid oscillation while still being able to react to changes in the environment.

4.5 Experiments

We carried out a series of experiments demonstrating the capabilities of our navigation system. Therefore, we used the Nao humanoid, as is described in Appendix B, with an attached Asus Xtion Pro Live RGB-D camera. The camera is mounted on the robot's head in a way such that its optical axis faces the floor in a 30° angle while walking. We found this to be the best compromise between observing the near range for obstacle detection and looking ahead for localization and path planning. The increased weight due to the mounted camera destabilizes the walking behavior of the robot. We therefore added thin plastic sheets to the robot's feet to increase the friction.

To allow for real-time performance, we set the camera's resolution to 320×240 and update the map from sensor data at approximately 6 Hz. All processing is done on a 2.5 GHz Intel quad core CPU. We conducted the experiments in a multi-level environment, scaled-down to match the size of a Nao humanoid (see Figure 4.2). We sketched the structure in a 3D CAD software and converted it to an *OctoMap*. This model is used for localization. Note that the 3D model does not perfectly match the actual scene due to imperfection in constructing the environment and, furthermore, the scene will contain non-static obstacles not included in the 3D CAD model. Therefore, our approach constructs a local map from depth camera data during navigation in real time. A video demonstrating our approach can be found at `http://youtube.com/srcx7lPoIfw`.

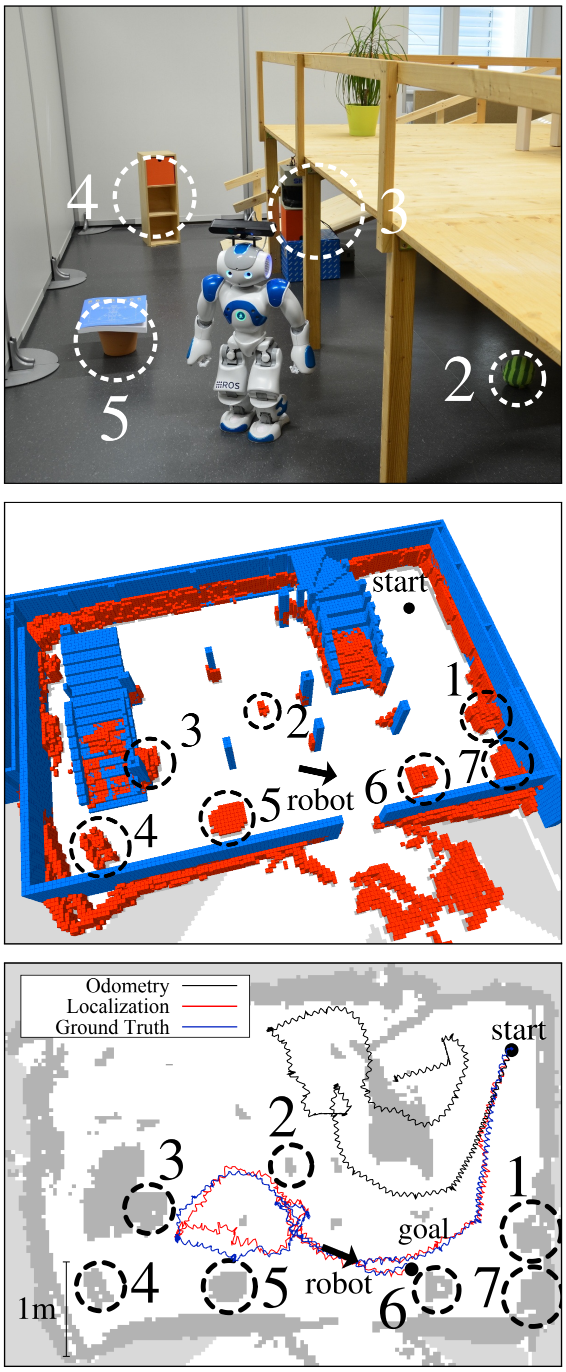


Figure 4.4: Top: Nao navigating in the lower level of our environment between obstacles (Scenario 2). Middle: Static (blue) and local (red) 3D map constructed by the robot while walking. Bottom: Two-dimensional projection of the local map used for collision avoidance and path planning. The lines show the robot's odometry estimate, the pose estimate from our localization system, and the ground truth. The arrow indicates the robot's pose in the top image. Numbered circles indicate obstacles that are not part of the static map.

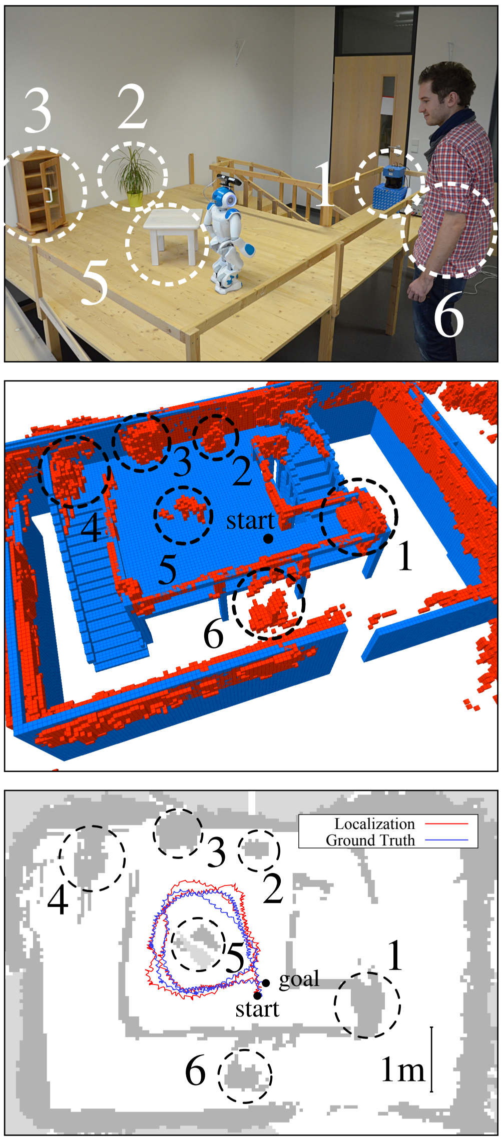


Figure 4.5: Top: Nao navigating in the topper level of our environment between obstacles (Scenario 3). Middle: Local three-dimensional map (red) constructed while the robot was walking on the top level of the environment, along with the static map (blue). Bottom: Two-dimensional projection of this map used for collision avoidance and path planning. As can be seen, also obstacles not contained in the original static representation are represented accordingly (numbered circles). The figure further shows the robot's trajectory estimated by our algorithm (red) and the ground truth (blue). The odometry plot has been left out for the sake of clarity.

Table 4.1: Aggregated localization error for three scenarios

	mean error [m]	std. dev. [m]
Scenario 1	0.07	0.04
Scenario 2	0.09	0.07
Scenario 3	0.07	0.05

4.5.1 Localization Accuracy

First, we performed a series of experiments to evaluate our localization system. We compared the resulting pose estimate to the ground truth in the 2D plane, which we obtained by tracking the humanoid with two stationary SICK laser range finders (Luber et al., 2011). Consequently, we evaluated the translational error in the horizontal plane.

We conducted experiments in three different scenarios. In Scenario 1, the robot navigated on the lower level of our environment. Except for the two laser range finders used to record the ground truth and their power supplies, the static map closely resembled the actual scenario as can be seen in Figure 4.2. Scenario 2 was similar to Scenario 1 but we additionally placed several obstacles such as books, balls, baskets, and shelves in the scene, as is shown in the top image of Figure 4.4. These obstacles are not part of the map used for localization and therefore are expected to decrease the performance. The robot followed a similar path in Scenario 1 and 2 (see bottom image in Figure 4.4). In Scenario 3, the robot walked two circles on the top level where we placed additional unmapped obstacles such as a table, a plant, and a cabinet (see Figure 4.5).

For all scenarios, we manually initialized the localization system. We used 500 particles for tracking the robot's pose and sampled 100 rays from the depth camera measurement for computing the likelihood as described in Section 4.2. The lower images in Figures 4.4 and 4.5 depict the trajectories of the pose estimates and the ground truth on top of the projected obstacle map. As one can see, the estimated pose closely resembles the ground truth. Figure 4.4 also shows the robot's odometry as reference which is clearly useless for reliably executing navigation tasks.

To evaluate the localization results quantitatively, we computed the mean error as well as the standard deviation over the robot's trajectory for each of the three scenarios. Table 4.1 summarizes the results. As can be seen, our system leads to robust and accurate pose estimation. As expected, the accuracy decreases slightly in Scenario 2 compared to Scenario 1, due to the additional obstacles. However, for all scenarios, the average error is still smaller than 10 cm.

Additionally, as Monte Carlo localization is a probabilistic technique, we repeated the accuracy evaluation five times over the same data set. Here, we used the same raw sensor

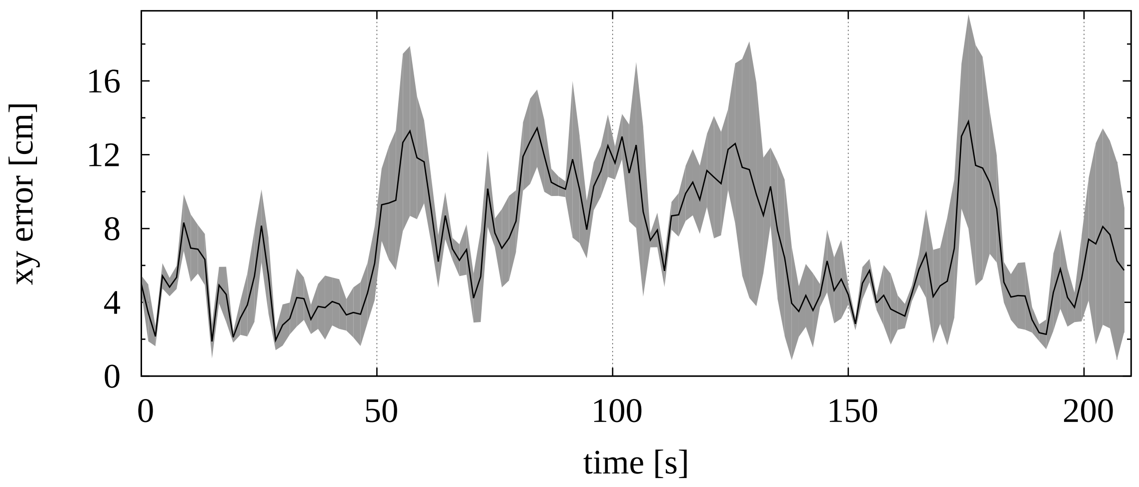


Figure 4.6: Localization error as mean and standard deviation for five runs in scenario 1.

data and initialization as for Scenario 1 and recorded the pose estimation errors relative to the ground truth. We then computed the mean error and standard deviation over the trajectory parameterized by time. In Figure 4.6 we present the obtained results. The error is generally small, however, for the time between 50 s and 130 s and 170 s to 180 s, the average error increased from approximately 6 cm to 10 cm, with a peak value of circa 20 cm. Here, the robot navigated in the hallway part parallel to the walls and the camera observed little structure in the environment that could help to reduce the translational uncertainty. A larger field of view or active sensing could improve the performance here.

4.5.2 Mapping

To demonstrate the mapping and obstacle detection capabilities of our system, we consider Scenarios 2 and 3 described in the previous section. Figure 4.4 and Figure 4.5 show the 3D map constructed from the depth camera data while walking (red), as well as the static map of the environment (blue). It is clearly visible that the constructed map closely follows the structure of the reference map. Furthermore, it also includes all the obstacles that are not part of the static map. The figures also show the 2D projection of the constructed 3D map used for collision avoidance. In both maps, the structure of the non-traversable area is clearly identifiable.

4.5.3 Path Planning and Obstacle Avoidance

In the last experiment, we evaluate the ability to react to changes in the environment and plan collision-free paths also with non-static obstacles. Figure 4.7 shows a scenario in which the robot reacted to a dynamic obstacle during walking. The left column shows the state of the projected obstacle map along with the robot's pose and the planned path. The right column depicts the current RGB camera image with an overlay of the state of the constructed 3D map. Initially, the robot planned a straight path to the goal location through the empty space (first row). While walking, a human entered the scene blocking the robot's initial path (second row). The robot immediately updated its obstacle map and planned a collision-free path to the goal. The robot followed that path accurately (third row) until it reached the goal.

4.6 Related Work

The work most closely related to our approach has been recently presented by Biswas and Veloso (2012). The authors developed an approach for indoor robot navigation based on depth camera data. They proposed to sample points from the depth data belonging to vertical planes. These points are down-projected to 2D and used to update the particle filter that estimates the robot's pose. The observation model hereby matches the projected points to a given map of walls. The projected points are further used for obstacle detection. One disadvantage of this approach is that it discards the 3D information of the sensor data. Therefore, robots based on these techniques cannot navigate in multi-level environments. Similar limitations hold for other approaches. For instance, Stachniss et al. (2008) presented a system for simultaneous localization and mapping (SLAM) to learn accurate 2D grid maps of large environments with a humanoid equipped with a laser scanner located in the neck. Such a map was subsequently used by Faber et al. (2009) for humanoid localization and path planning in 2D. During navigation, the robot avoids obstacles sensed with the laser scanner and ultrasound sensors located at the hip. Obstacles with a lower height are not detected which potentially leads to collisions. Also, Tellez et al. (2008) use laser data to construct a 2D occupancy grid map in which paths for a humanoid are planned. The authors use data from two laser scanners mounted on the robot's feet. Common to all these approaches is that they insufficiently represent the environment for navigation tasks in indoor scenarios with complex 3D structures.

Other authors proposed to use data structures more suitable for these tasks. For instance, Nakhaei and Lamiraux (2008) presented a technique to 3D environment modeling from stereo data for humanoid motion planning. Similar to our approach, the authors proposed to use a probabilistic voxel grid. However, their system has no localization component, which leads to inconsistencies in the learned map. Ozawa et al. (2005) developed a system that relies on stereo image sequences to construct a dense local feature map. This system performs real-time mapping with a humanoid robot based on 3D visual

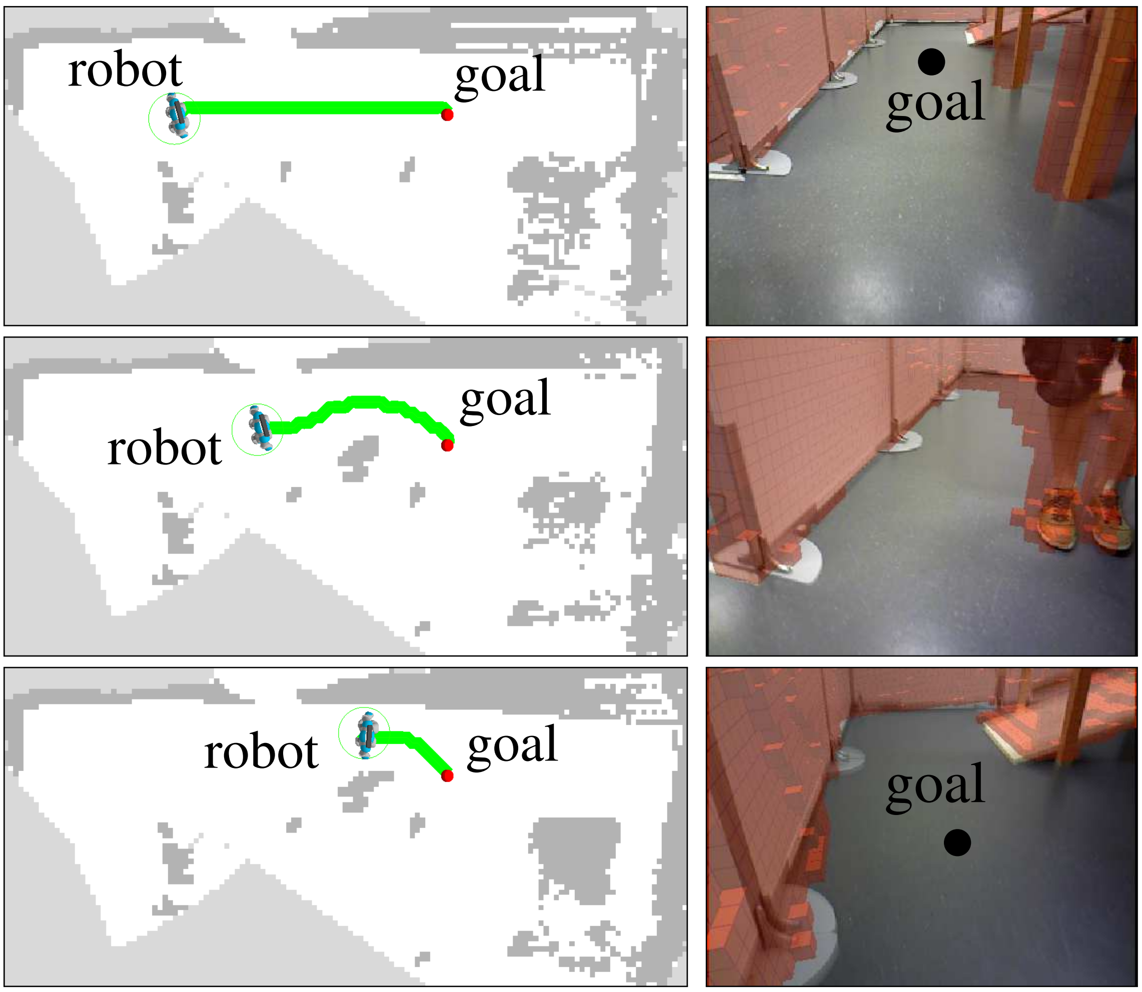


Figure 4.7: The robot avoids a dynamic obstacle. The first row shows the robot's initial path to the goal with the corresponding camera image. Then, a human blocks the robot's path, forcing the robot to detour (second row). The robot follows the updated path to the goal (last row). The camera images show an overlay of the current obstacle map (red).

odometry for short trajectories. Similarly, Gutmann et al. (2008) build a 2.5D heightmap given accurate stereo data and additionally update a 3D occupancy grid map to plan paths for a humanoid. Chestnutt et al. (2009) also suggested to use heightmaps for navigation. They obtain 3D laser data acquired with a constantly sweeping scanner mounted on a pan-tilt unit at the humanoid's hip. The authors fit planes through 3D point clouds and construct a heightmap of the environment. Afterwards, they distinguish between accessible areas and obstacles based on the height difference. Such a sensor setup can only be used on robots with a significantly larger payload than the Nao humanoid.

Hornung et al. (2010) presented a 3D localization method for humanoid robots, also based on 2D laser data. Similar to our method, they applied a particle filter to estimate the 6D pose of the robot in a given 3D volumetric map of the environment. Our work can be seen as an extension as our approach does not require an expensive laser range finder but uses comparably cheap depth cameras. Furthermore, our system additionally contains 3D obstacle mapping and path planning capabilities. To localize a wheeled robot, Cherubini and Chaumette (2013) proposed to use a view sequence of images. By specifying a sequence of images, the robot is able to follow the trajectory described by the images. The approach uses a laser to detect and avoid obstacles on the way while controlling the camera to guarantee visibility of the features required for localization. While this approach leads to close path following behavior even over large distances, in our work, we do not restrict the motion of the robot to a taught trajectory but allow more deliberate motions. Also targeted at wheeled robots, Kümmerle et al. (2008) developed a laser-based localization system operating on so-called multi-level surface (MLS) maps. These maps store multiple levels of the scene per 2D grid cell and compactly represent 3D environments. However, for accurate localization in complex scenarios, a volumetric representation is required that explicitly represents free space and models arbitrary structures. MLS maps cannot provide this. Other approaches rely on external sensing, e.g., the work by Baudouin et al. (2011). The authors propose a framework for footstep planning and collision avoidance in 3D environments. While the approach works in real-time and allows the robot to step over low obstacles, it relies on very accurate off-board sensing and applies a sampling technique for path planning that can result in complicated paths.

Pretto et al. (2009) estimate the 6D pose of a humanoid as well as the 3D position of features in monocular camera data. The authors designed feature detectors specifically to be able to deal with the effect of motion blur that typically occurs during humanoid walking. However, the features are sparse and thereby unsuitable for reliable obstacle avoidance. Einhorn et al. (2009) proposed to track features in consecutive images and recover the features' positions from the ego-motion of the camera. This requires an accurate estimate of the camera pose, which the authors obtain from the odometry of a wheeled robot. The system relies on the sparse features for obstacle detection. The system presented by Cupec et al. (2005) assumes that obstacles occur at contrast edges in the camera image of a humanoid robot. From the detected edges, the author's approach performs a scene analysis that expects rectangular objects and thereby estimates their poses. The inferred objects are then used to plan actions for the humanoid including stepping over, walking around, or climbing the object. However, this approach requires a controlled environment with restricted lighting and obstacle shapes.

Recently, multiple approaches have been presented that perform SLAM with RGB-D cameras (Endres et al., 2012; Huang et al., 2011; Newcombe et al., 2011b). These approaches are typically optimized for small workspaces such as desktops or small rooms but not for larger environments. Further, they require that the camera can see enough texture or structure to match the observations. Consequently, they are not appropriate for scenarios like ours, where the camera faces the lowly-textured floor most of the time, in order to sense obstacles in the robot's way. For the same reason, approaches targeted at larger environments (Whelan et al., 2012a) are also not directly applicable.

4.7 Conclusions

In this chapter, we demonstrated that affordable, consumer-level depth cameras are well-suited sensors for robot navigation tasks in complex indoor environments. We presented a real-time navigation system that allows the estimation of a humanoid's 6D pose and construction of an environment representation as a 3D map while walking. For efficiency, our approach projects the map within the robot's vertical extent to 2D and approximates the humanoid's shape by a disc. We described how collision-free paths through environments containing static and non-static obstacles can be generated from this representation.

In experiments with a Nao humanoid, equipped with an Asus Xtion Pro Live RGB-D camera, we thoroughly evaluated the performance of our system. As the results show, our approach leads to accurate pose estimates and reliable, collision-free navigation through the environment by using the acquired 3D map. Of course, depth cameras also have some drawbacks. In the near range of the camera (closer than 50 cm), no depth data is available. In this case, we could fall back to applying collision detection approaches based on monocular vision data using the techniques introduced in Chapter 3. Depth data could be used to train the visual classifiers when available and label parts of the image where depth data is missing. However, in practice, we rarely observed this situation, as by using the presented approach, the robot avoids getting close to obstacles.

In the next chapter, we extend the presented approach. We will lift the circular robot model assumption made in this chapter to allow the robot to better pass narrow passages and even step over or onto obstacles.

Chapter 5

3D Footstep Planning Among Clutter

In this chapter, we enhance the depth camera-based navigation approach introduced in the previous chapter. We describe an integrated navigation system that allows humanoid robots to autonomously navigate in unknown, cluttered environments. Our system employs data of an on-board depth camera to estimate the robot's pose and compensate for drift of the odometry. The system does not rely on a given map for pose estimation. For navigating in challenging environments, we lift the circular robot shape assumption and instead iteratively compute sequences of safe actions including footsteps and whole-body motions, leading the robot to target locations. Therefore, the system constructs a high resolution heightmap representation of the environment. The extended planner chooses from a set of actions that consists of planar footsteps, step-over motions, as well as parameterized step-onto and step-down actions. We developed a new approach for fast collision checking during planning. As we demonstrate in experiments with a Nao humanoid, our system leads to accurate navigation in cluttered environments and the robot is able to traverse highly challenging passages.

The human-like design and locomotion allows humanoid robots to step over or onto obstacles, to reach destinations only accessible by stairs or narrow passages, and to navigate through cluttered environments without colliding with objects. These abilities would make humanoid robots ideal assistants to humans, for instance in housekeeping or disaster management. However, there is a number of reasons, why up to today, we do not see such robots in practical applications.

First, many researchers developed or apply navigation algorithms that represent a humanoid's shape as a disc, for instance the works by Gutmann et al. (2005); Li et al.

(2009); Stilman et al. (2006). This model does not respect all the navigation capabilities of humanoid robots and therefore more appropriate approaches are necessary for navigation in cluttered, or very narrow scenarios. Second, while some researchers focus on planning locomotion for humanoid robots, they often neglect sensing. Instead, they assume a known model of the world (Hauser et al., 2005; Hornung et al., 2012a), or they use external sensing systems (Chestnutt et al., 2005; Perrin et al., 2012a; Stilman et al., 2006). Onboard sensing is, however, essential for autonomous navigation in unknown or only partially known environments. Finally, a seamless combination of individual system components including environment modeling, pose estimation, and gait generation is required for humanoids to carry out complex tasks. For all these individual aspects, promising approaches have been presented. Yet, an integrated system that combines the best solutions for all the subtasks, has not been demonstrated.

In the previous chapter, we presented a robust navigation framework that includes many state-of-the-art components. It utilizes a consumer-level depth camera for pose estimation in a given 3D model of the environment and for mapping of unknown objects. For planning collision-free paths, the approach projects the constructed 3D map onto the ground plane and checks for collisions with a circular approximation of the robot's shape. While this leads to very efficient collision checking, actions such as stepping over or onto objects cannot be considered. Recently, Nishiwaki et al. (2012) presented an impressive system that combines environment mapping, footstep planning, and gait control. However, for localization the system relies solely on odometry and collision checking is performed only on foot level, i.e., neglecting the body of the humanoid.

In this chapter, we therefore present an integrated navigation framework that combines pose estimation, mapping, and motion planning for autonomous navigation in unknown 3D environments. Our system relies only on the robot's onboard sensors, i.e., its joint encoders, an inertial measurement unit (IMU), and a head-mounted depth camera. The environment is represented as an accurate heightmap that is constructed by integrating multiple measurements over time while the humanoid navigates. A prior map of the environment is not needed. Our approach performs efficient whole-body collision checking and applies traversability analysis to determine safe footprints. To navigate collision-free in challenging scenes containing obstacles on the ground and narrow passages, our anytime planner computes a sequence of actions that consists of planar footsteps, step-over motions, as well as parameterized step-onto and step-down actions.

As we demonstrate in practical experiments with a Nao humanoid, our system leads to robust navigation in cluttered scenes containing objects of various shapes and sizes, including stairs. The left image of Figure 5.1 shows our Nao stepping over a slat and the right image shows the corresponding heightmap constructed by integrating multiple depth camera measurements while navigating. One can see that the map closely resembles the

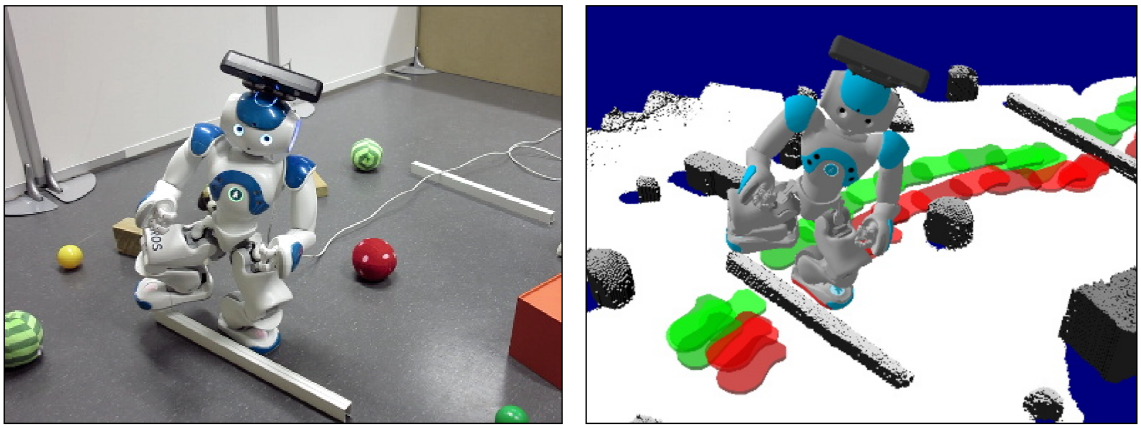


Figure 5.1: Left: A Nao humanoid autonomously traversing a cluttered scene. Right: The corresponding heightmap representation of the environment that the robot generates during navigation based on data from its head-mounted depth camera. The heightmap is used for planning safe footsteps.

scenario on the left and our system allows the robot to autonomously traverse the cluttered scene by planning a sequence of safe actions.

5.1 Pose Estimation

Many approaches to biped navigation rely only on odometric information to obtain the pose of the robot (Chestnutt et al., 2009; Gutmann et al., 2005; Nishiwaki et al., 2012). However, odometry is prone to drift over time and abrupt errors due to slipping of the robot's feet. To increase the accuracy of the pose estimation, we therefore combine odometry information with depth camera measurements to reduce such effects.

Our approach computes a corrected pose estimate $\mathbf{x}_t$ for the robot that aligns the current depth sensor observations at time t with the previous ones at time t-1. As in the previous chapter, $\mathbf{x}_t$ is the 3D pose $\begin{bmatrix} x_t & y_t & z_t & \varphi_t & \theta_t & \psi_t \end{bmatrix}^\top$ of the robot's torso at time t. To obtain $\mathbf{x}_t$ we seek to minimize the distance between corresponding points in the point clouds obtained from the depth camera readings. Formally, we estimate

$$\mathbf{x}_t = \operatorname*{arg\,min}_{\mathbf{x}'} \sum_i \left\| \left(\mathcal{T}(\mathbf{x}')\,\mathcal{F}_C^T(\boldsymbol{\theta},\mathbf{q}_t)\,\tilde{\mathbf{p}}_{t,i}\right) - \left(\mathcal{T}(\mathbf{x}_{t-1})\mathcal{F}_C^T(\boldsymbol{\theta},\mathbf{q}_{t-1})\,\tilde{\mathbf{p}}_{t-1,i}\right)\right\|^2, \tag{5.1}$$

where $\tilde{\mathbf{p}}_{t,i} \leftrightarrow \tilde{\mathbf{p}}_{t-1,i}$ are corresponding points in two consecutive point clouds computed from the depth camera at time t and $t-1$, respectively. All points are expressed in the camera frame, therefore they are transformed to the torso frame by means of forward kinematics (see Appendix A.3) as is expressed by $\mathcal{F}_C^T(\boldsymbol{\theta}, \mathbf{q}_t)$, where $\mathbf{q}_t$ are the joint positions at time t and $\boldsymbol{\theta}$ the robot's calibration parameters. The function $\mathcal{T}(\mathbf{x})$ yields the homogeneous transformation matrix corresponding to the pose $\mathbf{x}$.

Since the correspondences are not known in real sensor data, we estimate $\mathbf{x}_t$ by using a variant of the Generalized-ICP (GICP) algorithm proposed by Segal et al. (2009). GICP iterates between establishing correspondences between the two point clouds and fitting a motion transformation from the established correspondences, similar to the approach of homography estimation presented in Section 3.3. To bootstrap GICP, we initialize $\mathbf{x}_t$ from previous estimate $\mathbf{x}_{t-1}$ propagated by the odometry estimate, i.e., $\mathbf{x}_t = \mathbf{x}_{t-1} \oplus (\hat{\mathbf{x}}_t \ominus \hat{\mathbf{x}}_{t-1})$, where $\hat{\mathbf{x}}_t$ is the pose estimate at time t according the odometry (see Appendix B.1). Since the ground plane typically dominates the scene and thus also the alignment process, we filter out points belonging to the ground plane prior to applying GICP using the technique described in Section 4.2. Consequently, our approach corrects for the drift of the odometry and thereby enables the robot to navigate in areas where no prior map for accurate localization is available.

5.2 Environment Representation

For planning accurate footsteps through clutter with a small-sized robot such as the Nao humanoid (see Appendix B), we require a fine-detailed, high-resolution representation of the environment. In Chapter 4 we proposed to use an octree-based data structure to represent the environment. While this allows the construction of very memory-efficient maps of larger environments, very high-resolution 3D maps still occupy a large amount of memory. Further, the structure requires computational overhead when updating the map or accessing individual cells as the tree has to be traversed to find the corresponding leave nodes. However, searching for neighboring cells is a very frequent operation in collision-checking for footstep planning.

To this end, our system represents the environment as a high-resolution heightmap that is learned from depth camera data. Each cell c of the map stores a height value h_c (i.e., the z-coordinate in the fixed odometry frame) and a variance σ_c^2. The variance represents the uncertainty about the height of each cell resulting from small pose estimation errors and sensor noise. Hence, we interpret $\mathcal{N}(h_c(t), \sigma_c^2(t))$ as the belief about the height of c at time t. To update the map, the points from the current point cloud are binned into the cells of the heightmap. Let z_c be the maximum over the z-coordinates of all observed points falling into a cell c. Our approach then updates the belief about the height of c from

$\mathcal{N}(h_c(t), \sigma_c^2(t))$ and the observation z_c using a Kalman filter, assuming a state model with no underlying dynamics (Pfaff et al., 2007). Hence, we assign

$$h_c(t+1) = \frac{1}{\sigma_c^2(t) + \sigma_z^2} \left(\sigma_z^2 \, h_c(t) + \sigma_c^2(t) \, z_c \right), \tag{5.2}$$

$$\sigma_c^2(t+1) = \frac{1}{\sigma_c^2(t) + \sigma_z^2} \, \sigma_z^2 \, \sigma_c^2(t), \tag{5.3}$$

where σ_z^2 represents the uncertainty of the observation. As we employ a depth camera as sensor, σ_z^2 is best modeled proportional to the quadratic distance from the sensor to the observed point, as proposed by Khoshelham and Oude Elberink (2012).

5.3 Footstep Planning for 3D Environments

5.3.1 State Representation and Transition

For planning safe motions based on footsteps, we consider a discretized four-dimensional state space represented by a sparse graph. The graph consists of a set of nodes corresponding to the discrete states and a set of edges representing transitions between states. A state s is a tuple $\left(\bar{x}, \bar{y}, \bar{\psi}, f\right)$, where $\begin{bmatrix}\bar{x} & \bar{y} & \bar{\psi}\end{bmatrix}^\intercal$ describes the 2D pose of a foot with $\begin{bmatrix}\bar{x} & \bar{y}\end{bmatrix}^\intercal$ being a location and $\bar{\psi}$ an orientation. Further, $f \in \{\text{left}, \text{right}\}$ is a binary variable indicating whether the left or right foot acts as stance foot. The height z_s of a state s is determined uniquely from the underlying heightmap as the average over the height values covered by the robot's footprint placed at $\begin{bmatrix}\bar{x} & \bar{y} & \bar{\psi}\end{bmatrix}^\intercal$, and hence, it is not explicitly represented as part of the state space.

For planning motions, we consider a set of discrete actions $\mathcal{A}$. For an action $a \in \mathcal{A}$, $a(s)$ describes the transition $s \xrightarrow{a} s'$ from a state s to its successor s'. An action a is consequently parameterized by a tuple $\left(\Delta\bar{x}, \Delta\bar{y}, \Delta\bar{\psi}, f\right)$, where f indicates the stance foot for the action and the remaining parameters the displacement of the swing foot relative to the stance foot f. Furthermore, each action is parameterized over an interval $[\Delta z_{\min}, \Delta z_{\max}]$ that describes the admissible height differences from a state to its successor when executing this action. This is of importance when climbing onto or from objects. For planar footsteps, it is simply $\Delta z_{\min} = \Delta z_{\max} = 0$.

5.3.2 Safe Actions

During planning, our system evaluates which of the actions $a \in \mathcal{A}$ can be executed safely from a given state s. Our approach first checks whether the resulting footprint $s' := a(s)$

is accessible, i.e., whether the corresponding surface of the map is sufficiently flat and horizontal. Accordingly, we compute the difference between the minimum and maximum values in the heightmap under the footprint at state s and check whether it lies within a threshold (implied by the robot's hardware and walking controller).

Additionally, an action a is only allowed if the height difference $\Delta h(s,a) := z_{s'} - z_s$ from s to its successor s' lies within the limits Δz^a_{min} and Δz^a_{max} associated with a. Hence, we require the following inequality to hold:

$$\Delta z^a_{\text{min}} \leq \Delta h(s,a) \leq \Delta z^a_{\text{max}}. \tag{5.4}$$

Finally, for motion planning in three-dimensional environments it is important to check whether the motions are free of collisions, whereas in footstep planning approaches, often only the footprints are checked for collision (e.g., Hornung et al., 2012a; Nishiwaki et al., 2012). Our approach to whole-body collision checking is described in the following.

5.3.3 Whole-Body Collision Checking

We propose a new representation called inverse heightmap (IHM), which is computed for each action $a \in \mathcal{A}$. An IHM is a grid, centered at the stance foot, that stores for each cell the minimum height, relative to the stance foot, for any part of the body that falls into this cell while executing an action. It is constructed from an animated 3D model of the robot. The bottom image in Figure 5.2 shows an example of an IHM for a step-over motion, along with the 3D model used for generating the IHM (top row). The green volume is a 3D visualization of the IHM and similar to a swept volume. However, for efficient collision-checking, we only consider its projection in the IHM.

To evaluate whether it is safe to execute an action a at a state s, our system first aligns the corresponding IHM_a with the foot's pose according to s by an affine transformation. Then, a simple comparison between the height map and the IHM_a is used to decide whether a can safely be executed. Therefore, we apply bilinear interpolation on the transformed IHM_a to align the cells of both maps and check if

$$\forall c \in \text{IHM}_a : i_c + z_s > h_c. \tag{5.5}$$

Here, z_s is the height of state s according to the heightmap, i_c is the value stored in cell c of IHM_a, and h_c is the corresponding value in the heightmap. Because of this simple decision criterion, IHMs are an efficient way to perform whole-body collision checks. Note that they can be precomputed for all actions $a \in \mathcal{A}$.

The described approach requires knowledge about the trajectory of the feet for each action. Because the placement of the swing foot is not encoded in the state, the trajectory

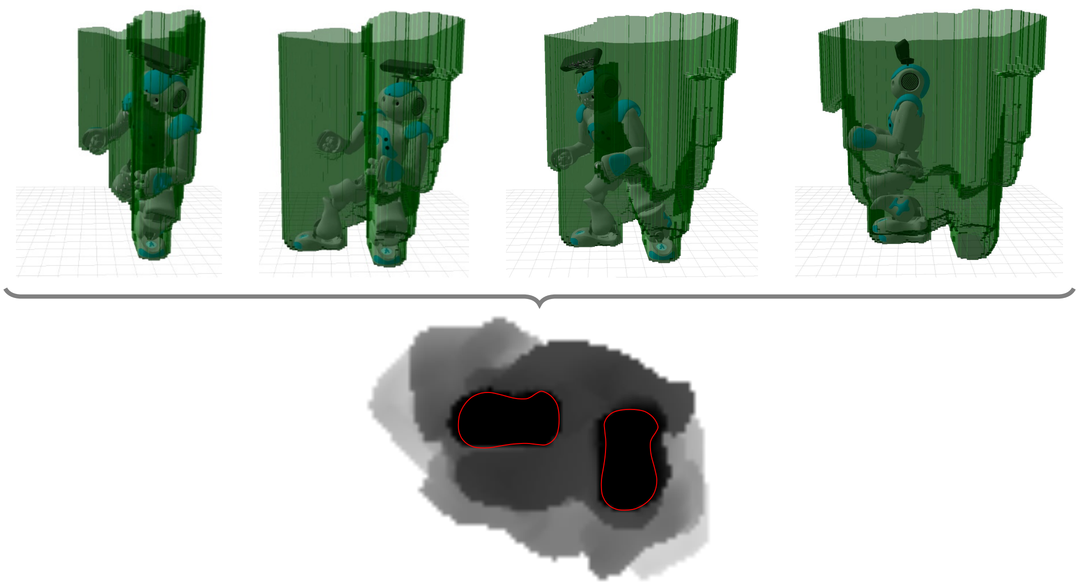


Figure 5.2: Generation of an inverse heightmap (IHM) for a step-over action. The top four images are snapshots of a 3D model of the robot executing the action along with the volume covered by the motion (green). The bottom image shows the resulting IHM (the darker the lower) that corresponds to the projection of the volume swept by the motion onto the ground plane. For reference, the footprints of the robot's initial and next stance foot are outlined in red.

is not fully known. Therefore, we assume that every footstep passes through a predefined via point configuration, similar to Kuffner Jr et al. (2001). The IHM for an action consequently consists of the swing foot's downwards phase from one via point configuration to the double support phase and of the other foot's upwards phase to the other via point configuration.

5.3.4 Footstep Planning with ARA*

Our planner searches for the optimal solution with respect to the time needed to reach a target state and plans a sequence of safe actions that ensure collision-freeness of the whole body. To plan footstep sequences, we rely on Anytime Repairing A* (ARA*) algorithm (Hornung et al., 2012a; Likhachev et al., 2004), which is an efficient anytime variant of A* (Hart et al., 1968). Efficient in the sense that it finds an initial solution as fast as possible, while guaranteeing a bound on its suboptimality, i.e., the deviation factor of the solution with respect to the optimal one. Afterwards, the algorithm tries to refine the solution in the remaining time. This type of algorithm has two advantages. First, it is goal directed and, second, an initial, valid solution is computed fast. The latter is especially useful if the plan has to be updated often, e.g., due to updates of the environment representation.

For the goal-directedness of ARA*, a heuristic has to inform the algorithm about the estimated costs to the goal from any state in the search space. Our approach aims at minimizing the travel time, thus, the heuristic predicts the remaining time to reach the goal location. We consider two different heuristics and compare their performance in the experimental evaluation presented in Section 5.5.1. To be able to compute the optimal path, the heuristic needs to be admissible, i.e., it may not overestimate the true costs. For both heuristics, we obtain the predicted remaining time $t(s)$ from a state s to the goal by the estimated remaining distance $d(s)$ divided by the velocity $v_{\max}$ of the fastest action the robot can perform, to avoid underestimating the actual costs. Thus, we define $t(s) := d(s)/v_{\max}$.

Euclidean Distance Path Cost Heuristic

One simple heuristic for obtaining the remaining distance $d(s)$ to the goal is the Euclidean distance. It promotes expanding states on the straight line to the goal. This heuristic is clearly admissible. On the down-side, it is often a poor approximation in cluttered environments where detours are inevitable (Hornung et al., 2012a) or when step-over actions are associated with higher costs, as is the case for small-sized humanoids including the Nao.

Dijkstra Path Cost Heuristic

We also developed a more informed heuristic that better approximates the true distance in presence of obstacles. The heuristic is based on Dijkstra's shortest path algorithm. In particular, we construct a graph over the heightmap, where each node represents a cell of the heightmap. The edges of the graph represent the eight-neighborhood of the nodes.

Our algorithm assigns traversal costs to each edge according to the difference of the height values in the neighborhood of the connected nodes. If the height difference exceeds the maximum step height of the robot, the edge is considered non-traversable and infinite costs are assigned. Edges in a planar neighborhood are assigned costs according to the euclidean distance. All other edges correspond to a change in the elevation where the robot could step up, down, or over. To account for the additional time to execute these actions, our approach assigns higher costs to such elevating edges. Afterwards a full Dijkstra search is performed on the graph. For a state s, the metric distance $d(s)$ is evaluated as the accumulated traversal costs for all the edges on the shortest path to the goal from the node corresponding to s.

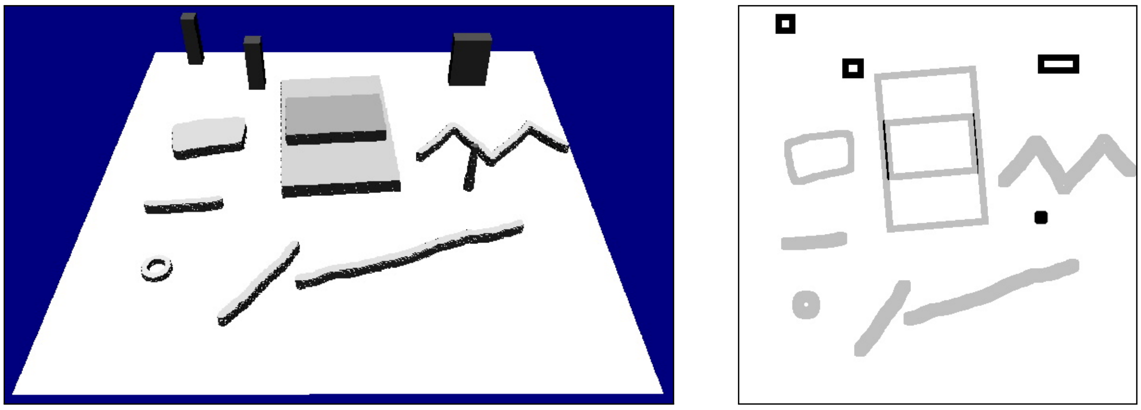


Figure 5.3: The left image shows an example heightmap, and the right image the corresponding traversability costs used to generate the Dijkstra heuristic.

Figure 5.3 shows an example for the traversability classification of a heightmap. Here, dark corresponds to non-traversable, bright to planar, and gray to elevating edges. The particular costs of the edges depend on the target hardware platform and should not overestimate the true costs. Otherwise, the heuristic would be inadmissible and, thus, optimality cannot be guaranteed.

5.4 Action Set for the Nao Humanoid

In this section, we describe the action set for the Nao humanoid that we used during the experimental evaluation (see Appendix B). With the provided walking controller, its swing foot can be placed at most 8 cm to the front and 16 cm to the side and the peak elevation is 4 cm. The size of the robot's feet is approximately 16 cm×9 cm. From these numbers, it is clear that Nao is not able to step over, onto, or down from obstacles using the standard motion controller. The discrete set consisting of 12 basic footsteps that we selected from the possible motions of the standard controller is shown in Figure 5.4 (a).

Using kinesthetic teaching, we designed motions that allow the robot to overcome these limitations. A special motion, the so-called T-step, where the feet are placed at an angle of 90°, as shown in Figure 5.4 (b), is the basis for the other actions. Our motivation for this action is to exploit the larger lateral foot displacement while moving forward. From this pose, the robot can perform a step-over action to overcome obstacles with a height and width of 6 cm, as demonstrated in Figure 5.4 (c). Furthermore, from the T-step, the robot is able to step onto or down from obstacles. Figure 5.4 (d) illustrates the robot stepping onto an obstacle exemplary for this motion. The motion is similar to the step-over action but the swing foot is placed closer to the stance foot and at a different height. The height is adjusted during execution using inverse kinematics based to the value in the heightmap.

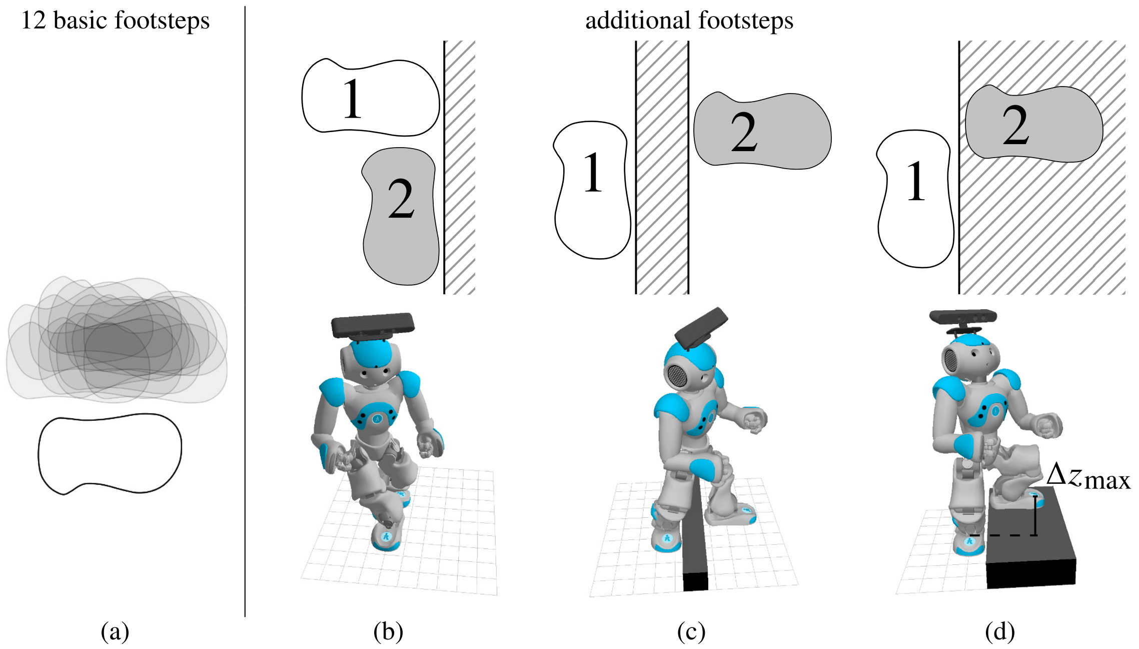


Figure 5.4: Footsteps set for Nao. (a) Basic planar footsteps, (b) T-step, (c) Step-over action, (d) Step-onto/down action. Step-over and Step-onto/down actions are preceded by a T-step. All actions are also mirrored for the other foot.

For this action, the height difference relative to the stance foot must be in the interval $[\Delta z_{\min}, \Delta z_{\max}]$ as defined in Section 5.3.2. In our implementation, we set $\Delta z_{\min} = -7\,\text{cm}$ and $\Delta z_{\max} = 7\,\text{cm}$. All motions also exist in a mirrored version for the other stance foot. Thus, in our experiments, the action set $\mathcal{A}$ consists of 16 actions per foot.

5.5 Experimental Evaluation

In this section, we present the results from a thorough evaluation of our system. We first evaluate our 3D planner quantitatively in experiments with simulated heightmaps. Afterwards, we demonstrate the capabilities of our navigation system in a series of real-world experiments. All experiments were carried out with a small-sized Nao humanoid with a head-mounted ASUS Xtion depth camera as described in Appendix B.

In the experiments presented in the following, the robot records a point cloud with its camera every second step[1]. The robot waits about 0.5 s before recording the point clouds to reduce disturbances caused by its shaking motion and delays between the joint encoder readings and the depth camera data. To obtain a larger field of view, the robot takes two depth images with its camera facing left and right and combines them to one large point

[1]Generally, it is also possible to increase the number of steps between two consecutive measurements

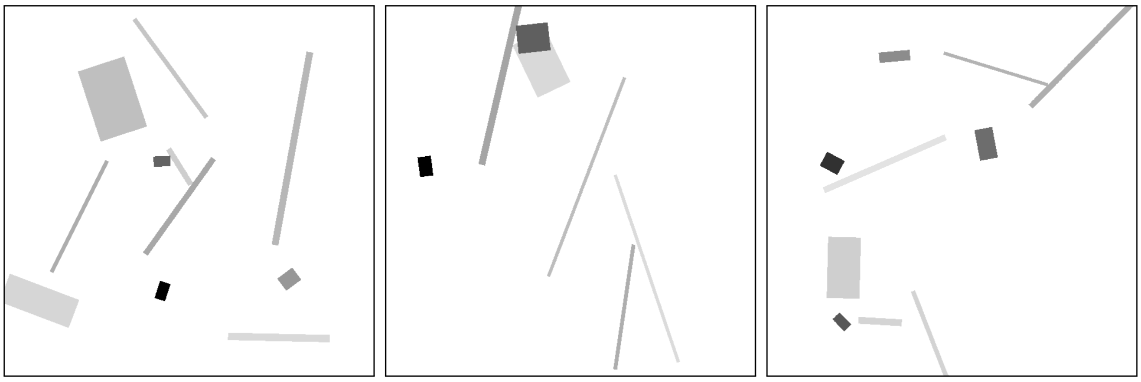


Figure 5.5: Three randomly sampled maps consisting of bars, platforms, and blocks used to quantitatively evaluate our 3D planner. The level of the color gray hereby indicates the height of the cells (the darker, the higher).

Table 5.1: Quantitative evaluation of the motion planner.

Heuristic t-Limit	Dijkstra 10 s	Euclid 10 s	Dijkstra 5 s	Euclid 5 s
Suboptimality	1.04±0.05	1.13±0.14	1.08±0.10	1.19±0.19
$t_{\text{init\_sol}}$ [s]	0.73±0.83	0.61±0.54	0.71±0.80	0.58±0.52

cloud. Here, the camera is pitched down by 50° and the yaw angle is 17° and −17°, respectively. Our planner treats unknown areas as free space to allow for planning into unknown areas, which often occurs when only onboard sensor data is used and no prior information on the environment is available. While executing the planned motions, the robot actively looks in direction of the next unknown area along the path to update the heightmap. After each new measurement the robot checks if the planned path is still valid. If not, the robot re-plans the path. A video demonstrating the system can be found online at `http://youtu.be/g2NZ_EasJv0`.

5.5.1 Quantitative Evaluation of the 3D Planner

To evaluate our planner and the two different heuristics quantitatively, we randomly generated ten different heightmaps. The sampled maps are all of size 2.5 m × 2.5 m with a resolution of 4 mm and contain obstacles such as bars, platforms, and blocks of varying width, length, and height. Three example heightmaps are shown in Figure 5.5. We sampled start and goal locations uniformly such that they were collision-free and their distance was between 1.5 m and 3.0 m. We then used our motion planner to generate safe trajectories on a computer with an Intel Core i5 3.1 GHz CPU. The initial suboptimality bound, i.e., the maximum allowed deviation from the optimal solution, was set to 8 in all experiments. All 100 planning problems could be solved by ARA* within the given time limits of 5 s and 10 s, respectively. As can be seen from Table 5.1, the extended Dijkstra heuristic leads to more efficient solutions that are closer to the optimal path compared to the straight-line Euclidean distance heuristic (significant at a 95% level). Shown are the mean and standard deviation of the path cost suboptimality (path costs divided by the costs of the optimal solution). On average, it took 97 s to compute the optimal plan with A*, whereas our anytime algorithm generates first, valid solutions within less than a second on average and afterwards improves the initial plan.

We also evaluated the planning performance when considering the 3D structure and the extended action set compared to 2D footstep planning (Hornung et al., 2012a). To this end, we applied ARA* using only the set of planar footsteps shown in Figure 5.4 (a) to the same set of 100 planning problems. The 2D planner was only able to solve 91% of the planning problems within the limit of 10 s. In most of the generated maps, our new 3D planner outperformed the 2D variant in terms of paths costs since the latter one had to choose detours around obstacles. However, in some simpler scenarios the generated solutions of the 2D planner were superior to the ones of the 3D planner. The reason is that the 3D planner has a higher branching factor and has to additionally perform whole-body collision checks, so that the 2D planner could expand more states and found better solutions within the given time limit.

5.5.2 Evaluation of Localization and Mapping

The following experiment is designed to evaluate the state estimation accuracy of our approach in terms of both localization and mapping performance. We tracked the pose of the robot with an optical motion capture system from Motion Analysis while the robot traversed the course shown in the top image of Figure 5.6. The bottom image shows the corresponding heightmap learned by the robot during navigation using our approach. The path traversed by the robot is indicated by the footprints and lead over a bar. As one can see, the map closely resembles the actual structure of the scenario shown in the top image.

Using the motion capture system, we measured the accumulated error between the tracked pose and the pose estimated by our approach and between the tracked pose and the odometry. Figure 5.7 illustrates the results in terms of the planar translational and rotational error, plotted over the actual traveled distance. We observed that the accumulated drift of the pose estimate was 0.21 m in *xy*-direction over the whole trajectory of 3.47 m. Hence, our system drifts approximately 5.9 cm per traveled meter. Analogously, the accumulated drift in the *yaw*-estimate was 3.46° and hence, on average our system drifts 0.99° per meter. For odometry, we noted an accumulated drift of 0.53 m in *xy*-direction and 6.10° in the *yaw*-angle. Hence, the average drift is 15.3 cm and 1.76° per

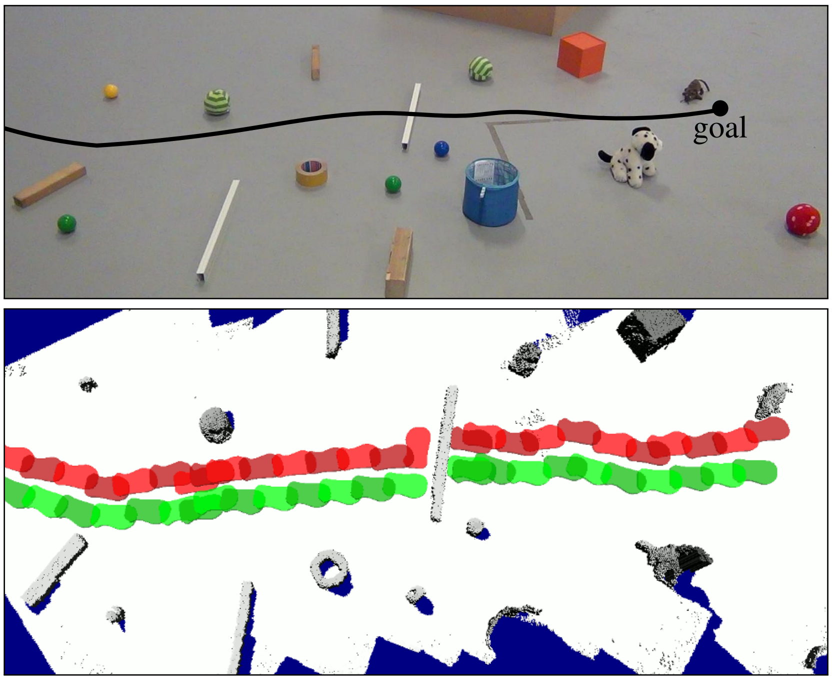


Figure 5.6: Top: Cluttered scenario autonomously traversed by the robot from left to right for evaluating the state estimation accuracy. The black line shows the path traversed by the robot (manually drawn). Bottom: The corresponding heightmap learned during navigation by our approach along with the path traversed by the robot. As can be seen, the environment is represented highly accurate. Dark parts correspond to unobserved area.

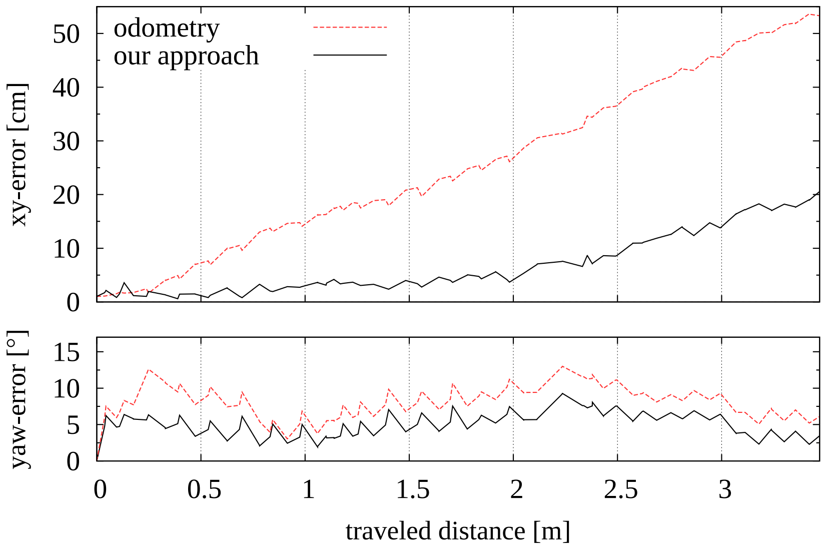


Figure 5.7: Localization accuracy relative to the ground truth from an optical motion capture system. Our approach clearly outperforms the odometry estimate and leads to an accurate pose estimate during navigation.

traveled meter, respectively. Thus, our approach clearly outperforms the pure odometry estimate obtained from forward kinematics of the measured leg joint angles. Furthermore, the experiment illustrates that our localization method is able to reduce the drift of the pose estimate and allows for constructing an accurate environment map.

We also measured the time for aligning the point clouds and updating the heightmap. On average, a heightmap update from a combined point cloud took 0.07 s ± 0.02 s. Combining the point cloud from the left and the right view, and aligning the resulting point cloud to the previous combined point cloud with GICP took 0.41 s ± 0.22 s.

5.5.3 Parametrized Stepping Over and Onto Motions

In the remaining two experiments, we present qualitative results of our framework with a Nao humanoid. Figure 5.8 depicts an experiment in which the robot climbed two stairs up and down again. No model of the stair was used, not even the height of the stairs was known beforehand. The heightmap was constructed online and the height of the footsteps was computed according to this representation.

5.5.4 Traversing Narrow Passages

The final experiment demonstrates the advantage of utilizing whole-body collision checking. Figure 5.9 shows a scenario where a Nao humanoid needed to traverse a passage that was so narrow that the humanoid could not walk through facing forwards without its arms colliding with the obstacles. Consequently, our motion planner computed a path where the robot traversed the passage sideways and without collisions. As the lower body fitted through in a forward direction, motion planners that consider only the feet or legs of the robot for collision checks might have found a solution leading to a collision during the execution of the plan.

5.6 Related Work

Autonomous biped navigation has been studied intensively in the last few years. For instance, Chestnutt et al. (2005) investigated footstep planning among flat obstacles using A*. Hornung et al. (2012a) have reasoned about the impact of different heuristics applied to anytime footstep planning. These approaches check for collisions only by considering rectangular footprints of the robot and do not consider volumetric obstacles. Furthermore, they neglect onboard sensing.

Perrin et al. (2012a) also investigated footstep planning and evolved it further to account for the 3D shape of the humanoid and the obstacles. They perform collision checks for

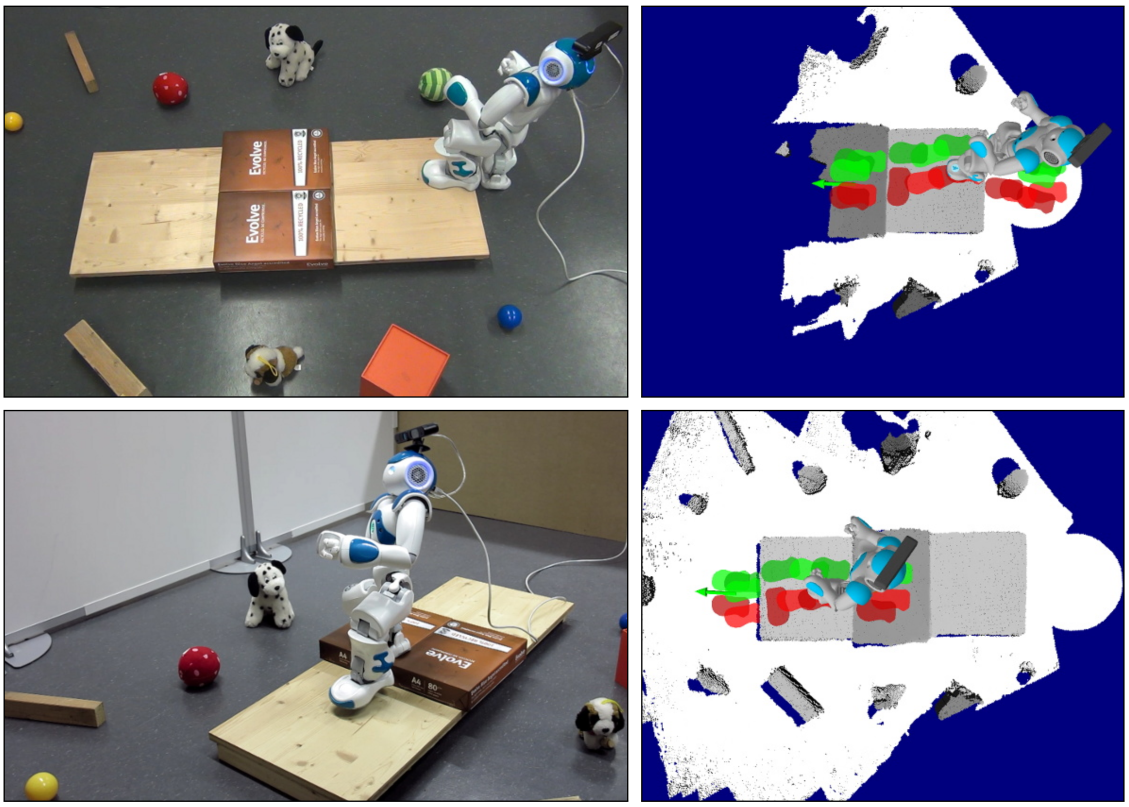


Figure 5.8: Nao climbing up and down steps by carrying out parameterized step onto and down actions. The right column shows the heightmap representation at the time of the image on the left along with the current footstep plan and the pose estimate.

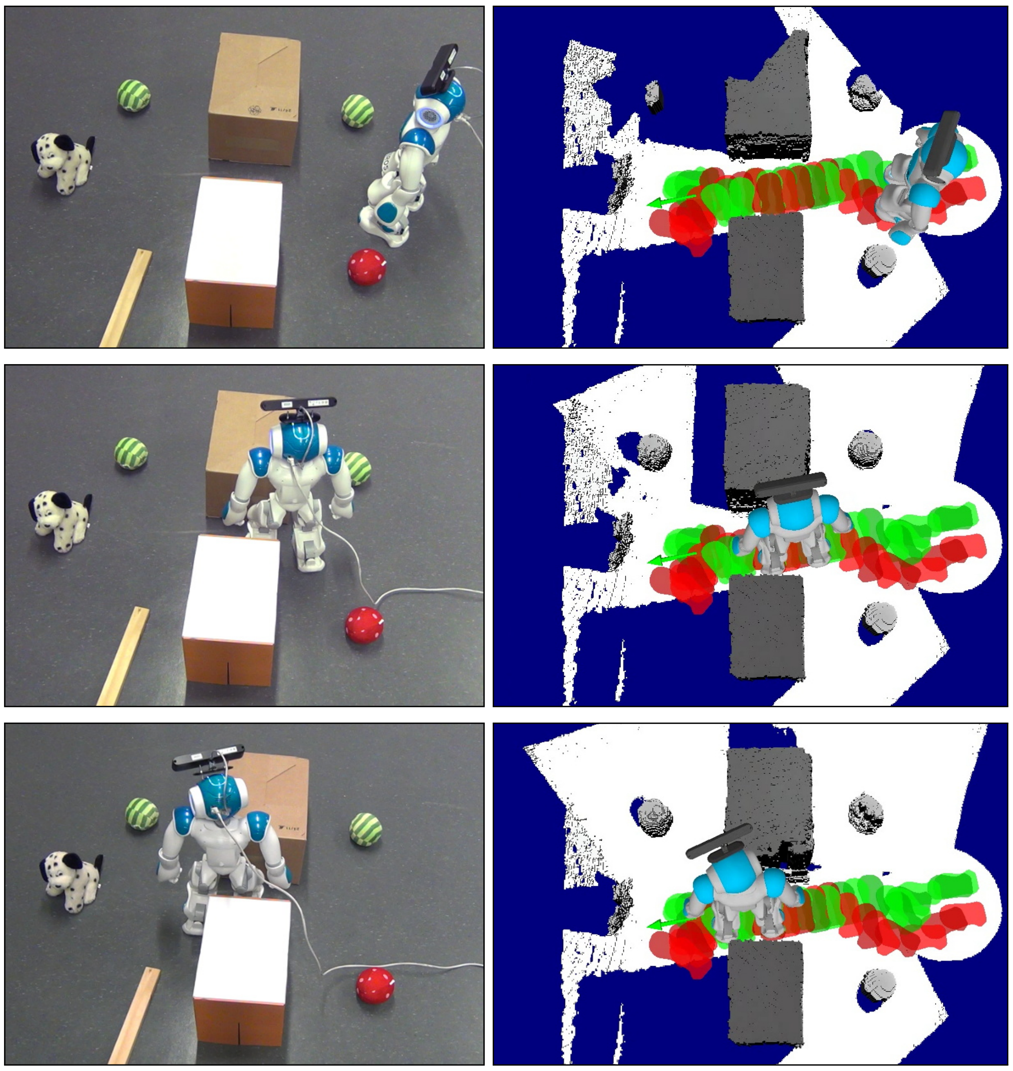


Figure 5.9: Traversing a narrow passage. The Nao humanoid cannot walk through the passage facing forwards because its arms would collide with the obstacles. Our planner performs whole-body collision checks and thus computes a solution where the robot traverses the passage sideways.

the legs of the robot by precomputing swept volume approximations of the swing leg trajectories. Perrin et al. (2012b) further suggested simplifying the collision check for near real-time performance by approximating the robot's shape with a combination of three boxes. While navigating, both approaches either rely on an external motion capture system (Perrin et al., 2012a) to localize the robot and the obstacles, or assume known, simulated environments (Perrin et al., 2012b).

Other authors combine footstep planning and sensing in one system to allow for autonomous navigation. For instance, Cupec et al. (2005) expect obstacles at contrast edges in the camera image of a humanoid robot. From the detected edges, they employ a scene analysis that assumes rectangular objects to estimate their poses and to separate the objects into three different classes. The classification is then used to plan footsteps for the humanoid. However, this approach requires a controlled environment with restricted lighting and obstacle shapes. Michel et al. (2007) presented a method to track objects in monocular images. This enabled a HRP-2 robot to accurately localize itself relative to a staircase and plan footsteps to climb it. However, the approach requires a detailed prior model of the object. Thus, both techniques are not generally applicable for collision-free navigation in unknown environments with arbitrary objects.

More general approaches try to construct a representation of the free and occupied space in the environment while navigating. For example, Gutmann et al. (2005, 2008) maintain a labeled heightmap and a 3D occupancy grid based on data from an onboard stereo camera. The mapping system relies solely on odometry. The map representation is classified into floor, stairs, borders, tunnels, or obstacles and is used for planning discrete actions to a target location. For collision checking, the required space of each action is approximated by cylinders. The coarse resolution of the map and the approximative collision checks do not allow planning actions such as to step over objects.

Nishiwaki et al. (2012) utilize a tilting laser scanner mounted on a humanoid robot for environment mapping. While navigating, their robot takes 3D scans of the area in front. The laser point clouds are binned into cells of a heightmap which is used for judging the quality of possible footprint locations and planning a sequence of safe stepping positions. Because this approach also relies solely on odometry for pose estimation, old data is deleted from the map to reduce artifacts resulting from accumulated errors. Furthermore, collision checks are only performed for the footprints, i.e., they disregard the body of the humanoid.

Recently, Ramos et al. (2014) presented another interesting method for walking on rough terrain with a humanoid using inverse dynamics control and 3D model reconstruction from a stereo vision. The model is implemented as truncated signed distance function and used to adjust the foot position and trajectory according to the ground terrain structure. The system uses a quadratic solver to compute the foot trajectories and keep the robot dynamically balanced. However, so far, the approach has only been demonstrated in simulation, which simplifies the problem substantially.

Recently, simultaneous localization and mapping (SLAM) systems that operate on RGB-D camera data have been presented (Endres et al., 2012; Henry et al., 2012). These approaches are concerned with global consistency of the constructed maps and are computationally demanding. In the presented work, we do not intend to solve similar problems but focus on maintaining a locally consistent, high-resolution map that can be used for 3D footstep planning and whole-body collision checking.

Newcombe et al. (2011a) presented *Kinect Fusion* and Whelan et al. (2012b) *Kintinuous*, which are approaches for dense surface modeling and pose tracking with RGB-D cameras. While in general, these systems provide very impressive results, they rely on the presence of sufficient variation in depth. In preliminary experiments, we observed that these systems struggle with typical humanoid robot navigation scenarios where only a dominant floor plane and sparse clutter is visible.

5.7 Conclusions

In this chapter, we presented valuable extensions and techniques to RGB-D camera-based navigation for humanoid robots. We showed an integrated approach that enables a humanoid to navigate in previously unknown, cluttered environments, without the need for a given map for localization. Instead, our system includes incremental pose estimation based on odometry and point cloud alignment using GICP. Further, it features mapping of the environment using a high-resolution heightmap representation that is consequently used for anytime footstep planning and whole-body collision checking using novel inverse heightmaps. To the best of our knowledge, this is the first system that combines these techniques in a unique framework.

As the experiments with a Nao humanoid show, our technique to pose estimation clearly outperforms the odometry estimate and allows for the construction of accurate heightmaps. Based on this, our robot plans and robustly executes sequences of actions that include stepping over and climbing up or down obstacles as well as passing through narrow passages, which previously was not possible. Our approach to planning and collision-checking based on a learned heightmap representation can be generally applied to any humanoid robot.

One interesting aspect for future work would be to obtain global consistency. This could for instance be achieved by using the localization approach presented in Section 4.2. The former requires a given prior map of the environment, which restricts the applicability of the approach to known areas, which is not an option for instance in disaster scenarios.

However, such a map could be acquired from a SLAM approach (Whelan et al., 2012b) using a horizontal camera setup for better depth variation and an additional sensor (e.g., a second camera) for obstacle avoidance with a coarser model for collision-checking. Maintaining multiple heightmaps would further allow the robot to navigate in multi-story environments. For larger robots that can take wider steps and consequently do not require such high-resolution maps, employing a volumetric map structure could also be an interesting alternative.

Chapter 6

Visual Perception for Accurate Manipulation Tasks

In this chapter, we present perception and motion generation techniques for accurate manipulation tasks. In particular, we consider the task of enabling a Nao humanoid to autonomously play a metallophone using these methods. The core of our approach is a model-based method to estimate the pose of the instrument and the beaters held by the robot using observations from the onboard monocular camera. We find valid configurations of the arms for beating the individual sound bars of the instrument. To determine these, we rely on the estimated pose of the instrument and the beaters and solve the inverse kinematics (IK) problem. Hereby, we use precomputed forward kinematics solutions represented by a reachability tree to accelerate the IK computation and compensate for local minima. The robot automatically validates the computed IK configurations from visual and auditory feedback using its sensors, and adapts its arm configurations if necessary. Our system parses MIDI-files of whole songs, maps the notes to the corresponding arm configurations for beating, and generates trajectories in joint space to hit the sound bars. As we show in the experiments with a Nao humanoid, our approach allows for clean and in-time playing of complete songs on a metallophone. The introduced techniques can be adapted to different robots and manipulation tasks.

In the previous chapters, we introduced valuable techniques for humanoid robot navigation based on visual information. So far, these techniques mainly concentrated on the robot's lower body, e.g., to plan a sequence of footsteps through the environment from a visually learned map. We presented strategies enabling the robot to reach distant target locations and avoid obstacles on its way.

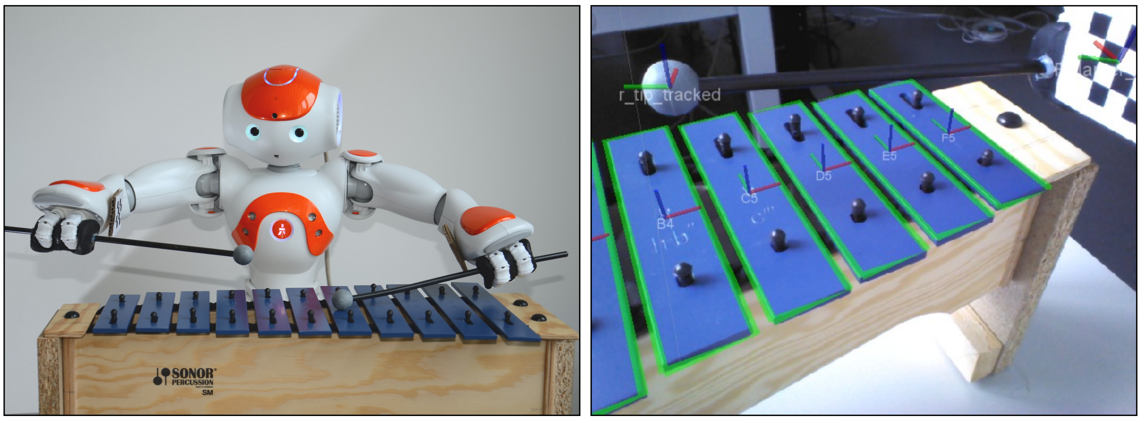


Figure 6.1: *Left:* Nao humanoid playing the metallophone. *Right:* View of the robot's onboard camera with estimated pose of the instrument, one beater's head (indicated r_tip_tracked), and calibration marker (coordinate frame inside the checkerboard).

However, in the big picture, the robot does not only need to reach a location. Reaching the location is just a preliminary condition in order to fulfill a given task. For instance, the robot could be told to deliver an object and place it on a table or it could be asked to repair a piece of furniture or play a musical piece on an instrument for entertainment. Generally, the ability to manipulate objects is a necessary capability for humanoid robots.

In this chapter, we present techniques that allow a humanoid robot to perform accurate manipulation tasks. We implement and demonstrate these techniques exemplary by enabling a humanoid to play a musical instrument, in this case, a metallophone.

In the last decade, there has been a raise of interest in employing robots in the entertainment field, including music. For instance, Chida et al. (2004) developed the robotic flutist WF-4, Weinberg et al. (2009) introduced the marimba-playing robot Shimon, and Mizumoto et al. (2009) demonstrated a theremin-playing HRP-2 robot, while Batula and Kim (2010) developed a small-scale humanoid pianist. In addition to entertainment, such robots could also be employed for teaching musical instruments to children or in the treatment of autistic people (Fujimoto et al., 2010; Ricks and Colton, 2010).

In contrast to the previous robotic musician applications, in our case visual observations are necessary to track the instrument and the beaters held by the robot. To this end, we use a particle filter and develop a novel observation model to enable the robot to accurately track the pose of these objects based on edge observations from its onboard camera.

Given the robot's kinematic parameters (see Chapter 2) and the pose estimates of the instrument and beaters, we use inverse kinematics (IK) to determine valid configurations of the arms to execute beating motions. To speed up the IK computation, we hereby use a so-called reachability tree in which we store precomputed forward kinematics solutions. Since the visual observations and the robot calibration might not be perfect, the robot

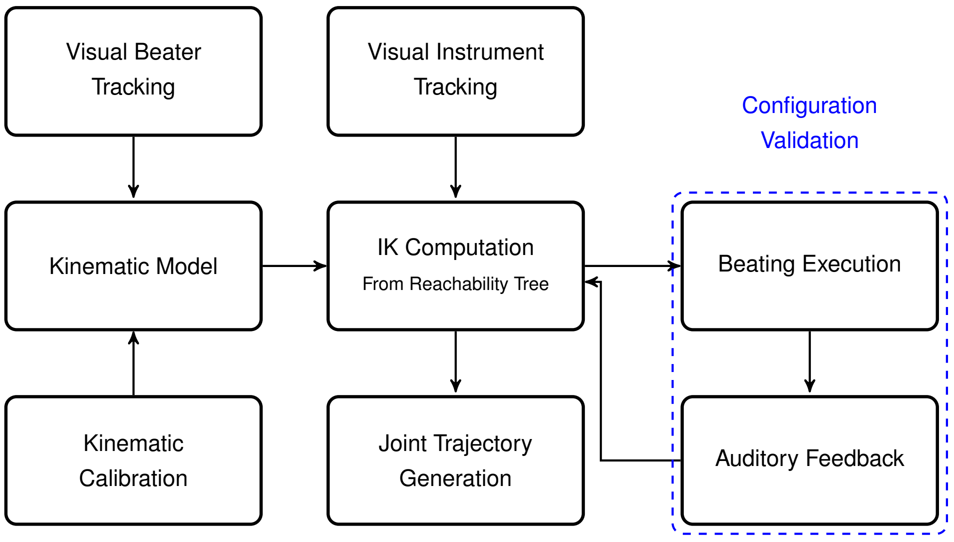


Figure 6.2: Overview of the proposed instrument playing system.

automatically validates the generated IK configuration by playing the notes and adapting its motions if necessary. Hereby, the robot uses auditory feedback from its onboard microphones and applies pitch detection based on the Fast Fourier Transform (FFT) to identify whether the note was hit correctly.

For playing whole songs, our system parses MIDI-files and maps the tones to the corresponding arm configurations that enable the robot to beat the individual sound bars. To reach these arm configurations, the robot generates joint trajectories by connecting the learned configurations. Figure 6.2 depicts an overview of our system.

We implemented our approach on a Nao humanoid (see Appendix B). Figure 6.1 shows the robot playing the instrument and illustrates tracking results for the metallophone and one beater's head in its camera image. As the experiments presented in this chapter the robot is able to play complete songs cleanly and in time using the methods presented in this chapter. A video that summarizes our approach and demonstrates some results can be found online at `http://youtu.be/XtUAiIpYUR8`.

6.1 Instrument and Beaters

We use a Sonor SM soprano-metallophone with 11 sound bars of 3 cm in width. The instrument has a size of 49 cm × 20 cm × 22 cm, including the resonating body. The smallest sound bar is playable in an area of 5.5 cm × 3 cm, the largest in an area of 9.5 cm × 3 cm. The instrument is diatonically tuned in C-Major. As beaters, we use

Sonor SCH2 with modified grips (see Figure 6.1) to allow the Nao's simple grippers to hold them. The beaters are approximately 26 cm in length with a head of 1 cm radius.

6.2 Model-Based Object Pose Tracking

To play the metallophone, the robot needs to be able to adjust its motions according to the estimated relative poses of the instrument and the heads of the beaters it is holding. Our approach uses a model-based technique within a particle filter framework to estimate these poses[1]. In Section 4.2, we already employed a particle filter for estimating the robot's pose within a given map from depth camera observations. Now, we seek to estimate the pose of an object relative to the robot. A depth camera would not work here, as the object is typically too close to the robot. Instead, we follow the idea that, given a hypothesis about an object's pose, one can project the contour of the object's model into the camera image and compare it to the actually observed contour. In this way, it is possible to evaluate the likelihood of the pose hypothesis. Using independent particle filters for the instrument and the beaters' heads, our system maintains multiple hypotheses for each and propagates them over time. Our approach hence shares the same idea as e.g., the work by Choi and Christensen (2012); Gonzalez-Aguirre et al. (2014); Michel et al. (2007).

As in Section 4.2, the particle filter represents the posterior density function over an object's pose at time t by a set of weighted samples $\mathcal{X}_t = \{(\mathbf{x}_t^1, w_t^i), \cdots, (\mathbf{x}_t^n, w_t^n)\}$. Each sample i represents a hypothesis about the pose $\mathbf{x}_t^i$ of the object. The weight w_t^i is proportional to the likelihood of the hypothesis given the history of observations. At each update step, the particles are first propagated according to a random walk as

$$\mathbf{x}_t^i = \mathbf{x}_{t-1}^i \oplus \boldsymbol{\varepsilon}_t^i, \tag{6.1}$$

$$\text{where} \quad \boldsymbol{\varepsilon}_t^i \sim \mathcal{N}(\mathbf{0}, \Sigma_t). \tag{6.2}$$

Here, Σ_t is the motion covariance at time t, from which $\boldsymbol{\varepsilon}_t^i$ is sampled. This step corresponds to the prediction step in Section 4.2. From the current camera image I_t, the importance weights of the particles are updated according to the observation model as

$$w_t^i = p(\mathrm{I}_t \mid \mathbf{x}_t^i), \tag{6.3}$$

[1]It is possible to integrate the pose estimation in the kinematic calibration described in Chapter 2. However, we chose a particle filter framework, as it provides continuous updates of the pose. In this way, it is possible to adjust the motions while playing in case the relative poses change. We plan to address this issue in future work.

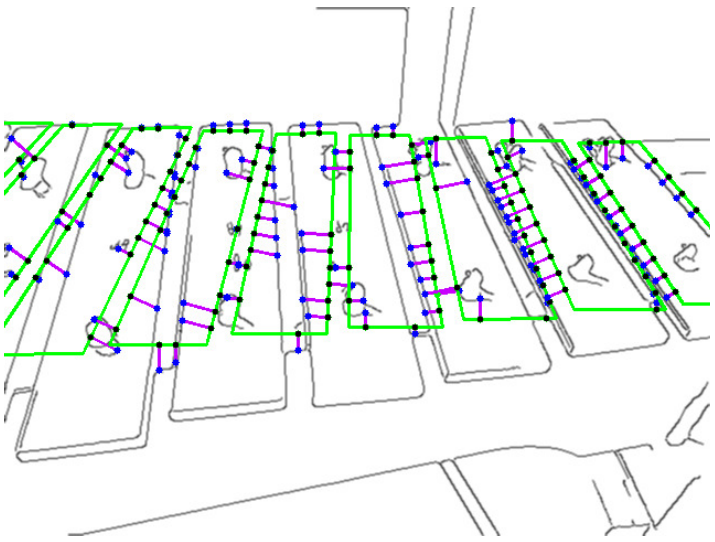


Figure 6.3: Example for likelihood computation in instrument tracking. Green lines are the model edges according to a pose hypothesis, with sampled points (black dots). Gray indicates canny edges, blue dots are detected matches to the green dots. The magenta lines indicate the residuals used in the likelihood computation. Only successful matches smaller than a maximum distance are shown.

which is described in the next two sections for the metallophone and the beaters' heads. This step corresponds to the correction step in Section 4.2. Finally, the filter applies resampling, i.e., a new set of particles is drawn according to the distribution of the importance weights $\{w_t^i\}$, as described in Section 4.2. Our algorithm uses the mean of the particles after the resampling step as pose estimate for the given time step. Note, in the remainder of the section, we only consider the undistorted image I_t, as obtained from the corresponding OpenCV routine (Bradski and Kaehler, 2008) and the camera calibration parameters (see Chapter 2).

6.2.1 Tracking of the Instrument

In this section, we describe the tracking of the instrument in the particle filter framework. We denote by $\mathbf{y}_t^i$ the ith pose hypothesis at time t. Fundamental to the localization capabilities of the particle filter is the choice of the observation model $p(\mathrm{I}_t \mid \mathbf{y}_t^i)$ for a pose hypothesis $\mathbf{y}_t^i$. Here, the pose represents the transform of the instrument relative to the robot's torso. To evaluate the likelihood of a hypothesis $\mathbf{y}_t^i$ about the pose of the metallophone, our approach matches the camera image I to the projection of the instrument's edge model according to $\mathbf{y}_t^i$. To this end, our system uniformly samples points $\mathbf{P}_m^k$ from the model's edges and projects them into the image according to the

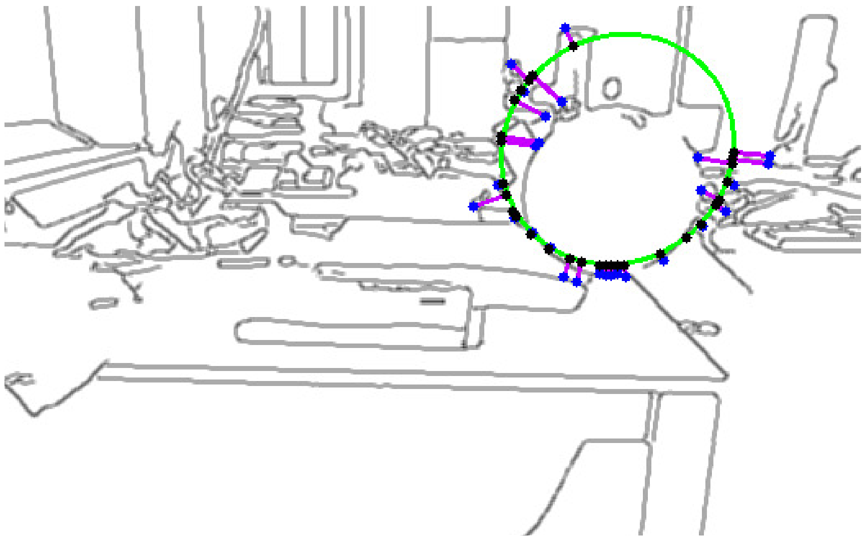


Figure 6.4: Example for the likelihood computation for tracking a beater's head. The green ellipse corresponds to the contour of a pose hypothesis, with sampled points (black dots). Gray indicates canny edges, blue dots are detected matches to the green dots. The magenta lines indicate the residuals, used in the likelihood computation. Only successful matches smaller than a maximum distance are shown.

pinhole model, after transforming the points into the camera frame (see (A.9)). Thus, we obtain

$$\mathbf{p}_m^k = K \begin{bmatrix} R & -R\,\mathbf{c} \end{bmatrix} \mathcal{F}_T^N(\boldsymbol{\theta}, \mathbf{q}_t)\, \mathcal{T}(\mathbf{y}_t^i)\mathbf{P}_m^k, \tag{6.4}$$

where $(R, \mathbf{c})$ are the orientation and position of the camera relative to the neck joint obtained via calibration (see Chapter 2). $\mathcal{F}_T^N(\boldsymbol{\theta}, \mathbf{q}_t)$ describes the forward kinematics of the torso with respect to the neck frame according to the calibration parameters $\boldsymbol{\theta}$ and the joint configuration $\mathbf{q}_t$ at time t (see Appendix A.3). Further, K is the camera matrix as in (A.6). Finally, $\mathcal{T}(\mathbf{y}_t^i)$ gives the homogeneous 4-by-4 transformation matrix corresponding to the pose hypothesis $\mathbf{y}_t^i$.

The projected model edge points $\mathbf{p}_m^k$ are then matched to the closest edge points $\mathbf{p}_e^k$, obtained from the Canny edge detector (Canny, 1986). To find the correspondences, our method follows the model edge normal at $\mathbf{p}_m^k$ until the line intersects with a Canny edge point. If the difference in orientation between the model edge and the corresponding image edge at $\mathbf{p}_e^k$ falls below a threshold, we consider the points matching and store the residual $r^k = \|\mathbf{p}_m^k - \mathbf{p}_e^k\|$. Otherwise, we follow the line until the next Canny edge with matching orientation. If no match can be found within a maximum distance, this maximum distance is assigned to r^k. Our algorithm approximates the image edge orientation by convolving the image with a horizontal and a vertical Sobel operator and computing the `atan2` of the two resulting gradients[2].

With the residual vector $\mathbf{r} = \left[r^1, \cdots, r^{N_p}\right]^\top$, where N_p is the total number of sampled points, our approach then computes the observation likelihood from the arithmetic mean $\bar{\mathbf{r}}$ of $\mathbf{r}$. We assume the observation likelihood to be distributed according to an exponential distribution over $\bar{\mathbf{r}}$ where the distribution parameter λ_y was determined experimentally:

$$p(\mathrm{I}_t \mid \mathbf{y}_t) = \lambda_y \, e^{-\lambda_y \bar{\mathbf{r}}} \tag{6.5}$$

Figure 6.3 shows an illustration of the likelihood computation. The green lines are the projected model edges, gray are the canny edges, green dots are the $\mathbf{p}_m^k$, blue dots the $\mathbf{p}_e^k$, and magenta lines the residuals r^k. Only edge points that could be matched are shown in the image.

6.2.2 Tracking of the Beaters' Heads

To estimate the pose of the beaters attached to the robot's grippers, we use the same particle filter framework[3] described in the previous section. As we are tracking the spherical head of the beater, we do not estimate the rotational component of the pose, but only the translation vector $\mathbf{b}$ relative to the corresponding robot's gripper. We denote by $\mathbf{b}_t^i$ the estimate of particle i at time t.

The observation model $p(\mathrm{I}_t \mid \mathbf{b}_t^i)$ we employ is based on a mathematical model of the spherical head of the beaters. Using the model, we project the outline of the head into the image and match points sampled from the projection to the Canny edge features. A similar mathematical model was previously used to calibrate the camera's intrinsics (Agrawal and Davis, 2003). For readability, in the following we omit the particle index i and time index t in $\mathbf{b}_t^i$, and consequently we just write $\mathbf{b}$. We describe a beater's head as quadric $S \in \mathbb{R}^{4\times4}$ using homogeneous coordinates, with

$$S = \begin{bmatrix} I_3 & -\mathbf{b} \\ -\mathbf{b}^\top & \mathbf{b}^\top \mathbf{b} - r^2 \end{bmatrix}. \tag{6.6}$$

[2]One underlying assumption for finding the correspondences is that the poses represented by the particles are not too far away from the true pose of the instrument. We currently achieve that by a rough manual initialization.

[3]Again, it would also be possible to integrate the pose estimation into the kinematic calibration described in Chapter 2, however, we observed suboptimal estimation results from such an approach. This is likely due to non-linearities in the encoder readings near the joint limits, unmodeled effects such as joint elasticity, and local minima in the optimization. Furthermore, the placement of the beater in the gripper is not always the same for each trial or might even change while playing. The particle filter approach automatically handles such problems by constantly updating the pose estimate between gripper and beater.

Here, r is the radius of the beater's head. To evaluate the likelihood of $\mathbf{b}$, we consider the matching of the camera image I_t, with the projection of S into I_t. The projection of S is a conic $C \in \mathbb{R}^{3\times3}$ (Hartley and Zisserman, 2004, Chapter 8), with

$$C = \left(\mathcal{P}\, S^{-1}\, \mathcal{P}^\top\right)^{-1}, \tag{6.7}$$

where $\mathcal{P}$ is the projection matrix that projects points from the gripper frame into the camera. It is defined as

$$\mathcal{P} = K \begin{bmatrix} R & -R\,\mathbf{c} \end{bmatrix} \mathcal{F}_G^N(\boldsymbol{\theta}, \mathbf{q}_t). \tag{6.8}$$

Here, $(R, \mathbf{c})$ are the orientation and position of the camera relative to the neck joint obtained via calibration (see Chapter 2) and $\mathcal{F}_G^N(\boldsymbol{\theta}, \mathbf{q}_t)$ describes the forward kinematics of the gripper with respect to the neck frame according to the calibration parameters $\boldsymbol{\theta}$ and the joint configurations $\mathbf{q}_t$ at time t (see Appendix A.3). Further, K is the camera matrix as in (A.6). The conic C describes an ellipse that is the outline of the projected sphere, i.e., $\tilde{\mathbf{p}}\, C\, \tilde{\mathbf{p}}^\top = 0$ for all $\tilde{\mathbf{p}}$ in homogeneous coordinates on the ellipse. In inhomogeneous coordinates, the ellipse can be described by

$$0 = \mathbf{p}^\top A\, \mathbf{p} + \mathbf{a}^\top \mathbf{p} + d \tag{6.9}$$

$$\text{where } C =: \begin{bmatrix} A & \mathbf{a} \\ \mathbf{a}^\top & d \end{bmatrix}, \tag{6.10}$$

for all points $\mathbf{p} \in \mathbb{R}^2$ on the ellipse and $A \in \mathbb{R}^{2\times2}$, $\mathbf{a} \in \mathbb{R}^2$, and $d \in \mathbb{R}$ obtained from C.

From (6.9), we compute the ellipse's center $\mathbf{o} = \left[h_x, h_y\right]^\top$, major and minor axes (l_1, l_2) and rotation ϕ, which are needed to sample points from the ellipse. To do so, we bring the ellipse in center-oriented form and perform an eigendecomposition to obtain the parameters. Thus, with the definitions

$$\mathbf{o} := -A^{-1}\frac{1}{2}\,\mathbf{a} \tag{6.11}$$

$$\text{and } B := A\,(\mathbf{a}^\top A^{-1}\mathbf{a} - d)^{-1} \tag{6.12}$$

we can rewrite (6.9) as

$$(\mathbf{p} - \mathbf{o})^\top B\,(\mathbf{p} - \mathbf{o}) = 1\,. \tag{6.13}$$

By the eigendecomposition

$$B = R\,D\,R^\top, \quad \text{with } D = \mathrm{diag}(v_1, v_2) \tag{6.14}$$

we finally obtain the ellipse parameters as: $\begin{bmatrix} h_x & h_y \end{bmatrix}^\top = \mathbf{o}$, $l_1 = \sqrt{\frac{1}{\nu_1}}$, $l_2 = \sqrt{\frac{1}{\nu_2}}$, and $\phi = \frac{1}{2}\operatorname{atan2}(-2A_{12}, A_{22} - A_{11})$. This allows us to write the ellipse equation as

$$x(u) = l_1 \cos\phi \cos u - l_2 \sin\phi \sin u + h_x \tag{6.15}$$

$$y(u) = l_1 \sin\phi \cos u + l_2 \cos\phi \sin u + h_y, \tag{6.16}$$

where $\begin{bmatrix} x(u) & y(u) \end{bmatrix}^\top$ are the contour points of the ellipse, with $u \in [0, 2\pi]$. Our algorithm uniformly samples k values for u and follows the normals at $\begin{bmatrix} x(u) & y(u) \end{bmatrix}^\top$ until they intersect with the Canny edges. From there, we proceed similar to Section 6.2.1, and use an exponential distribution as in (6.3) to evaluate the likelihood from the residuals $\mathbf{r} = \begin{bmatrix} r^1 & \dots & r^k \end{bmatrix}^\top$. The normals of the ellipse are computed from the gradients as

$$\mathbf{n}(u) = \begin{bmatrix} y'(u) & -x'(u) \end{bmatrix}^\top, \text{ with} \tag{6.17}$$

$$x'(u) = -l_1 \cos\phi \sin u - l_2 \sin\phi \cos u \tag{6.18}$$

$$y'(u) = -l_1 \sin\phi \sin u + l_2 \cos\phi \cos u. \tag{6.19}$$

We only consider Canny edge points, where the difference between the orientation of the image gradient and the gradient of the ellipse at $\begin{bmatrix} x(u) & y(u) \end{bmatrix}^\top$ fall below a threshold. If no matching point can be found within a maximum distance, we set the residual to the maximum value. Finally, the approach assigns the likelihood

$$p(\mathrm{I} \mid \mathbf{b}_t^i) = \lambda_b e^{-\lambda_b \bar{\mathbf{r}}}, \tag{6.20}$$

with the distribution parameter λ_b. Figure 6.4 shows an example for the likelihood computation of a single particle, while tracking a beater's head. Even though the background contains clutter, the sampled model points are matched to the contour of the beater's head, as a result of considering the edge orientations. Note, because we are using the forward kinematics function in (6.8), motions of the arms or the head are automatically handled by the system, as the projection matrix $\mathcal{P}$ operating on the sphere is updated accordingly.

6.3 Inverse Kinematics and Beating

Based on the estimated pose of the instrument, the beaters' heads, and the calibrated kinematic model, our system computes for each sound bar a suitable beating configuration for the arm kinematic chain. Suitable means that the beater's head can be placed on the surface of the sound bar at a desired angle ϕ_S. Here, ϕ_S is the angle between the stick of the beater and the instrument plane, which we assume to be parallel to the robot's transverse plane. From this configuration, the control points of a predefined beating motion are updated. To compute the corresponding joint configuration, the system applies IK based on a resolved-rate approach (Buss, 2004). To specify the target configurations for the individual sound bars, we provide 4D target poses (position $\mathbf{x_S}$ and angle ϕ_S) for the beaters, as well as two general reference joint configurations $\mathbf{q}_r$ (for the left and the right arm), from which the computed arm configurations should deviate as little as possible. The latter is to ensure that the robot does not perform too large motions when switching between different notes and to exploit the redundancy. $\mathbf{q}_r$ is automatically generated once per arm as the beating configuration for the each side's respective reference sound bar, which is calibrated first.

In more detail, for a robot configuration $\mathbf{q}$, we consider the corresponding translation $\mathbf{t_q}$ from the torso's center to the tip of the beater, and the angle $\phi_\mathbf{q}$ between the stick of the beater and the robot's transverse plane. Both $\mathbf{t_q}$ and $\phi_\mathbf{q}$ depend on the joint configurations $\mathbf{q}$, the forward kinematic model, and the estimated translation from the gripper to the tip of the beater $\mathbf{b}$ (Section 6.2.2). Furthermore, we extract the rotation matrix R_S and the center of the sound bar $\mathbf{x}_S$, relative to the torso frame, from the pose estimate for the instrument (Section 6.2.1) and its model. We define the vector-valued error function $\mathbf{e}$ for a configuration $\mathbf{q}$ from these terms as

$$\mathbf{e}(\mathbf{q}) := \begin{bmatrix} R_S \mathbf{w}_{[1:3]} \circ R_S^{-1}(\mathbf{x_S} - \mathbf{t_q}) \\ \mathbf{w}_4(\phi_S - \phi_\mathbf{q}) \\ 0 \\ 0 \end{bmatrix}. \tag{6.21}$$

Here, $\circ$ is the Hadamard product, i.e., for $\mathbf{v} = \mathbf{w} \circ \mathbf{x}$, it is $\mathbf{v}_i := \mathbf{w}_i \mathbf{x}_i$ and $\mathbf{w}$ is a vector of error weights. The idea is to give more weight to deviations in direction of the shorter side of the sound bar, as a misplacement in the other direction is less crucial for hitting the bar. Therefore, the displacement vector $(\mathbf{x_S} - \mathbf{x_B})$ is transformed into the frame of the sound bar, weighted, and transformed back. Further, we consider the deviations of the beater angle $\phi_\mathbf{q}$ from the desired angle ϕ_S. The latter controls the inclination or steepness at which the beater will hit the sound bar. Our system does not impose a target value for the remaining two degrees of freedom, therefore the two last entries in $\mathbf{e}(\mathbf{q})$ are zero.

To reach the sound bar with the beater's head, we iteratively compute joint velocities $\dot{\mathbf{q}}_t$ for a discrete time interval Δ_t and update the joint positions $\mathbf{q}_t$ until $||\mathbf{e}(\mathbf{q})||$ falls below a threshold. The control laws are consequently

$$\dot{\mathbf{q}}_t = \mathcal{J}_t^\dagger\, \mathbf{e}(\mathbf{q}_t) + (I - \mathcal{J}_t^\dagger\, \mathcal{J}_t)(\beta_l \nabla f_t + \beta_d \nabla r_t) \tag{6.22}$$

$$\mathbf{q}_t = \mathbf{q}_{t-1} + \Delta_t\, \dot{\mathbf{q}}_t. \tag{6.23}$$

Here, $\mathcal{J}_t$ is the manipulator Jacobian of the beater for configuration $\mathbf{q}_t$ with its pseudo-inverse $\mathcal{J}_t^\dagger$, the β_l and β_d are weights for the secondary tasks, and $(I - \mathcal{J}^\dagger \mathcal{J})$ is the null-space projector. Furthermore, ∇f_t is the joint limit gradient at $\mathbf{q}_t$ and ∇r_t the gradient of the function $r(\mathbf{q})$ evaluated at $\mathbf{q}_t$. The function $f_t(\mathbf{q})$ restricts the joint configurations to the physical limits of the robot and is implemented as a combination of a parabola and a constant function (Chaumette and Marchand, 2001). The function $r(\mathbf{q})$ punishes deviations from the reference joint configuration $\mathbf{q}_r$ and is defined as

$$r(\mathbf{q}) := (\mathbf{q} - \mathbf{q}_r)^\intercal \, (\mathbf{q} - \mathbf{q}_r). \tag{6.24}$$

Such resolved-rate IK solvers suffer from problems such as singularities and local minima. Consequently, a good seed configuration is essential. To this end, we developed an approach, the so-called reachability tree (see Figure 6.5), to precompute a set of joint configurations, similar to Vahrenkamp et al. (2013), along with the corresponding beater positions. These configurations are stored for efficient lookup in a k-d tree (Bentley, 1975). In particular, we use the k-d tree to lookup the configurations $\{\mathbf{q}_n\}$ in a neighborhood of the target position $\mathbf{x}_S$ for each IK query. The closest configuration might not be the best for the IK computation, as the robot calibration or the beater placement in the gripper potentially changed since the tree creation. Therefore, we pick as seed the configuration $\mathbf{q} \in \{\mathbf{q}_n\}$ that minimizes $\|\mathbf{e}(\mathbf{q})\|$ with $\mathbf{e}$ as defined in (6.21) from the current kinematic state and the pose estimate for the beater.

After the robot adapts the joint configuration, our system checks whether the beater's head is placed at the desired location, i.e., the center of the sound bar. Otherwise, the IK computation is restarted with the current pose estimates for beater and instrument. When the desired pose is reached, the robot performs a predefined beating motion to hit the sound bar and then analyzes the audio signal, as described in the following.

6.4 Auditory Feedback

To identify whether the executed beating motion successfully hit the sound bar, we rely on auditory feedback, as visual feedback would require a high speed camera which is still uncommon for robots. To this end, our system analyzes the audio signal of the integrated microphones, sampled at 48 kHz. The system considers a window of circa 2 s after the beating motion is executed. To analyze this signal, we follow a straight-forward approach, based on the Fast Fourier Transform (FFT) introduced by Cooley and Tukey (1965). Let **a**

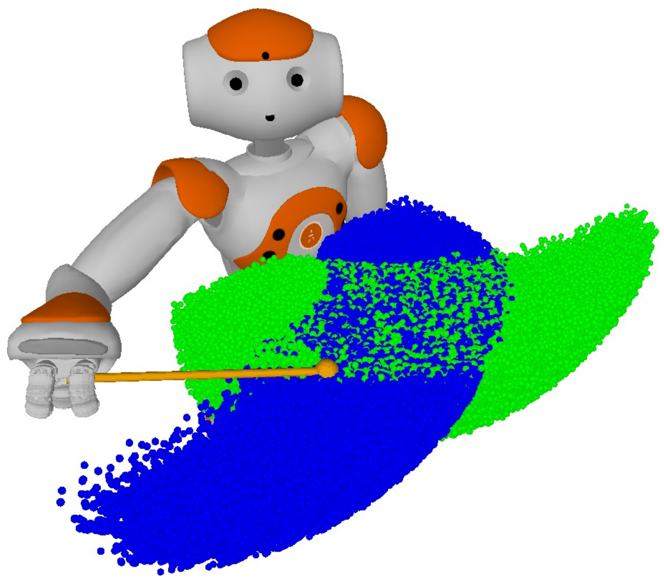


Figure 6.5: Illustration of the reachability tree. Blue dots indicate tip locations that can be reached with the beater in the robot's right gripper. Green dots indicate the same for the beater in the robot's left gripper. The corresponding configurations were sampled from the forward kinematic model and are used in the IK computation. The samples are restricted to the robot's typical workspace.

be a discrete, normalized audio signal of length N_a, with $\mathbf{a}_i \in [-1, 1]$ for $i \in \{1, \cdots, N_a\}$. Our algorithm first computes

$$\mathbf{y} = \frac{1}{N_a}\mathsf{FFT}(\mathbf{a}) \tag{6.25}$$

$$\text{and} \quad \mathbf{m} = \operatorname{abs}\mathbf{y}, \tag{6.26}$$

where $\mathbf{m}$ is the magnitude of $\mathbf{y}$. The algorithm then searches for local maxima in $\mathbf{m}$ based on a sliding window. The peaks are ordered according to the magnitude values, beginning with the strongest magnitude, i.e., $\mathbf{m}_{i_j} \geq \mathbf{m}_{i_{j+1}}$. We assume that the frequency $f(i_1)$ with the maximum magnitude $\mathbf{m}_{i_1}$ is the note that was supposedly played, where $f(i) = \frac{i}{N_a - 1}\, 48\,\text{kHz}$. Our approach employs four different heuristics to identify whether the note was played correctly:

1. The ratio of $\mathbf{m}_{i_1}$ to the median of $\mathbf{m}$ is larger than a threshold.

2. The ratio $\frac{\mathbf{m}_{i_1}}{\mathbf{m}_{i_2}}$ is larger than a threshold.

3. The deviation of $f(i_1)$ from the frequency f_r of the closest reference note for the instrument in frequency space is smaller than a threshold. The deviation is specified in cent as $1200 \log \frac{f(i_1)}{f_r}$.

4. The detected frequency $f(i_1)$ matches the frequency f_d of the desired note.

Only if all heuristics are satisfied, the note is considered successfully played. If only the fourth heuristic fails, the robot repeats the complete IK computation, as the beater is obviously positioned wrongly, e.g., due to wrong pose estimates. Because the pose estimates are continuously updated, repeating the IK computation can compensate for temporary errors in the pose estimates. If any of the other heuristics fail, the audio signal is likely to contain only noise, and the robot first tries to beat harder prior to restarting the complete IK computation. This way, the robot learns the strongness for beating a note, and we achieve a robust beating calibration for playing the instrument.

6.5 Joint Trajectory Generation

Our system parses single-track MIDI files to obtain the sequence of notes to play. It converts the notes into a joint trajectory using the beating configurations obtained from IK as control points. The timestamps for the control points are extracted from the MIDI file as well. Our approach calls the NaoQi-API provided by the manufacturer to compute a trajectory from the determined beating configurations. The API therefore applies Bezier interpolation in joint space. Immediately afterwards, our system sends the trajectory to the robot controller for execution. This way, the robot plays in-time with the song.

6.6 Experiments

6.6.1 Qualitative Evaluation

First, we evaluate the performance of our system as a whole. The results are also demonstrated in the video at `http://youtu.be/XtUAiIpYUR8` as well as in Figure 6.6. Initially, our system performed a fresh kinematic calibration procedure as outlined in Chapter 2. Afterwards, we placed the beaters in the robot's grippers, as autonomous grasping is not in the scope of this work. We then manually initialized the localization system and moved the instrument closer to the robot, such that it can reach all the bars with the beaters. The video shows that the tracker was following the motion of the instrument closely. The robot started the automatic beating calibration, i.e., it estimated the pose of the heads of the beaters, moved the head to the first sound bar, calibrated the beating motion, and proceeded to the next bar. Each sound bar was hit correctly. As can be seen in the video, the robot was then immediately able to play complete songs such as Beethoven's *Ode to Joy* and *Jingle Bells* provided as MIDI files. Each note in both songs was hit correctly. In the video, one can hear some dissonances that occur when the

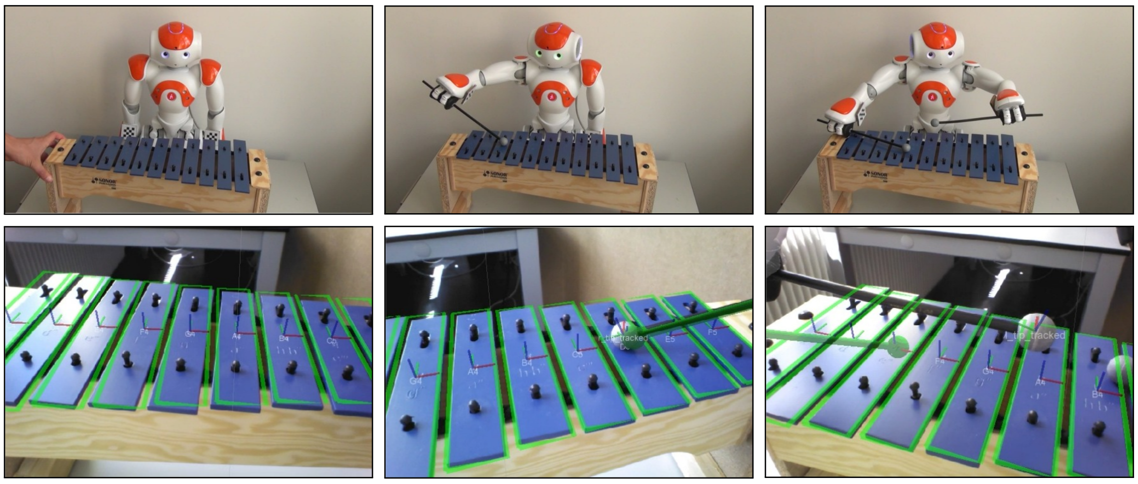


Figure 6.6: Film frame from a video of the beating calibration procedure. The top row shows an external camera video for reference, the lower the robot's onboard perspective along with the tracking results. Left: The instrument is moved in front of the robot, while it keeps track of the instrument. Second column: The robot calibrates the beating configuration for the D-sound bar from the visual pose estimation. Third column: The robot calibrates the beating motion for another sound bar with its left arm. The green beater visual indicates the determined IK solution.

previous note still sounds while another note is hit and both notes are dissonant. Such effects can be prevented by advanced players by stopping the previous note at the same time as playing the current one. In future work, we will investigate such techniques in our system.

6.6.2 Pose Estimation and Calibration

Figure 6.1 depicts an example for the pose estimation of the instrument and one beater's head. The edge model closely fits the camera image, and the estimated center of the beater's head (indicated by the coordinate frame) aligns with the head in the image as well. Furthermore, one can see the result of the kinematic calibration (see Chapter 2), as the coordinate frame indicating the center of the checkerboard marker nicely overlays with the checkerboard in the camera image. In addition to that, the referenced video contains a sequence demonstrating tracking results for the beater and the instrument.

The localization performance is generally good, but the markerless method can fail, once it loses track of the instrument. This is because there is a strong symmetry in the appearance of the object. If the pose estimate is shifted by one (or more) sound bar plus a small deviation in depth, the appearance is very similar and hence, the likelihood function will also return similar values. This can be avoided by additional markers on the object, which we chose to avoid. In practice, the localization performance has proven sufficient for our application.

6.6.3 Auditory Feedback

To test the auditory feedback, we had a novice human player play the notes of the instrument while the robot ran the audio analysis. We evaluated whether the detected notes were the actually played one. The human played notes randomly, with a pause of circa 5 s in between two notes. We placed the instrument at three different locations (left, right, center) relative to the robot. Each of the 11 notes were played in each position. From 33 played notes, the system was able to recognize 30 correctly, for the three failures, the notes were determined correctly, but heuristic two was not satisfied. Consequently, in the beating calibration context, the robot would unnecessarily have tried to solve the IK again. Further, we tested whether the analyzer is able to identify when two notes were hit at the same time. In practice, this is useful, as the robot sometimes hit in between two notes due to imprecise motion execution or minor pose estimation errors. To this end, the novice player was asked to hit two neighboring notes at the same time, thereby simulating the in-between-sound-bars effect (as hitting in between on purpose proved to be difficult). In all three positions, all the eleven notes were correctly identified as unacceptable.

6.6.4 Beating Calibration Accuracy

To test the accuracy of the beating motion calibration, we let the robot perform the beating calibration five times, similar to the one shown in the video and described in Section 6.6.1. For each trial, the instrument's location was shifted slightly within the reachable limits for the robot. For each location, the robot managed to find a configuration for all the eleven sound bars and beat each note correctly as part of the auditory feedback. Therefore, one can conclude, that the accuracy of the visual tracking is sufficiently good for the task of calibrating the beating motions.

6.7 Related Work

The work presented in this chapter combines various techniques into one unique system. Consequently, the related work ranges from visual pose estimation over musical robots and pitch detection up to inverse kinematics. In this section, we will restrict to the most relevant aspects, i.e., visual pose estimation as the main underlying technique and musical robots as an application for our framework.

Musical robots have been studied previously. For instance, Chida et al. (2004) presented a robotic flutist called WF-4. The development focused on the engineering point of view, i.e., the creation of artificial lungs and lips. Weinberg et al. (2009) introduced the marimba-playing robot Shimon. The authors focused on the interaction of the robot with human musicians, i.e., for improvisation. The robot is fixed to the instrument via a slider, and thus, pose estimation is not explicitly addressed. Mizumoto et al. (2009) demonstrated a theremin-playing HRP-2 robot. The authors also assume a known pose of the robot relative to the instrument, but calibrate the pitch of the theremin relative to the motion of the arm and transfer the knowledge to other robots. Batula and Kim (2010) developed a small-scale humanoid pianist. Similar to our approach, the system detects the pitch of the played note to adjust the posture of the robot. However, the authors assume that the robot is placed parallel to the instrument and do not estimate its pose but instead move the robot to the side in steps of the width of the keys. Further literature exists, that focuses on other aspects of musical robots, e.g., acoustic beat tracking. To the best of our knowledge, none of these works estimate the pose of the instrument accurately from exteroceptive sensors.

Tracking of objects has been widely studied in the computer vision literature. A comprehensive survey on these techniques has been published by Lepetit and Fua (2005). Therefore, we will only summarize some of the most closely related and recent works.

Greenspan and Fraser (2003) track a sphere dipole in the camera images using weak perspective of the spheres and compare them to edges in the camera images. Drummond et al. (2002) employ edge models of objects to track them in the camera images. They rely on iteratively reweighted least-squares (IRLS) to estimate the pose of the camera and minimize the error between corresponding edge image and model features. Similarly, Michel et al. (2007) use the GPU to accelerate the tracking of stairs from monocular camera images. They also follow the IRLS approach on the edge model of the stairs, thereby enabling a HRP-2 robot to localize itself with respect to the stairs and climb them. One disadvantage with such optimization approaches is that they suffer from local minima and can typically not recover well from false data associations. Klein and Murray (2006) and Pupilli and Calway (2006) presented approaches that rely on edge models and particle filters to track objects in the camera image. Particle filters typically are more robust to wrong data associations and errors in the observations. However, particle filters are also computationally demanding if many particles have to be maintained and evaluated simultaneously. Therefore, Choi and Christensen (2012) proposed to combine particle filters for tracking with IRLS to refine the estimate of few particles and to consequently reduce the number of required particles. In our case, we do not require many particles, because we exploit the robot's forward kinematics to propagate the particles accordingly. Similarly, Gonzalez-Aguirre et al. (2014) recently applied particle filters to localize a humanoid robot within an environment given its CAD model and the forward kinematics of the robot.

6.8 Conclusions

In this chapter, we described techniques for visual arm navigation and manipulation requiring a high accuracy. At the core, we employ particle filters to estimate the pose of objects from the camera observations and edge models. We implemented our framework for a humanoid robot playing the metallophone by using the beaters. Therefore, our system estimates the pose of the beaters and the instrument. We compute target configurations for the beaters in joint space from a resolved-rate IK solver to reach the sound bars of the instrument with the beaters. To accelerate the IK computation and compensate for local minima, we further presented reachability trees that contain joint configurations and the corresponding beater positions. To increase the robustness of the application and to determine whether a note was hit cleanly, we further validate each of the beating configurations from auditory and visual feedback.

As we showed in the experiments, we achieve good accuracy in tracking the relevant object poses and in the beating calibration. This allows to cleanly play complete songs. Hereby, the robot adapts its motions, if necessary, based on visual and auditory feedback. In future work, we want to update the beating motions while the robot is playing, which relies on a constant update of the instrument's pose as provided by the particle filter.

Our methods are not restricted to the Nao humanoid, but could easily be adapted for other robots as well. The employed techniques such as visual tracking using a model-based approach are general and can be used in the context of other tasks. For instance to estimate the pose of other tools held by the robot. A similar task would for instance involve forcing a nail into a wooden board. The visual tracking could estimate the pose of the nail in one gripper and a hammer in the other one. We could compute possible hammer placements via IK, and validate them visually prior to hitting the nail.

Chapter 7

Conclusions

7.1 Summary

Within this thesis, we introduced valuable advancements to the state of the art in humanoid robot navigation. We hereby focused on cameras as primary sensors for perception during navigation. Specifically, our methods enable humanoids to acquire better knowledge about themselves and their environment, allow them to determine safe areas to step onto, localize within a given map, navigate in challenging environments, and perform manipulation tasks requiring high accuracy.

In summary, we first introduced a self-calibration method based on camera observations of markers attached to the robot's end-effectors. By formulating and solving a graph-optimization problem, we determine the robot's calibration parameters consisting of joint offsets and the camera's extrinsics and intrinsics. Additionally, we showed an approach to generating and selecting meaningful robot postures that minimize the uncertainty in the calibration parameters. Thereby, we reduce the number of required observations and achieve a high accuracy. Accurate robot calibration is a prerequisite for all navigation tasks and consequently all aspects of navigation can profit from it.

Another crucial capability of a robot navigating in unknown environments is the determination of safe areas to walk onto. We presented a method that matches sparse features between two monocular camera images by exploiting the humanoid's specific motion, a model of the ground plane, and odometry information. From the sparse features we train appearance-based classifiers that allow a dense labeling of the camera image from which we construct an occupancy grid. The approach is self-supervised, i.e., it requires no prior training phase, and yields accurate classification while walking or even turning on the spot on different grounds.

For robots equipped with a depth camera, we described a complete navigation framework consisting of a localization component, a mapping subsystem and a path-planning module. We demonstrated accurate localization results in a given map. Furthermore, our system constructs a voxel-based representation of the non-static parts of the scene to plan collision-free paths to a target location. The system is specifically targeted at navigation in complex indoor scenarios featuring multiple levels and handles uncertainty and sensor noise robustly.

As extensions to this system, we introduced anytime footstep planning for navigation in cluttered, narrow scenarios. This way, the robot is able to step over, onto, or from obstacles. Our system considers the whole body of the robot when checking for collisions while planning a sequence of actions. Therefore, we presented a novel, efficient *inverse heightmap* representation of the discrete motions to accelerate the collision checks. Our anytime planner quickly finds initial solutions and subsequently improves them. Furthermore, we compensate for the drift of the odometry by aligning consecutive depth camera observations to allow navigation in unknown scenarios.

Finally, we presented techniques allowing a robot to perform manipulation tasks requiring high accuracy. We implemented our framework for a humanoid playing the metallophone. At the core of our approach, we developed a model-based pose estimation framework for the instrument and the beaters, based on robust particle filters. From the determined poses, we compute beating configurations for both arms. We accelerate the computation and compensate for local minima by precomputing a set of configurations via forward kinematics that are refined via a resolved-rate IK solver. Our system uses auditory and visual feedback to validate the beating configurations. By enabling the robot to play complete songs, we demonstrated the high accuracy of our system.

In summary, we answered the following questions:

- How can a robot obtain knowledge about its own parameters?
- How does a robot know which areas are safe to set foot in?
- How can a robot perceive its environment and maintain an internal representation of it?
- How can a robot locate itself with respect to that representation?
- How can a robot avoid obstacles on its way?
- How can a robot utilize its locomotive capabilities to overcome challenging environments?
- How can a robot manipulate objects in its environment to achieve a goal?

All of our approaches were implemented and thoroughly evaluated on a Nao humanoid. We demonstrated the practicability and utility of the proposed methods even for such an affordable robot platform with limited hardware. We believe that our contributions constitute important advances to the robotics community and will be adopted by other researchers. Many of the presented approaches have or will be released as open source code. Consequently, the 6D localization code has already been used in further publication by different authors, e.g., by Hornung et al. (2014a).

7.2 Outlook

The methods presented in this thesis lay an important foundation for humanoid robot navigation and could inspire manifold follow-up research.

One research direction could feature a marker-less extension to our self-calibration framework, which currently relies on distinguished markers attached to the end-effectors. A possible solution could be to use natural features occurring on the robot's body, either geometric or appearance-based ones. Another option could be to utilize motion induced by moving the robot's limbs in front of the camera. Furthermore, it is imaginable that the robot automatically determines when a recalibration is necessary and induces the calibration routine or even continuously updates its internal models. This is an aspect that is similar to life long learning in the SLAM literature (Kretzschmar et al., 2010).

Based on our technique to self-supervised learning of appearance-based classifiers, one could extend the approach and also add a long term memory (Dayoub et al., 2011). In our method, we considered a short-term memory that forgets previously learned appearance features of the traversable area, once new information is available. By adding models for different floor types and maintaining them over a long time, the system had access to prior knowledge as soon as the robot enters unknown environments. This would allow for more forward-looking planning of exploration steps and might further increase the robustness of the systems.

For constructing highly detailed maps of the environment in the context of 3D footstep planning, we considered a static environment in the vicinity of the robot. A natural extension would thus be to identify moving objects and treat them specially. It would also be useful to equip our framework with means to obtain global consistency in the constructed map, e.g., by integrating it with a SLAM system (Whelan et al., 2012a). This would allow a robot to explore and construct a detailed map of *large*, unknown scenarios, which could be useful in disaster operation. The map could consequently not only be used by the robot but also by human co-workers prior to entering a collapsed building. Furthermore, low-level control could be more tightly integrated with the proposed system.

For instance, by actively controlling the balance of the robot, it could walk on rough or inclined terrain (Nishiwaki et al., 2012).

We showed a method to estimate the pose of objects from their models in the context of manipulation tasks requiring high accuracy. One relevant extension could include autonomous learning of the object model from observations (Martínez et al., 2014) prior to tracking and using the object within our framework. This would increase the autonomy of robots, as no prior model needed to be specified. Service robots in households could particularly profit, as robots need object models for manipulation but the manufacturer cannot provide these models for every potential object in a household.

In this thesis, we treated navigation techniques focusing on the locomotion aspect and techniques for manipulation rather separately. One of the great challenges of the future will be to combine these aspects in a unified framework, allowing complex operations like mobile manipulation, e.g., for picking up debris, carrying it across rough terrain, and placing it on a deposit. Within the same framework, doors or drawers could be opened, as both generally require the integration of locomotion and manipulation aspects.

Finally, at some point, sophisticated navigation capabilities like walking among clutter and climbing of objects have to be combined with high-level task planning. For full autonomy, the robot needs to be able to reason about which steps to take in which order to achieve a given goal (Erdogan and Stilman, 2013). A possible scenario could be that the robot pushes an object that it can step onto, like a chair, under a broken light bulb so it can reach it for maintenance.

Appendix A

Mathematical Background

A.1 Homogeneous Coordinates

In this thesis, homogeneous coordinates are often used for the sake of simplicity. This is because by using homogeneous coordinates, all rigid transformations and even projective transformations can be expressed by one matrix multiplication. Hence, in the following, we briefly recapitulate homogeneous coordinates and their application in projective geometry. In this thesis, we denote homogeneous coordinates by a $\tilde{}$, as for instance, $\tilde{\mathbf{x}}$.

A homogeneous representation of a point $\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^\mathsf{T} \in \mathbb{R}^2$ can be obtained by adding an additional coordinate z, i.e.,

$$\tilde{\mathbf{x}} = \begin{bmatrix} x\,z \\ y\,z \\ z \end{bmatrix}. \tag{A.1}$$

It is important to note that there exists an equivalence class of homogeneous coordinates that all represent the same point. Two homogeneous points are equivalent if and only if they differ only by a scaling factor $\lambda \in \mathbb{R} \setminus \{0\}$. Hence, all choices for $z \neq 0$ represent the same point $\mathbf{x}$. From a homogeneous representation $\tilde{\mathbf{x}} = \begin{bmatrix} x' & y' & z' \end{bmatrix}^\mathsf{T}$, we can obtain the unique, inhomogeneous representation by

$$\mathbf{x} = \begin{bmatrix} x'/z' \\ y'/z' \end{bmatrix}. \tag{A.2}$$

As multiple representations for the same point exist, a common convention, that we follow in this thesis, is to define $z = 1$. Thus, we write $\begin{bmatrix} \mathbf{x}^\mathsf{T} & 1 \end{bmatrix}^\mathsf{T}$ to indicate a homogeneous representation of the point $\mathbf{x}$.

One particularly useful application of homogeneous coordinates is to express rigid transformations as a linear operation. A rigid transform f, with

$$\mathbf{y} = f(\mathbf{x}) := R\,\mathbf{x} + \mathbf{b}, \tag{A.3}$$

where $\mathbf{b} \in \mathbb{R}^2$ and $R \in SO(2)$, can be formulated conveniently by

$$\tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} R & \mathbf{b} \\ \mathbf{0}^\top & 1 \end{bmatrix}}_{=:T} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}. \tag{A.4}$$

The matrix T is referred to as a *homogeneous transformation matrix*.

In general, all projective transformations (also called *homographies*) can be expressed in matrix notation using homogeneous coordinates $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ by

$$\tilde{\mathbf{y}} = \underbrace{\begin{bmatrix} A & \mathbf{b} \\ \mathbf{u}^\top & v \end{bmatrix}}_{=:H} \tilde{\mathbf{x}}, \tag{A.5}$$

where $A \in \mathbb{R}^{2\times 2}$, $\mathbf{u} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^\top$ and $u_1, u_2, v \in \mathbb{R}$. Note that multiplying H by a scaling factor $\lambda \in \mathbb{R} \setminus 0$ yields the same transformation as $\lambda\,\tilde{\mathbf{y}} = \lambda\,H\,\tilde{\mathbf{x}}$ represents the same points $\mathbf{y}$ for all non-zero λ. In Chapter 3 we make use of homographies to express relationships between two images.

The principle of homogeneous representations and transform matrices for points in $\mathbb{R}^2$ is easily extended to higher dimensions. This especially includes the case of points in $\mathbb{R}^3$, which can be represented by equivalence classes of four-dimensional points.

A.2 Pinhole Camera Model

The *pinhole camera model* describes the projection of a point from the 3D-space into the camera image. More specifically, we define the camera's *intrinsics* or *calibration matrix*

$$K := \begin{bmatrix} f_x & 0 & k_x \\ 0 & f_y & k_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{A.6}$$

where f_x, f_y, k_x, and k_y are the camera's intrinsic parameters. With this definition and using homogeneous coordinates, we can express the projection of a point from the 3D-space into image space by a matrix multiplication as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{A.7}$$

The same relationship can be expressed using inhomogeneous coordinates as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \frac{x}{z} + c_x \\ f_y \frac{y}{z} + c_y \end{bmatrix}. \tag{A.8}$$

So far, we assumed that the point in 3D-space is given relative to the camera. However, the point is often described in a different coordinate frame, sometimes called *world frame*. Assuming that the camera's center is located at $\mathbf{c}$ and the camera's orientation is given by $R \in SO(3)$ relative to the world frame, we can express the image of a point $\tilde{\mathbf{x}}$ in homogeneous world-coordinates by

$$\tilde{\mathbf{y}} = K\,[R \mid t]\,\tilde{\mathbf{x}}, \tag{A.9}$$

where $\mathbf{t} = -R\,\mathbf{c}$. The matrix $[R \mid t] \in \mathbb{R}^{3\times 4}$ is often referred to as the *extrinsic matrix*.

The projection in (A.7) and (A.8) assumes a perfect rectilinear camera, where straight lines remain straight. In practice, cameras use imperfect lenses which introduce distortion in the image. To model distorted image coordinates, we use a mapping that assumes that the distortion correlates with the square of the distance to the center:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} (1+\kappa r^2)\,x \\ (1+\kappa r^2)\,y \\ z \end{bmatrix}, \text{where } r^2 = \frac{x}{z}^2 + \frac{y}{z}^2. \tag{A.10}$$

Here, $\kappa \in \mathbb{R}$ models the amount of radial distortion, or the deviation from the rectilinear camera. For $\kappa \to 0$, the model is identical with (A.7). This mapping can also be expressed in inhomogeneous coordinates by the function

$$\begin{aligned} \mathsf{proj} : \mathbb{R}^3 &\to \mathbb{R}^2 \\ \begin{bmatrix} x \\ y \\ z \end{bmatrix} &\mapsto \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} K \begin{bmatrix} (1+\kappa r^2)\,\frac{x}{z} \\ (1+\kappa r^2)\,\frac{y}{z} \\ 1 \end{bmatrix}. \end{aligned} \tag{A.11}$$

The latter is used often throughout this thesis. In Chapter 2, we present a method to determine the parameters f_x, f_y, k_x, k_y, and κ.

A.3 Forward Kinematics

Often, there is the need to express one coordinate frames of the robot relative to another, for instance the end-effector frame relative to the base frame. The actual transform depends on the kinematic structure of the robot and the position of the joints along the kinematic chain. Determining the transform is called *forward kinematics*. We briefly describe this process, mostly following the notation by Siciliano et al. (2010, Chapter 2).

Consider two coordinate frames B and E on the robot that are connected by a kinematic chain consisting of n joints referred to as $\left(j_1, j_2, \ldots, j_n\right)$ and $n+1$ links, referred to as $\left(L_0, \ldots, L_n\right)$. Here, link L_i connects the joints j_i and j_{i+1}. Furthermore, link L_0 connects the fixed robot frame B with the first joint j_1, and L_n connect the last joint j_n with the fixed frame E, respectively. Their transforms are described by the constant homogeneous transform matrices T_0^B and T_E^n. Then, the forward kinematics are given by

$$\mathcal{F}_E^B(\boldsymbol{\theta}, \hat{\mathbf{q}}) \coloneqq T_0^B \left(\prod_{i=1}^{n} A_i^{i-1}(q_i) \right) T_E^n, \tag{A.12}$$

where $A_i^{i-1}(q_i)$ is also a homogeneous 4-by-4 transformation matrix describing the transform between the joints j_{i-1} and j_i. The transform depends on the position q_i of joint j_i. The position q_i is given by the corresponding encoder reading $\hat{q}_{j_i} \in \hat{\mathbf{q}}$ and, if available, a corresponding joint offset $q_{j_i}^{\text{off}} \in \boldsymbol{\theta}$ from the calibration parameters (see Chapter 2). In this case, $q_i = \hat{q}_{j_i} + q_{j_i}^{\text{off}}$, otherwise $q_i = \hat{q}_{j_i}$. The transform matrices can then be expressed by

$$A_i^{i-1}(q_i) = \begin{bmatrix} R_i^{i-1}(q_i) & \mathbf{o}_i^{i-1} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{A.13}$$

where $R_i^{i-1}(q_i)$ is a 3-by-3 rotation matrix describing the rotation around the rotation axis of j_i by q_i, and $\mathbf{o}_i^{i-1}$ is the origin of the joint j_i, with respect to j_{i-1}. Similarly, we can obtain the transform matrices T that do not depend on a joint position. A method for obtaining the transform matrices with application to robotics is described in the classical work by Denavit and Hartenberg (1955).

As one application, the forward kinematic function can be used to express the coordinates of a point $\mathbf{p}_E$ given in the frame E in coordinates of the frame B by

$$\tilde{\mathbf{p}}_B = \mathcal{F}_E^B(\boldsymbol{\theta}, \hat{\mathbf{q}})\, \tilde{\mathbf{p}}_E. \tag{A.14}$$

Appendix B

The Humanoid Robot Nao

This section describes the humanoid robot Nao that we employed during all the experiments performed throughout this thesis. The robot is manufactured by Aldebaran Robotics and available in different versions. In our work, we use a fourth generation (V4) robot, officially entitled *Nao Next Gen*.

The robot is powered by a single-core 1.6 GHz Intel Atom processor for high-level computation such as the walking engine and an embedded ARM CPU for low-level processing, i.e., the communication with the servos. Nao is 58 cm in height, weights 4.8 kg and has 25 degrees of freedom. Its kinematic structure is outlined in Figure B.1. As can be seen, each arm has five joints and each leg six joints. However, the HipYawPitch joint is a mimic one, i.e., there is only one servo controlling LHipYawPitch and RHipYawPitch, and thus, the legs only have eleven degrees of freedom in total. Furthermore, the robot has two joints in the neck and one cable-driven servo in each hand to allow opening and closing of the grippers. The arms have a lenght of circa 23 cm from the ShoulderPitch joints to the grippers, and the legs have a length of circa 29 cm from the HipYawPitch joint to the AnkleRoll joints.

The joint positions are measured using magnetic rotary encoders, that exploit the Hall-effect. Thereby, the sensors have a precision of approximately 0.1°. However, the accuracy of the sensors is not known and depend on the manufacturing and wear. In Chapter 2 we present an approach to estimate the offsets of the joints.

Additionally, the robot is equipped with an Inertial Measurement Unit (IMU) mounted in its chest. It consists of a two-axis gyroscope and a three-axis accelerometer. Their measurements are fused with a Kalman Filter to yield the robot's orientation. However, the yaw angle (rotation about the axis of gravity) cannot be estimated, as the gyroscope lacks the corresponding axis.

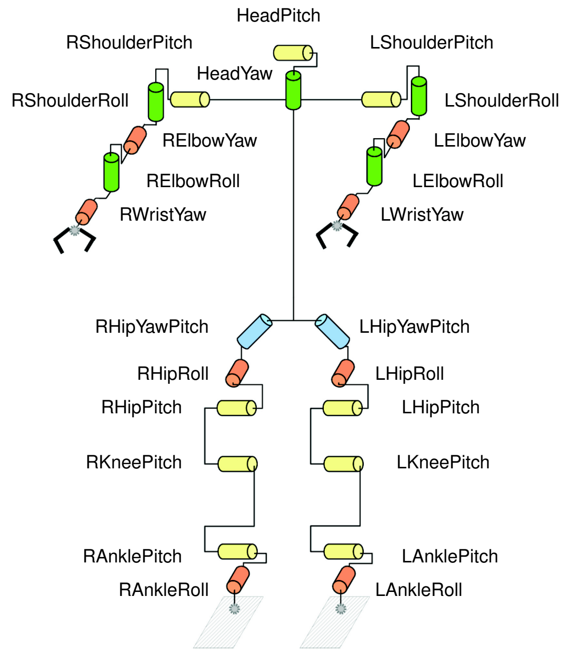
The robot features two monocular cameras integrated in its head. One is parallel to the HeadPitch joint, the second one is mounted circa 4.6 cm below and faces downwards at an angle of 40°. Each camera has a field of view of 60.9° horizontally and 47.6° vertically. Given the camera poses, it is clear, that the overlap between both cameras is very narrow and thus, they cannot efficiently be used as stereo camera pair. The cameras output images with a resolution of 640×480 pixel at 30 Hz. However, as the data needs to be (de-)serialized and transferred between different processes while the limited processing power is shared with other operations, the practically achievable frame rate is much slower.

The robot comes with an API called *NaoQI*, which we used in version 1.12 and 1.14. The API allows access to the sensors and actuators, and to generate trajectories for the end-effectors from a sequence of control points in joint space using Bezier interpolation. Furthermore, the API features a omni-directional walking engine (Gouaillier et al., 2010), which we used throughout this work to control the walking behavior of the robot. With the current firmware, the robot is able to walk up to 10 cm/s. In Section 5.4 we illustrate the set of possible footsteps it can perform with the given API, as well as our extensions that allow the robot to step over or onto obstacles.

Additionally, when it became available, we mounted an Asus Xtion Pro Live RGB-D sensor to the robot's head (see Figure B.1). The camera has a field of view of 58° horizontally and 45° vertically. It is attached on the robot's head in a way such that its optical axis faces the floor at an angle between 30° and 50° while walking. The value is adjustable. The sensor outputs RGB-D images, i.e., pairs of RGB- and depth images, in a resolution of 640×480 pixel at 30 Hz. Like the popular Microsoft Kinect device, the senor generates depth images by observing a known projected infrared pattern with its infrared camera. The baseline between projector and camera is approximately 7.5 cm, and thus, there is a near range clipping at circa 50 cm. Furthermore, the measurements become unusable after 5 m. The error is assumed to grow quadratically with the distance to the sensor (Khoshelham and Oude Elberink, 2012).

B.1 Odometry Computation From Forward Kinematics

For a humanoid robot, an odometry estimate can be obtained via forward kinematics. To estimate the robot's torso pose $\hat{\mathbf{x}} = \begin{bmatrix} x & y & z & \varphi & \theta & \psi \end{bmatrix}^\top$, we keep record of the accumulated transform to the current stance foot, starting with the identity, and assume it to be fixed while the swing foot moves. Using forward kinematics, the poses of all the robot's joints and links, including its sensors like monocular or depth camera, can be computed relative to the stance foot for an arbitrary time t. We denote this transform by $\hat{\mathbf{x}}_t$. The transform to the stance foot is updated whenever the swing foot becomes the new stance foot by concatenating on the accumulated transform, the relative transform from the previous stance foot to the new one. In this way, we are able to estimate the robot's



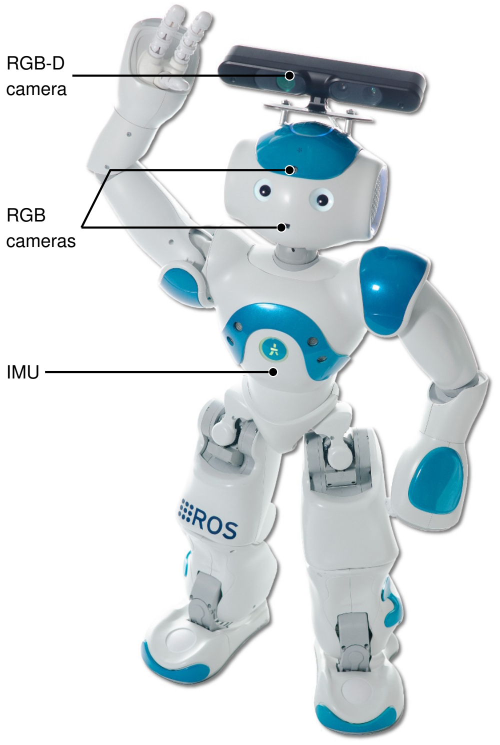


Figure B.1: Left: Kinematic structure including joint names of the Nao humanoid robot. The image is based on a figure by Gouaillier et al. (2010). Right: Nao with a head-mounted RGB-D camera and location of the main sensors used in this thesis. Not shown are the joint encoders.

pose even if the height changes, e.g., when the robot steps onto objects. To compensate for small errors in the joint encoder readings or slightly inclined terrain, our system uses the pitch and roll measurements from the onboard IMU. Consequently, the algorithm rotates the pose estimate around the current stance foot such that the reference frame of the IMU, as computed by forward kinematics, aligns with the IMU measurements. The IMU is typically installed in the humanoid's chest or head. This method is sensitive to drift as errors accumulate, e.g., from slipping of the feet or noisy encoder measurements. In Section 5.1 we therefore present a technique that reduces the drift by employing also depth camera readings.

List of Figures

List of Tables

List of Algorithms

Bibliography

M. Agrawal and L. Davis. Camera calibration using spheres: a semi-definite programming approach. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2003.

N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transfom. *IEEE Transactions on Computers*, 23(1):90–93, 1974.

H. Alvarez, L. M. Paz, J. Sturm, and D. Cremers. Collision avoidance for quadrotors with a monocular camera. In *Proc. of the Int. Symp. on Experimental Robotics (ISER)*, 2014.

T. Asfour, K. Welke, P. Azad, A. Ude, and R. Dillmann. The karlsruhe humanoid head. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2008.

F. Basso, A. Pretto, and E. Menegatti. Unsupervised intrinsic and extrinsic calibration of a camera-depth sensor couple. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014.

A. M. Batula and Y. Kim. Development of a mini-humanoid pianist. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2010.

L. Baudouin, N. Perrin, T. Moulard, F. Lamiraux, O. Stasse, and E. Yoshida. Real-time replanning using 3d environment for humanoid robot. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2011.

H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2006.

D. Bennett and J. Hollerbach. Autonomous calibration of single-loop closed kinematic chains formed by manipulators with passive endpoint constraints. *Robotics and Autonomous Systems*, 1991.

M. Bennewitz, D. Maier, A. Hornung, and C. Stachniss. Integrated perception and navigation in complex indoor environments. In *Proc. of the Humanoids 2011 Workshop on Humanoid Service Robot Navigation in Crowded and Dynamic Environments*, 2011.

J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9), 1975.

O. Birbach, B. Bäuml, and U. Frese. Automatic and self-contained calibration of a multisensorial humanoids upper body. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

J. Biswas and M. Veloso. Depth camera based localization and navigation for indoor mobile robots. In *RSS 2011 Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2011.

J. Biswas and M. Veloso. Depth camera based indoor mobile robot localization and navigation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

J.-H. Borm and C.-H. Menq. Experimental study of observability of parameter errors in robot calibration. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1989.

G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, 2008.

C. Braillon, C. Pradalier, J. Crowley, and C. Laugier. Real-time moving obstacle detection using optical flow models. In *IEEE Intelligent Vehicles Symp.*, 2006.

S. R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. `http://euclid.ucsd.edu/~sbuss/ResearchWeb/ikmethods/index.html`, 2004.

E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *Proc. of Robotics: Science and Systems (RSS)*, 2013.

J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

H. Carrillo, O. Birbach, H. Taubig, B. Bauml, U. Frese, and J. Castellanos. On task-oriented criteria for configurations selection in robot calibration. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

F. Chaumette and E. Marchand. A redundancy-based iterative approach for avoiding joint limits: Application to visual servoing. *IEEE Trans. on Robotics and Automation*, 17 (5), 2001.

A. Cherubini and F. Chaumette. Visual navigation of a mobile robot with laser-based collision avoidance. *Int. Journal of Robotics Research (IJRR)*, 32(2), 2013.

J. Chestnutt, M. Lau, K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade. Footstep planning for the Honda ASIMO humanoid. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2005.

J. Chestnutt, Y. Takaoka, K. Suga, K. Nishiwaki, J. Kuffner, and S. Kagami. Biped navigation in rough environments using on-board sensing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

K. Chida, I. Okuma, S. Isoda, Y. Saisu, K. Wakamatsu, K. Nishikawa, J. Solis, H. Takanobu, and A. Takanishi. Development of a new anthropomorphic flutist robot WF-4. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2004.

C. Choi and H. I. Christensen. Robust 3d visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features. *Int. Journal of Robotics Research (IJRR)*, 31(4), 2012.

J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics Computation*, 1965.

R. Cupec, G. Schmidt, and O. Lorch. Experiments in vision-guided robot walking in a structured scenario. In *IEEE Int. Symp. on Industrial Electronics*, 2005.

H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski. Self-supervised monocular road detection in desert terrain. In *Proc. of Robotics: Science and Systems (RSS)*, 2006.

D. Daney. Optimal measurement configurations for gough platform calibration. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2002.

F. Dayoub, G. Cielniak, and T. Duckett. Long-term experiments with an adaptive spherical view representation for navigation in changing environments. *Robotics and Autonomous Systems*, 59(5), 2011.

J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *ASME Trans., J. of Applied Mechanics*, 22(2), 1955.

T. Drummond, I. C. Society, and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(27), 2002.

E. Einhorn, C. Schröter, and H.-M. Gross. Monocular scene reconstruction for reliable obstacle detection and robot navigation. In *Proc. of the European Conf. on Mobile Robots (ECMR)*, 2009.

F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

C. Erdogan and M. Stilman. Planning in constraint space: Automated design of functional structures. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

B. Espiau. Effect of camera calibration errors on visual servoing in robotics. In *Proc. of the Int. Symp. on Experimental Robotics (ISER)*, 1993.

F. Faber, M. Bennewitz, C. Eppner, A. Goeroeg, A. Gonsior, D. Joho, M. Schreiber, and S. Behnke. The humanoid museum tour guide Robotinho. In *18th IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2009.

M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 1981.

I. Fujimoto, T. Matsumoto, P. R. S. De Silva, M. Kobayashi, and M. Higashi. Study on an assistive robot for improving imitation skill of children with autism. In *Proc. of the Int. Conf. on Social Robotics (ICSR)*, 2010.

D. Gonzalez-Aguirre, M. Vollert, T. Asfour, and R. Dillmann. Robust real-time 6d active visual localization for humanoid robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014.

D. Gouaillier, C. Collette, and K. C. Omni-directional closed-loop walk for nao. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2010.

M. Greenspan and I. Fraser. Tracking a sphere dipole. In *Proc. of the Int. Conf. on Vision Interface (VI)*, 2003.

J.-S. Gutmann, M. Fukuchi, and M. Fujita. Real-time path planning for humanoid robot navigation. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2005.

J.-S. Gutmann, M. Fukuchi, and M. Fujita. 3D perception and environment map generation for humanoid robot navigation. *Int. Journal of Robotics Research (IJRR)*, 27(10): 1117–1134, 2008.

P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Proc. of the IEEE Int. Conf. on Systems Science and Cybernetics*, 4(2), 1968.

R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

K. Hauser, T. Bretl, and J.-C. Latombe. Non-gaited humanoid locomotion planning. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2005.

P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *Int. Journal of Robotics Research (IJRR)*, 31(5):647–663, 2012.

J. M. Hollerbach and C. W. Wampler. The calibration index and taxonomy for robot kinematic calibration methods. *Int. Journal of Robotics Research (IJRR)*, 1996.

A. Hornung, K. M. Wurm, and M. Bennewitz. Humanoid robot localization in complex indoor environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz. Anytime search-based footstep planning with suboptimality bounds. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2012a.

A. Hornung, M. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, and S. Chitta. Navigation in three-dimensional cluttered environments for mobile manipulation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012b.

A. Hornung, D. Maier, and M. Bennewitz. Search-based footstep planning. In *Proc. of the ICRA Workshop on Progress and Open Problems in Motion Planning and Navigation for Humanoids*, 2013a.

A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34, 2013b.

A. Hornung, S. BÖttcher, J. Schlagenhauf, C. Dornhege, A. Hertle, and M. Bennewitz. Mobile manipulation in cluttered environments with humanoids: Integrated perception, task planning, and action execution. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2014a.

A. Hornung, S. Osswald, D. Maier, and M. Bennewitz. Monte Carlo localization for humanoid robot navigation in complex indoor environments. *Int. Journal of Humanoid Robots (IJHR)*, 11(2), 2014b.

A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual odometry and mapping for autonomous flight using an RGB-D camera. In *Proc. of the Int. Symp. of Robotics Research (ISRR)*, 2011.

U. Hubert, J. Stückler, and S. Behnke. Bayesian calibration of the hand-eye kinematics of an anthropomorphic robot. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2012.

K. Khoshelham and S. Oude Elberink. Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors: Journal on the Science and Technology of Sensors and Biosensors*, 12:1437–1454, 2012.

D. Kim, J. Sun, S. M. Oh, J. M. Rehg, and A. F. Bobick. Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2006.

Y.-G. Kim and H. Kim. Layered ground floor detection for vision-based mobile robot navigation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2004.

G. Klein and D. Murray. Full-3d edge tracking with a particle filter. In *Proc. of the British Machine Vision Conf. (BMVC)*, 2006.

G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. of the IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR)*, 2007.

H. Kretzschmar, G. Grisetti, and C. Stachniss. Lifelong map learning for graph-based SLAM in static environments. *KI – Künstliche Intelligenz*, 24, 2010.

J. Kuffner Jr, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Footstep planning among obstacles for biped robots. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2001.

R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard. Monte Carlo localization in outdoor terrains using multilevel surface maps. *Journal of Field Robotics (JFR)*, 25:346–359, 2008.

R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

V. Lepetit and P. Fua. Monocular model-based 3D tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, 2005.

T. Li, K. Sun, Y. Jin, and H. Liu. A novel optimal calibration algorithm on a dexterous 6 dof serial robot-with the optimization of measurement poses number. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011a.

T. Li, K. Sun, Z.-w. Xie, and H. Liu. Optimal measurement configurations for kinematic calibration of six-DOF serial robot. *Journal of Central South University of Technology*, 18, 2011b.

X. Li, S. Zhang, and M. Sridharan. Vision-based safe local motion on a humanoid robot. In *Workshop on Humanoid Soccer Robots*, 2009.

M. Likhachev, G. Gordon, and S. Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 2004.

T. Low and G. Wyeth. Obstacle detection using optical flow. In *Aust. Conf. on Robotics and Automation (ACRA)*, 2005.

M. Luber, G. D. Tipaldi, and K. O. Arras. Place-dependent people tracking. *Int. Journal of Robotics Research (IJRR)*, 30(3), 2011.

D. Maier and M. Bennewitz. Appearance-based traversability classification in monocular images using iterative ground plane estimation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

D. Maier, A. Hornung, and M. Bennewitz. Real-time navigation in 3D environments based on depth camera data. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2012.

D. Maier, C. Lutz, and M. Bennewitz. Integrated perception, mapping, and footstep planning for humanoid navigation among 3d obstacles. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013a.

D. Maier, C. Lutz, and M. Bennewitz. Autonomous biped navigation through clutter. In *Proc. of the RSS Workshop on Robots in Clutter: Preparing Robots for the Real World*, 2013b.

D. Maier, C. Stachniss, and M. Bennewitz. Vision-based humanoid navigation using self-supervised obstacle detection. *International Journal of Humanoid Robots*, 10(2), 2013c.

D. Maier, R. Zohouri, and M. Bennewitz. Using visual and auditory feedback for instrument-playing humanoids. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2014.

D. Maier, S. Wrobel, and M. Bennewitz. Whole-body self-calibration via graph-optimization and automatic configuration selection. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015. To appear.

P.-A. Martínez, D. Varas, M. Castelán, M. Camacho, M. F., and A. G. 3D shape reconstruction from a humanoid generated video sequence. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2014.

P. Michel, J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner, and T. Kanade. GPU-accelerated real-time 3D tracking for humanoid locomotion and stair climbing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

J. Michels, A. Saxena, and A. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Int. Conf. on Machine Learning (ICML)*, 2005.

T. Mizumoto, H. Tsujino, T. Takahashi, T. Ogata, and H. Okuno. Thereminist robot: Development of a robot theremin player with feedforward and feedback arm control based on a theremin's pitch model. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2), 1988.

H. P. Moravec and A. E. Elfes. High resolution maps from wide angle sonar. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1985.

A. Nahvi and J. Hollerbach. The noise amplification index for optimal pose selection in robot calibration. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1996.

A. Nakhaei and F. Lamiraux. Motion planning for humanoid robots in environments modeled by vision. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2008.

R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc. of the IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR)*, 2011a.

R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2011b.

K. Nishiwaki, J. Chestnutt, and S. Kagami. Autonomous navigation of a humanoid robot over unknown rough terrain using a laser range sensor. *Int. Journal of Robotics Research (IJRR)*, 2012.

L. Ott and F. Ramos. Unsupervised incremental learning for long-term autonomy. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

R. Ozawa, Y. Takaoka, Y. Kida, K. Nishiwaki, J. Chestnutt, J. Kuffner, S. Kagami, H. Mizoguchi, and H. Inoue. Using visual odometry to create 3d maps for online footstep planning. In *IEEE Intl. Conf. on Systems, Man, and Cybernetics*, 2005.

J. Pan, S. Chitta, and D. Manocha. FCL: A general purpose library for collision and proximity queries. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

I.-W. Park, B.-J. Lee, S.-H. Cho, Y.-D. Hong, and J.-H. Kim. Laser-based kinematic calibration of robot manipulator using differential kinematics. *IEEE/ASME Trans. on Mechatronics*, 2012.

N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida. Fast humanoid robot collision-free footstep planning using swept volume approximations. *IEEE Transactions on Robotics (T-RO)*, 28(2), 2012a.

N. Perrin, O. Stasse, F. Lamiraux, Y. J. Kim, and D. Manocha. Real-time footstep planning for humanoid robots among 3D obstacles using a hybrid bounding box. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012b.

P. Pfaff, R. Triebel, and W. Burgard. An efficient extension of elevation maps for outdoor terrain mapping and loop closing. *Int. Journal of Robotics Research (IJRR)*, 2007.

C. Plagemann, C. Stachniss, J. Hess, F. Endres, and N. Franklin. A nonparametric learning approach to range sensing from omnidirectional vision. *Robotics and Autonomous Systems*, 58:762–772, 2010.

V. Pradeep, K. Konolige, and E. Berger. Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach. In *Proc. of the Int. Symp. on Experimental Robotics (ISER)*, 2010.

A. Pretto, E. Menegatti, M. Bennewitz, W. Burgard, and E. Pagello. A visual odometry framework robust to motion blur. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2009.

M. Pupilli and A. Calway. Real-time camera tracking using known 3d models and a particle filter. In *Proc. of the Int. Conf. on Pattern Recognition (ICPR*, 2006.

O. Ramos, M. Garcia, N. Mansard, O. Stasse, J.-B. Hayet, and P. Soueres. Towards reactive vision-guided walking on rough terrain: an inverse-dynamics based approach. *Int. Journal of Humanoid Robots (IJHR)*, 2014.

D. Ricks and M. Colton. Trends and considerations in robot-assisted autism therapy. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2010.

A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEEE Trans. Systems. Man. Cybernet*, 6(6):420–433, 1976.

S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert. Learning monocular reactive uav control in cluttered natural environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

G. A. F. Seber and C. J. Wild. *Nonlinear Regression*. Wiley-Interscience, 2003.

A. Segal, D. Hähnel, and S. Thrun. Generalized-ICP. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 2010.

R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, 1987.

C. Stachniss, M. Bennewitz, G. Grisetti, S. Behnke, and W. Burgard. How to learn accurate grid maps with a humanoid. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008.

M. Stilman, K. Nishiwaki, S. Kagami, and J. Kuffner. Planning and executing navigation among movable obstacles. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

Y. Sun and J. M. Hollerbach. Observability index selection for robot calibration. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008.

N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *European Conference on Computer Vision (ECCV)*, Crete, Greece, 2010.

R. Tellez, F. Ferro, D. Mora, D. Pinyol, and D. Faconti. Autonomous humanoid navigation using laser and odometry data. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2008.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT-Press, 2005.

A. Traslosheros, J. Sebastian, E. Castillo, F. Roberti, and R. Carelli. One camera in hand for kinematic calibration of a parallel robot. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

I. Ulrich and I. Nourbakhsh. Appearance-based obstacle detection with monocular color vision. In *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, 2000.

N. Vahrenkamp, T. Asfour, and R. Dillmann. Robot placement based on reachability inversion. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

H. Wang, K. Yuan, W. Zou, and Y. Peng. Real-time obstacle detection with a single camera. In *Proc. of the IEEE Int. Conf. on Industrial Technology (ICIT)*, 2005.

G. Weinberg, T. Mallikarjuna, and A. Ramen. Interactive jamming with Shimon: A social robotic musician. In *Proc. of the Int. Conf. on Human Robot Interaction (HRI)*, 2009.

T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012a.

T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended kinectfusion. In *RSS 2012 Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012b.

K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.

K. Yamane. Practical kinematic and dynamic calibration methods for force-controlled humanoid robots. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2011.

Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 1999.

J. Zhou and B. Li. Robust ground plane detection with normalized homography in monocular sequences from a robot platform. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP)*, 2006.