Motion Imitation and Generation for Mobile Robotic Systems

Felix Burget

Technische Fakultät
Albert-Ludwigs-Universität Freiburg

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Betreuer: Prof. Dr. Wolfram Burgard

Motion Imitation and Generation for Mobile Robotic Systems

Felix Burget

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

Dekan	Prof. Dr. Oliver Paul
Erstgutachter	Prof. Dr. Wolfram Burgard
	Albert-Ludwigs-Universität Freiburg
Zweitgutachter	Prof. Dr. Maren Bennewitz
	Rheinische Friedrich-Wilhelms-Universität Bonn
Tag der Disputation	10. Juli 2018

Zusammenfassung

Im industriellen Umfeld gelten stationäre Robotersysteme als zuverlässige, präzise und hochproduktive Maschinen, die die aufgetragenen Arbeiten mit konstanter Qualität ausführen. Typischerweise liegt in diesem Einsatzgebiet eine strukturierte Umgebung vor. Außerdem ist der Arbeitsbereich des Roboters genau definiert und von dem Arbeitsraum der menschlichen Mitarbeiter abgetrennt. Darüber hinaus sind die Aufgaben, die dem Roboter erteilt werden, üblicherweise vorab definiert.

Mit der Einführung mobiler Roboterplattformen hat sich dieses Bild eines Roboterstereotyps in den letzten Jahrzehnten zunehmend verändert. Mobilität hat hier zu einer enormen Zunahme an potenziellen Anwendungen für Robotersysteme geführt, die über den Bereich der industriellen Automatisierung hinausgehen. In Folge dessen ist deren Einsatz heutzutage auch im häuslichen Bereich und in Katastrophenschutzszenarien denkbar geworden. Die Kombination mobiler Plattformen mit klassischen industriellen Roboterarmen führte zu agilen Robotereinheiten, die sowohl Navigations- als auch Manipulationsfähigkeiten besitzen und in der Robotik als sogenannte *mobile Manipulatoren* bezeichnet werden. Die durch den mobilen Unterbau realisierte Erweiterung des Arbeitsbereichs des Roboterarms führte zugleich zu gesteigerten Erwartungen die an diese Systeme gestellt werden. So sollen diese heute in vielen verschiedenen Bereichen arbeiten können, in anspruchsvollen Umgebungen zurecht kommen und flexibel bezüglich der an sie gestellten Aufgaben sein.

Die Nutzung der besonderen Fähigkeit dieser Roboterplattformen, in der Umgebung navigieren und simultan Objekte manipulieren zu können bringt jedoch auch zahlreiche Herausforderungen mit sich. Die erste Schwierigkeit besteht darin, dass diese Systeme eine hohe Anzahl von Freiheitsgraden besitzen, was die Erstellung effizienter Bewegungsplanungs- und Regelungsstrategien erheblich komplexer macht. Klassische Ansätze aus der Literatur folgen deshalb häufig der Strategie, komplexe mobile Manipulationsaufgaben in einfachere Teilaufgaben, die lediglich Navigation mit der mobilen Basis oder Manipulation mit dem Roboterarm erfordern zu zerlegen. Anschließend werden diese Teilprobleme sequentiell und unabhängig voneinander gelöst. Obwohl diese *Divide-and-Conquer* Strategie für viele Anwendungen zulässig ist, wird dieses Vorgehen in keiner Weise dem Potential solcher agilen Robotersysteme gerecht. Zudem gestatten diese Ansätze nicht, Aufgaben auszuführen, bei denen eine koordinierte Bewegung von mobiler Basis und Roboterarm zwingend erforderlich ist. Ein weiterer wichtiger Aspekt in Bezug auf mobile Manipulationsaufgaben sind Anzahl und Art der Bedingungen, die der Bewegung des Roboters auferlegt werden. Letztere können weiter in Bedingungen unterteilt werden, die mit der kinematischen Struktur des Roboters zusammenhängen und in solche, die von den zu manipulierenden Objekten auferlegt werden. Neben allgemeinen kinematischen Randbedingungen wie Gelenkgrenzen und Kollisionsvermeidung gibt es weitere plattformspezifische Bedingungen, die sich auf das Bewegungsmodell der mobilen Basis beziehen. Radroboterplattformen, zum Beispiel, weisen eine kinematische Struktur auf, deren Bewegung entweder durch ein *Omnidirektional-*, *Differentialantrieb-* oder *Achsschenkellenkung*-Bewegungsmodell beschrieben werden kann. Robotern mit omnidirek-

tionalen Navigationsfähigkeiten ist es möglich, sich jederzeit in beliebige Richtungen zu bewegen. Bei letzteren genannten Modellen hingegen, ist eine instantane Bewegung in eine orthogonal zur Sagittalachse des Roboters verlaufenden Richtung nicht möglich. Humanoide Roboterplattformen besitzen naturgemäß omnidirektionale Mobilität, für deren Navigation jedoch besondere Bedingungen gelten. Um zu laufen, muss bei diesen Robotern die Schleife, die durch die kinematischen Ketten der Beine mit dem Untergrund gebildet werden wiederholt geschlossen werden. Außerdem muss der Körpermassenschwerpunkt kontinuierlich geregelt werden, um zu vermeiden, dass der Roboter die Balance verliert und umfällt. Weitere Bedingungen werden aktiv, sobald der Roboter mit einem zu manipulierenden Objekt in Kontakt kommt. In diesem Fall müssen nicht nur Kollisionen zwischen dem Objekt, dem Roboter und der Umgebung vermieden werden, sondern auch aufgabenbezogene Bedingungen erfüllt werden. Viele Beispiele für mobile Manipulationsaufgaben, die solche Bedingungen beinhalten sind im Bereich der Objektzustellung zu finden. Die Lieferung eines mit Flüssigkeit gefüllten Gefäßes könnte zum Beispiel erfordern, dass das Objekt während des Transports stets aufrecht gehalten wird. Das Ziehen eines Werkzeugwagens von einem Ort zum anderen würde zusätzlich zu einer Einschränkung der Orientierung erfordern, dass der Greifer des Roboters auf Höhe des Griffs des Transportwagens verbleibt. Andere Beispiele ergeben sich aus Manipulationaufgaben für artikulierte Objekte. Das Öffnen von Schubladen, Türen oder Ventilen setzt voraus, dass der Greifer des Roboters einer linearen Bahn bzw. einem Kreisbogen um die Drehachse des Objekts folgt.

In der Forschung wurden bereits einige der oben genannten Herausforderungen behandelt und bedeutende Fortschritte in den Bereichen Roboternavigation und -manipulation erzielt. Trotzdem verfolgten bisher nur wenige Forschungsarbeiten das Ziel, das mit den Fähigkeiten moderner mobiler Manipulatoren einhergehende Potential umfassend auszuschöpfen. Für die nächste Generation mobiler Robotersysteme ist zu erwarten, dass Navigation- und Manipulationaufgaben immer mehr miteinander verschmelzen und somit kinematische als auch aufgabenbezogene Bedingungen in der Bewegungsgenerierung gleichzeitig berücksichtigt werden müssen. Der Hauptbeitrag der vorliegenden Arbeit ist es, mobile Robotersysteme mit Fertigkeiten auszustatten, die es ihnen möglich machen, komplexe mobile Manipulationsaufgaben durchzuführen. Hierbei betrachten wir verschiedene Stufen der Autonomie für den Roboter, beginnend mit der Imitation menschlicher Bewegung hin zur autonomen Bewegungsplanung und -ausführung.

Der Mensch ist aus technischer Sicht ein hervorragendes Beispiel eines mobilen Systems, das in der Lage ist eine große Anzahl von Freiheitsgraden zu kontrollieren und gleichzeitig die Bewegungseinschränkungen, die sich aus Kontakt mit der Umgebung ergeben zu berücksichtigten. Darüber hinaus besitzt der Mensch ein hohes Maß an Intelligenz bezüglich der Interpretation visueller Informationen und des Prozesses der Entscheidungsfindung. Diese Fähigkeiten machen den Wunsch den Menschen als Quelle für die Regelung komplexer künstlicher kinematischer Strukturen miteinzubeziehen offensichtlich. Im Verlaufe des letzten Jahrzehnts sind mehrere erschwingliche, leichte und präzise Bewegungserfassungssysteme auf dem Markt verfügbar geworden, die eine Aufnahme und Untersuchung menschlichen Bewegungsverhaltens sowie die Nutzung von Bewegungsdaten für Roboterapplikationen ermöglichen. Beispiele für solche Systeme sind mechanische Exoskelette, auf Interialsensoren basierende Ganzkörperbewegungserfassungsanzüge und kamerabasierte Bewegungsverfolgungssysteme. Für die Weiterverarbeitung der aus Bewegungsdemonstrationen gewonnen Daten werden typischerweise verschiedene Ansätze verfolgt. Eine erste Möglichkeit besteht darin zunächst Daten mehrerer Demonstrationen zu sammeln und anschließend umfassend zu analysieren um Rückschlüsse auf die zugrundeliegenden Prinzipien menschlicher Bewegung zu ziehen. Alternativ können diese Daten auch genutzt werden, um ein Bewegungsmodell einer Aktion für den Roboter zu lernen. Andere Ansätze verarbeiten die Bewegungsdaten in Echtzeit, um Roboterplattformen direkt anzusteuern. Während die erstgenannten Verfahren darauf abzielen, Roboter mit der Fähigkeit auszustatten, autonom menschenähnliche Bewegungen zu erzeugen, bieten letztere den Vorteil, dass der menschliche Proband fortlaufend Einfluss auf die Bewegung des Roboters nehmen kann. Somit können in jeder Situation angemessene Bewegungskommandos vom Menschen vorgeben werden, was dem Roboter ein hohes Maß an Anpassungsfähigkeit verleiht. Des Weiteren kann das Bild einer auf der Roboterplattform montierten Kamera kontinuierlich zu dem Probanden übertragen werden, um ihm eine Interpretation der aktuellen Lage zu ermöglichen. So kann dieser anhand der visuellen Informationen zum Beispiel den aktuellen Standort des Roboters bestimmen oder zu greifende Objekte auffinden. Zusätzlich ermöglicht dieser Ansatz eine dynamische Anpassung der Roboterbewegung, falls dies aufgrund äußerer Einflüsse erforderlich ist. Auf der anderen Seite gehen mit diesen imitationsbasierten Ansätzen auch eine Menge von Herausforderungen und Einschränkungen einher. Unterscheidet sich zum Beispiel die kinematische Struktur oder Größe des Menschen erheblich von der des Roboters ist eine Abbildung der menschlichen Bewegung auf den Roboter deutlich schwieriger. Wenn die Imitation noch dazu in Echtzeit stattfinden soll ist außerdem nur ein sehr begrenztes Zeitfenster für die Ausführung von Berechnungen verfügbar. Generell gibt es jedoch auch viele Anwendungen, bei denen eine auf dem Menschen basierte Regelung der Roboterbewegung unerwünscht ist. Insbesondere ist dies bei Roboter-Assistenzsystemen der Fall, die als Museumsführer, Haushaltsroboter oder für die Altenpflege eingesetzt werden sollen. In anderen Anwendungsszenarien, bei denen die Kommunikation zwischen Bewegungserfassungssystem und Roboter eine hohe Latenz aufweist oder gänzlich unmöglich ist, sind imitationsbasierte Methoden nur schwierig oder gar nicht umsetzbar. Schwierigkeiten dieser Art treten typischerweise häufig bei dem Einsatz von Roboterplattformen in Katastrophenhilfeszenarien auf.

All diese Szenarien haben gemein, dass sich der Roboter nicht länger auf menschliches Know-How stützen kann und somit viele rudimentäre Fähigkeiten selbst erlernen muss. Eine Grundvoraussetzung, um autonom handeln zu können ist dabei die Fähigkeit seine eigene Position und Orientierung in der Umgebung zu bestimmen. Des Weiteren muss er in der Lage sein, selbständig den Ort zu bestimmen an den er sich begeben muss um eine bestimmte Objektmanipulationsaufgabe vorzubereiten oder erfolgreich abzuschließen. Um dieses Navigationsziel ausgehend von seiner aktuellen Position zu erreichen, muss wiederum ein mit seinem Bewegungsmodell kompatibler Pfad geplant werden. Außerdem muss der Roboter darauf achten, dass die von einem Objekt in seinem Greifer resultierenden Bewegungsbedingungen entlang des geplanten Pfades eingehalten werden. Durch ungenaue Bewegungsausführung kann es zudem zu Abweichungen vom geplanten Weg kommen, die zu einer Verfehlung des Ziels oder im schlimmsten Fall zu einer Kollision mit anderen Objekten in der Umgebung führen können. Aus diesem Grund muss der Roboter seine Pose anhand von Sensordaten kontinuierlich schätzen und seine Bewegung gegebenenfalls anpassen um eine sichere und präzise Ankunft am gewünschten Zielort gewährleisten zu können.

Anhand der beschriebenen Herausforderungen wurden folgenden Fragestellungen aus dem Bereich der Bewegungsimitation und -generierung abgeleitet und im Rahmen der vorliegenden Arbeit beantwortet:

- Wie können menschliche Ganzkörperbewegungen in Echtzeit von einem humanoiden Roboter imitiert werden?
- Wie können motorische Defizite von Menschen mit neurologischen Erkrankungen, z.B. der Parkinson-Krankheit, unter Verwendung von Bewegungsimitationstechniken quantifiziert werden?
- Wie positionieren sich Menschen relativ zu einem Objekt in Vorbereitung einer Manipulationsaktion und wie kann diese Strategie auf mobile Robotersysteme übertragen werden?
- Wie können Bewegungen mobiler Roboter für die Umsetzung komplexer Greif- und Transportaufgaben, unter Berücksichtigung von Kollisionsvermeidung sowie von durch Objekte auferlegte geometrische Bewegungsbedingungen, effizient geplant werden?
- Was sind die Anforderungen und Herausforderungen in der Entwicklung eines Roboter-Assistenzsystems für Nutzer die lediglich über eingeschränkte Kommunikationsfähigkeiten verfügen?

Zunächst stellen wir in dieser Arbeit einen Ansatz vor, der es humanoiden Robotern ermöglicht, menschliche Ganzkörperbewegungen in Echtzeit zu imitieren. Im Gegensatz zu bereits existierenden Ansätzen aus der Literatur berücksichtigt unser Ansatz dabei nicht nur Bewegungen bei denen der Roboter mit beiden Füßen auf dem Boden verbleiben kann, sondern auch die Imitation von Sequenzen die mehrere Standfußwechsel sowie längere Zeitperioden, in denen auf einem Bein balanciert werden muss, beinhalten. Um die Bewegung eines menschlichen Probanden aufzunehmen, nutzen wir einen Ganzkörperanzug, der aus mehreren Interialsensoren besteht und eine drahtlose Übertragung von Bewegungsdaten anbietet. Des Weiteren greifen wir auf eine kompakte Repräsentation des menschlichen Bewegungsmodells zurück um die Echtzeitfähigkeit unseres Systems zu gewährleisten. Jeder von dem Bewegungserfassungssystem zu einem bestimmten Zeitpunkt aufgenommene Datenpunkt entspricht hierbei einer Ganzkörperpose des Menschen und wird zur Imitation wie folgt weiterverarbeitet. In einem ersten Schritt wird die aufgenommene Pose unter Berücksichtigung der Größenunterschiede zwischen den kinematischen Modellen auf den humanoiden Roboter abgebildet. Anschließend wird die abgebildete Körperhaltung in eine statisch stabile Ganzkörperkonfiguration für den Roboter überführt. Dieser Schritt ist erforderlich, um dem Unterschied der beiden Modelle hinsichtlich ihrer Massenverteilung Rechnung zu tragen. Darüber hinaus muss die Bewegung des Robotermassenschwerpunktes aktiv geregelt werden, um eine sichere Transition zwischen Posen mit ein- und zwei auf dem Boden aufliegenden Fußflächen sicher zu stellen. Ein Schwachstelle imitationsbasierter Ansätze ist jedoch, dass die Qualität der vom Menschen vorgegebenen Bewegung nicht in Frage gestellt wird. Somit haben suboptimale menschliche Bewegungsmuster ein entsprechend schlechtes Bewegungsverhalten des Roboters zur Folge.

Um diese Problematik zu behandeln stellen wir als nächstes einen Ansatz vor, der dem menschlichen Bewegungsverhalten zugrundeliegende Prinzipien untersucht und quantitativ evaluiert. Dabei stützen wir uns wie zuvor auf Bewegungsdaten von menschlichen Aufgabendemonstrationen, die mittels eines Ganzkörperanzugs aufgezeichnet wurden. Die Grundlage für die vorgestellte Untersuchung bildet eine Handkoordinationsaufgabe, die ausschließlich Bewegungen des Oberkörpers erfordert. Als Referenzgruppe betrachten wir Probanden,

deren Krankheitsvorgeschichte nachweislich keinerlei Erkrankungen beinhaltet, die das Bewegungsverhalten beeinflussen. Die Vergleichsgruppe hingegen wird durch Parkinson-Patienten (PD-Patienten) repräsentiert, deren neurologische Beeinträchtigungen bekanntermaßen mit motorischen Bewegungsdefiziten einhergehen. Ziel unseres Ansatzes ist es, die aufgenommenen Bewegungen quantitativ auszuwerten und daraus gruppenspezifische Bewegungsstrategien abzuleiten. Zur Umsetzung verfolgen wir einen zweistufigen Ansatz. Zuerst übertragen wir die aufgenommenen Bewegungen auf ein vereinfachtes virtuelles kinematisches Modell des menschlichen Oberkörpers. Danach wird dieses Modell verwendet, um mittels eines *Jacobischen* Regelalgorithmus die vorgegebene menschliche Bewegung zu reproduzieren. Zur Parametrisierung des Reglers wird hierbei eine Gewichtung der individuellen Robotergelenkbewegungen eingesetzt. Zur Bestimmung der zur Reproduktion der Bewegung erforderlichen Gelenkgewichte wurde ein iteratives Lernverfahren entwickelt, welches die aufgenommenen menschlichen Gelenktrajektorien als Referenz verwendet. Abschließend werden die Datensätze gelernter Gelenkgewichte für die beiden Gruppen statistisch ausgewertet und mit ihnen assoziierte Bewegungsstrategien abgeleitet.

Als nächstes stellen wir einen Ansatz vor, der es mobilen Roboterplattformen ermöglicht, eine optimale Standpose zur Vorbereitung oder Ausführung einer nachfolgenden Manipulationsaufgabe autonom auszuwählen. Um bei der Auswahl einer angemessenen Standpose nicht weiter auf den Menschen angewiesen zu sein, beschreiben wir in dieser Arbeit die Greif- und Manipulationsfähigkeiten des Roboters durch eine diskretisierte Repräsentation seines Arbeitsbereichs. Die damit assoziierte Datenstruktur, die als *Erreichbarkeitskarte* bezeichnet wird, entspricht dabei einem räumlichen Gitter, das aus gleich großen Arbeitsraumzellen besteht und die Fähigkeit des Roboters beschreibt, bestimmte Greifziele in seiner unmittelbaren Umgebung zu erreichen. Für die Vorbereitung oder Ausführung von Manipulations- und Greifaktionen ist jedoch exakt die inverse Repräsentation gefragt, d.h. das Greifziel ist gegeben und eine dafür angemessene Standpose soll ermittelt werden. Um diese Repräsentation zu erhalten, erzeugen wir in unserem Ansatz sogenannte *inverse Erreichbarkeitskarten*, deren Datenstruktur potenzielle Standposen der mobilen Basis relativ zu einem gegebenen Greifziel beschreibt. Anschließend können die in der Karte gespeicherten Qualitätsinformationen genutzt werden, um eine bestmögliche Standpose aus der Menge potenzieller Standposen zu identifizieren.

Des Weiteren stellen wir in dieser Arbeit ein probabilistisches Bewegungsplanungssystem namens *Bidirectional Informed RRT** (BI^{2}RRT*) vor, das zur Erzeugung optimaler Pfade für mobile Manipulationsaufgaben dient. Dieses baut auf dem bereits existierenden *Informed-RRT** Planungsalgorithmus auf und erweitert diesen um weitere Eigenschaften. So ist es mit BI^{2}RRT* möglich eine bidirektionale Pfadplanung durchzuführen. Außerdem können beliebige, von Objekten im Greifer auferlegte, geometrische Bewegungseinschränkungen während der Planung berücksichtigt werden. Ist eine Start- und Zielkonfiguration gegeben, führt unser Planungssystem zunächst eine gleichmäßige Abtastung des Konfigurationsraums durch um neue Konfigurationsstichproben zu generieren. Sobald ein erster, jedoch suboptimaler, Lösungspfad gefunden wurde wird auf eine intelligentere Abtaststrategie gewechselt. Durch diese wird in der verbleibenden Planungszeit eine deutlich höhere Konvergenzrate in Richtung der optimalen Lösung erzielt. Geometrische Bewegungseinschränkungen werden zudem durch die Anwendung einer Projektionsmethode auf die genommenen Konfigurationsstichproben während der Planung automatisch behandelt. Eine Evaluierung unseres Ansatzes anhand unterschiedlicher Planungsszenarien ergab, dass unser System in der Lage ist, schnell und zuverlässig qualitativ hochwertige Lösungspfade für komplexe Manipulationsaufgaben zu generieren.

Zuletzt wird in dieser Arbeit ein mobiles Roboter-Assistenzsystem vorgestellt, das auch von Menschen mit stark eingeschränkten Bewegungs- und Kommunikationsmöglichkeiten genutzt werden kann. Das Gesamtsystem besteht hierbei aus mehreren interagierenden Komponenten. Dazu gehören eine nichtinvasive Aufzeichnung und Decodierung neuronaler Signale, eine semantische Aufgabenplanung, eine Bewegungs- und Manipulationsplanung sowie Umgebungswahrnehmungskomponente. Für die Aufnahme neuronaler Signale wird eine Elektroenzephalographie (EEG) -Kappe verwendet. Zur Decodierung der Signale folgen wir einem auf künstlichen neuronalen Netzwerken basierenden Ansatz. Die aus den EEG-Signalen decodierten Informationen werden anschließend zur Steuerung einer grafischen Benutzerschnittstelle (GUI) verwendet, die von einem semantischen Planer zur Verfügung gestellt wird. Die Benutzeroberfläche wiederum zeigt das Spektrum an möglichen Roboteraktionen an, welches kontinuierlich anhand von Informationen, die von der Roboterplattform oder einem Kamerasystem geliefert werden aktualisiert wird. Sobald der Benutzer mittels der GUI ein gewünschtes Ziel ausgewählt hat, zum Beispiel *"bringe mir ein Glas Wasser"*, ermittelt der semantische Planungsalgorithmus die für die Umsetzung erforderliche Sequenz von Roboteraktionen. Anschließend werden die Aktionen sequentiell durch Bewegungsplanungs- und Ausführungstechniken in der realen Welt umgesetzt.

Alle in dieser Arbeit vorgestellten Verfahren wurden praktisch implementiert und sorgfältig evaluiert. Zudem wurde ihre Anwendbarkeit neben Experimenten in Simulation durch den Einsatz auf realen Roboterplattformen belegt. Zusammenfassend stellt die vorliegende Arbeit somit einen wichtigen Beitrag dar, um mobile Roboter mit der Fähigkeit auszustatten, komplexe Bewegungen imitieren sowie autonom generieren zu können.

Abstract

Mobile manipulators are highly dexterous robotic units, unifying the navigation capabilities of mobile platforms with the manipulation capabilities of classical industrial robotic arms. Thus, they are nowadays expected to be able to operate in versatile domains, cope with challenging environments and to be flexible regarding the tasks assigned to them. Establishing efficient motion planning and control strategies for such systems, on the other hand, is particularly challenging due to their high number of degrees of freedom and the multitude of task and platform related constraints involved.

The core capabilities required by a mobile robotic system to successfully complete a mobile manipulation task are to be able to determine where it needs to go in the environment, how to get there without colliding with obstacles and to ensure that its motions are compliant with possible task-related constraints. Moreover, it needs to provide an appropriate interface to permit human operators to specify what it needs to do. In this context, we present in this thesis several novel contributions to the field of motion imitation and generation for mobile robotic systems. We hereby consider different levels of autonomy for the robot, initially relying on a human operator to provide the knowledge required to complete a task successfully towards a robotic service assistant capable of autonomously planning and executing mobile manipulation actions. Moreover, we incorporate motion imitation techniques to compare motion demonstrations of healthy subjects with the ones of patients exhibiting motor control deficits. Motion imitation and generation are both valuable approaches, as each of them offers its own individual advantages. Therefore, preference to the appropriate technique should be given depending on the intended field of application.

At the beginning, we introduce an approach that permits humanoid robots to imitate whole-body motions captured from a human operator in real time. Hereby, the robot is able to perform motions involving extended periods of time in which the robot needs to balance on a single leg. For our investigation on the underlying principles of human motor control behavior, we rely on motion capture data recorded from human demonstrations. More specifically, we quantitatively evaluate and compare the motion control strategies adopted by two groups, i.e., healthy subjects and Parkinson's disease patients. Additionally, we develop an approach that lets mobile robotic platforms autonomously select an optimal stance pose for preparation or execution of a subsequent mobile manipulation task by adopting the concept of inverse reachability maps. In the following, we present a probabilistic motion planning framework for generating asymptotically optimal paths for mobile manipulation tasks. This framework extends previous planning approaches in the field towards bidirectional search and satisfaction of arbitrary geometric end-effector task constraints. Finally, we present a mobile robotic service assistant framework composed of several interdisciplinary components that permits users with limited communication skills to express their desire using only thoughts.

All techniques developed in this thesis were practically implemented and thoroughly evaluated. The overall contribution of the present work is to equip mobile robotic platforms with the ability to imitate complex whole-body motions as well as to generate them autonomously.

Acknowledgments

Pursuing a PhD requires a high level of stamina, which over the years is impossible to maintain without the support and encouragement by many people. In the following, I would like to thank all those people who have contributed to the positive development of this journey.

First of all, I would like to thank my advisers Wolfram Burgard and Maren Bennewitz for guiding my scientific work and providing me with many valuable suggestions along the way. During my work in the Humanoid Robots Lab and the Autonomous Intelligent System Lab I learned a lot from their experience not only regarding scientific matters, but also concerning the presentation of own research findings on international conferences and workshops. Furthermore, they gave me the unique opportunity to gain insights into various research fields by getting involved into several interdisciplinary projects within the BrainLinks-BrainTools Cluster of Excellence.

My thank also goes to the former and current members of the Autonomous Intelligent System Lab and my former colleagues at the Humanoid Robots Lab. Due to the great atmosphere and the many fruitful discussions it was always a pleasure to work with you. Here, I would particularly like to mention my office mates Tim Welschehold, Benjamin Suger, Jörg Röwekämper, Tayyab Naseer, Ayush Dewan, Andreas Eitel, Oier Mees, Armin Hornung and Daniel Meier.

For their efforts and contributions in collaborative research projects, I furthermore thank my co-authors Daniel Kuhner, Johannes Aldinger, Lukas Fiederer, Martin Völker, Robin Schirrmeister, Chau Do, Joschka Bödecker, Bernhard Nebel and Tonio Ball.

I also would like to thank Tobias Schubert, Christian Dornhege, Chau Do, Daniel Kuhner and Tim Welschehold for proof-reading earlier versions of this document and providing valuable feedback.

In addition, I would like to appreciate the assistance of Susanne Bourjaillat and Michael Keser in the course of technical and administrative issues.

My deepest gratitude goes to my friends and family for supporting me in all decisions. Most of all, I would like to thank Britta, Robin and Malia for their consideration, support and love throughout the years.

Last but not least, I also would like to gratefully acknowledge that the work on this thesis has been generously supported by the German Research Foundation (DFG) within the BrainLinks-BrainTools Cluster of Excellence (grant number EXC 1086).

Contents

Chapter 1

Introduction

In industrial settings, stationary robotic systems are regarded as reliable, accurate and highly productive machinery that perform the tasks given with constant quality. In this scope, the workspace of the robot is well defined and isolated from human coworkers. The environment is assumed to be structured and the task given to the robot is specified beforehand.

With the introduction of mobile robotic platforms, this robotic stereotype became less applicable over the last decades. Mobility has led to a vast growth in the field of potential applications for robotic systems beyond the scope of industrial automation, including their deployment in the domestic domain and disaster relief scenarios. Combining the navigation capabilities of mobile platforms with the manipulation capabilities of classical industrial robotic arms led to highly dexterous robotic units, referred to as *mobile manipulators*. With the mobile platform extending the workspace of the manipulator, these systems are nowadays expected to be able to operate in versatile domains, cope with challenging environments and to be flexible regarding the tasks assigned to them.

However, the desire to exploit the advanced set of capabilities of such platforms, gained by the ability to navigate in the environment as well as to manipulate objects therein, to full extent introduces several new challenges. The major issue is that these systems typically possess a high number of degrees of freedom, which significantly increases the complexity of establishing efficient motion planning and control strategies. Classical approaches often follow the idea of circumventing the complexity of mobile manipulation tasks by decomposing them into the simpler subproblems of navigation and manipulation, which are solved sequentially and independent from each other. While this divide-and-conquer strategy is sufficient for many applications, it does not meet the demands facing such dexterous systems. Furthermore, it prohibits the execution of tasks requiring a coordinated motion of the mobile base and manipulator, that are in theory kinematically feasible. Another important aspect regarding mobile manipulation tasks is the number and type of constraints imposed on the robot's motion. The latter can be further distinguished into constraints related to the kinematic structure of the robot and those being imposed by the objects to be manipulated. Aside from general kinematic constraints, such as joint limits and collision avoidance, there are more platform-specific ones concerning the type of mobility. Wheeled robotic platforms, for example, exhibit a kinematic structure whose movement can be described by either an *omnidirectional*, *differential drive* or *ackermann steering* motion model. Omnidirectional mobility permits a robot to move in arbitrary directions at any time, whereas it is not possible for the latter models to instantly perform a motion in a direction orthogonal to the sagittal axis of the robot. For legged robotic systems, although naturally exhibiting omnidirectional mobility, particular constraints need to be taken into account. Humanoid platforms, for example, need to repetitively generate

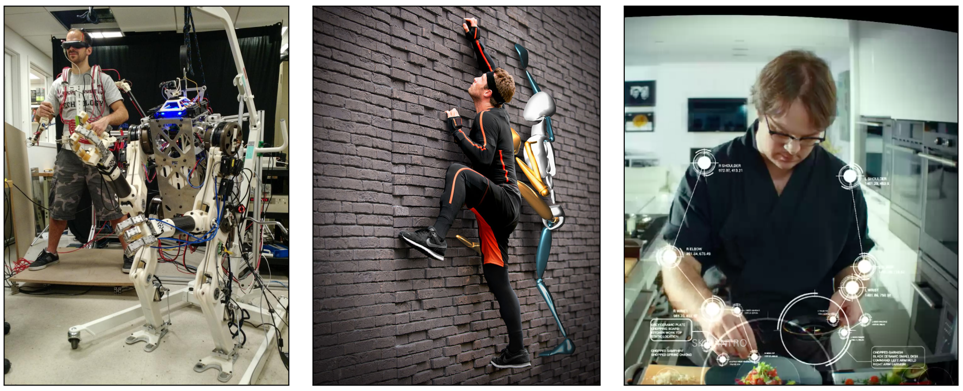


Figure 1.1: *Left:* The HERMES system for imitation of complex whole-body motions with a humanoid robot (Source: Biomimetic Robotics Lab, MIT). *Center:* The Xsens MVN full-body, wearable motion capture suit (Source: Xsens Technologies). *Right:* Marker-less motion capture system that allows a robotic chef to replay the movements recorded from a human chef (Source: Moley Robotics).

kinematic loop-closure establishing ground contact with both feet in order to navigate through the environment. Moreover, they are required to actively control the point representing the projection of their center of mass onto the floor in order to maintain balance and to avoid falling. Further constraints arise, once the robot has made contact with an object to be manipulated. Supplementary to the obvious requirement of avoiding collisions between the object, robot and the environment, the motion of the robot needs to comply with task specific constraints. The domain of object delivery provides a rich set of mobile manipulation tasks involving such constraints. A vessel containing liquid, for example, needs to be kept upright throughout its transition from one location to another. Pulling a tool trolley to a desired destination would require the robot's gripper to remain at the height of the trolley's handle in addition to keeping a proper orientation. Other examples arise from tasks involving manipulation of articulated objects. Opening drawers, doors or valves, demands the robot's gripper to follow a linear trajectory and a circular arc around the object's axis of rotation, respectively.

The research community has addressed several of the challenges outlined above and achieved significant progress in the fields of robot navigation and manipulation. However, despite the dexterity of modern mobile robotic platforms only little research has been conducted exploiting the advanced set of capabilities inherent to such systems. With the next generation of mobile robotic systems, navigation and manipulation will become inseparably intertwined, making the ability to simultaneously deal with kinematic and task-related constraints a necessity. The main contribution of this thesis is to provide mobile robotic systems with the capability to perform complex mobile manipulation actions. We hereby consider different levels of autonomy, starting from imitation of human motions and moving towards autonomous motion planning and execution.

Humans are, from a technical perspective, an excellent example of mobile systems that are capable of controlling a high number of degrees of freedom while simultaneously accounting for constraints arising from contact with the environment. Moreover, they possess a level of intelligence with respect to decision making and interpretation of visual information that is desirable for the control of artificial kinematic structures. Within the last decade, several affordable, lightweight and accurate motion capture systems have become available,

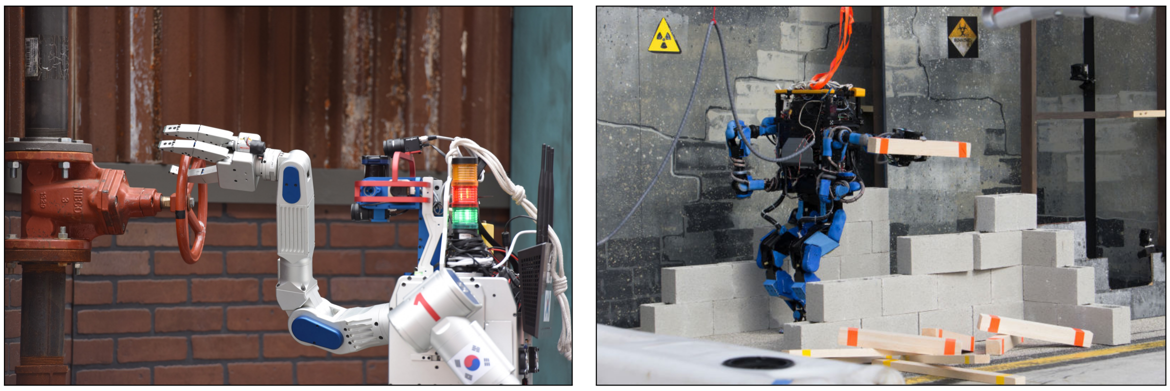


Figure 1.2: *Left:* Hubo robot of Team KAIST autonomously opens a valve. *Right:* Google's SCHAFT robot removes clutter from the path in a disaster scenario (Source: DARPA)

that permit a closer examination of human motor control behavior and make some of these human skills accessible for robotic applications. Examples of such systems include mechanical exoskeletons, inertial sensor-based motion capture suits and marker-less visual motion tracking systems, as depicted in Figure 1.1. The data captured from motion demonstrations of an expert human operator is typically processed using one of the following approaches. The first possibility is to gather data from multiple demonstrations, which are subsequently batch processed to investigate the underlying principles of human motion or to teach a robot how to execute a task in a correct manner. Other approaches process the motion capture data online in order to control robotic platforms in real time. While the former method aims to equip robots with the ability to autonomously generate human-like motion, the latter offers several benefits by keeping the human operator in the control loop. The first advantage of these approaches is that the robot remains flexible regarding the task, by relying on the operator to provide appropriate motion commands. Secondly, visual feedback from a robot's on-board camera permits the operator to infer the current location of the robot and potential manipulation targets. Finally, the motion of the robot can be dynamically adapted when changes in the environment make this necessary. However, there are also several challenges and limitations in the context of motion imitation based approaches. Huge differences between the human body and the robotic platform regarding their scale and kinematic structure make an intuitive motion mapping difficult. Moreover, only a strictly limited time window is available for the computations, if real-time motion imitation performance is to be achieved. Furthermore, there are many applications where human-based control is undesirable, e.g., for robotic service assistants deployed as tour-guides, household robots or for elderly care. The usage of online motion imitation techniques becomes difficult to impossible in other applications, when the communication between the motion capture system and the robot exhibits large delays or is impossible, as is frequently the case in disaster relief scenarios such as the ones illustrated in Figure 1.2.

Common to these settings is that the robot can no longer rely on the abilities of a human operator, implying that many rudimentary skills need to be explicitly modeled. A basic requirement for an autonomous agent is that it is capable of determining its own position and orientation in the environment. Then, it needs to be able to decide where to go in order to prepare or accomplish a given mobile manipulation task. To reach that target, it subsequently needs to plan a path which is kinematically feasible and collision-free, initiating from its current pose estimate. With an object attached to its gripper it additionally needs to ensure that all configurations along the planned path are compliant with the constraints imposed by the manipulated object. Once the robot starts moving towards the goal, deviations from the planned path may occur due to imprecise motion execution, which lead to different target poses or in the worst case to collisions with the environment. To compensate for these inaccuracies and therefore to guarantee a safe and precise arrival at the desired target pose, the robot needs to continuously monitor its pose based on sensor data and adapt its motion accordingly.

From the described challenges, we identified the following research questions in the field of motion imitation and generation for dexterous mobile robotic systems that we will address in this thesis:

- How can we capture and transfer whole-body human motions to a humanoid robot in real time?
- How can we quantify motor control deficits in human subjects induced by neurological impairments, e.g., arising from Parkinson's disease, using motion imitation techniques?
- How do humans position themselves relative to an object in preparation for a subsequent manipulation action and how can that strategy be transferred to mobile robotic systems?
- How can we efficiently plan for challenging reaching and delivery tasks for mobile robots, thereby avoiding collisions and respecting geometric constraints imposed by the objects to be manipulated?
- What are the requirements and challenges to build a robotic service assistant system for users with limited communication skills?

The structure of this thesis is as follows. In Chapter 2, we present an approach that permits humanoid robots to imitate whole-body motions captured from a human operator in real time. We hereby do not constrain the robot to remain with both feet on the ground. Instead, our work allows the robot to perform motions involving extended periods of time in single support mode, i.e., when the robot needs to balance on a single leg. For capturing human motions, we rely on a whole-body motion capture suit composed of several inertial measurement units that offer wireless transmission of motion data. To achieve real-time performance, we have chosen a compact representation for the human motion model. The data obtained from the suit is processed sequentially in our approach. For every captured frame, we first perform a motion mapping step which accounts for the differences in scale between the human and humanoid kinematic model. In a second step, we consider the difference between the two models regarding their mass distribution in order to convert the mapped body pose into a statically stable whole-body configuration. To safely switch between single and double support mode, we furthermore actively control the motion of the robot's center of mass. A general shortcoming, however, is that the value of the prescribed motion is not put into question. Sub-optimal human motion patterns will therefore deteriorate the resulting task performance of a humanoid robot.

Chapter 3 addresses this issue by examining the underlying principles of human motor control behavior. Again, we rely here on motion capture data recorded from human demonstrations. In this context, we consider a hand coordination task involving only motions of the human upper body. For our investigation, we consider subjects which have provably no history of prior diseases affecting the motion behavior, as a control group. The comparison group is represented by Parkinson's disease (PD) patients, whose neurological impairments are well known to go along with motor control deficits. In this work, we propose an approach to quantitatively evaluate the motion recordings in order to infer specific motion strategies for both groups. To do so, we follow a two step approach. In the first step, we map the recorded motions to a simplified artificial kinematic model of the human upper body. Thereafter, this model is used to reproduce the same motions using a jacobian damped least-squares controller with joint weight parameterization. To determine the joint weights required to closely reflect the observed motions, we develop an iterative learning method that uses the mapped human joint trajectories as reference input. Considering the resulting set of joint weights, we are finally able to infer associated motion control strategies for the two groups.

In Chapter 4, we introduce an approach that lets mobile robotic platforms autonomously select an optimal stance pose for preparation or execution of a subsequent mobile manipulation task. In order to no longer depend on a human operator to provide an appropriate stance pose, we describe the robot's reaching and manipulation capabilities by a discretized representation of its workspace. The resulting data structure, referred to as the *forward reachability map*, is a spatial grid composed of equally sized workspace voxel and indicates the robot's capability of reaching certain end-effector targets from different robot base poses. Manipulation and reaching actions, however, require exactly the inverse information to be available. Therefore, we generate *inverse reachability maps* in this work, which permit the selection of optimal stance poses for a mobile base relative to a given grasping target.

Chapter 5 presents a probabilistic motion planning framework, called Bidirectional Informed RRT*, for generating asymptotically optimal paths for mobile manipulation tasks. It extends the Informed RRT* algorithm towards bidirectional search and satisfaction of arbitrary geometric end-effector task constraints. Given a pair of terminal robot configurations, our planner uses uniform sampling in the configuration space until an initial, though sub-optimal, solution path is found. Afterwards, the remaining planning time is used for path refinement adopting an informed sampling strategy which provides a higher rate of convergence towards the optimal solution. Task-related end-effector constraints are automatically handled during planning by employing a first-order projection method for configuration space samples.

Afterwards, we introduce in Chapter 6 a mobile robotic service assistant framework dedicated to users with limited communication skills. The overall system is composed of several interacting components, i.e., non-invasive neuronal signal recording and decoding, high-level task planning, motion and manipulation planning as well as environment perception. Neuronal signals are recorded with an electroencephalography (EEG) cap and decoded using a convolutional neural network approach. The decoded information is subsequently used to control a graphical user interface (GUI), provided by a high-level planner instance. The GUI in turn displays the current set of feasible actions which is continuously updated according to information provided by the robot and a camera perception system. Tasks selected by the user are finally decomposed into an atomic action sequence and resolved into robot motion trajectories using low-level motion and manipulation planning techniques.

In Chapter 7, we finally summarize the results of this thesis and elaborate on open research questions for future work.

1.1 Key Contributions

In this thesis, we present several scientific contributions to the field of motion imitation and generation for mobile robotic systems. In summary, our key contributions are:

- a real-time posture mapping approach to transfer human motion data to a humanoid robot (Chapter 2),
- a method to evaluate the motor control deficits in human subjects introduced by neurological impairments, such as Parkinson's disease (Chapter 3),
- an approach for automatic stance pose selection for mobile robotic systems based on inverse reachability maps (Chapter 4),
- a flexible and efficient path planning framework for task-constrained mobile manipulation (Chapter 5),
- a modular robotic service assistant framework controllable by users with limited communication skills (Chapter 6).

1.2 Publications

The contents of this thesis have been published in international conferences and workshop proceedings. In the following, a summary of the publications is given in chronological order.

Conference Proceedings

- F. Burget, L.D.J. Fiederer, D. Kuhner, M. Völker, J. Aldinger, R.T. Schirrmeister, C. Do, J. Boedecker, B. Nebel, T. Ball, and W. Burgard. Acting Thoughts: Towards a Mobile Robotic Service Assistant for Users with Limited Communication Skills. In *European Conference on Mobile Robots (ECMR)*, 2017
- F. Burget, M. Bennewitz, and W. Burgard. BI^{2}RRT*: An Efficient Sampling-Based Path Planning Framework for Task-Constrained Mobile Manipulation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016
- F. Burget, C. Maurer, W. Burgard and M. Bennewitz. Learning motor control parameters for motion strategy analysis of Parkinson's disease patients. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015
- F. Burget, and M. Bennewitz. Stance Selection for Humanoid Grasping Tasks by Inverse Reachability Maps. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015
- J. Koenemann, F. Burget, and M. Bennewitz. Real-time Imitation of Human Whole-Body Motions by Humanoids. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014

Workshop Proceedings

- F. Burget, M. Cenciarini, B. Meier, H. Bast, M. Bennewitz, W. Burgard, and C. Maurer. A Closed-Loop System for Real-Time Calibration of Neural Stimulation Parameters using Motion Data. In *Proc. of the ICRA Workshop on Wearable Robotics for Motion Assistance and Rehabilitation - RoboAssist*, 2014

The following publications are not covered by this thesis

- F. Burget, A. Hornung, and M. Bennewitz. Whole-body motion planning for manipulation of articulated objects. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013

1.3 Collaborations

This thesis was carried out within the projects *Calimotion* and *NeuroBots* of the interdisciplinary cluster of excellence *BrainLink-BrainTools* and includes several collaborations with other researchers. The following listing summarizes these collaborations for the individual chapters of this work.

- Chapter 2: The motion imitation system presented was originally developed during the work of Jonas Koenemann as a research assistant and extends prior work published in [74]. The related publication is [75].

- Chapter 3: The work on motion strategy analysis has been conducted in collaboration with Massimo Cenciarini, Christoph Maurer, Wolfram Burgard and Maren Bennewitz. The experimental protocol and the database of motion capture data has been established together with Massimo Cenciarini and Christoph Maurer. The joint weights learning approach is the outcome of fruitful discussions with Maren Bennewitz and Wolfram Burgard. Medical implications are drawn from the results with the support of Christoph Maurer. The related publications are [14, 15].

- Chapter 4: The approach for selecting optimal stance poses for mobile robotic platforms based on inverse reachability maps has been developed in cooperation with Maren Bennewitz. The related publication is [12].

- Chapter 5: The framework for task-constrained mobile manipulation planning, presented in this chapter, is the result of a collaboration with Maren Bennewitz and Wolfram Burgard. The related publication is [16].

- Chapter 6: The mobile robotic assistant system presented in this chapter is a result of a collaboration with Lukas Fiederer, Daniel Kuhner, Martin Völker, Johannes Aldinger, Robin Schirrmeister, Chau Do, Joschka Boedecker, Bernhard Nebel, Tonio Ball and Wolfram Burgard. The author of this thesis has contributed the component for planning and executing navigation actions for the mobile base. Moreover, the central knowledge base interfacing all components as well as the perception system and its logic layer has been jointly developed with Daniel Kuhner. Recording and decoding of neuronal signals is work conducted by Lukas Fiederer, Martin Völker, Robin Schirrmeister and Tonio Ball. The high-level planning component and its graphical user interface has been developed by Johannes Aldinger and Bernhard Nebel. Daniel Kuhner contributed the component for arm manipulation planning. The approach for liquid level detection has been developed by Chau Do. The related publication is [17].

1.4 Notation

The following notation will be used throughout this work.

Symbol	Description
$a, b, \dots$	scalar value
$\mathbf{x}, \mathbf{y}, \dots$	vector (column vector by default)
$\mathbf{x} = (x, y, z)$	vector with the scalar values x, y, z
$\mathbf{A}, \mathbf{B}, \dots$	matrix
$\mathbf{x}^\top, \mathbf{A}^\top, \dots$	transpose of a vector or a matrix
$\lVert\mathbf{x}\rVert$	norm of a vector (i.e. its length)
$\mathbf{A}_{ij}$	entry of matrix A at row i and column j
$\mathbf{x}_i$	i-th entry of a vector
$\bar{\mathbf{x}}$	mean of the elements of $\mathbf{x}$
$\tilde{\mathbf{x}}$	range of $\mathbf{x}$
$\hat{\mathbf{x}}$	a value in the neighborhood of $\mathbf{x}$
$a \leftarrow b$	assignment of value b to variable a
φ, θ, ψ	Euler angles roll, pitch and yaw
$\mathcal{S} = \{s_1, s_2, \dots\}$	set
$\lvert\mathcal{S}\rvert, \lvert\mathbf{x}\rvert$	cardinality (number of elements) of a set or dimension of a vector
$\mathbf{x} \oplus \mathbf{y}, \mathbf{x} \ominus \mathbf{y}$	concatenation and difference of 6D rigid body transforms [123]

Chapter 2

Real-time Imitation of Human Whole-Body Motions

The system presented in this chapter enables humanoid robots to imitate complex whole-body motions of humans in real time. Based on a compact representation of the human kinematic model, we consider the positions of the hands and the motions of the center of mass as the most important aspects to imitate. Since direct imitation of the prescribed human motion would quickly yield unstable motion behavior of the humanoid robot, our system proposes a method to actively control the motion of the center of mass over the support polygon. For every motion capture frame recorded from the human operator, our approach generates a statically stable whole-body pose. Hereby, we do not constrain the configurations to be in double support. Instead, we allow for changes of the support mode according to the motions to imitate. To achieve safe imitation, we adapt the target poses of the robot's feet if necessary. Subsequently, we find corresponding statically stable configurations by means of inverse kinematics. We present experiments using human data, captured with an inertial sensor-based whole-body motion capture suit. The results show that a Nao humanoid is able to reliably imitate complex whole-body motions in real time. This considers also human demonstrations involving extended periods of time in single support mode, i.e., when the robot has to balance on a single leg.

The advancements in sensor technology, achieved within the last decade, has enabled the development of a variety of highly accurate and lightweight motion capture systems such as mechanical exoskeletons, inertial sensor-based whole-body suits and visual motion tracking systems. The data obtained by recording multiple human task demonstrations with such a system can be subsequently used, for example, to teach a robot how to generate human-like motion patterns. Furthermore, these systems are capable of recording human motion data at very high frequency rates, thus opening up interesting perspectives for online control of robotic platforms exhibiting a high number of degrees of freedom. Humanoid robots, for instance, can be teleoperated by imitating the motions captured from a human operator. This capability might be particularly useful for implementing motion control for telepresence systems or humanoids to be deployed in hazardous environments.

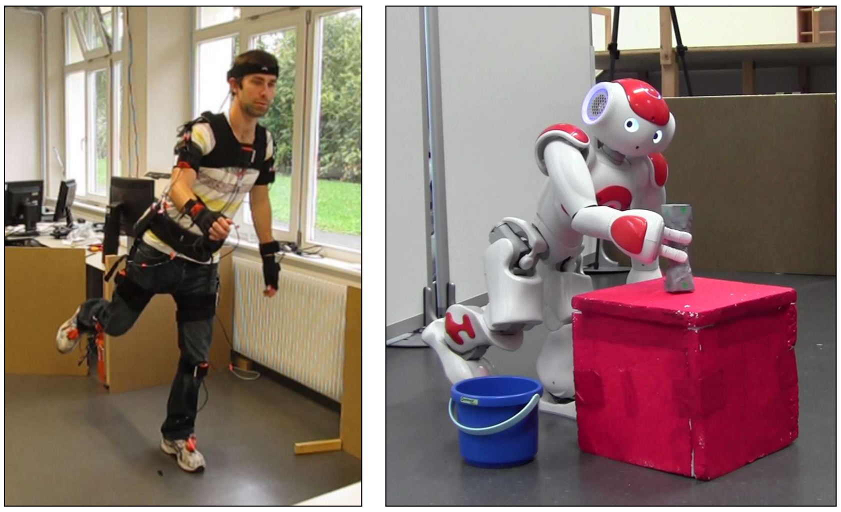


Figure 2.1: Imitation of a complex whole-body motion with a humanoid for a teleoperated manipulation task. Note that the robot imitates configurations in which it is required to balance on a single leg over a longer period of time.

However, transferring the captured human motion to a humanoid robot is not straightforward. There are mainly two factors that prevent a direct imitation of the prescribed motion. First, the kinematic structure of the robot is typically very different from the human body regarding scale and agility. Secondly, there are large differences concerning the body mass distribution between the two models. Therefore, it can be challenging generating feasible motions for the robot and ensure stable execution when dealing with complex motion demonstrations. Particularly, when the human motion involves multiple support mode changes or even extended periods of time in single support, safe imitation on the humanoid robot is difficult to achieve.

To this day, a variety of approaches towards imitation of human whole-body or upper body motions have been presented. Many of them rely on an offline step that performs optimization on the human data to adapt it to the robot's kinematic structure and constraints [20, 73, 93, 110, 131, 134]. On the other hand, several systems that allow for real-time imitation have been presented. Most of them focus on generating upper-body motions while the legs are neglected or mainly used for balancing [37, 42, 95], others do not take support mode changes into consideration [127, 145]. In contrast to previous work, our system enables a real humanoid robot to imitate complex whole-body motions that include support mode changes in real time while ensuring static stability. Instead of relying on expensive preprocessing steps and a high number of variables to optimize, our approach uses a compact human model to reduce the computational effort. In particular, we consider the positions of the end-effectors, i.e., the position of the hands and feet, as well as the position of the center of mass (CoM). By doing so, we render the robot's motion as similar as possible to the demonstrated motion. Our approach applies inverse kinematics (IK) to generate joint angles for the four kinematic chains given the end-effector positions as input. Afterwards, we adapt the joint angles of the leg kinematic chains to match the human's CoM position and to ensure static stability of the robot. To do so, we use a retargeting technique for the robot's feet which generates statically stable whole-body configurations by employing numerical inverse kinematics.

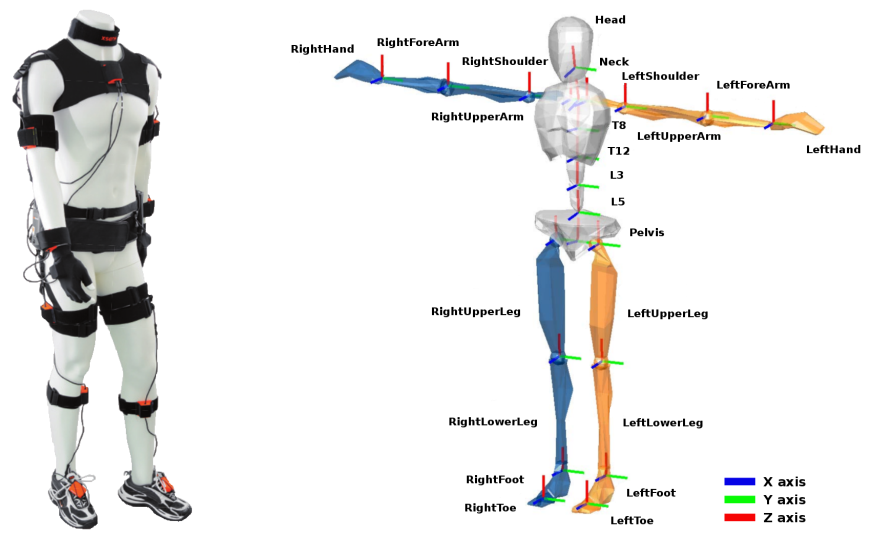


Figure 2.2: *Left:* Xsens MVN motion capture suit. *Right:* Biomechanical model with anatomical landmarks (Source: Roetenberg et al. [105]).

The approach we present in this chapter is, to the best of our knowledge, the first one that explicitly allows imitation of motion sequences involving extended periods of time in single support mode, in which balancing on one foot is inevitable. We present experiments with a Nao humanoid reliably imitating complex whole-body motions in real time (see Figure 2.1). The human motion is captured with a motion capture suit consisting of several inertial sensors attached to the individual body segments. We thoroughly evaluate our approach regarding stability, similarity to the human motion and computational effort. As the results show, our system generates safe motions for the robot while achieving high similarity to the human and allows for teleoperation in real time. Preliminary results of this work have been published in [74].

2.1 Whole-Body Motion Capture

As our approach relies on accurate motion data recorded from human demonstrations in real time, we describe in the following the motion capture system used in our work as well as the associated data acquisition process.

2.1.1 Motion Capture Suit

To record human motion data, we use the *Moven* full-body, wearable motion capture system by Xsens Technologies [105]. The suit is composed of 16 inertial measurement units of the type *MTx*[1], which are attached to specific anatomical landmarks on the human body, as depicted in Figure 2.2 (left). The measurement units are equipped with accelerometers, gyroscopes and magnetometers measuring the earth's magnetic field. The data recorded from these units is collected by two wireless transmitters located at the human operator's hip. A computer on the receiver side processes the incoming data based on advanced Kalman filter algorithms and

[1]A detailed specification is available under http://www.xsens.com/products/mtx

displays the resulting body posture in the *MVN Studio* user interface provided by Xsens. Here, the biomechanical model shown in Figure 2.2 (right), composed of 23 body segments and 22 revolute joints, is used for graphical representation of the captured motion.

Furthermore, *MVN Studio* offers to store the captured motion data offline or to transmit it in real time for further processing. For offline storage of the recorded motion, the accumulated data is exported to a *Moven Open XML* (MVNX) file after recording. On the other hand, real-time transmission is achieved by sending the data of each motion capture frame over the network, as will be described in more detail in Section 2.1.2.

The data representation of each motion capture frame, however, is independent from the modality chosen for the transmission. Although the system provides several motion quantities such as angular velocities and accelerations, only the orientation and position information of body segments is considered relevant in this work. Therefore, we describe for both modalities a motion capture frame $\mathbf{f}_i$, recorded at time i, by a vector of 161 data elements, i.e., 7 values for each of the $N = 23$ body segments,

$$\mathbf{f}_i = (\mathbf{q}_{seg1}, \mathbf{p}_{seg1}, \ldots, \mathbf{q}_{segN}, \mathbf{p}_{segN})^\top, \tag{2.1}$$

where $\mathbf{q}_{seg}$ and $\mathbf{p}_{seg}$ describes the orientation represented as a quaternion (4 values) and the position (3 values) of a segment, respectively.

2.1.2 Data Acquisition

A premise to achieve real-time motion imitation performance for our approach is that motion capture frames are aquired from *MVN Studio* immediately after they have been recorded. To permit the transmission of motion capture data, Xsens has developed a real-time network streaming protocol, called the *Moven Xsens Transfer Protocol* (MXTP). It builds upon the *IP* (Internet Protocol) and *UDP* (User Datagram Protocol) network layer already available for network clients. The IP layer is used to defined the source and destination of the packets within the network. The *UDP* layer is used to encapsulate the data. Contrary to the *TCP* (Transmission Control Protocol), the *UDP* protocol is unidirectional and stateless, i.e., does not require the receiver to answer incoming data packets. Nevertheless, with our local network being composed of only two computers, we experienced nearly no or only very little packet loss during our experiments.

2.2 Human Motion Model

The captured motion data allows for precisely reproducing the human motions on a virtual model of the human body. However, the execution of the same motion on a robot platform is virtually impossible due to the differences in the number of degrees of freedom and joint range between the two models. Based on the received data $\mathbf{f}_i$ (see Eq. (2.1)), we consider in this work a motion as a sequence of postures $\mathbf{p}_i$

$$\mathbf{p}_i = (\mathbf{p}_{LHand}^{LShoulder}, \mathbf{p}_{LFoot}^{LHip}, \mathbf{p}_{RHand}^{RShoulder}, \mathbf{p}_{RFoot}^{RHip})^\top. \tag{2.2}$$

Thus, each posture is defined as the 3D position of the end-effectors, i.e., the hands and feet, relative to the left or right shoulder/hip frame at time i (see Figure 2.3). So far, we just include

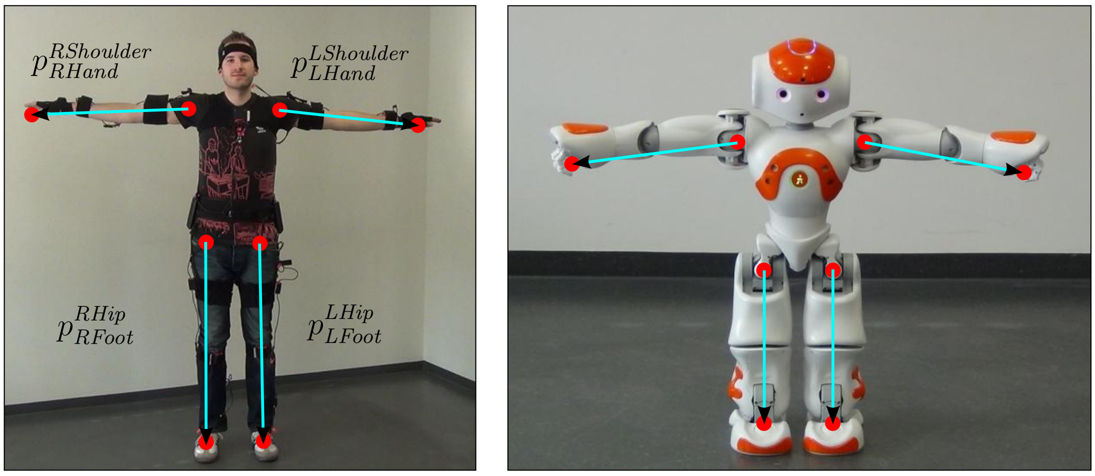


Figure 2.3: T-pose of the human and the robot model used as reference for posture mapping.

the end-effectors' positions in the model. However, the model can be easily extended to include the end-effectors' orientations provided by $\mathbf{f}_i$ as well as further features such as the elbow and knee positions. As the proposed method is based on inverse kinematics, additional constraints or tasks can be included in an augmented Jacobian or can be solved by projection into the Nullspace of the Jacobian. For each posture we additionally take into account the position of the CoM and the support mode of the demonstrator, which can be estimated from the motion capture data. By adopting this compact representation of body postures, we account for the limited physical capabilities of humanoid robot platforms with respect to the human body.

2.3 Human to Humanoid Posture Mapping

The first step of our approach aims to generate a whole-body configuration for the humanoid robot that reflects the captured human posture as closely as possible while initially neglecting constraints arising from the humanoid's kinematic structure. To do so, we make use of a common reference posture which is subsequently used in the posture mapping process. In the following this procedure will be described in detail.

2.3.1 Reference Posture

In order to account for the difference in scale between the two models, an initialization is performed prior activating real-time imitation. During that initialization, the human and humanoid are adopting a specific body posture, referred to as the T-pose (see Figure 2.3). Besides being suitable for comparing the lengths of the respective kinematic chains, this pose is also used to calibrate the biomechanical models depicted in Figure 2.2 in order to obtain more accurate human body posture estimates. In the following, we will refer to $\mathbf{p}_{ref,H}$ and $\mathbf{p}_{ref,R}$ as the posture of the human and the robot in the T-pose, defined according to Eq. (2.2).

2.3.2 Posture Mapping

To continuously update the robot posture, we consider the change of the human body posture $\mathbf{p}_{i,H}$, recorded at time i, relative to its reference posture $\mathbf{p}_{ref,H}$. The deviation of the end-effector positions relative to their references is determined by

$$\Delta\mathbf{p}^c_{i,H} = \mathbf{p}^c_{i,H} \ominus \mathbf{p}^c_{ref,H}, \tag{2.3}$$

with $c \in [1, 4]$ being the respective kinematic chain. In order to imitate the motion of the human, we expect the deviation of the robot's end-effector positions $\Delta\mathbf{p}^c_R$ from their positions in $\mathbf{p}_{ref,R}$ to be proportional to the values obtained from Eq. (2.3), as given by the following equation

$$\Delta\mathbf{p}^c_{i,R} = m^c \cdot \Delta\mathbf{p}^c_{i,H}, \tag{2.4}$$

where m^c represents the proportionality constant, given by the ratio between the limb length of the robot $l_{limb,R}$ and the human $l_{limb,H}$ for the c-th kinematic chain, defined as

$$m^c = \frac{l^c_{limb,R}}{l^c_{limb,H}}. \tag{2.5}$$

The length of the limbs is extracted from motion data recorded while the human and humanoid are performing the T-pose depicted in Figure 2.3. Given $\Delta\mathbf{p}_{i,R}$, the robot's posture $\mathbf{p}_{i,R}$ at time i is updated as

$$\mathbf{p}_{i,R} = \mathbf{p}_{ref,R} \oplus \Delta\mathbf{p}_{i,R}. \tag{2.6}$$

For the target position of the c-th end-effector contained in $\mathbf{p}_{i,R}$, we find the joint angles for the corresponding kinematic chain using a numerical inverse kinematics solver, which contrary to analytical solvers generates continuous motion transitions. The technique used for this purpose is based on the damped least-squares (DLS) control method, which considers a weighting matrix for joint limits avoidance in addition to avoiding kinematic singularities [21]. The control law applied to generate joint velocities that iteratively lead to a kinematic chain configuration matching the desired target position for a robot end-effector is given by

$$\dot{\mathbf{q}} = \mathbf{W}^{-1}\mathbf{J}^\top(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^\top + \lambda^2\mathbf{I}_m)^{-1}\mathbf{e}, \tag{2.7}$$

where $\lambda, \mathbf{e}$ is a constant damping factor and the error between the current and desired target position $\mathbf{p}^c_{i,R}$, respectively. The Jacobian of the kinematic chain is denoted by $\mathbf{J}$. $\mathbf{I}_m$ is an $m \times m$ identity matrix, where m is the number of task coordinates. The diagonal weighting matrix used for joint limits avoidance is defined as

$$\mathbf{W} = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & w_n \end{bmatrix}, \tag{2.8}$$

with n being the number of joints for the respective chain. The individual entries w_j are computed as follows

$$w_j = \begin{cases} 1 + |\frac{\partial H(q)}{\partial q_j}|, & \text{if } \Delta|\frac{\partial H(q)}{\partial q_j}| \geq 0 \\ 1, & \text{if } \Delta|\frac{\partial H(q)}{\partial q_j}| < 0 \end{cases}, \tag{2.9}$$

where the gradient of the performance criterion $H(q)$ is defined as

$$\frac{\partial H(q)}{\partial q_j} = \frac{(q_{j,max} - q_{j,min})^2(2q_j - q_{j,max} - q_{j,min})}{4(q_{j,max} - q_j)^2(q_j - q_{j,min})^2}. \tag{2.10}$$

Here, q_j refers to the angle of the j-th joint. The quantities $q_{j,min}$, $q_{j,max}$ denote the upper and lower joint limit, respectively. From Eq. (2.9), it follows that a joint is free to move given that it is moving away from its limits, i.e., the associated weight factor w_j is set to 1. Contrarily, motions towards the joint range borders are damped according to the current distance to the limit or entirely stopped once a critical proximity is detected.

Executing the resulting joint angles for the individual chains will lead to a robot posture that is similar to the captured human posture considering the differences between the two models in scale and kinematic structure. Nevertheless, the mapping procedure is insufficient for safe imitation of human motions. So far, only the feet *positions* have been considered for the mapping. Therefore, the support feet of the robot may not be parallel to the ground. Moreover, differences in the mass distribution of the robot and the human have not been considered to this stage. Finally, the dynamics of the human and the robot are different. For simplification, our approach does not explicitly consider the dynamics of the robot. Instead, it generates statically stable robot poses for every point in time as described in the following.

2.4 Humanoid Posture Stabilization

As it is desirable to maintain maximal similarity between the robot's motion and the human motion, the mapped, though unstable robot postures should be modified as little as possible. To ensure this requirement, we present a posture stabilization method that modifies only the configurations of the robot's lower body kinematic chains. Another important criteria for a realistic reproduction of the demonstrated motion is that the support mode and the trajectory of the CoM of the robot closely follows the one given by the human operator. Thus, posture stabilization aims to transfer the unstable robot pose obtained after posture mapping into a statically stable robot configuration, given the current CoM and support mode of the human.

To fulfill these demands, our approach performs the following steps. Initially, a common representation of the CoM position for the robot and the human is established. However, direct mapping of the CoM trajectory to the humanoid robot is prohibited due to the fact that the human may change his support mode dynamically, whereas the dynamics of the robot have not been modeled in this work. Therefore, the recorded CoM trajectory is adapted in order to generate statically stable support mode transitions for the robot. Furthermore, the change of the CoM position per time step is constrained to ensure safe motion execution. The support mode for the robot is determined based on the support mode of the human and the designated position of the CoM. In the last step, the end-effector pose of the feet is retargeted to generate a statically stable whole-body pose. The corresponding joint configurations are found using the numerical inverse kinematic solver, defined according to Eq. (2.7). The outlined steps are explained in detail in the following.

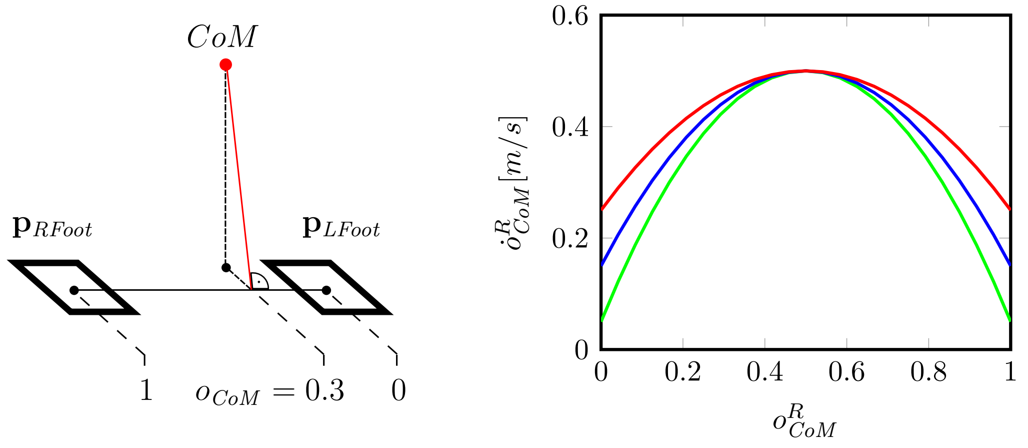


Figure 2.4: *Left:* Determination of the normalized offset given the projection of the center of mass onto the connection line between the feet. *Right:* Different choices for the maximal permitted offset velocity $\dot{o}^R_{CoM}$ as a function of the current offset o^R_{CoM}. The colors of the curves indicate their impact in terms of stable motion execution, ranging from safe transitions with larger imitation delays (green) to critical transitions yielding lower imitation delays (red).

2.4.1 Controlling the Center of Mass

We use a low-dimensional projection of the CoM to describe the position of the CoM $\mathbf{p}_{CoM}$ relative to the positions of the feet as a scalar factor. This offset o_{CoM} is determined by the orthogonal projection of the CoM onto the connection line between the feet, as depicted in Figure 2.4 (left). The offset is normalized between 0 and 1 so that it describes the relative distances of the projected CoM to the feet center positions $\mathbf{p}_{LFoot}$ and $\mathbf{p}_{RFoot}$. With this normalization, the offset from the human motion data can be directly translated to the robot. The offset is computed as follows:

$$o_{CoM} = \frac{(\mathbf{p}_{CoM} - \mathbf{p}_{LFoot}) \cdot (\mathbf{p}_{RFoot} - \mathbf{p}_{LFoot})}{\|\mathbf{p}_{RFoot} - \mathbf{p}_{LFoot}\|^2} \tag{2.11}$$

As the CoM of the human is not necessarily over a single support foot when changing to the single support mode, the trajectory of the offset needs to be adapted for the robot to allow statically stable support mode transitions. For example, the offset is required to be 0 before the robot can safely lift its right foot to balance on the left leg. Thus, whenever the human stands on a single foot, the offset for the robot is forced to be 0 or 1. Obviously, this would result in fast changes in the trajectory of the CoM. Thus, the velocity of the offset from its current value o^R_{CoM} towards the desired offset o_{CoM} is limited to generate smooth and safe trajectories of the CoM. Here, we use the negative quadratic function, illustrated in Figure 2.4 (right, green curve), with maximum velocity at $o^R_{CoM} = 0.5$ and close to zero velocity at the borders of the offset range to determine a new target offset $\hat{o}^R_{CoM}$ for the robot. In practice, this results in safe motion imitation, but on the other hand causes a slight delay in the imitation process when support mode changes occur. Reducing the delay by adapting the function parameters is generally possible, though comes along with a higher risk of loosing balance. A sample trajectory of the human and robot offset is illustrated in Figure 2.11 within the experimental section.

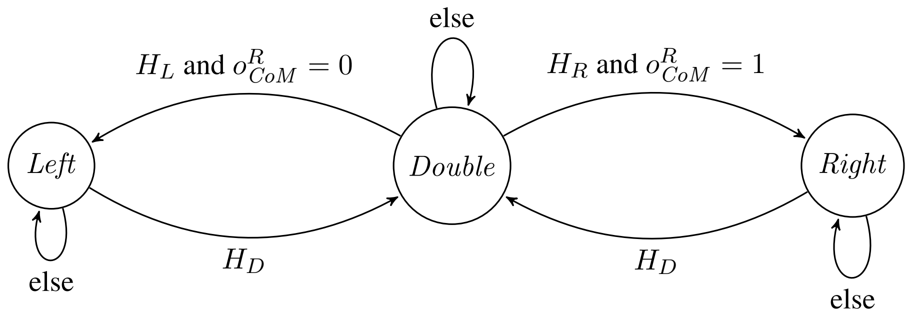


Figure 2.5: Robot support mode states and transitions. H_D, H_L, and H_R indicate whether the human is in double, left, or right support mode. o^R_{CoM} refers to the normalized offset (see Figure 2.4). Before a change to single mode occurs, the CoM is smoothly shifted to the corresponding foot.

2.4.2 Controlling the Support Mode

Having a good knowledge about their own body dynamics, humans are capable of almost instantaneously switching from double to single support and vice versa. With the ability to recover from unstable body postures, i.e., where the projection of their CoM lies outside of the support polygon, they can safely perform motions such as dynamic walking or jumping. Aside from the fact that many motion sequences are impossible to imitate by a humanoid robot due to the limited dynamic properties of its mechanical structure, we do not explicitly consider the dynamics of the humanoid in this work. Thus, the trajectory of the projection of the CoM is enforced to stay within the support polygon at any time in order to achieve a safe statically stable motion execution behavior. To do so, the CoM of the robot is carefully shifted over the desired support foot prior initiating a support mode change. Our approach uses a finite state machine to model the support mode of the robot based on the support mode of the human (which is either double (H_D), left (H_L), or right (H_R)), the robot's current support mode and the offset of the robot o^R_{CoM}. The state transitions are illustrated in Figure 2.5. Support mode changes of the human are detected once the height difference between the two feet has exceeded or fall below a specific threshold. The current pose of the feet is obtained from motion capture data. When the robot is in double support, for example, it will only change to single support if the human is in single support and if its own offset o^R_{CoM} is 0 or 1. If the offset has a value in between, the robot is not allowed to change to single support, even if the human is already on a single foot. Instead, the offset is smoothly shifted towards 0 or 1 by the offset control, described in Section 2.4.1, in order to achieve the desired support mode. Accordingly, the robot needs a few frames to change its support mode.

2.4.3 Endeffector Retargeting

Finally, we describe how statically stable whole-body configurations are generated from the robot postures (obtained after posture mapping), the determined CoM positions and support modes. As explained in the following, our approach follows the idea of adapting the pose of either one foot or both feet such that the target offset $\hat{o}^R_{CoM}$ is met.

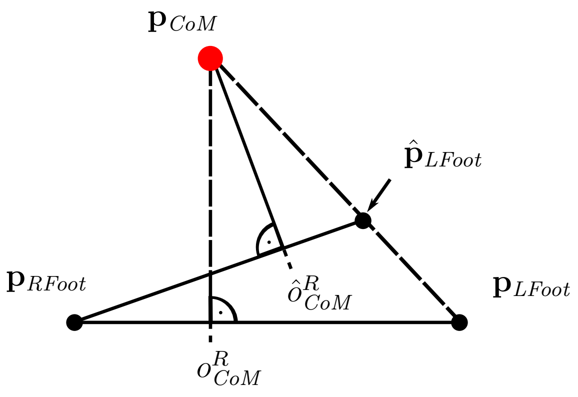


Figure 2.6: Double support posture stabilization. Depending on the current CoM and the target offset $\hat{o}^R_{CoM}$, the pose of one foot is retargeted and the joint angles of the corresponding leg chain are recalculated such that the resulting posture is statically stable.

Double Support

Given the current offset of the robot o^R_{CoM} and the desired offset o_{CoM} specified by the human demonstrator, we want to achieve statically stable double support for the robot, with o^R_{CoM} being ideally equal to o_{CoM}. However, as outlined above, the change of the human offset from one posture to the next is often quite large due to the dynamic nature of human motion. Trying to instantaneously shift the robot's offset by the same amount would very likely cause the robot to become unstable or even to fall over. Therefore, we allow only for a limited maximum shift of the robot's offset in each imitation step. This issue is taken care of by our offset controller, introduced in Section 2.4.1. In double support mode only one foot is repositioned such that the CoM, projected on the connection line between the feet, equals the target offset $\hat{o}^R_{CoM}$. The repositioning process is illustrated in Figure 2.6. Here, the left foot position $\mathbf{p}_{LFoot}$ is shifted along the line pointing to the CoM position $\mathbf{p}_{CoM}$ until the CoM's orthogonal projection onto the line connecting the feet centers yields the desired target offset. Whether the right or the left foot position is adapted depends on the target offset $\hat{o}^R_{CoM}$ and the current offset o^R_{CoM}. If $\hat{o}^R_{CoM} < o^R_{CoM}$, i.e., the CoM is required to move to the left from its current position, the left foot is repositioned. Similarly, the right foot position is adapted given that the target CoM moves towards the right.

Afterwards our approach calculates new target orientations for the feet such that they have the same orientation and span a plane on the ground. The orientation is given by the direction of the up-pointing vector of the feet, which is the normal $\mathbf{n}_{\{L/R\}Foot}$ of the desired plane

$$\mathbf{n}_{\{L/R\}Foot} = \mathbf{p}_{CoM} - (\hat{\mathbf{p}}_{LFoot} + \hat{o}^R_{CoM} \cdot (\mathbf{p}_{RFoot} - \hat{\mathbf{p}}_{LFoot})). \tag{2.12}$$

Accordingly, it points from the target offset $\hat{o}^R_{CoM}$ corresponding to the projected CoM, to the new CoM position.

Single Support

Provided that the human operator is in single support mode and $o^R_{CoM} = 0$ or 1, i.e., the projection of the robot's CoM lies within the left or right foot support polygon, it is sufficient to adapt the target orientation for the supporting foot for each new incoming human posture. The target positions, on the other hand, obtained after posture mapping can be adopted without modification in this mode. An appropriate support foot orientation is computed according to Eq. (2.12), with $\hat{o}^{R}_{CoM}$ being either 0 or 1.

Finally, given a modified target position and orientation for one or both feet, we determine a joint configuration for the respective kinematic chain, using the numerical IK solver defined in Eq. (2.7). Here, we solve the IK for the 5 dimensional task, specified by the target position $\hat{\mathbf{p}}_{\{L/R\}Foot}$ and orientation $\mathbf{n}_{\{L/R\}Foot}$. In contrast to posture mapping, precision is crucial for stabilization. Therefore, we run the damped least-squares IK solver for this operation until the error is found to be within a certain tolerance.

2.5 Posture Postprocessing

For increasing the robustness of our approach, we additionally implemented a set of safety routines taking into account stability issues arising from fast movements of a non-supporting end-effector, collisions between the feet and the ground as well as *self-collisions* among different end-effectors of the robot. To address the former problem, our approach limits the velocity of non-supporting end-effectors if their scheduled Cartesian velocity is found to induce a critical deviation of the CoM's motion from its desired statically stable trajectory. For the latter, we evaluate and adapt the target position of the non-support foot in single to double support mode transitions if needed, in order to avoid motions of the foot towards a position penetrating the ground. Lastly, we keep track of the distances between the robot's end-effectors and push the target positions apart in case they are found to be to close to each other. This is particularly important considering motion sequences where the hands of the operator get in touch with each other.

2.6 Implementation Details

The MVN motion capture suit by Xsens used in this work is generally capable of recording motions at a rate of 120 Hz. In our approach, however, we found a sampling rate of 30 Hz to be sufficient for accurately capturing the demonstrated human motions. Moreover, it comes along with a reduced computational effort, yielding a real-time performance with even lower motion imitation delays. For mapping postures from the human operator to the humanoid robot, as described in Section 2.3, we run the damped least-squares based IK solver (DLS) with a fixed number of 30 iterations for the arms and 5 iterations for the feet. A rough tracking of the feet is sufficient in the posture mapping process as the positions are subsequently adapted in the stabilization step to achieve static stability. The same IK solver is used in our stabilization method and executed until an accuracy of at least 1 mm or 0.033 rad is reached. For the damping factor λ used in the DLS method, we experimentally determined a value of 0.3. This choice has shown to be a good compromise between stability and convergence of the IK method. Computations were performed on a single core of a standard desktop CPU (Intel Quadcore i5-2400, 3 GHz).

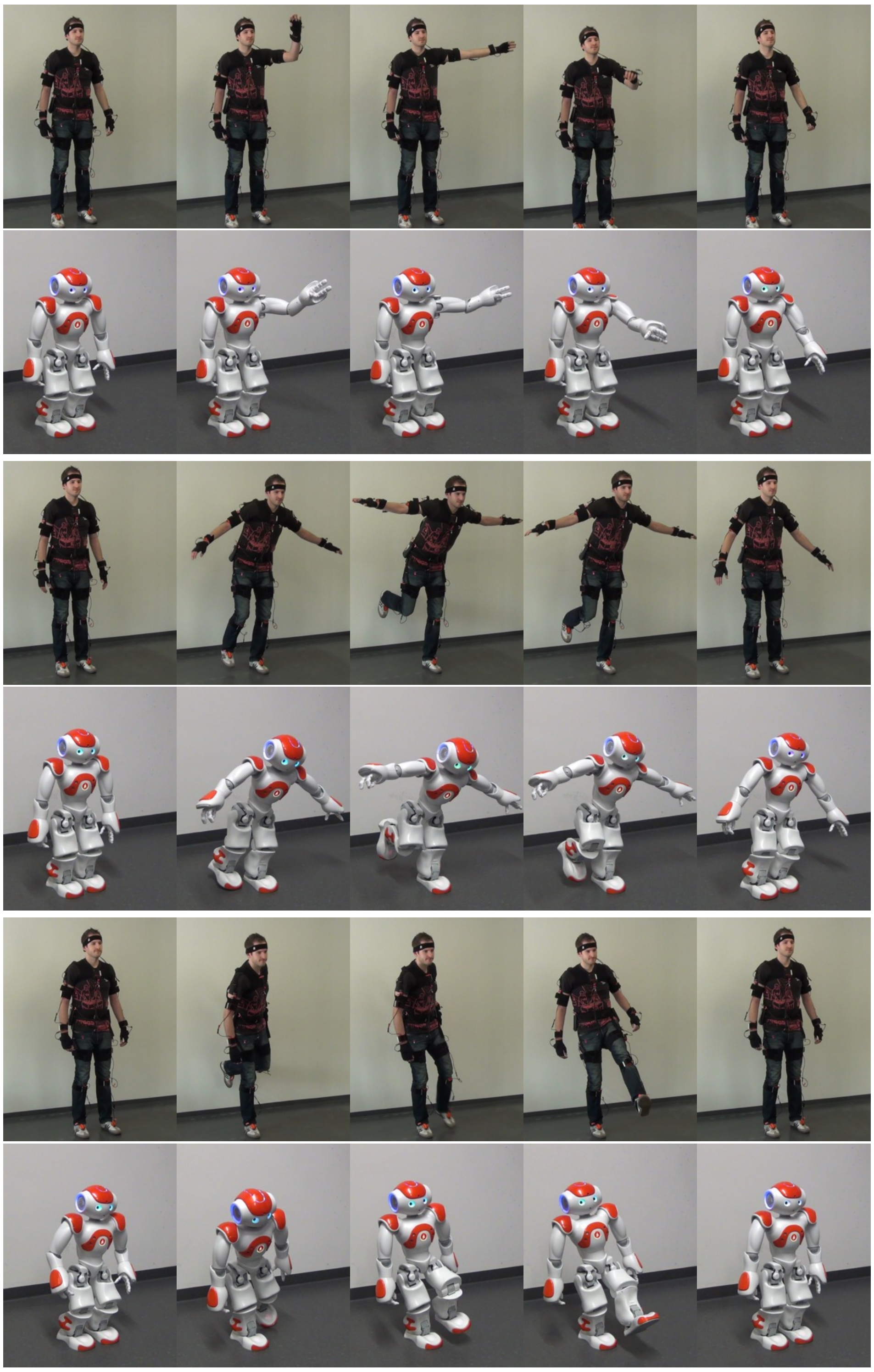


Figure 2.7: Samples of motion imitation sequences generated by our approach. The examples include arm and leg motions as well as coordinated whole-body motions in single support.

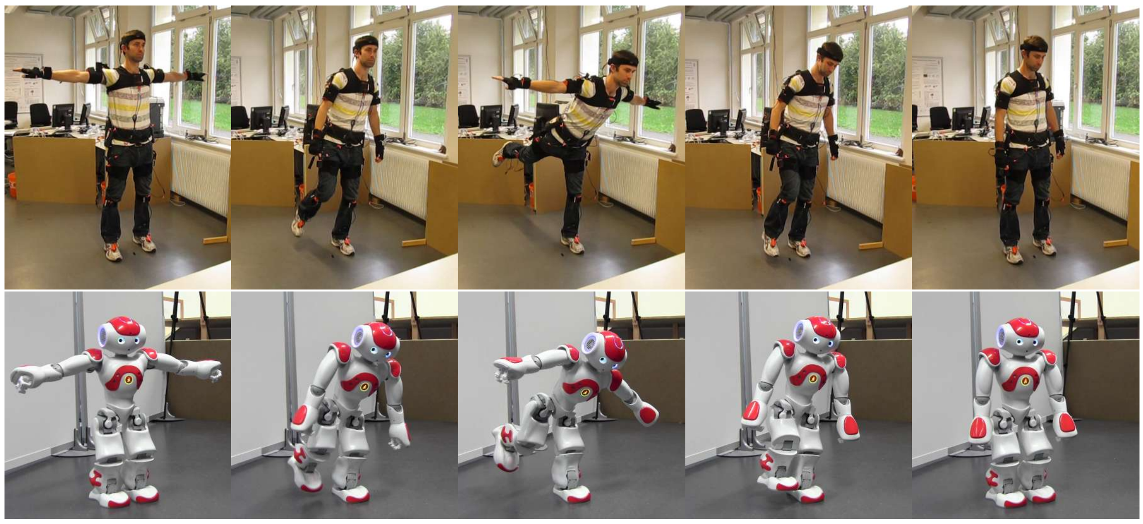


Figure 2.8: Nao humanoid imitating a human performing a motion to reach a complex single support posture. Using our approach, the robot can even keep its balance when it is in single support for longer periods of time. The leftmost image shows the calibration posture for the mapping process, introduced in Section 2.3.1.

2.7 Experimental Results

For the evaluation of our motion imitation approach, we used a Nao v4 humanoid robot developed by SoftBank Robotics. Nao is 58 cm tall, weighs 5.2 kg and has 25 degrees of freedom. A detailed description of the humanoid robot platform can be found in Appendix A.1. The human was wearing an MVN suit by Xsens for capturing the motions. This inertial sensor-based tracking device provides an accurate estimate of the human's posture from which the target positions of the end-effectors, used as input for posture mapping, are extracted.

Several samples of motion imitation sequences generated by our approach are depicted in Figure 2.7. In the following, we evaluate our system in terms of similarity of the robot motion to the demonstrated human motion, stability and computational cost. Finally, we present a teleoperation experiment as an application scenario for our approach.

2.7.1 Similarity to Human Motion

In order to quantitatively evaluate our approach regarding the similarity of the generated robot motion to the demonstrated motion of the human operator, we measured the deviation of the respective end-effector positions in a complex whole-body motion sequence involving fast arm movements and support mode changes. More detailed, the human first stretched his right foot backwards followed by raising his arms in order to maintain balance in single support. Afterwards, he returned to a calm statically stable double support posture with the arms adjacent to the body. Some snippets of the described motion sequence are illustrated in Figure 2.8. Here, the reference human postures given as input are shown alongside with the corresponding humanoid robot postures generated by our method. The first frame shows the human and robot adopting the reference posture, referred to as the T-Pose, used for the calibration process in posture mapping.

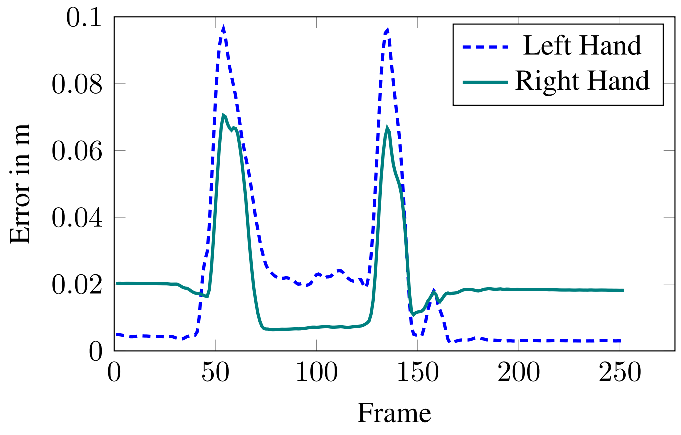


Figure 2.9: Deviation of the hand positions from the desired positions for the motion depicted in Figure 2.8. The average error is only 1.4 cm.

A measure for the similarity between the human and humanoid motion is obtained by comparing the desired end-effector trajectories generated by the human to humanoid posture mapping step (see Section 2.3) with the actual end-effector trajectories obtained after posture stabilization and postprocessing. The progression of the end-effector deviation in course of the motion sequence, depicted in Figure 2.8, is shown for the hands and feet of the robot in Figure 2.9 and Figure 2.10, respectively. The accumulated error observed for the lower and upper body end-effectors can be traced back to various sources. A first error component is induced by the joint limits weighting matrix and the constant damping factor used within the numerical inverse kinematic solver applied in posture mapping. Particularly, when imitating motions that drive the robot's joints in proximity to their range limit, slow convergence and therefore increased errors occur. Another error component, related to the feet, arises from the posture stabilization method, where stability is enforced at the expense of imitation delays. These delays are especially apparent in the case of support mode changes. Further deviations occur due to the limitation of the end-effector velocities in the stabilization and postprocessing method.

Figure 2.9 shows that the deviation of the hands from their nominal positions increases when the operator raises (Frame No.40) and lowers (Frame No.125) its arms due to our limitation of non-supporting end-effector velocities. Figure 2.10 plots the corresponding errors of the foot positions. Here, our stabilization method shifts the robot's center of mass by incrementally adapting the leg configurations in order to realize the demonstrated support mode change. Initially being in double support mode, the observed deviations from the target positions are rather small. When the operator changes to left foot single support mode (Frame No.40), the robot needs some time to shift its CoM statically stable to the left foot, while the operator continues with the motion. Simultaneously to changing the support mode, the operator performs a fast backwards movement of the right leg, inducing a growth of the error for the right foot. This effect is amplified by the end-effector velocity control of the postprocessing step. In the left support phase, the left foot keeps on balancing while the right foot attempts to catch up with its desired position (Frame No.48-152). At the end of the left support phase, the human operator moves the right foot with high acceleration forwards, i.e., towards the support foot (Frame No.125-145), before placing it back to the ground (Frame No.150). Finally, the

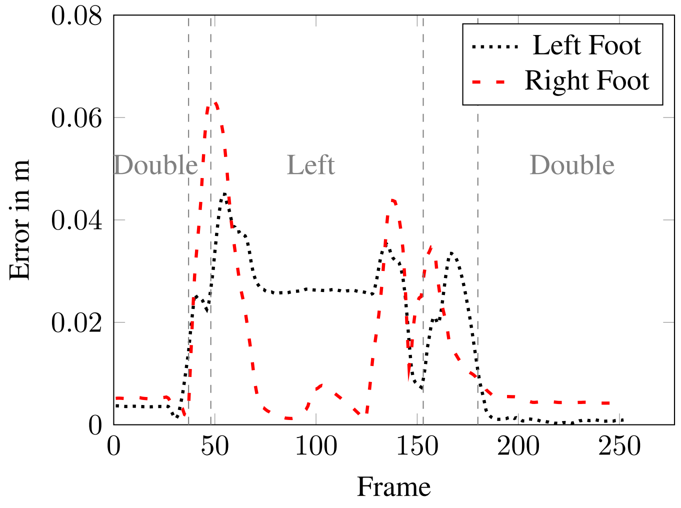


Figure 2.10: Deviation of the generated feet positions from the desired ones for the sequence depicted in Figure 2.8. The dashed vertical lines specify the beginning and end of support mode changes. The increased error in the left foot during support mode changes results from shifting the CoM within our stabilization method. Furthermore, the human right leg is moved fast backward and forward, respectively, and the robot needs a few time steps to catch up. The average error is only 1.6 cm.

CoM is shifted towards the neutral position by the stabilization procedure using both feet, resulting in a temporarily increased error in the left and the right foot (Frame No.153-178).

For evaluating the average error with respect to the given target end-effector trajectory for the complex single support sequence in Figure 2.8, we repeated the experiment 10 times in total. The average error over all repetitions is found to be 1.4 cm for the hands and 1.6 cm for the feet. Note that although the mapped human postures are adapted for the humanoid robot to ensure stability, there is almost no noticeable difference observable between the human and the corresponding humanoid motion.

2.7.2 Ensuring Stability

In this experiment, we examine our motion imitation approach regarding its performance in generating statically stable support mode transitions from the human CoM trajectory and support mode given as input. In particular, we consider the offset value o_{CoM} introduced in Section 2.4, which is the low-dimensional representation of the projected CoM. The temporal evolution of the human and robot offset for the sequence in Figure 2.8 is depicted in Figure 2.11. During the initial double support phase, the robot and human possess an identical offset, which results from their center of mass being slightly shifted towards the right foot. Afterwards, the offset of the demonstrated motion rapidly approaches a value of 0, before the left foot lifts off and single support mode is entered (Frame No.37). Once the human is in single support, the offset of the robot is forced to approach a value of 0, even though the desired human offset might be slightly different. The admissible offset velocity defined according to Figure 2.4 (right), is relatively high in the mean interval. Therefore, the robot's offset is capable to closely follow the demonstrated trajectory until it gets close to its range

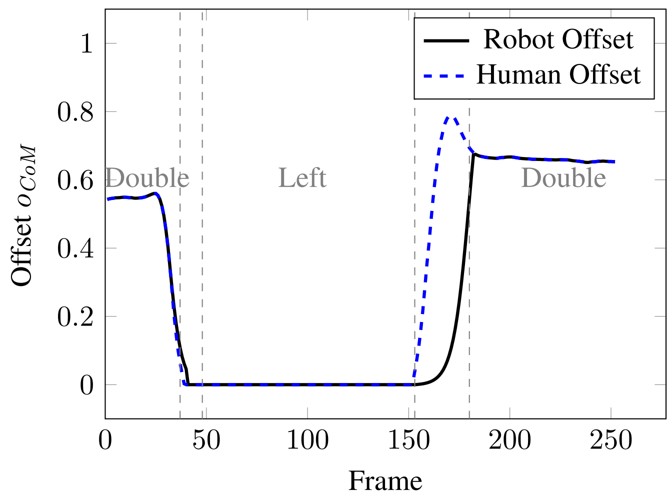


Figure 2.11: Evolution of the human and robot offset value for the motion sequence depicted in Figure 2.8. The robot's offset closely follows the human's offset. During support mode changes, the speed of shifting the robot's CoM is limited to ensure stability.

limit. In the proximity of 0 (or 1) the motion is slowed down by our offset control method, thus guaranteeing a safe statically stable arrival at the target value 0 (Frame No.37-48). Throughout the left foot single support phase the offset value remains at a value of 0. Finally, the human operator places the right foot back to the ground (Frame No.152). Analogous to the previous support mode change, our offset control forces the offset to approach its desired value with limited velocity, inducing a delay of a couple of frames (Frame No.152-180).

As shown by the experiments, our technique is capable of controlling the robot's CoM such that the generated motion is as closely as possible to the human motion while at the same time ensuring static stability. More specifically, our system generates a trajectory of the robot's CoM that closely follows the human CoM trajectory, achieves the desired support mode changes and limits the velocity of the CoM to ensure safe execution.

2.7.3 Computational Costs

In order to evaluate the computational demands of our motion imitation technique, we measured the computation times required for the individual processing steps. Here, we considered a long motion sequence composed of 1000 frames in order to obtain meaningful results. The sequence contains different movements such as stepping and reaching, as depicted in the teleoperation experiment described in Section 2.7.4. Table 2.1 specifies the mean and standard deviation as well as the maximum value encountered for the individual steps. The times listed for posture mapping include the calculation of all end-effector target positions and the determination of the corresponding joint configurations for the robot using the inverse kinematic solver. Note that the target end-effector transforms can be calculated in constant time and the number of IK iterations is fixed to a small value. The time required by posture stabilization, on the other hand, is dominated by the inverse kinematics calculations. As opposed to posture mapping, it is here crucial to reach the desired target positions with sufficient accuracy. Therefore, the IK solver is run until the error is found to be below a certain threshold. The total time includes all computations of our system, beginning when the

	Mean/Std	Max
Posture mapping	$1.04 \pm 0.35\,\text{ms}$	$1.73\,\text{ms}$
Posture stabilization	$1.85 \pm 3.50\,\text{ms}$	$27.08\,\text{ms}$
Total time	$3.30 \pm 3.57\,\text{ms}$	$29.21\,\text{ms}$
# IK iterations	54 ± 178	1784

Table 2.1: Computational effort and number of IK iterations.

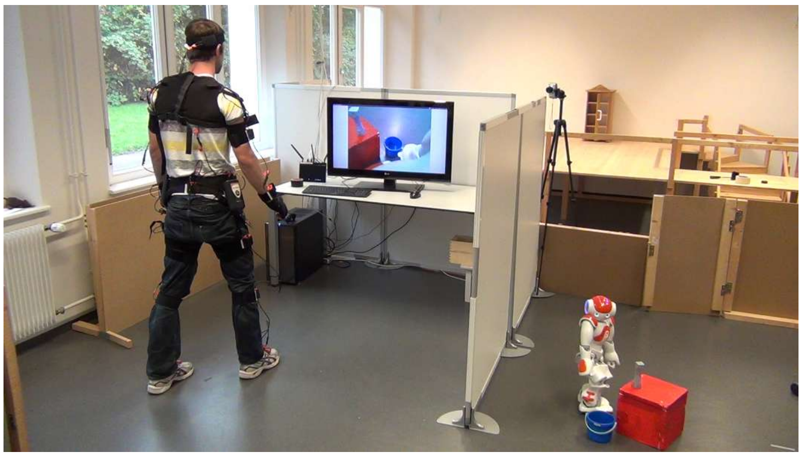


Figure 2.12: Teleoperation with visual feedback. *Left*: Human operator and live view of the robot's camera displayed on a monitor. *Right*: Teleoperated Nao humanoid.

captured human data is obtained and ending when a statically stable pose has been generated for the robot. The last row in Table 2.1 specifies the number of iterations performed by the damped least-squares solver per motion capture frame.

The calculation times are within a few milliseconds on average. Some configurations require a larger amount of iterations to converge to the desired accuracy. However, even in the worst case the computation time does not exceed the 30 ms limit required to achieve real-time performance. Given the complexity of the demonstrated motion, we can therefore conclude that a large set of whole-body motions can be safely imitated with our approach at a rate of 30 frames per second.

2.7.4 Teleoperation

In the last set of experiments, we evaluated the applicability of our system towards teleoperation of a humanoid robot for manipulation of objects in its surrounding. What makes this scenario particularly challenging is that the scene and the robot are not directly observable by the human operator. Instead, he needs to rely only on visual feedback provided by the robot's on-board camera. The camera itself, located in the head, is always pointing to the right hand of the robot, which is intended to be used for the manipulation task. This way, the operator is able to continuously monitor the distance between the hand and the target object. The setup for the two teleoperation experiments we conducted in this context is shown in Figure 2.12. Here, the operator is located in front of a screen displaying the live-view of the robot's camera.

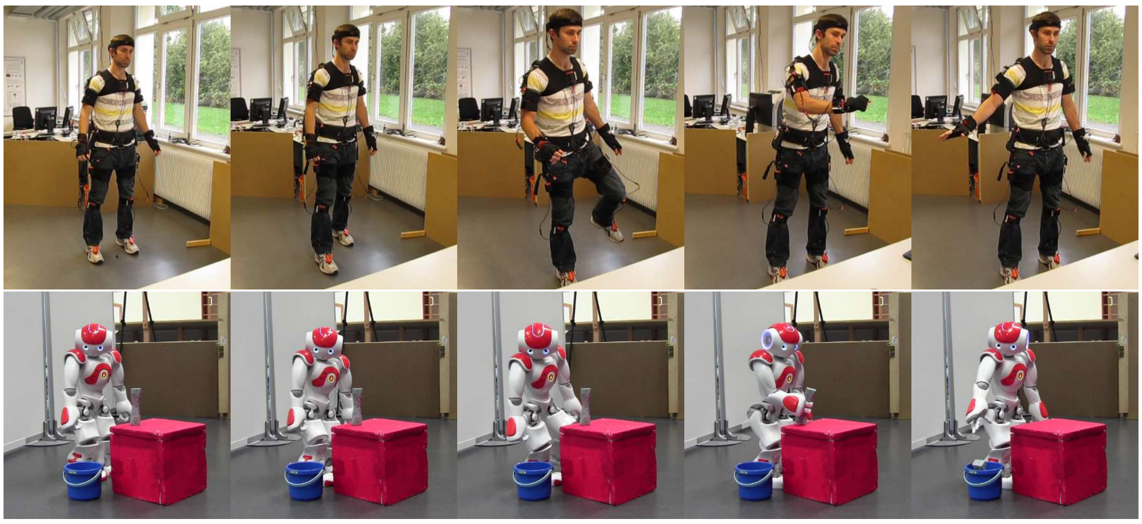


Figure 2.13: Teleoperated walking and object manipulation. A video demonstrating our imitation approach is available online at http://www.youtube.com/watch?v=AWdvffiw5Ug.

The two sequences include stepping, balancing in single support and grasping an object. In the first experiment, depicted in Figure 2.1, the operator balances on the left foot while picking up the object. Here, the robot is able to reach the distant object by leaning forward and counterbalancing the CoM shift by simultaneously stretching the right foot backwards. After successfully grasping the object, the operator returns to double support and drops the object into a bucket. For simplicity, we set the desired target orientation for the right hand to remain upright in this experiment. In the second experiment, shown in Figure 2.13, the robot is initially too far away from the object in order to instantly reach it. Nevertheless, the operator is capable to get the object within the robot's reachable workspace by performing a sequence of two steps. Afterwards, he bends forwards, stretches his right arm to pick up the object and finally drops it into the bucket. Both teleoperation experiments were successfully conducted multiple times on the real robot platform, without encountering unstable motion behavior. Note, that the robot and human operator do not necessarily need to share the same location, which makes our approach interesting for situations where manipulation actions need to be carried out in hazardous environments.

2.8 Related Work

One of the first approaches for real-time control of a humanoid robot by imitation is presented by Riley et al. [104]. For motion data acquisition, a simple visual marker system attached to the upper body is used. The authors apply inverse kinematics to estimate the human's joint angles and map them afterwards to the robot. Ott et al. [95] proposed to use a spring model, in which control points on the robot's skeleton are virtually connected to the markers on the human body. Based on the forces acting on the springs, joint angles are determined that consider the robot dynamics. The authors present experiments, in which a humanoid imitates human upper-body motions in real time. However, the legs are only actuated for the purpose of maintaining balance. The work by Dariush et al. [37] considers imitation as task space control based on low-dimensional motion primitives. The authors use a separate *Zero Moment*

Point (ZMP)-based balance controller and the lower body is only controlled so as to ensure stability.

Also Yamane and Hodgins [145] presented a control-based approach to imitate human motions with a force-controlled robot. Using this technique, joint trajectories for the whole-body of a humanoid can be generated online. The legs are also controlled to follow the human motion while maintaining stability. The only assumption is that both feet remain in ground contact. In the future, the authors plan to lift this assumption by integrating techniques to detect stepping motions and adapting the CoM trajectory in the controller according to [146]. In the latter approach, the authors proposed to predict the trajectory of the desired *Center of Pressure* (CoP) for a number of frames based on the captured motion and accordingly modify the CoM to ensure stability.

Cela et al. [19] presented a motion capture system consisting of eight sensors to measure joint angles of the legs and accelerations of the arms. The authors ensure stability during real-time imitation using a feedback control system based on data of an accelerometer placed on the robot's back. With this system, changes from double support to singe support are possible, however, due to the limited set of sensors, no complex motions can be imitated. Recently, Vuga et al. [141] introduced an approach for dynamical stable imitation of human motions. The authors use a separate controller for the lower body that ensures stability by permitting imitation in the null space of the balance controller only. This way, imitation of walking motion sequences are possible.

Stanton et al. [127] described a learning approach to determine a kinematic mapping between the human and the robot. This technique relies on an initial training phase in which the human is asked to imitate the motions of the robot. Afterwards, the human can teleoperate the robot in real time. Since no balance controller is used in this approach, the range of motions the robot is able to imitate is very limited.

Suleiman et al. [131] focus on the imitation of upper body motions. The authors treat the imitation as a constrained optimization problem on a given sequence of captured human motions. Nakaoka et al. [92] consider dance movements. In this approach, motion primitives and their parameters are learned offline from observed human motions. Here, the leg motions are not directly imitated, but generated from the primitives. In a latter work, Nakaoka et al. [93] use a set of models for different leg motions of the robot to ensure that characteristic motions are executed stably during imitation of the dance movement. Also here, the motion models and their parameters are learned in an offline step. For imitation, the particular type of motion primitive is recognized from captured motion data and the leg trajectory is chosen accordingly, so that the robot can safely execute the corresponding sequence. Kim et al. [73] also focus on dance movements and use an offline optimization step for determining a kinematic mapping between the human and the robot, used to ensure stability of imitated whole-body motions, including support mode changes. During execution, three online controllers are used for balancing and soft stepping. Chalodhorn et al. [20] proposed to apply dimensionality reduction that transforms the high-dimensional human motion data in a low-dimensional subspace. The offline optimization of the motions, which takes into account the robot dynamics and stability, is subsequently performed in the reduced subspace. The authors applied the approach to the task of ”learning to walk” by imitation.

2.9 Conclusion

In this chapter, we presented a technique for real-time imitation of human whole-body motions by humanoid robots. In order to find robot postures that resemble the demonstrated human motion, our approach uses a compact representation of the human kinematic model in conjunction with numerical inverse kinematics computations. To achieve safe imitation of challenging motion sequences in real time, we introduced a method that generates statically stable motion transitions by incrementally shifting the robot's CoM towards its target location. At the same time, the mapped human posture is modified as little as possible to maintain maximal similarity to the captured motion. The major scientific contribution of our system is that it also allows for single support phases, where the center of mass needs to be actively balanced over the support foot for an extended period of time. Real world experiments with a Nao humanoid and a MVN suit by Xsens demonstrated the capability of our approach to reliably generate safe motions for the robot, closely following the human reference also for long and complex motion sequences. The maximum time required to generate a robot posture for a single motion capture frame is less than 30 ms on average. Therefore, real-time performance is achieved, which is a necessary prerequisite for the deployment in teleoperation applications. A video demonstrating the capabilities of our approach is available online[2].

An interesting aspect for future work is the inclusion of the end-effector orientations for the upper body end-effectors in the imitation process. Due to the fact that the arm kinematic chains are composed of only 5 DOF, whereas the task is 6 dimensional in this case, this would require using whole-body control in the posture mapping step. In turn, posture stabilization would aim to achieve the desired CoM position in the Nullspace of the upper body Jacobian or defines the CoM shift as the highest-priority goal in a task priority control scheme.

Another extension, though complex to implement, would be to consider the robot body dynamics in the imitation process. Although this would come with a higher risk of falling due to imprecise estimates of the dynamics, it may yield a further reduction of the imitation delays and an increased similarity between the robot motion and the demonstrated motion. An issue to be considered is that the robot body dynamics, as defined by the hardware specification, cannot keep up with the skills of a human operator to dynamically change between whole-body postures. Thus, the range of motions the robot is capable to imitate always remains limited with respect to the human.

An aspect not taken into consideration in this chapter is the actual value of the prescribed motion. Here, we assumed that the motion samples are representative for a desirable, natural human kinesic behavior that should be adopted by robotic platforms without putting them into question. In the following chapter, we will investigate on the differences between the motions of different human subject. In particular, we will examine the effect of neurological impairments, which are well known to induce motor control deficits, on the human musculoskeletal system.

[2]http://www.youtube.com/watch?v=AWdvffiw5Ug

Chapter 3

Learning Motor Control Parameters for Motion Strategy Analysis

The neurological impairments of Parkinson's disease (PD) patients are well known to go along with motor control deficits, e.g., tremor, rigidity, and reduced movement. However, not much is known about the motor control parameters affected by the disease. In this chapter, we present a novel approach to human motion analysis using motor control strategies with joint weight parameterization. We record the motions of healthy subjects and PD patients performing a hand coordination task with the whole-body XSens MVN motion capture system. For our motion strategy analysis we then follow a two step approach. First, we perform a complexity reduction by mapping the recorded human motions to a simplified kinematic model of the upper body. Second, we reproduce the recorded motions using a Jacobian weighted damped least-squares controller with adaptive joint weights. We developed a method to iteratively learn the joint weights of the controller with the mapped human joint trajectories as reference input. Finally, we use the learned joint weights for a quantitative comparison between the motion control strategies of healthy subjects and PD patients. Other than expected from clinical experience, we found that the joint weights are almost evenly distributed along the arm in the PD group. In contrast to that, the proximal joint weights of the healthy subjects are notably larger than the distal ones.

Within the last decade, recording and analyzing human motion data has gained an increased interest in a variety of research fields, ranging from medical science, neuroscience, computer graphics, to robotic applications. The way the data is used, however, differs between these fields. The former fields are mainly interested in understanding human motion and its underlying principles in order to improve therapy methods for patients with neurological or physiological diseases, whereas computer graphics and robotics aims at generating human-like motions for artificial multi-joint robotic systems to improve the appearance, coexistence, collaboration, and safety in human-robot interaction scenarios. So far, robotic research has used human motion data to map hand/end-effector and joint trajectories to robotic platforms for teleoperation applications, similar to the work described in Chapter 2, or applied the data as reference input for motion planning algorithms and control schemes, rather than

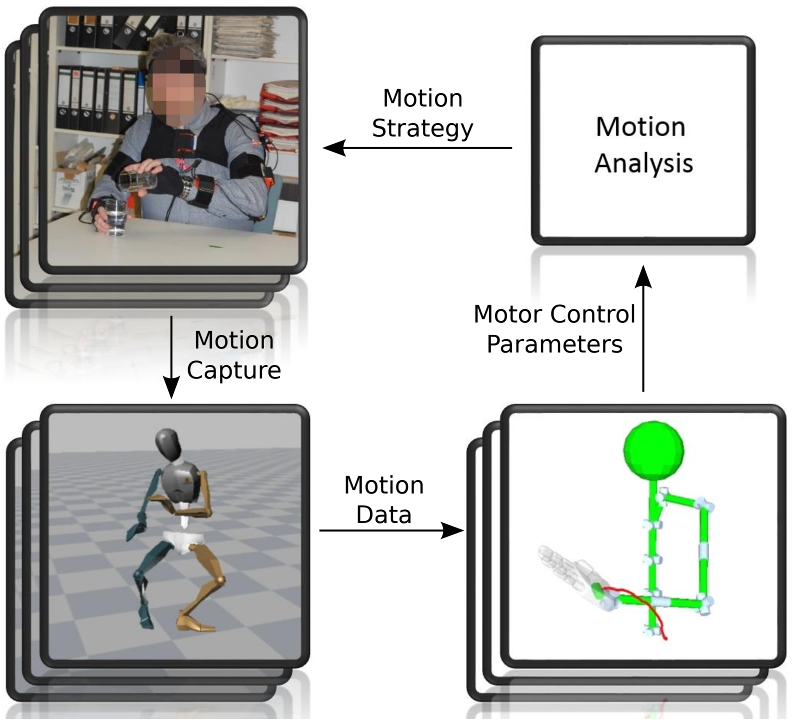


Figure 3.1: Motion strategy analysis: experimental setup for data collection (top left), motion representation (bottom left), and motion model used to learn the motor control parameters (bottom right).

investigating the underlying motion control strategy that generated the observed human trajectories. Altogether, these approaches have in common that they consider the human motion samples as a desirable, natural human kinesic behavior to be adopted by robotic platforms or reflected by schemes modeling the human musculoskeletal system. Contrary, we aim in this work towards analyzing the distinction between the human motion samples and separate them into different groups, namely the motion of healthy subjects and Parkinson's disease (PD) patients whose motor control is affected by a neurological disorder. Note that the end-effector trajectories of different subjects performing the same task, e.g., moving an object from one location to another, are typically similar, whereas the joint trajectories generated by the individual motor control scheme might differ significantly depending on the constitution of the subject.

In this chapter, we present a novel approach to investigate the hypothesis that members of the respective group share a common motor control strategy to select among the infinite set of joint trajectory solutions achieving a given task. We focus our analysis on the hand coordination task of pouring water from one glass into another, as depicted in Figure 3.1. Our approach relies on motion data of healthy and PD subjects collected using a whole-body motion capture suit. In a first step, we map the data to a simplified scale-adaptive artificial model of the human upper body in order to perform dimensionality reduction. Afterwards, we use this model to track the mapped human end-effector trajectories based on a variable damped least-squares control scheme with adaptive joint weights. The joint weight parameterization of the control scheme allows executing the same end-effector trajectory with arbitrary joint trajectories that, in turn, reflect different motor control strategies. To determine the respective

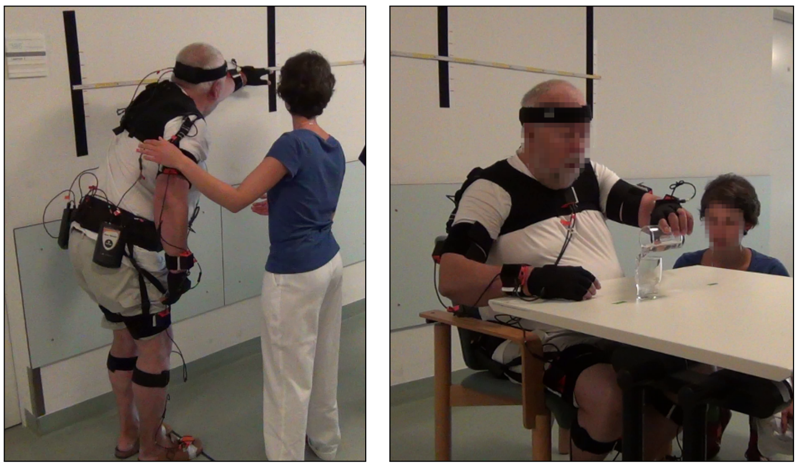


Figure 3.2: XSens MVN motion capture suit used to record motions of healthy and PD subjects in clinical experiments. *Left:* PD subject performing a functional reach test. *Right:* Hand-coordination task of pouring water from one glass into another.

motor control strategy for each subject, we developed a technique to iteratively learn the joint weights of the controller to match the observed joint trajectories. Based on the resulting joint weights, we carry out a quantitative comparison of the differences between the motions of PD patients and healthy subjects and infer their motion control strategy principles. According to our results, it turns out that we can differentiate between two motor control strategies, referred to as the *proximal* and *distributed* motion strategy, adopted by the two groups for task achievement. To the best of our knowledge, this is the first approach learning indicative motor control parameters in order to explain the effects of neurological diseases onto the musculoskeletal system in human subjects.

3.1 Motion Database

The basis of our motor control analysis is a database of motions recorded from healthy and PD subjects using the whole-body motion capture suit, described in Section 2.1.1 (XSens MVN [105]). The deployment of the suit for clinical trials is shown in Figure 3.2. In the following, we will briefly explain how the motion capture data is acquired and give details about the motor control task considered for the purpose of our investigation.

3.1.1 Data Acquisition

In order to examine the underlying human motor control behavior, we sample the IMU sensor data provided by the motion capture suit at full resolution, i.e., with 120 Hz. A visualization of the corresponding motions is provided by the *MVN Studio* software, which maps the recorded motions to an artificial human avatar composed of 23 segments and 22 joints. Since we are investigating motor control deficits of PD patients in hand coordination tasks, we are using only the data of the upper body, i.e., the trajectories of the spine and arm joints $\mathbf{q}^h$. Instead of acquiring the data online as done in Chapter 2, we export it in a database of *Moven Open XML* (MVNX) files.

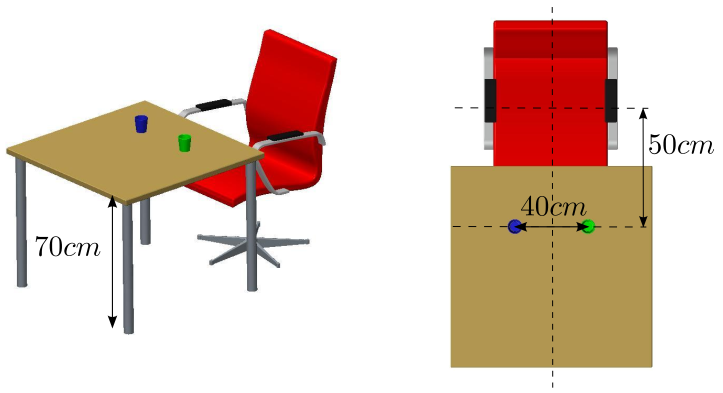


Figure 3.3: Hand coordination task: Subjects are asked to pour water from one glass (green, at the left of the subject) into another, empty glass (blue).

3.1.2 Motor Control Task

To analyze the motions of PD patients and compare them to motions of healthy subjects, we set up a task, where water needs to be poured from one glass into another using the left hand (see Figure 3.3). The task has been designed such that the resulting motions are composed of two sections, the coarse subtask of transitioning the glass filled with water to the empty glass and the delicate subtask of pouring the water from one glass into the other without spilling.

3.2 Motion Representation

The intrinsic model of the motion capture system for representing the motions recorded from the upper body of subjects, i.e., the spine and arm kinematic chain, is composed of 8 spherical joints, and thus 24 degrees of freedom (DOF) (see Figure 2.2). Empirical analysis of the recorded data, however, revealed that contrary to the model presented some human joints have less than three rotation axes, thus resulting in only 19 DOF actively contributing to the observed motions. Due to this fact, we built a simplified artificial model used as a compact representation for the motions of a human's upper body in hand coordination tasks.

3.2.1 Artificial Model of the Human Upper Body

The artificial kinematic model, shown in Figure 3.4, is composed of the previously mentioned 19 DOF dominating the execution of the hand coordination task. Here, the first 8 DOF represent the motion of the spine and the remaining 11 DOF the motion of the kinematic arm chain. The configuration of the entire model is defined by the following vector of joint angles

$$\mathbf{q}^h = (\mathbf{q}_{spine}, \mathbf{q}_{arm})^\top \in \mathbb{R}^{19}, \tag{3.1}$$

with

$$\mathbf{q}_{spine} = (s_1^\varphi, s_1^\theta, s_2^\varphi, s_2^\theta, s_3^\varphi, s_3^\theta, s_4^\varphi, s_4^\theta) \in \mathbb{R}^8, \tag{3.2}$$

$$\mathbf{q}_{arm} = (a_{C7}^\varphi, a_{C7}^\psi, a_{sh}^\theta, a_{sh}^\varphi, a_{up\_arm}^\varphi, a_{el}^\psi, a_{el}^\theta, a_{f\_arm}^\varphi, a_{wr}^\psi, a_{wr}^\theta, a_{wr}^\varphi) \in \mathbb{R}^{11}, \tag{3.3}$$

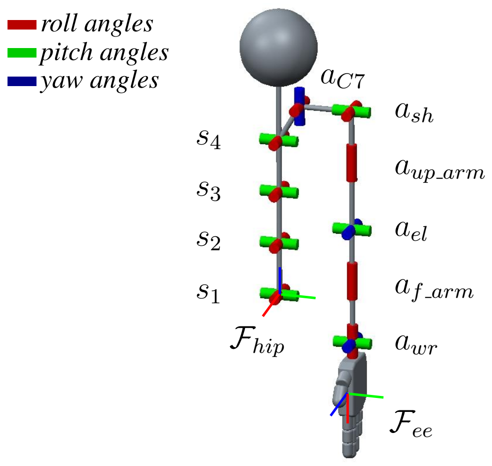


Figure 3.4: Artificial kinematic model of the human upper body.

where $\mathbf{q}_{spine}$ and $\mathbf{q}_{arm}$ denote the configurations of the spine and arm chain, respectively. Here, the vector elements s_j^k, a_j^k refer to the roll, pitch, and yaw angle $k \in \{\varphi, \theta, \psi\}$ of joint j. Moreover, we use for each subject measurements from the T-pose, depicted in Figure 2.3, in order to set the length of the links for the artificial kinematic model equal to the length of corresponding links of the human.

3.2.2 Motion Mapping

In order to map the recorded motions to our simplified representation, we assign the joint trajectories $\mathbf{q}_i^h$ of the 19 dominant human joints to the respective joints of our artificial model. Furthermore, we record the trajectory of the hand $\mathbf{x}_e^h = (\mathbf{x}_e, \dot{\mathbf{x}}_e)^\top \in \mathbb{R}^{12}$, referred to as the end-effector in the following, over the entire motion sequence. The end-effector pose trajectory $\mathbf{x}_e \in \mathbb{R}^6$ of frame $\mathcal{F}_{ee}$, expressed with respect to the fixed frame $\mathcal{F}_{hip}$, is obtained by solving the forward kinematics for each configuration $\mathbf{q}^h(t)$ captured by the system at time t. The end-effector velocity trajectory follows from the backward difference $\dot{\mathbf{x}}_e(t) = (\mathbf{x}_e(t) - \mathbf{x}_e(t-1))/\Delta t$, where $\Delta t \approx 8$ms considering a sampling rate of 120Hz.

We use the same kinematic representation with the trajectory information $\mathbf{x}_e^h$ and $\mathbf{q}^h$ in the following to implement a mathematical model that learns the underlying characteristics of the motions in terms of motor control parameters.

3.3 Learning Motor Control Parameters

When applying a mathematical model for motor control, there exist infinite solutions of joint trajectories that achieve the desired human end-effector trajectory $\mathbf{x}_e^h$. A common approach is to select a solution, generated through the optimization of a specific objective function, that is assumed to mirror the underlying motor control principles and thus the resulting joint motions $\mathbf{q}^h$ adopted by human beings. While the minimum-jerk model [133] has found to yield a close fit to natural human arm motions, it does not allow to make any statements concerning the differences between the motion of two different subjects. The motor control analysis presented in this work relies on an iterative method for fitting a mathematical model to the observed human motion by adapting motor control parameters based on joint trajectory error information.

3.3.1 End-Effector Trajectory Tracking

As a first step, our approach reproduces the recorded end-effector trajectory $\mathbf{x}_e^h$ using a mathematical model, i.e., a controller, for motion generation. A classical approach from the literature to generate a desired end-effector trajectory for a kinematic structure is based on the inverse differential kinematics control scheme [121]. In this approach, a desired end-effector trajectory $\mathbf{x}_d$ is tracked by numerical integration of joint velocities over a given interval Δt, with $\mathbf{q}(0)$ being the initial configuration. The joint values required at time t_{k+1} to move the end-effector along the desired trajectory from pose $\mathbf{x}_d(t_k)$ to $\mathbf{x}_d(t_{k+1})$ are computed as

$$\mathbf{q}(t_{k+1}) = \mathbf{q}(t_k) + \dot{\mathbf{q}}(t_k)\Delta t, \tag{3.4}$$

with

$$\dot{\mathbf{q}}(t) = \mathbf{J}^{-1}(\mathbf{q}(t))(\dot{\mathbf{x}}_d(t) + \mathbf{K}\mathbf{e}(t)), \tag{3.5}$$

where $\mathbf{J}^{-1}$ is the Jacobian inverse evaluated in configuration $\mathbf{q}$, $\dot{\mathbf{x}}_d$ the desired velocity along the end-effector path, and $\mathbf{K}$ a positive definite diagonal gain matrix whose scalar values can be chosen to give individual weights to the components of the error $\mathbf{e}$. Note, that the time dependency of the variables is omitted from now on in favor of a compact notation. The operational space error between the desired $\mathbf{x}_d$ and the current $\mathbf{x}_c$ end-effector position and orientation is denoted as $\mathbf{e}$ and defined as follows

$$\mathbf{e} = \mathbf{x}_d - \mathbf{x}_c. \tag{3.6}$$

$\mathbf{e}$ accounts for the numerical drift of the solution involved in the integration process in Eq. (3.4). This ensures that the end-effector pose corresponding to the computed joint variables matches the desired one. Inversion of the Jacobian, however, is only feasible if the number of operational space variables r is equal to the number of joint space variables n, i.e., if $\mathbf{J}$ is a square matrix. When $r < n$, as in our case where $r = 6$ and $n = 19$, a manipulator is said to be redundant and we need to refer to a modified control scheme. A solution scheme for redundant manipulators is obtained by

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}_{dls}(\dot{\mathbf{x}}_d + \mathbf{K}\mathbf{e}) + (\mathbf{I}_n - \mathbf{J}^{\dagger}_{dls}\mathbf{J})\dot{\mathbf{z}}, \tag{3.7}$$

where the Jacobian inverse in Eq. (3.5) has been replaced with the damped least-squares pseudoinverse, defined as

$$\mathbf{J}^{\dagger}_{dls} = \mathbf{J}^{\top}(\mathbf{J}\mathbf{J}^{\top} + \lambda^2\mathbf{I}_r)^{-1}. \tag{3.8}$$

The term λ^2 represents a dynamic damping factor used for stabilization of the solution in the vicinity of kinematic singularities, where the Jacobian becomes ill-conditioned from a numerical viewpoint. In accordance with [23], we use the following definition

$$\lambda^2 = \begin{cases} 0, & \text{if } \sigma \geq \epsilon \\ (1 - (\frac{\sigma}{\epsilon})^2)\lambda^2_{max}, & \text{if } \sigma < \epsilon \end{cases} \tag{3.9}$$

where σ is the manipulability measure evaluated at each configuration [148], λ_{max} is the maximum damping factor, and ϵ is the activation threshold, respectively. Using an activation threshold ensures that damping is only applied when needed. Moreover, we consider a second term in Eq. (3.7) projecting a gradient $\dot{\mathbf{z}}$ into the null-space of the inverse differential kinematics solution. This gradient can be used to fulfill additional tasks without perturbing the end-effector trajectory tracking performance. Here, we choose $\dot{\mathbf{z}}$ such that the joint values of our model are kept within the range of human joints, thus respecting the natural kinematic constraints of the human musculoskeletal system. Following the approach presented by Chaumette and Marchand [22], we define the gradient as follows

$$\dot{z}_i = \begin{cases} \frac{q_i - \hat{q}_{i_{max}}}{\tilde{q}_i}, & \text{if } q_i > \hat{q}_{i_{max}} \\ \frac{q_i - \hat{q}_{i_{min}}}{\tilde{q}_i}, & \text{if } q_i < \hat{q}_{i_{min}} \\ 0, & \text{else,} \end{cases} \tag{3.10}$$

where $\dot{z}_i$ is the joint limit gradient for the i-th joint. Here, q_i indicates the current joint value and $\tilde{q}_i = q_{i_{max}} - q_{i_{min}}$ its respective absolute joint range. The variables $\hat{q}_{i_{min}}$ and $\hat{q}_{i_{max}}$ are used to select an upper and lower threshold for the joint limit gradient activation, defined as

$$\hat{q}_{i_{min}} = q_{i_{min}} + \gamma\tilde{q}_i, \tag{3.11}$$

$$\hat{q}_{i_{max}} = q_{i_{max}} - \gamma\tilde{q}_i, \tag{3.12}$$

with a fixed value for $\gamma \in [0.0, 0.5]$. To track the human end-effector trajectory with the damped least-squares control scheme, we set $\mathbf{x}_d = \mathbf{x}_e$ and $\dot{\mathbf{x}}_d = \dot{\mathbf{x}}_e$ and choose the first configuration of the recorded motion $\mathbf{q}^h(0)$ at time $t = 0$ as the initial configuration $\mathbf{q}(0)$ in our control setup.

3.3.2 Adaptive End-Effector Trajectory Tracking

The joint trajectories generated by Eq. (3.7) follow from the minimization of a specific cost functional and represent only one possible solution to track a desired end-effector trajectory $\mathbf{x}_e^h$. Alternative solutions can be obtained by adding further objective functions, i.e., optimizing the motion with respect to different criteria. In this work, we follow the approach of Schinstock et al. [113], that parameterizes the control scheme using joint weights in order to be able to generate arbitrary joint trajectory solutions. To do so, we use in Eq. (3.7) an extended variant of $\mathbf{J}_{dls}^{\dagger}$, referred to as the weighted damped least-squares pseudoinverse

$$\mathbf{J}_{wdls}^{\dagger} = \mathbf{J}_w^{\top}(\mathbf{J}_w\mathbf{J}_w^{\top} + \lambda^2\mathbf{I}_r)^{-1}, \tag{3.13}$$

with

$$\mathbf{J}_w = \mathbf{J}\mathbf{W}_q, \tag{3.14}$$

where $\mathbf{W}_q$ is a $n \times n$ diagonal matrix of joint weights for a kinematic model such as the one depicted in Figure 3.4, defined as

$$\mathbf{W}_q = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & w_n \end{bmatrix}. \tag{3.15}$$

Algorithm 1: Joint Weights Learning Algorithm

```
Input  : Reference joint and end-effector trajectory of the human $\mathbf{q}^h$, $\mathbf{x}_e^h$
Output : Joint weight vector $\mathbf{w}$
1  $\mathbf{w}$, $\mathbf{w}^{inc}$, $\mathbf{w}^{grad}$ $\leftarrow$ initJointWeights()
2  for $i = 1$ to maxIter do
3      jointIdx $\leftarrow 0$
4      while jointIdx != numJoints do
5          $\mathbf{q}^c \leftarrow$ runControlJWDLS($\mathbf{w}$, $\mathbf{x}_e^h$, $\mathbf{q}^h(0)$)
6          weightUpdate $\leftarrow$ updateJointWeight($\mathbf{q}^c_{jointIdx}$, $\mathbf{q}^h_{jointIdx}$, $w^{grad}_{jointIdx}$)
7          if (weightUpdate = FALSE) then
8              jointIdx $\leftarrow$ jointIdx + $1$
9          end
10     end
11 end
12 return $\mathbf{w}$
```

Solving Eq. (3.7) using weighted damped least-squares yields $\dot{\mathbf{q}}_w$, from which an approximated solution for $\dot{\mathbf{q}}$ is obtained by

$$\dot{\mathbf{q}} = \mathbf{W}_q \dot{\mathbf{q}}_w. \tag{3.16}$$

Such a parameterization of the control law allows for modeling different motor control strategies that all track the same desired end-effector trajectory. In case of $\mathbf{W}_q$ being the identity matrix, Eq. (3.13) coincides with the damped least-squares solution defined in Eq. (3.8). On the other hand, lower weights can be chosen for the joints of the spine to generate a motion that is dominated by the joints of the arm kinematic chain. In the following, we present an iterative scheme to determine the values for the joint weights w_i, required to replicate the motions recorded from healthy and PD subjects as closely as possible.

3.3.3 Joint Weights Learning

The approach to determine the joint weights, given a human end-effector trajectory $\mathbf{x}_e^h$ and joint trajectory solution $\mathbf{q}^h$, is described as pseudocode in Alg. 1. In line 1 of Alg. 1 the joint weight $\mathbf{w}$, weight increment $\mathbf{w}^{inc}$ and gradient direction $\mathbf{w}^{grad}$ vectors are initialized. Afterwards, *maxIter* runs of the joint weights learning algorithm are performed, where each iteration corresponds to the stepwise optimization of all joints weights along the kinematic chain of our model. At the beginning of each iteration the index *jointIdx* is set to 0, indicating that we start the weights optimization process for the first joint of the spine segment, i.e., s_1^{φ} in Eq. (3.2). Using the initial configuration $\mathbf{q}^h(0)$ and the current joint weights $\mathbf{w}$, we track the human end-effector trajectory $\mathbf{x}_e^h$ by running the weighted damped least-squares control scheme presented in Section 3.3.2 and obtain the joint trajectory solution $\mathbf{q}^c$ (line 5 of Alg. 1). Subsequently, the joint trajectories $\mathbf{q}^c_{jointIdx}$ and $\mathbf{q}^h_{jointIdx}$ of the *jointIdx*-th joint, generated by the controller and the human, respectively, are used as input for the *updateJointWeight* function, described in Alg. 2. The additional input parameters $rmse_{thr}$ and $\Delta rmse_{thr}$ define thresholds for the error between the control-based and the recorded human joint trajectories.

Algorithm 2: Update step of the joint weights learning algorithm

```
Input  : Reference joint trajectory of human $\mathbf{q}_i^h$ and trajectory generated by the
         controller $\mathbf{q}_i^c$, joint weight gradient direction $w^{grad}$, thresholds for absolute
         value and change of root-mean-square error $rmse_{thr}$, $\Delta rmse_{thr}$
Output : Updated joint weight $w_i$, return status of update
1  $rmse_i^t \leftarrow$ computeJointTrajectoryError($\mathbf{q}_i^c$, $\mathbf{q}_i^h$)
2  $\Delta rmse_i^t \leftarrow rmse_i^t - rmse_i^{t-1}$
3  if ($rmse_i^t < rmse_{thr}$ or $\Delta rmse_i^t < \Delta rmse_{thr}$) then
4      return FALSE
5  else
6      if ($\Delta rmse_i^t > 0$) then
7          $w_i^{grad} \leftarrow -w_i^{grad}$
8          $w_i^{inc} \leftarrow$ incScaleFactor $* w_i^{inc}$
9      end
10     $w_i \leftarrow w_i + (w_i^{grad} * w_i^{inc})$
11     return TRUE
12 end
```

Here, we determine in a first step the root-mean-square error $rmse_i^t$ between the trajectories $\mathbf{q}_i^c$ and $\mathbf{q}_i^h$ of joint i and the change of that error $\Delta rmse_i^t$ with respect to the one computed in the previous iteration $rmse_i^{t-1}$ (lines 1-2 of Alg. 2). If $rmse_i^t$ or $\Delta rmse_i^t$ is below the thresholds $rmse_{thr}$ or $\Delta rmse_{thr}$, respectively, the algorithm returns *FALSE*, indicating that the similarity between the control-based and human solution for the motion of joint i has either reached a satisfactory level or cannot be further improved through joint weight adaption (lines 3-4 of Alg. 2). Otherwise, we evaluate whether the root-mean-square error has been decreased by the last joint weight modification performed at $t-1$ (line 6 of Alg. 2). If that is not the case, the joint weight gradient direction $w_i^{grad} \in \{1, -1\}$ is switched and the current joint weight increment w_i^{inc} is reduced by multiplying it with a constant factor *incScaleFactor* (lines 7-8 of Alg. 2). This factor helps to avoid undesired oscillation of the joint weight and to ensure convergence of the learning algorithm. If the joint weight update performed in the last iteration, has improved the similarity between the control-based and human joint trajectory the values of the variables w_i^{grad} and w_i^{inc} remain the same. In a final step, the weight w_i of joint i is increased or decreased depending on the current gradient direction w_i^{grad} and *TRUE* is returned, indicating that the joint weight has been modified (lines 10-11 of Alg. 2).

Depending on the weight update status *weightUpdate* returned by the *updateJointWeight* function, the joint weights learning algorithm proceeds in two different ways (line 6 of Alg. 1). If *weightUpdate* $=$ *TRUE*, the weighted damped least-squares controller is run again with the new joint weight w_i followed by another joint trajectory error evaluation. On the other hand, if *weightUpdate* $=$ *FALSE*, the variable *jointIdx* is incremented to consider the weight of the next joint along the chain for the optimization process (lines 7-8 of Alg. 1). When the maximum number of iterations *maxIter* is reached the final joint weight vector $\mathbf{w}$ learned for the subject is returned (line 12 of Alg. 1). Note, that the joint trajectories generated by the control scheme are not independent from each other, i.e., an improvement in similarity to the human motion achieved for a specific joint trajectory solution $\mathbf{q}_i^c$ by modification of the respective joint weight w_i may deteriorate the joint trajectory solution $\mathbf{q}_j^c$ of another joint j. Therefore, we run the entire weight optimization process *maxIter* times to find a compromise between conflicting joint weights, mutually deteriorating each others similarity to the recorded human joint trajectory solutions.

3.4 Implementation Details

The artificial human upper body model is generated in ROS (Robot Operating System) by defining an *urdf* file (Universal Robot Description Format). For recording the human end-effector trajectories, we perform forward kinematics using the KDL library [124]. The error gain matrix $\mathbf{K}$ used in Eq. (3.5) is set to the identity. For the joint limits avoidance task, we set the activation parameter $\gamma = 0.2$. The parameters used to determine the damping factor in Eq. (3.9) are set to $\lambda_{max} = 0.04$ and $\epsilon = 0.0008$. For the weight learning algorithm, we assign an initial value of 1 to all vector elements of $\mathbf{w}$ and $\mathbf{w}^{grad}$. The weight increment $\mathbf{w}^{inc}$ is set to 0.8 for all joints. For the weight reduction factor, we use *incScaleFactor* $= 0.8$ and for the joint trajectory error thresholds $rmse_{thr} = 0.02$ and $\Delta rmse_{thr} = 0.001$. In total, we perform *maxIter* $= 10$ runs of the weight optimization process.

3.5 Experimental Results

In the following, we present experimental results for a database composed of motions recorded from healthy and PD subjects. The results include an evaluation of the trajectory tracking performance of our controller as well as an analysis of the joint weights learned by our algorithm for two different groups, PD patients and healthy subjects.

3.5.1 Experimental Setup

The two groups were recruited from the clinic's movement disorders outpatient clinic for the study of the hand coordination task, described in Section 3.1.2. All participants gave their written informed consent and their data was pseudonomized at study inclusion, all in accordance with the Helsinki Declaration and to the local ethics committee (Ethikkommission der Medizinischen Fakultät der Ludwig-Maximilians-Universität).

Six PD patients participated in this study. They were 2 female and 4 male ranging from 48 to 74 (mean 65) years of age. None had any additional disorder influencing postural control. Patients were on their regular medication in ON state, 2 had deep brain stimulation. Half of the patients had pathological side *left* and *right*, respectively. The momentary state of patients' mobility was assessed just prior to the experiment with the Unified Parkinson's Disease Rating Scale (UPDRS mean 22.33±12.94 SD). Six control subjects were recruited from relatives of the authors and (former) university personnel, 3 female and 3 male ranging from 48 to 62 (mean 57) years of age. None had history of neurological disorders of any sort or orthopedic disorders requiring surgery or regular medication. All subjects were right-handed. Each subject performed six repetitions of the task, leading to an overall database of 36 motions for the healthy and PD group, respectively.

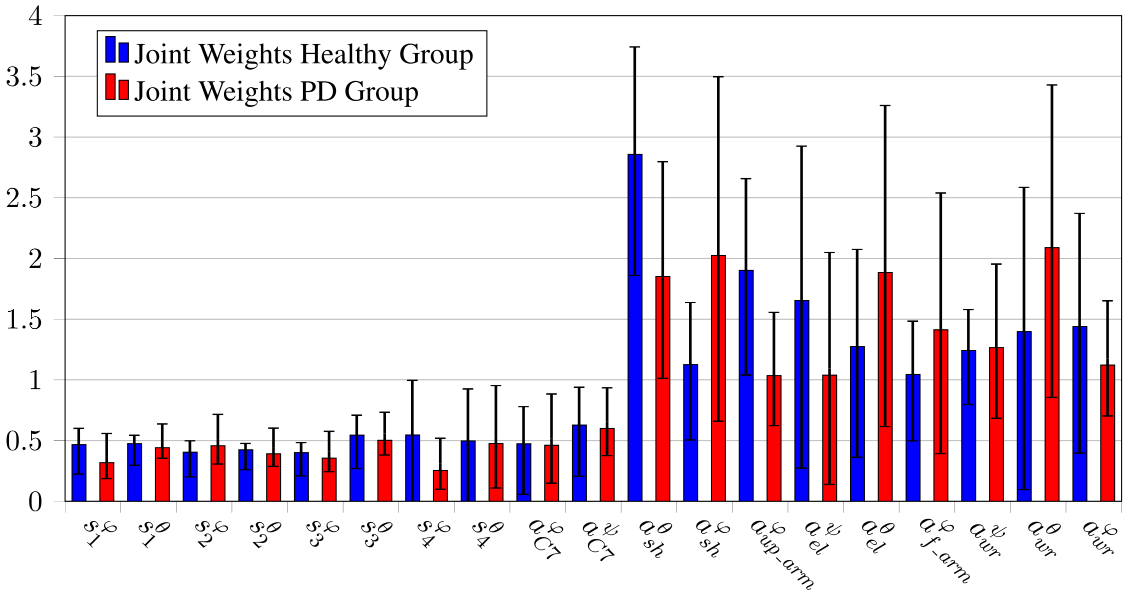


Figure 3.5: Average joint weights learned for the healthy (blue) and PD (red) subject group. The *y*-axis values correspond to the final weights for the joints indicated on the *x*-axis.

3.5.2 Trajectory Tracking Performance

The mean Cartesian tracking error between the recorded human end-effector position trajectories $\mathbf{x}_e$ and the associated end-effector position trajectories obtained from the weighted damped least-squares control scheme using the final joint weights $\mathbf{w}$, learned from Alg. 1, is 0.64 cm. Using the final joint weights, the mean residual error between the control-based and the corresponding human reference joint trajectories is found to be 0.029 rad, i.e., 1.69°.

3.5.3 Motion Strategy Analysis

The primary goal of our work is to investigate the hypothesis that healthy subjects and PD patients follow two different motion strategies to complete the same task successfully. Here, we consider the joint weights as indicative motor control parameters from which we want to infer underlying motion strategies adopted by the two groups. In this context, learning those parameters can be considered a prerequisite for the following analysis. In total, we learned the joint weights for 36 healthy and PD affected hand coordination motions, respectively. Figure 3.5 shows the mean and standard deviation for the weight of each joint and group. These first results suggest, that the motions of healthy and PD subjects primarily differ in the way the arm kinematic chain is actuated for task achievement. Healthy subjects show a strong tendency towards a *proximal* motion strategy, with decreasing joint weights along the arm chain, whereas PD subjects follow a *distributed* motion strategy, with balanced joint weights along the arm chain.

In order to determine whether there is a significant difference between the joint activity in healthy and PD subjects, we additionally performed an *independent sample t-test* for each weight. The results depicted in Figure 3.6 confirm that, in particular, the weights for the proximal joints, shoulder pitch a_{sh}^{θ} and upper arm roll $a_{up\_arm}^{\varphi}$, are significantly higher for

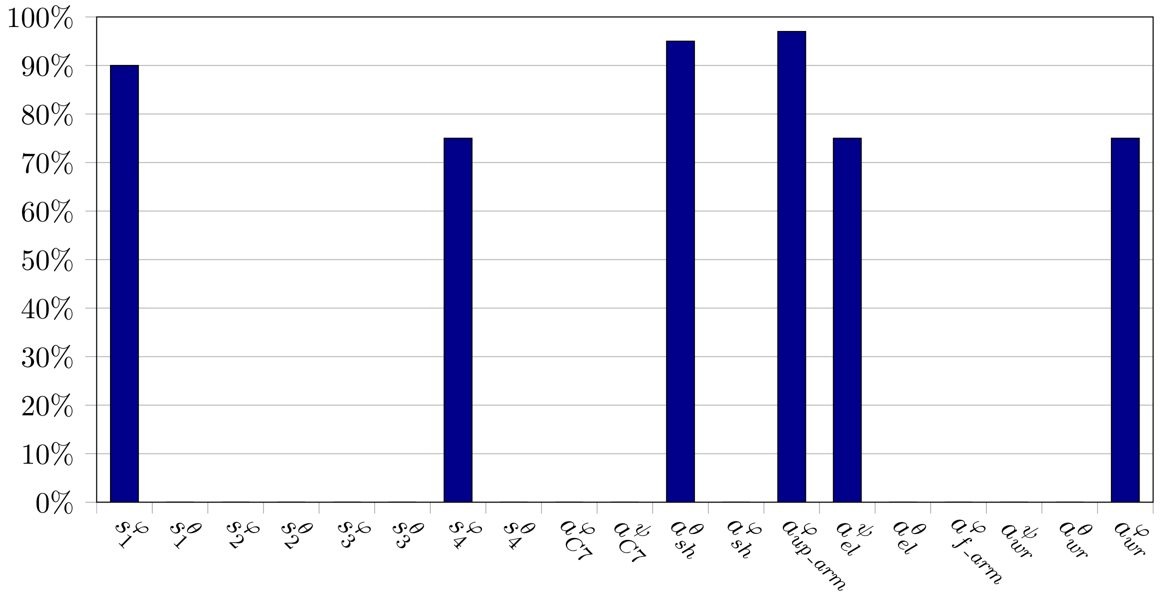


Figure 3.6: Results from the *independent sample t-test*: Comparing the joint weight means of the healthy and PD group against each other.

healthy subjects than for PD patients (with a significance level of $\alpha = 0.05$ and $\alpha = 0.025$, respectively). Given these insights, it seems that healthy subjects naturally use primarily the proximal joints to minimize the distance to a given target hand pose. On the other hand, distal joint activity is set aside for delicate hand pose adjustments. PD seems to affect this motion strategy in terms of reduced proximal joints activity. In order to complete the same task successfully the reduced proximal motion is compensated by raising the motion contribution of the remaining joints.

Note that in our experiments, the activation of the spine joints were similar between the two groups, although PD is well known to affect the postural stability of subjects. This finding may result from patients recovering stability by making use of the arm and back rest provided by the chair they are sitting on while performing the task. Investigation of whether the omission of arm and back rest support yields significant differences between the weights learned for the spine joints for healthy and PD subjects is subject to future work.

3.6 Related Work

Previous approaches dealing with human motion analysis can be subdivided into two main categories. The first category deals with the segmentation of the motion data into different actions or emotions. Approaches from the second category try to build computational models reflecting the underlying principles of human motion through optimization of different criteria or objective functions. In this work, both categories are relevant because the motor control deficits of PD patients can be interpreted as a result of a specific permanent emotional state or a motor control strategy following an objective function that is different from the one adopted by healthy subjects. In the following, we discuss representative approaches for each category.

Rahimi et al. [103] presented an approach that uses principal component analysis (PCA) to identify kinematic variables that best represent mobility tasks performed by PD patients. This method uses motion data of patients at different stages of PD recorded in their home environments using a full-body motion capture suit. Subsequently, the data were analyzed to determine possible variability between tasks, subjects, and trials. The results, however, state that no specific movement profile among patients for each task has been found.

Das et al. [38] use a support vector machine (SVM) to discriminate mild versus severe Parkinson's disease symptoms. The authors recorded motions of PD patients performing various motor control tasks and trained a motor task specific SVM classifier based on different sets of features. Das et al. report an average classification accuracy of approximately 90%.

Day et al. [39] tested the hypothesis that predictive motor behavior is abnormal in Parkinson's disease by recording the performance of healthy and PD subjects tracking a repeated and an unpredictable pattern of a moving spot with their hand. Despite of the obvious motor control deficits of the patients, the authors found that their tracking performance, evaluated w.r.t. the measured tracking lags, is comparable to that of healthy subjects.

Barbic et al. [7] investigated three techniques for automatic segmentation of motion capture data into distinct actions, e.g., walking, drinking, or sitting down. The two presented online segmentation techniques are PCA and probabilistic PCA. The third approach is a batch process using Gaussian mixture models for segmentation. All methods achieved good results in the experiments, though probabilistic PCA has found to provide the overall best performance. Based on the work in [7], Zhou et al. [151] proposed aligned cluster analysis, an extension of standard kernel *k*-means clustering for temporal segmentation of human motion data into actions. Here, the extension allows a variable number of features in the cluster means and the use of a dynamic time warping kernel to achieve temporal invariance. In a further extension [152], Zhou et al. developed an approach to implement a hierarchical decomposition of human motion data, where actions such as running or walking can be further decomposed into motion primitives of smaller temporal scale.

Cimen et al. [30] presented a technique using a set of posture, dynamic, and frequency-based descriptors for emotion classification of motion data. Based on different feature combinations, this approach applies a SVM learning algorithm to classify recorded motions into four distinct emotional states. Aristidou and Chrysanthou [6] propose a method to automatically extract motion qualities from dance performances, in terms of *laban movement analyses* (LMA) for motion analysis and indexing purposes. Using the four LMA components body, effort, shape, and space, the authors analyze correlations between the performer's acting emotional states.

Campos and Calado [18] provided an overview of human arm movement control theories and the different paradigms that have been used in modelling arm control. The authors distinguish between descriptive, dynamic, stochastic and motor execution models and analyze their relevance for rehabilitation practices.

Flash and Hogan [45] presented an approach for modelling voluntary human arm movements mathematically by defining an objective function representing the rate of change of acceleration. By minimizing the objective function using dynamic optimization, the method predicts trajectories for point-to-point and curved motions that resemble the observed motions of human subjects. Based on this work, Todorov and Jordan [133] proposed a novel mathematical model that accurately predicts the speed profiles of a human arm in straight reaching and extemporaneous drawing movements. The results indicate that the relationship between end-effector path and speed profile of a complex arm movement is stronger than

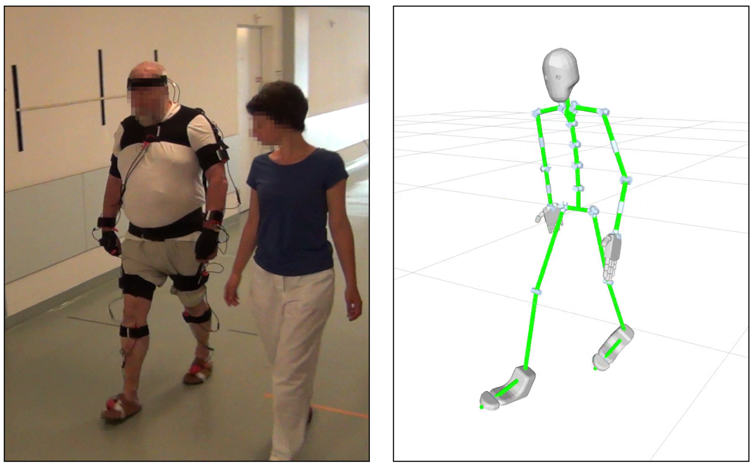


Figure 3.7: Possible extension of our approach towards lower limb and whole-body motion strategy analysis. *Left:* PD subject performing a 10 m walk trial. *Right:* Walking motion mapped to a whole-body artificial kinematic model.

previously thought. Albrecht et al. [3] developed an approach that uses physically inspired optimization principles describing a human's motion based on bi-level optimization methods. These principles are subsequently used to generate reaching motion trajectories for a humanoid robot that are similar to the recorded human behavior.

3.7 Conclusion

In this chapter, we presented an approach to differentiate between the underlying motion strategies adopted by healthy subjects and patients whose motor control is affected by a neurological disorder. We propose to learn indicative motor control parameters of a control scheme based on captured motion data. Our technique relies on a parameterization of the control scheme by means of joint weights, reflecting the activity level of joints contributing to the motion task. As we have shown in the experiments, the chosen control scheme is capable of closely replicating the recorded human end-effector and joint trajectories using the learned joint weights obtained from our algorithm.

According to our results on a motion database of healthy and Parkinson's disease subjects, there exist different motion strategies adopted by the two groups, referred to as the *proximal* and *distributed* motion strategy. Healthy subjects follow a hierarchical joint activation paradigm, whereas PD subjects show a balanced joint activation pattern. In general, the advantage of this novel measure of motor behavior lies in its independence of movement amplitudes and volition. Because joint contributions are not easily visible even for experienced neurologists, it might open a new field of motion analysis, yielding to new measures of motor deficits, which might be also used for the evaluation of therapeutic interventions such as deep brain stimulation in Parkinson's disease, even in a closed-loop fashion.

As the presented work can be considered as a basis for further investigations, there are also several issues to be addressed in future work. First, and most important, it would be valuable to increase the number of motion samples considered for the motion strategy analysis. Here, care should be taken to ensure that the number of male and female subjects as well as the mean age of the respective groups are as similar as possible. Moreover, it is necessary to record from subjects with the same handedness in order to avoid a bias of the results. Considering a larger database would decrease the amount of intra-group variability and overlap between the groups for each joint weight, as evidenced in Figure 3.5, thus consolidating the findings. However, building up a control group with a comparable mean age is not easy, as the average age of patients, diagnosed with PD, is with 64 years[1] relatively high and therefore only a low number of healthy subjects with the same age are available in the immediate work environment.

Another interesting aspect for future work would be to examine whether there is a relationship between motor symptom severity, i.e., the UPDRS scores and the resulting joint weight means. This may provide some insight into whether there is a gradual shift in the motion strategy as the motor symptoms of the Parkinson's disease worsen.

Finally, additional motion capture data could be used to investigate the motor control strategy also for other tasks involving the lower limbs or even whole-body motions. Of course, this would require to use an extended artificial kinematic model, as depicted in Figure 3.7, and simultaneous control of multiple kinematic chain.

[1]More information is available under http://www.parkinson.org

Chapter 4

Stance Pose Selection by Inverse Reachability Maps

Knowledge about the robot's reachable workspace is of great importance when considering grasping tasks carried out with a humanoid platform. Without this knowledge, it might be necessary to repeatedly adapt the stance location and call an inverse kinematics solver before a valid robot configuration suitable for reaching a given grasping pose can be found. In this chapter, we present an approach to select an optimal stance location in SE(2) for a mobile robotic platform relative to a desired grasp pose. We use a precomputed representation of the robot's reachable workspace that stores quality information in addition to spatial data. By inverting this representation we obtain a so-called inverse reachability map (IRM) containing a collection of potential stance poses for the robot. The generated IRM can subsequently be used to select a statically stable, collision-free stance configuration to reach a given grasping target. We evaluated our approach with a Nao humanoid in simulation and in experiments with the real robot. As the results show, using our approach optimal stance poses can easily be obtained. Furthermore, the IRM leads to a substantially increased success rate of reaching grasping poses compared to other meaningful foot placements within the vicinity of the desired grasp.

The inherent anthropomorphic kinematic structure of humanoid robots is a basic property, allowing them to conduct mobile manipulation tasks in environments originally designed for humans. A prerequisite for successful task completion in such settings is that the robot is equipped with sufficient knowledge about the relevant aspects of the scene and its own capabilities. This also includes the ability to decide where to place itself relative to an object to be manipulated in order to achieve an admissible grasp configuration. In Chapter 2, a human operator was supposed to provide that knowledge as well as appropriate motion commands to reach a convenient stance location relative to a specific grasping target. However, human assistance depended mobile robotic systems are considered impractical or even undesirable for many applications, for the reasons previously mentioned in Chapter 1. In such scenarios, the robot needs to be able to independently infer potential stance poses, suitable to carry out the upcoming manipulation task, from a desired end-effector grasping target given as input. To decide where to place itself, the robot needs to account for obstacles in the environment

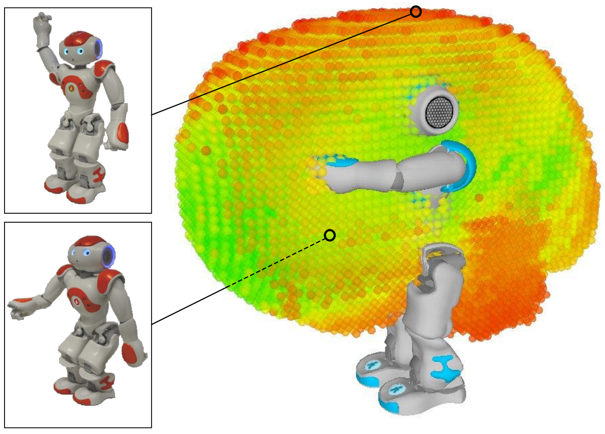


Figure 4.1: Representation of reachable right hand locations from statically stable double support poses. Voxels are colored by the maximum value of the manipulability index among the configurations stored in them (green = high, red = low).

scattered around the prescribed grasping target . Moreover, the decision is strongly influenced by the structure of the robot's kinematic model, including the number of joints and their value range as well as its mass distribution. Based on these parameters, the robot's manipulation capabilities can be uniformly represented by a spatial data structure, called *reachability map* [150], which is exemplary depicted in Figure 4.1.

Many existing techniques follow the approach of dealing with locomotion and manipulation tasks as two distinct problems (e.g., [9, 107]). Thus, first a motion for the lower body is planned, i.e., for the base of a mobile platform or the legs of a humanoid, in order to bring an object to be manipulated within the extent of the robot's upper body workspace. Subsequently, the reaching task is performed using only the upper body joints. Whether the object can actually be reached from the current stance location significantly depends on the number of available joints and constraints involved. Most mobile manipulator and humanoid robot platforms are designed to have at least one degree of redundancy in the upper body end-effector chains. This choice allows the robot to arrive at almost all of its reachable end-effector poses with arbitrary orientation without the need to reposition or reconfigure the mobile base or lower body, respectively. In other words, a suboptimal choice of the stance location relative to a grasping target is compensated by the redundant kinematic structure of the upper body. However, when considering robots with an upper body equipped with only six or less degrees of freedom (DOF), the number of achievable end-effector orientations for the targets within the reachable workspace of the upper body is limited. As opposed to mobile manipulators, that are forced to reposition their base once the desired end-effector orientation has been determined to be unachievable from the current stance location, humanoid robots are capable of adjusting their lower body configuration to extend the set of achievable end-effector orientations for a given grasping target. The actuation of the legs of a humanoid, however, introduces balance constraints and thus cannot fully compensate for the limited kinematic properties of the upper body chains. Figure 4.2 illustrates this issue by comparing the capability of a humanoid robot and mobile manipulator in reaching a grasping target at a specific height with a fixed upright end-effector orientation. Here, the mobile manipulator reaches a large set of end-effector poses

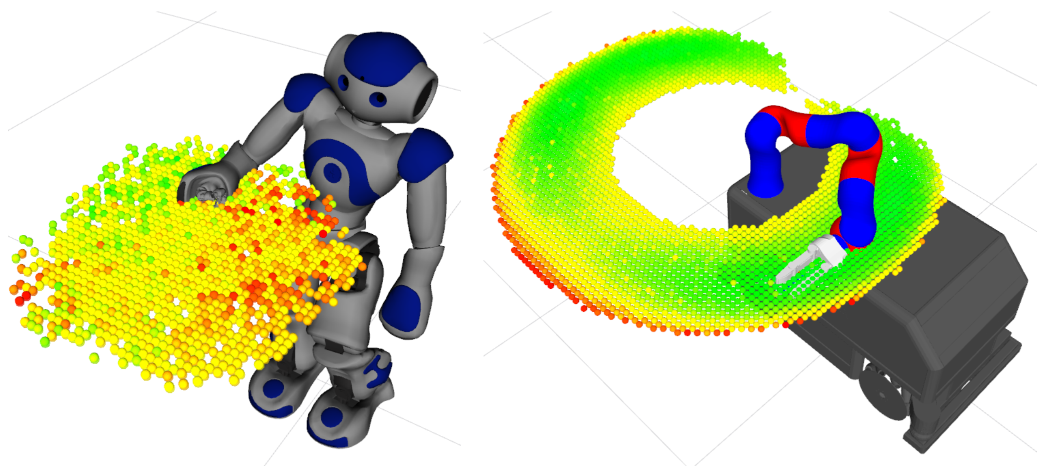


Figure 4.2: Grasping targets at a specific height, reachable with a fixed upright end-effector orientation. Targets reached by the *NAO* humanoid (left) and the *omniRob* mobile manipulator (right). Voxels are colored by their manipulability index (red = low, green = high).

with dexterous configurations, whereas the number and quality of configurations covering the workspace is much lower for the humanoid platform. This indicates, that the choice of a stance pose relative to an object becomes a crucial factor for humanoids, if repetitive adaptions of the stance location and calls to an inverse kinematics solver for the arm chain are to be avoided.

In this chapter, we present an approach that is based on the concept of *inverse reachability maps* (IRM), proposed by Vahrenkamp et al. [136, 138], to select an optimal stance pose for the feet of a humanoid robot in SE(2), i.e., a 2D position and 1D orientation, relative to the grasping target. Nevertheless, the approach is not limited to humanoids and can be easily applied to less constrained robotic platforms, such as mobile manipulators. As we show in experiments with a Nao humanoid, we can easily select optimal stance poses from our IRM given a desired grasp. By comparing our approach to random stance pose selection within a meaningful extent around the target grasp, we achieve a substantially increased success rate. Note, that such an IRM can also be used for humanoids and wheeled mobile manipulators with a higher number of DOF's to choose optimal stance locations for manipulation tasks.

4.1 Workspace Representation

Inspired by the work of Vahrenkamp et al. [138], we describe the robot's reaching and manipulation capabilities by a discretized representation of its workspace. The resulting spatial data structure, referred to as the *reachability map* (RM), is composed of voxels of constant size and needs to be computed only once offline. Each voxel of the grid represents a set of robot configurations for which the end-effector pose lies within the extent of the voxel.

4.2 Generation of Forward Reachability Maps

A simple strategy to generate a representation of the robot's reachability in workspace can be obtained by iterating through all the 3D voxels of the spatial data structure while trying to solve the inverse kinematics (IK) problem. If a IK solution is found, the corresponding voxel is marked as reachable. Otherwise, the voxel is marked as being out of reach. Following this approach, a binary representation of the reachability capabilities is obtained. Whereas this information is sufficient to determine whether an end-effector pose is reachable or not, it does not permit concluding how well a voxel is reachable compared to other voxels in the spatial grid. A more informative representation can be constructed by sampling the configuration space $\mathcal{C}$ of the robot and determining the voxel containing the corresponding end-effector poses using forwards kinematics. This results in a data structure, from which the quality of a voxel can be defined as the volume in $\mathcal{C}$-space that maps to end-effector poses lying within the extent of the voxel. Nevertheless, this measure is not sufficient for uniquely identifying an appropriate configuration due to the fact that undesirable singular configurations are equally contributing to the quality measure assigned to the voxels in the map construction process. To avoid this false information and therefore to represent the robot's reaching capabilities more accurately, we adopt in this work the manipulability measurement, introduced by Yoshikawa [148], to assign quality information to the voxels of the reachability map. This way, the ability of the end-effector to maneuver in workspace is directly encoded in the quality index of the resulting reachability data.

4.2.1 Configuration-Space Sampling

To build a reachability map, we need to generate samples from the robot's configuration space $\mathcal{C}$. To do so, values can be either sampled uniformly from the respective joint range or they can be selected iteratively by stepping through the joint value range with a specific increment. The decision of how to sample the $\mathcal{C}$-space significantly depends on the number of DOF's and constraints involved. Considering fixed base and mobile manipulators most of the samples drawn will fall in $\mathcal{C}_{free} \subset \mathcal{C}$, i.e., the subspace of collision-free and hence valid configurations. However, in the case of humanoid robots additional loop-closure and stability constraints arise when leg joints are involved in the sampling process. Hence, the set of admissible configurations $\mathcal{C}_{stable} \subset \mathcal{C}_{free}$ for a humanoid covers only a small portion of the entire $\mathcal{C}$-space. In this work, we follow the approach of generating samples by stepping through the value ranges of the joints actuating the kinematic chains. Here, we define for humanoid platforms two different increments, a coarse increment for upper body joints and a finer one for lower body joints that directly affect the compliance with loop-closure and stability constraints. In contrast, a constant increment can be chosen for wheeled mobile manipulators, which typically remain statically stable throughout the entire $\mathcal{C}$-space of the robotic arm mounted at its base.

4.2.2 Quality Information on Configurations

In this work, we evaluate the quality of reachability map voxels based on manipulability measurements obtained for the configurations, leading to end-effector poses lying within its extent. According to Yoshikawa [148], the manipulability of a robotic mechanism can be generally understood as the ease of arbitrarily changing the position and orientation of the end-effector located at the manipulator's tip. This freedom of motion plays an important role when a subsequent interaction between the robot and the environment is intended. Since the ability to manipulate strongly depends on the configuration of the robot, it is necessary to recall some general mathematical relationships. Let us consider a task $\mathbf{r}$, typically defined by a vector of m dimensions, and a configuration $\mathbf{q}$, i.e., a vector of n joint values. The mapping from configuration space to task space is defined by

$$\mathbf{r} = f(\mathbf{q}), \tag{4.1}$$

which is also known as the forward kinematics equation. Accordingly, the relation between the joint velocities $\dot{\mathbf{q}}$ and the task velocities $\dot{\mathbf{r}}$ follows from the derivative of Eq. (4.1), given by

$$\dot{\mathbf{r}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}, \tag{4.2}$$

where $\mathbf{J}(\mathbf{q})$ denotes the Jacobian matrix of the manipulator in configuration $\mathbf{q}$. Analyzing the rank of $\mathbf{J}(\mathbf{q})$ reveals some important properties of the kinematic structure. When the following condition is satisfied

$$\max_{\mathbf{q}} \operatorname{rank} \mathbf{J}(\mathbf{q}) = m, \tag{4.3}$$

the manipulator is said to have a degree of redundancy of $(n-m)$. Otherwise, if the manipulator Jacobian looses rank in a certain configuration $\mathbf{q}^*$, i.e.,

$$\operatorname{rank} \mathbf{J}(\mathbf{q}^*) < m \tag{4.4}$$

the robot is said to be in a singular configuration. In this case, the task vector $\mathbf{r}$ specifying the end-effector pose cannot move in a certain direction and thus the manipulability is reduced. An explicit measure of the manipulability, represented by a scalar value w, has been introduced by Yoshikawa in [148]:

$$w = \sqrt{\det \mathbf{J}(\mathbf{q})\mathbf{J}^\top(\mathbf{q})}. \tag{4.5}$$

When considering non-redundant manipulators, i.e., when the number of task and configuration dimensions are equal, Eq. (4.5) can be reduced to

$$w = |\det \mathbf{J}(\mathbf{q})|. \tag{4.6}$$

The manipulation capabilities of a kinematic chain in a specific configuration can also be assessed using a graphical representation. When considering the set of joint velocities of unit norm, defined according to

$$\dot{\mathbf{q}}^\top \dot{\mathbf{q}} = 1, \tag{4.7}$$

and substituting in the above equation $\dot{\mathbf{q}}$ with the expression obtained by rearranging Eq. (4.2), we obtain

$$\dot{\mathbf{r}}^\top (\mathbf{J}(\mathbf{q})\mathbf{J}^\top(\mathbf{q}))^{-1}\dot{\mathbf{r}} = 1, \tag{4.8}$$

which is the equation of points on the surface of an ellipsoid in the end-effector velocity space [121]. The direction and dimension of the principal axes of the ellipsoid, centered at the end-effector frame, then follow by computing the *singular value decomposition* (SVD) of the matrix $\mathbf{J}\mathbf{J}^\top$. Note, that the argument $\mathbf{q}$ of the Jacobian is omitted in favor of a compact representation in the following. The directions of the principal axes are given by the eigenvectors of the matrix $\mathbf{J}\mathbf{J}^\top$ while their magnitude is determined by the singular values σ_i of $\mathbf{J}$, computed as

$$\sigma_i = \sqrt{\lambda_i(\mathbf{J}\mathbf{J}^\top)}, \tag{4.9}$$

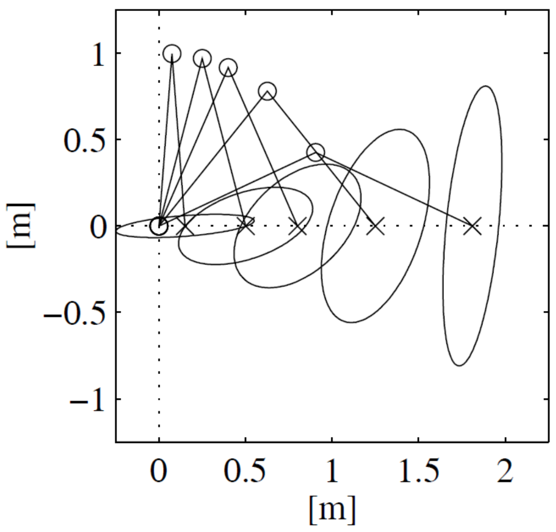


Figure 4.3: Manipulability ellipsoid for a planar manipulator with two revolute joints in different configurations (Source: [121]).

where λ_i denotes the i-th eigenvalue of $\mathbf{J}\mathbf{J}^\top$. Examples of the velocity ellipsoid for an 2R planar manipulator in different configurations are shown in Figure 4.3. The shape of the ellipsoid indicates that large velocities can be applied in the direction of the major principal axis while only small velocities are possible in the direction of the minor axis. In singular configurations, one of the principles axis vanishes and the ellipse or ellipsoid collapses to a line in 2D and a plane in 3D, respectively. Finally, multiplying the singular values yields a manipulability measure w that is proportional to the ellipsoid's volume

$$w = \sigma_1 \sigma_2 \cdots \sigma_m, \tag{4.10}$$

and thus offers an appropriate approximation to be used for assessing the distance of robot postures to singular configurations. Assigning quality information to voxels on the basis of this measure, instead of naively counting the number of voxel *hits*, in the map construction process therefore generates reachability information that better reflects the true manipulation capabilities of the manipulator.

Note, that manipulability can be similarly evaluated considering forces instead of velocities. This measure is of particular importance, e.g., when the robot is intended to lift or push heavy objects. The principal axes of the corresponding force ellipsoid have the same direction as the axes of the velocity ellipsoid. The magnitude of the axes, however, are interchanged with respect to the velocity ellipsoid. Thus, only small forces can be applied in directions allowing large velocities and vice versa. For further reading about velocity manipulability and a detailed explanation of the force manipulability the reader is referred to the literature [28, 120, 121, 148].

Furthermore, additional refinements of the manipulability measure can be achieved by considering further relevant properties of the robot's kinematic structure. For example, Vahrenkamp et al. [137], consider joint limits under redundancy, the distance to obstacles and between robot links for determining voxel qualities in the map construction process. In case of humanoid platforms, other meaningful measures such as the distance of the projected CoM to the support polygon boundaries or the emerging torque at the ankle joints could be included.

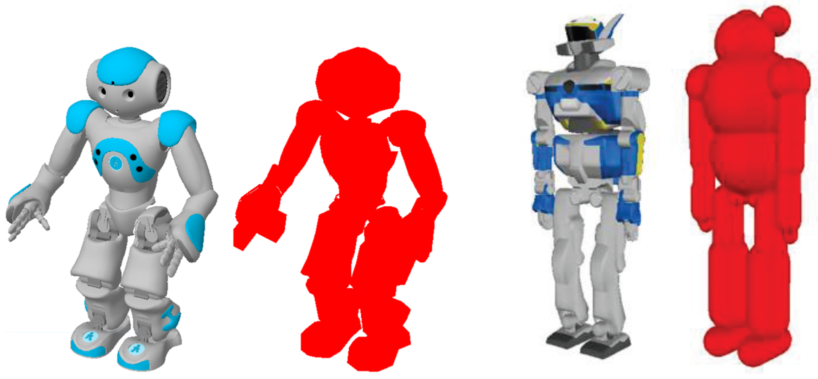


Figure 4.4: Original model and approximated model used for collision detection and distance computation for the humanoid robot NAO (left) and HRP-2 (right) (Source: Kanehiro et al. [68]).

4.3 Kinematic Constraints

The extent of a robot's reachable workspace highly depends on the number of degrees of freedom it is composed of and the range of admissible values its actuators can adopt. Beyond that, configurations need to be checked for *self-collisions*, i.e., robot postures where different links of the kinematic structure are getting into contact with each other. Further, humanoid specific, constraints adopted in our approach consider end-effector poses only reachable if loop-closure for the kinematic chain formed by the two legs with the ground as well as static stability can be achieved for the corresponding configurations. In the following, a detailed description of the kinematic constraints involved in our map construction process will be given.

4.3.1 Collision Detection

Detecting collisions among the links of a robot (*self-collision*) and between the robot and the environment (*obstacle-collision*) is a fundamental element of sampling-based configuration generation schemes. At the same time, collision detection represents a problem of considerable computational complexity and can therefore be seen as a separate field of research. Depending on the number of collision checks to be performed, the choice of a collision detection algorithm may have a major influence on the overall performance of frameworks building upon it. This is particularly relevant considering probabilistic motion planning frameworks, which typically need to perform many collision checks until a valid solution is found. A common approach to determine whether a certain configuration is in collision, is to perform an interference check among the robot link geometries and between the robot and obstacle geometries. In practice, this is done by finding the minimum distance between the geometries. If the distance is found to be negative, i.e., the geometries overlap, the configuration is determined to be in collision. Often, however, robots are constituted by complex geometries and finding the minimum distance becomes a computational expensive operation. In the field of motion planning this issue is traditionally addressed by approximating the robot geometry with simple geometric shapes, such as cylinders and spheres. Additionally, these shapes are enlarged so as to maintain a safety margin for each robot link against other geometries. An example of an

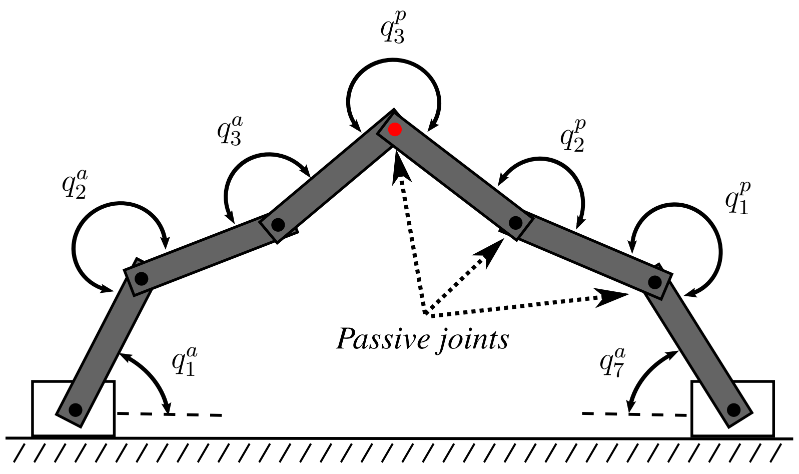


Figure 4.5: Example of a link decomposition into active and passive vectors of joints.

approximated robot geometry, generated for the humanoid robot HRP-2, is depicted in Fig. 4.4 (right). Whereas this approach yields a significant reduction of the computational demands, it comes at the cost of discarding configurations which are actually valid considering the original robot geometry. Contrary to motion planning algorithms, collision detection is employed only in the forward reachability map construction process of our framework. Therefore, only very subtle geometries of the original robot mesh, e.g., the hands of a humanoid, are approximated by simple geometric shapes in our work. Moreover, a higher computational expense is not critical here, considering that these maps need to be build only once in an offline step for the respective robotic platform. The resulting geometry for the Nao robot used for collision detection is illustrated in Fig. 4.4 (left).

For practical applications, there are already several collision detection frameworks available, such as *FCL*, *PQP* or *ODE* [97, 122, 135]. Due to their complexity, we consider in this work collision detection as a black box, providing validity information for the configurations given as input.

4.3.2 Kinematic Loop-Closure

When sampling configurations for serial kinematic chains, joint angles are assumed to be constraint only by their admissible value interval, but not to depend on the values chosen for the preceding joints along the chain. This assumption, however, is no longer valid when considering kinematic structures whose links are arranged to form loops. In this case, some joints angles must be chosen such that the loop remains closed. In the case of a humanoid robot, for example, closed kinematic chains occur when an object is manipulated using both hands, or the legs of the robot are desired to remain fixed on the ground during a manipulation task in favor of maintaining robust stability. In order to account for the closure constraint in the process of generating reachability maps for legged humanoid robots, the classical configuration sampling scheme needs to be adapted. A first approach is presented by the *Active-Passive Link Decomposition* method [54]. With this approach, the configuration vector $\mathbf{q}$ is split into a vector of *active* joints $\mathbf{q}^a$ and a vector of so-called *passive* joints $\mathbf{q}^p$. A possible

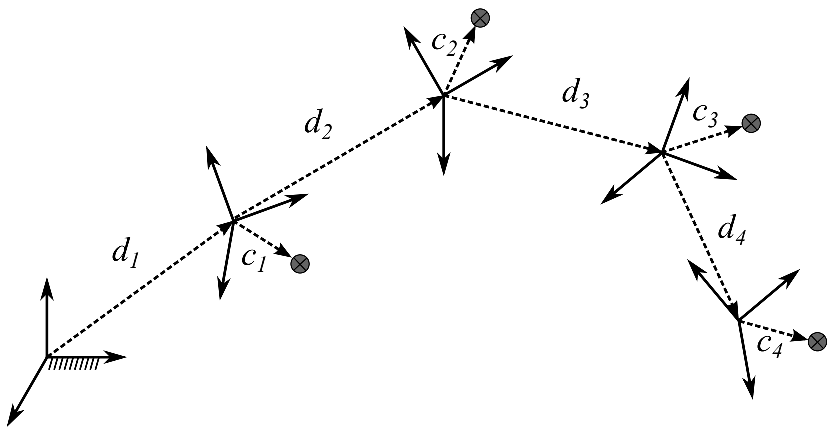


Figure 4.6: Parameter of a 4 link serial chain manipulator.

decomposition for a robot composed of seven revolute joints is shown in Fig. 4.5. With this choice, the kinematic structure has four degrees of freedom. In a first step, the method proceeds by randomly sampling values for the active joint variables $\mathbf{q}^a \in \mathbb{R}^4$. Afterwards the remaining passive joint variable $\mathbf{q}^p \in \mathbb{R}^3$ are determined by solving the inverse kinematics (IK) problem in order to satisfy the closure constraint. Note, that while for some choices of $\mathbf{q}^a$ multiple IK solutions for $\mathbf{q}^p$ exist, most of the sampled configurations $\mathbf{q}^a$ will not allow to find a solution to the IK problem.

Another method, called the *Random Loop Generator* [32], improves the constraint satisfaction success rate by iteratively choosing values for the variables in $\mathbf{q}^a$ that allow an inverse kinematic solution for the passive variables in $\mathbf{q}^p$. A prerequisite for this approach is that the chosen active joints appear sequentially along the kinematic chain, i.e., they are not interrupted by a passive joint. In a first step, an interval I_1 of admissible values for the first active joint q_1^a is computed. According to some geometrical analysis, it is known that for all values q_1^a outside this interval, no IK solution for the passive chain exists. In practice, this analysis can be performed by checking whether the passive chain can reach the accessible workspace spanned by the active chain, given a value for q_1^a. Once a value for q_1^a has been sampled from I_1, an interval I_2 of admissible joint values for q_2^a is computed. As before, a value of q_2^a is sampled from I_2 and an interval for the subsequent active joint is determined. This procedure is repeated until a value has been assigned to each active joint or the generator has detected that no admissible assignment is left for one of the joints. If the set of admissible values I_i for an active joint q_i^a is found to be empty, all previous assignments are discarded and the whole procedure is repeated. Otherwise, the vector $\mathbf{q}^a$ is used to find values for the passive joints in $\mathbf{q}^p$ through inverse kinematics.

4.3.3 Static Stability

In order to determine the validity of whole-body poses sampled from the $\mathcal{C}$-space of a humanoid robot it is a necessity to evaluate whether the corresponding configurations provide a stable posture or cause the robot to lose balance. From a geometrical point of view, the stability constraint implies that the projection of the Center of Mass (CoM) of the robotic structure is required to be located inside the support polygon. According to the literature, this

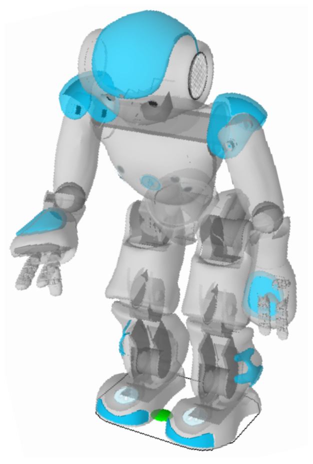


Figure 4.7: Support polygon (black line) and projected CoM (green dot).

formulation of the constraint is also referred to as *static stability*. An extension of this concept is presented by considering *dynamic stability*, where the dynamics of a system are taken into account to maintain equilibrium [142]. In motion generation schemes, as the one presented in Chapter 2, it is of great importance to account for the system dynamics in order to ensure stable trajectory execution. In contrast, a general representation of the reaching capabilities of a robot does not depend on a specific motion. Thus, only static stability is involved in the process of generating forward reachability maps for humanoid platforms.

As a first step, which is required to evaluate the validity of a configuration concerning the stability constraint, the location of the CoM needs to be determined. For simplicity, let us consider the serial kinematic chain composed of four links joined by revolute joints, as depicted in Figure 4.6. For each link of the kinematic chain, the length l_i and the mass m_i are assumed to be given. Furthermore, it is supposed that the robot description, generated from a 3D CAD model, provides the location of the center of mass $\mathbf{c}_i = (c_{i,x}, c_{i,y}, c_{i,z}, 1)^\top$ for each link, expressed in homogeneous coordinates in the respective reference frame. From the geometric description of the chain bodies, we can derive the homogeneous transformation matrices $\mathbf{T}_i$, defined as

$$\mathbf{T}_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{d}_i \\ \mathbf{0} & 1 \end{bmatrix}, \tag{4.11}$$

where $\mathbf{R}_i$ and $\mathbf{d}_i$ denotes the 3×3 rotation matrix and the translation vector of the i-th link frame with respect to the global frame, respectively. The total mass of the system is simply obtained by summing up the individual masses of the links $M = m_1 + m_2 + m_3 + ... + m_i$. The CoM location for the entire system in Figure 4.6 is obtained by computing the sum of each link's CoM divided by the total mass M [33]:

$$\mathbf{p}_{CoM} = \sum_{i=1}^{4} \frac{m_i \mathbf{T}_i \mathbf{c}_i}{M} \tag{4.12}$$

A humanoid robot is typically composed of several such kinematic chains. In this case, the overall CoM location can be obtained by computing the weighted average of the individual chain CoMs, determined according to Eq. (4.12).

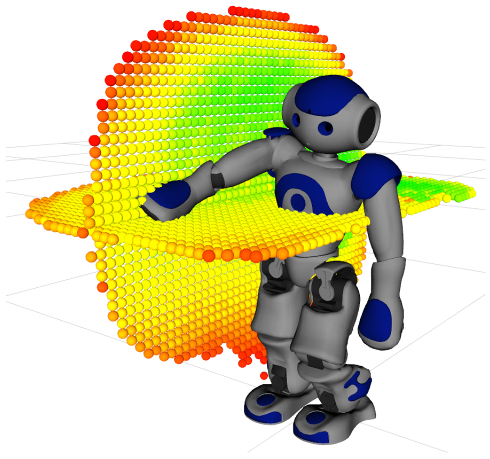


Figure 4.8: Horizontal and vertical cross-section through the RM, showing right hand poses reachable from statically stable double support configurations. Voxels are colored by their manipulability index (red = low, green = high).

Once the CoM for the anthropomorphic structure is known, we need to check whether its projection onto the support plane is within the boundaries of the so-called support polygon. The support polygon is defined as the convex hull established by the pressure point of the structure on the ground. Therefore, the shape and size of this polygon depends on whether the robot is currently in single leg (one foot on the ground) or double (both feet on the ground) support mode. An example of the support polygon for a double support configuration and the location of the projected CoM is shown in Figure 4.7. If the projection of the CoM is inside the support polygon, as depicted, the configuration is said to be statically stable. Otherwise, if the point is found to be outside the boundaries of the polygon the robot would fall over when adopting that pose. In the process of generating forward reachability maps for humanoid platforms, this check is iteratively performed for the sampled configurations. Based on the outcome of the validity evaluation, configurations are discarded or added to the corresponding voxel of the spatial reachability grid.

4.4 Reachability Maps for Whole-Body Humanoids

Constructing a reachability map for a humanoid robot equipped with multiple end-effectors is a challenging problem due to the high number of degrees of freedom and the number of constraints involved. As opposed to a fixed base and most mobile manipulators for which samples of the joint space are valid as long as they are self-collision free, additional stability issues arise when also lower body joints of a humanoid are considered. Although our framework allows representing reachability information for arbitrary chains of a humanoid robot, e.g., reachable location of the torso when actuating the leg chains, we are in this work particularly interested in reaching and manipulation tasks and hence in the workspace volume covered by the gripper from statically stable double support configurations. The individual steps for the construction of a reachability map are shown as pseudo code in Alg. 3 and will be explained in detail in the following.

Algorithm 3: Construction of Reachability Map

```
Input  : Root and tip link $l_{root}$, $l_{tip}$, sampling step width $\Delta q$, fixed desired pose of
         swing foot $\mathbf{p}^{SUF}_{SWF}$
Output : Reachability Map $RM$
1  $chain$ ← getChain($l_{root}, l_{tip}$)
2  while $\mathbf{q}_c$ ← sampleChainConfig(chain, $\Delta q$) do
3      $\mathbf{q}_{SUL}$ ← getSupportLegConfig($\mathbf{q}_c$)
4      $\mathbf{p}_{hip}$ ← computeHipPose($\mathbf{q}_{SUL}$)
5      $\mathbf{p}_{SWF}$ ← desiredSwingFootPose($\mathbf{p}_{hip}$, $\mathbf{p}^{SUF}_{SWF}$)
6      $\mathbf{q}_{SWL}$ ← solveSwingLegIK($\mathbf{p}_{SWF}$)
7      if checkConfigValidity($\mathbf{q}_c, \mathbf{q}_{SWL}$) then
8          $w$ ← computeManipulabilityMeasure($\mathbf{q}_c$)
9          $\mathbf{p}_{tcp}$ ← computeFK($\mathbf{q}_c$)
10         $idx$ ← findEEvoxel($\mathbf{p}_{tcp}$)
11         $RM$ ← addConfigToVoxel($idx, \mathbf{q}_c, \mathbf{q}_{SWL}, w$)
12     end
13 end
```

4.4.1 Building Whole-Body Reachability Maps

The algorithm takes as input a root and tip link l_{root}, l_{tip} for the chain for which sampling is performed. Here, $\Delta\mathbf{q}$ specifies the step width for sampling. Furthermore, a fixed desired pose $\mathbf{p}^{SUF}_{SWF}$ of the swing foot parallel and expressed with respect to the support foot, which corresponds also to the root of the sampled chain, is defined. After sampling a configuration of the chain (line 2 of Alg. 3) the part of the configuration vector $\mathbf{q}_{SUL}$ storing the support leg configuration is extracted (line 3 of Alg. 3) and the forward kinematics is solved to obtain the pose of the hip $\mathbf{p}_{hip}$ with respect to the support foot (line 4 of Alg. 3). For a better understanding, Figure 4.9 shows the joints and frames of the Nao humanoid robot. Given the hip and the desired swing foot pose expressed in the support foot frame we can easily determine the pose for the swing foot relative to the hip frame $\mathbf{p}_{SWF}$ required to achieve a double support configuration with the feet being placed parallel to each other (line 5 of Alg. 3). Afterwards, the algorithm tries to solve the inverse kinematics problem for the swing leg chain (line 6 of Alg. 3). If an IK solution is found it is stored in $\mathbf{q}_{SWL}$ and the whole-body configuration of the robot is checked for validity, i.e., we determine whether the whole-body pose is statically stable and collision-free (line 7 of Alg. 3). If no IK solution exists or the pose is found to be invalid, the algorithm proceeds by sampling a new configuration for the kinematic chain and repeats the previous steps. Otherwise, if the configuration is valid the algorithm continues by computing the manipulability measure (see Section 4.2.2) for the sampled configuration (line 8 of Alg. 3). As described in Section 4.2, the forward kinematics is subsequently computed to obtain the end-effector pose $\mathbf{p}_{tcp}$ for configuration $\mathbf{q}_c$ of the chain and the index idx of the voxel containing $\mathbf{p}_{tcp}$ is determined (lines 9-10 of Alg. 3). In a final step the sampled configuration, its manipulability measure, as well as the IK solution for the swing leg chain is stored in the voxel with index idx (line 11 of Alg. 3). Figure 4.8 depicts a horizontal and vertical cross-section of a RM, build for the kinematic chain rooted at the

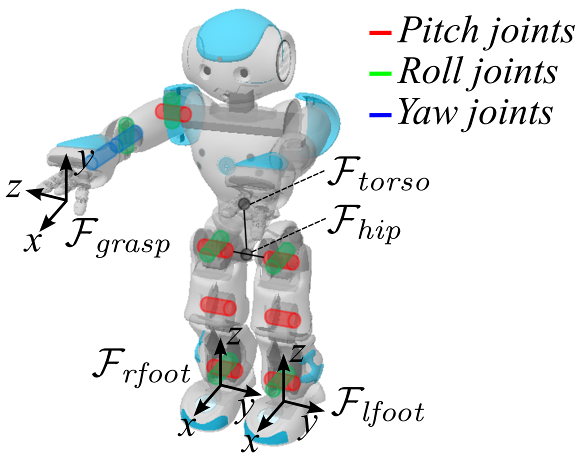


Figure 4.9: Kinematic chains and frames of the NAO robot considered in the process of building reachability maps.

right foot and terminating at the right gripper. Note, that the reachability map need to be built only once in an offline step.

4.4.2 Double Support Generation

In our approach, a voxel of the spatial data structure is reachable when the end-effector can be placed within its extent from a statically stable and collision-free double support configuration. Since sampling is performed only for a serial chain of the robot, e.g., for the joint between the foot and gripper link, the loop-closure and stability constraint must be additionally enforced. The former requires the adaption of the swing leg configuration such that the feet of the robot are placed parallel to each other on the floor. Here, we apply the *active-passive link decomposition* method, introduced in Section 4.3.2, to achieve a closed-loop configuration for the legs, where the active chain corresponds to the support leg for which joint values are sampled and the passive chain is represented by the swing leg. Let us assume w.l.o.g., that the right leg is the support leg whose configuration is given by $\mathbf{q}_{SUL}$ (line 3 of Alg. 3). By computing the forward kinematics, we obtain the pose $\mathbf{p}_{hip}$ of the hip frame $\mathcal{F}_{hip}$ with respect to the support foot frame $\mathcal{F}_{rfoot}$ (see Figure 4.9). Then, using the fixed transformation $\mathbf{p}_{SWF}^{SUF}$, expressing the desired pose of the swing foot frame $\mathcal{F}_{lfoot}$ with respect to the support foot, we can infer the desired pose $\mathbf{p}_{SWF}$ of the swing foot in the frame $\mathcal{F}_{hip}$. Finally, we apply an inverse kinematics solver to find a configuration $\mathbf{q}_{SWL}$ for the swing leg.

4.5 Reachability Maps for Mobile Manipulators

As opposed to humanoid robots, which are preferably deployed in the domestic domain, wheeled mobile manipulators are specifically designed to operate in industrial settings. As the required skills vary a lot between different industrial sectors, they need to be able to cope with a large spectrum of tasks. To fulfill these demands, the kinematic structure of such platforms is defined in a manner establishing high versatility while keeping the complexity of the control

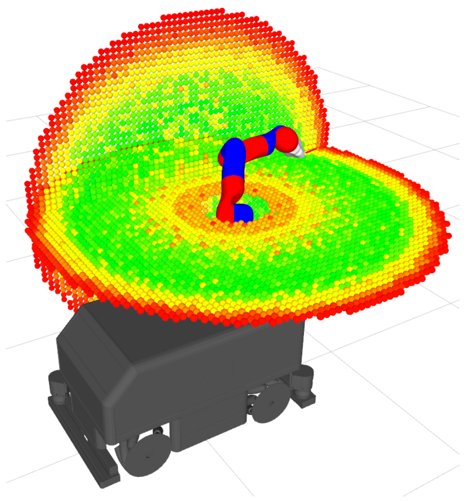


Figure 4.10: Horizontal and vertical layer of the RM, approximating the workspace volume reachable by the end-effector of the *omniRob* mobile manipulator. Voxels are colored by their manipulability index (red = low, green = high).

problem manageable. A first measure taken to equip these platforms with high dexterity is related to the number of joints and their associated admissible value interval. In practice, the manipulators used for mobile robotic platforms exhibit a joint range which is notably wider than the intervals specified for humanoid kinematic structures. Additionally, it typically possesses 7 degrees of freedom, thus offering a redundant degree of freedom considering 6D grasping tasks. Another characteristic property of mobile manipulators is that the mass ratio between the mobile base and the manipulator is often chosen in a way, such that the center of mass remains within the support polygon, defined by the footprint of the mobile base, throughout the entire $\mathcal{C}$-space of the robotic arm.

Taking these properties into account, the reachability map construction process described in Alg. 3 becomes far less constraint considering mobile manipulator platforms. Here, the input parameter specifying the desired pose of the swing foot relative to the support foot as well as the steps performed in lines 3-6 of Alg. 3 can be omitted for obvious reasons. With regard to the configuration validity evaluation performed in line 7 of Alg. 3, it is sufficient to check whether the configurations are self-collision free assuming static stability and the absence of external forces acting at the robot's end-effector. The spatial data structure, resulting from the reachability map construction process for the *omniRob* mobile manipulator (see Appendix A.2), is depicted in Figure 4.10.

4.6 Reachability Map Inversion

The reachability maps generated according to Section 4.4 and Section 4.5 represent the robot's capability of reaching certain end-effector poses from statically stable and collision-free configurations. Moreover, all configurations stored in the forward map for humanoids are guaranteed to be in double support. In manipulation and reaching tasks, however, we face exactly the inverse problem. Namely, the required end-effector pose is predefined by the pose of an object to be grasped and we aim at finding a base or feet configuration that maximizes

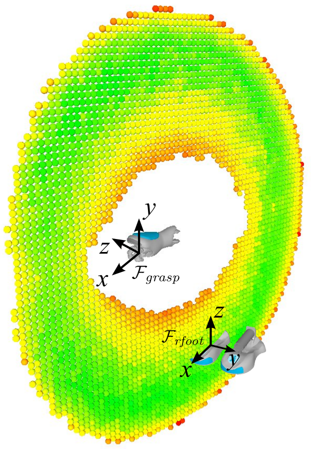


Figure 4.11: Cross-section through the IRM showing potential right foot locations (left foot is parallel) relative to the right hand of the robot. Voxels are colored by their manipulability index (red = low, green = high).

Algorithm 4: Reachability Map Inversion

```
Input  : Reachability Map $RM$
Output : Inverse Reachability Map $IRM$
1  while $v \leftarrow$ getVoxel($RM$) do
2      $n_c \leftarrow$ getNumConfigs($v$)
3      for $i = 1$ to $n_c$ do
4          $(\mathbf{q}_c, \mathbf{q}_{SWL}, w) \leftarrow$ getConfigData$(v, i)$
5          $\mathbf{p}_{tcp} \leftarrow$ computeTCPpose($\mathbf{q}_c$)
6          $\mathbf{p}_{base} \leftarrow (\mathbf{p}_{tcp})^{-1}$
7          $idx \leftarrow$ findEEvoxel($\mathbf{p}_{base}$)
8          $IRM \leftarrow$ addConfigToVoxel$(idx, \mathbf{q}_c, \mathbf{q}_{SWL}, w)$
9      end
10 end
```

the probability of successful task execution. For this purpose, we use an *inverse reachability map* (IRM) that represents potential base or feet poses relative to the end-effector. The IRM is generated by inverting the previously generated reachability map. The individual steps performed for inverting the reachability information are shown as pseudo code in Alg. 4 and will be explained in detail in the following.

Given the reachability map RM as input, we iterate through its voxels v and for each of them in turn through the n_c configurations stored in it (line 1 of Alg. 4). For wheeled mobile manipulators, the i-th configuration of voxel v is represented by a data structure composed of the configuration of the sampled chain $\mathbf{q}_c$ and the manipulability measure w (line 4 of Alg. 4). For humanoids, the configuration of the swing leg $\mathbf{q}_{SWL}$ is additionally contained. By computing the inverse of the end-effector transformation $\mathbf{p}_{tcp}$, obtained by solving the forward kinematics for the sampled chain, we obtain the pose $\mathbf{p}_{base}$ of the support foot or mobile base with respect to the end-effector frame (lines 5-6 of Alg. 4). Equivalent to the

Algorithm 5: IRM-based Stance Pose Selection

```
Input  : Inverse Reachability Map $IRM$, target grasp pose $\mathbf{p}^{global}_{grasp}$
Output : Optimal Stance pose $\mathbf{p}^{global}_{stance}$
1 $IRM_{grasp} \leftarrow$ transformIRM($IRM, \mathbf{p}_{grasp}$)
2 $IRM_{floor} \leftarrow$ intersectIRMwithFloor($IRM_{grasp}$)
3 $IRM_{stance} \leftarrow$ checkStanceFeasibility($IRM_{floor}$)
4 $\mathbf{p}^{global}_{stance} \leftarrow$ getMaxManipVoxel($IRM_{stance}$)
```

reachability map construction in Alg. 3, we afterwards determine the index idx of the IRM voxel containing the support foot or mobile base in pose $\mathbf{p}_{base}$ and add the configuration to the inverse reachability map data structure (lines 7-8 of Alg. 4). Note, that the map inversion process does not invalidate any configurations of the RM. Thus, no additional check for constraint violation is required. The generated IRM represents a set of valid stance poses relative to the end-effector independent from any specific grasp configuration. A cross-section of the IRM, representing all right foot poses with respect to the right hand/gripper is shown in Figure 4.11. As with the RM, the IRM needs to be built only once in an offline step and can be subsequently used in all stance pose selection queries for arbitrary target grasp poses.

4.7 Inverse Reachability Map based Stance Pose Selection

Once the IRM has been computed, it can be used to determine the optimal stance pose for a given grasping pose. Here, we assume a 6D target pose for the end-effector given as

$$\mathbf{p}^{global}_{grasp} = (x, y, z, \varphi, \theta, \psi)^\top, \tag{4.13}$$

where $(x, y, z)^\top$ and $(\varphi, \theta, \psi)^\top$ is the position and orientation of the desired grasp pose with respect to the global frame $\mathcal{F}_{global}$. To obtain the set of potential stance poses for a specific grasp pose from the IRM, we perform the steps described in Alg. 5.

First, the IRM is transformed in order to align its center with the grasp frame $\mathcal{F}_{grasp}$ (line 1 of Alg. 5). Afterwards, we determine the intersection of the transformed map IRM_{grasp} with the ground plane on which the feet must be placed planar (line 2 of Alg. 5). The resulting layer IRM_{floor} of ground floor voxels represents all support foot positions from which the grasp pose $\mathbf{p}^{global}_{grasp}$ is reachable. However, the orientation of the support foot poses stored in IRM_{floor} is not necessarily planar to the ground plane. Therefore, our algorithm iterates through the voxels of the IRM_{floor} and eliminates invalid configurations. To do so, it determines for each configuration $\mathbf{q}$ stored in a voxel of IRM_{floor}, the pose of the support foot $\mathbf{p}^{global}_{SUP}$ with respect to the global frame as

$$\mathbf{p}^{global}_{SUP} = \mathbf{p}^{global}_{grasp} \oplus \mathbf{p}^{grasp}_{SUP}, \tag{4.14}$$

where the transformation $\mathbf{p}^{grasp}_{SUP}$ is obtained by solving the forward kinematics for the sampled chain in configuration $\mathbf{q}$. Given the z-axis of $\mathcal{F}_{global}$ being oriented perpendicular to the ground plane, we only need to check whether the roll and pitch angle of $\mathbf{p}^{global}_{SUP}$ is sufficiently close to zero in order to determine whether the stance pose is feasible (line 3 of Alg. 5). Note, that the roll and pitch angles are never exactly zero due to the discretization of the joint range used

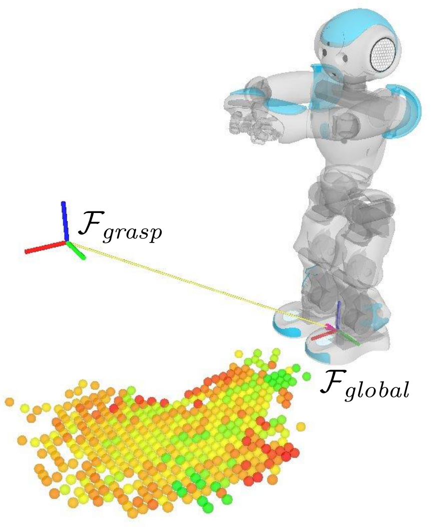


Figure 4.12: Potential stance poses for the right foot for reaching a desired grasping target with the right gripper. The left foot is always set parallel to the right. Voxels are colored by the maximum achievable manipulability among the configurations stored in the voxel (red = low, green = high).

within the forward reachability map generation process. Thus, we consider a certain orientation offset for the support foot relative to the ground plane to be acceptable. Furthermore, we exclude all support foot configurations, though being kinematically admissible, whose yaw angle relative to the grasp frame exceeds $\pm 90°$ (line 3 of Alg. 5). This operation eliminates unnatural configurations, for which the robot is heading in the opposite direction with respect to the object to be grasped. The resulting data structure IRM_{stance}, depicted in Figure 4.12, finally represents the set of statically stable, collision-free double support configurations from which the given grasp pose is reachable. Based on IRM_{stance}, an optimal stance pose $\mathbf{p}_{stance}^{global}$ is retrieved by finding the voxel containing the whole-body configuration with the highest manipulability measure among all voxel configurations stored in IRM_{stance} (line 4 of Alg. 5). For application in humanoid grasping tasks, such a pose could be subsequently fed to a footstep planning algorithm [49], generating a sequence of footsteps required to arrive at the desired stance location. Assuming perfect navigation capabilities of the robot, the whole-body configuration stored in the voxel could be furthermore used as a goal configuration for a bidirectional whole-body motion planner [13], once the stance destination has been reached. Thus, solving the inverse kinematics problem for the whole-body of a humanoid robot would no longer be required. Regarding wheeled mobile manipulators, optimal stance poses can be similarly used as terminal configurations for bidirectional motion planning algorithms, such as the BI^{2}RRT* algorithm, which we will introduce in Chapter 5.

4.8 Implementation Details

We implemented our approach for stance pose selection for mobile robotic platforms based on inverse reachability maps in ROS (Robot Operating System) and use FCL (Flexible Collision

Library) [97] to perform self-collision checks. We evaluate the stability of humanoid whole-body configurations by checking whether the projection of robot's CoM onto the ground plane is within the support polygon, assuming the robot to remain in double support. For solving the forward and inverse kinematics, we use KDL (Orocos Kinematics and Dynamics Library) [124] and find swing leg configurations placing the robot's foot on the ground using the Newton-Raphson numerical inverse kinematics solver. Additional, we developed functions that permit to build arbitrary serial chains from the kinematic tree structure representing the robot. Previously, it was only possible to generate kinematic chains going forwards along the branches of the kinematic tree, which is typically rooted at the torso for humanoid platforms.

4.9 Experimental Results

Wheeled mobile robotic platforms typically possess a kinematically redundant manipulator mounted at their base, that is capable of reaching a large set of end-effector poses within the extent of their workspace with an arbitrary orientation. On the other hand, humanoids typically exhibit limited reaching capabilities due to their relatively small joint ranges and the loop-closure and stability constraint involved. Therefore, we consider for the experimental evaluation of our approach only anthropomorphic kinematic structures, that particularly benefit from the use of an inverse representation of their reachability capabilities. In the following, we use the Nao V4 humanoid platform by SoftBank Robotics to demonstrate the improved success rate in reaching given grasping targets, achieved using our approach. The robot is 58 cm tall and has 25 DOF: 2 in the neck, 6 in each arm (including one to open and close the hand), and 5 in each leg. In addition, the legs share a common (linked) hip joint that cannot be controlled independently. Inertia, mass, and CoM of each link are known from CAD models. For efficient collision checks, we created a low-vertex collision mesh model for each of the robot's links from the CAD models, as depicted in Figure 4.4 (left). A detailed description of the humanoid platform is provided in Appendix A.1.

Generally, our approach is capable of performing reachability analysis for arbitrary sub-chains of the robot, e.g., for the chain leading from the torso down to one of the feet. However, here we specifically consider only kinematic chains rooted at one of the feet and ending at one of the grippers/hands, thus possessing 10 DOF. For building the reachability map for the Nao humanoid, we used a sampling resolution of 0.3 rad for the upper body joints and 0.2 rad for the lower body joints, which turned out to be a good compromise between performance and computational demands. The resulting IRM has a memory consumption of 5GB.

4.9.1 Selecting a Stance Pose for Grasping

In the first experiment, we analyze how the robot benefits from the knowledge about its own manipulation capabilities represented by the IRM. For the Nao robot, IRM_{stance} contains only 17% of the IRM_{floor} voxels and represents the stable, collision-free double support configurations from which the given grasp pose is reachable. Figure 4.12 shows the robot initially located at the global frame and all potential stance poses for the support foot (here right foot, left foot is always parallel) for a given grasping pose $\mathbf{p}_{grasp}^{global} = (0.5, 0.0, 0.3, 0.0, 0.0, 0.0)^\top$, specified for the robot's right gripper. Note, that each of the voxels (shown as spheres) can represent multiple stance poses of different orientations. The color of the voxels corresponds to

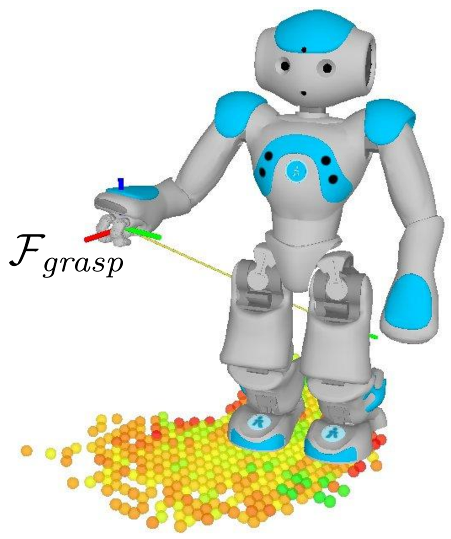


Figure 4.13: Example of a stance pose and whole-body configuration for reaching a grasping target $\mathbf{p}^{global}_{grasp}$.

the maximum manipulability measure encountered among the configurations stored in it. An optimal stance pose $\mathbf{p}^{global}_{stance}$ is easily obtained by selecting the first configuration stored in the voxel providing the highest manipulability measure among all IRM_{stance} voxels. Figure 4.13 shows a possible stance location and whole-body configuration for reaching the desired grasp pose $\mathbf{p}^{global}_{grasp}$.

4.9.2 IRM vs. RM based Stance Pose Selection

In order to quantitatively evaluate the improvement obtained regarding the ability of reaching specific grasping targets using inverse reachability information, we compared the success rate of our approach with respect to the strategy of selecting stance poses directly from the forward reachability map. To obtain meaningful foot placements from the RM, generated according to Section 4.4.1, we randomly selected voxels containing at least one support foot orientation from which the grasping target is reachable. Without further knowledge, this is a reasonable approach. In total, we sampled 10000 double support stance poses and checked whether the grasp pose lying within the extent of the RM is actually reachable from the individual poses with the correct end-effector orientation. As a result, only 17% of the sampled stance locations allowed to reach the correct 6D grasp pose. When using the IRM, a success rate close to 100% can be reached, depending on the joint sampling resolution chosen within the RM construction process (see Δq in Alg. 3). Thus, using the IRM, the number of target grasps successfully reached without the need for a repeated stance pose adaption is substantially increased. This shows that a humanoid can highly benefit from the knowledge about its own manipulation capabilities represented by the IRM. On average, it took 820 ms to compute IRM_{floor}, IRM_{stance} and to retrieve the optimal stance pose on an Intel Core i7 3.4GHz.

4.10 Related Work

In the literature, several techniques for generating robot configurations that achieve desired grasping poses with the end-effector already exist. For example, Stulp et al. [130] introduced the concept of *Action-Related Place* (ARPlace). ARPlace represents a probability mapping that specifies the expected probability that the target object will be successfully grasped, given the positions of the target object and the robot. The authors present results for a wheeled mobile base equipped with two 6-DOF manipulators.

Jamone et al. [61] proposed a method to allow a humanoid robot to learn a representation of its own reachable space, referred to as the *reachable space map*, from motor experience. To do so, the authors first learn an arm-gaze model, mapping end-effector poses to a specific gaze configuration for the head and eyes of the robot. Using the inverse of that model, an estimate of the reachable space map, i.e., a set of visually detected 3D points, for the robot's hand is obtained by solving a large number of IK queries. The authors differentiate between a basic and an enhanced map. While the former only specifies whether a fixated point in the gaze frame is reachable with the hand, the later additionally provides a quality measure about the degree of reachability.

Müller et al. [91] use a precomputed reachability map for the arm chain of a humanoid robot for efficiently planning grasping motions based on A* search. While this approach allows to quickly find collision-free motion trajectories and to adapt them in real-time, it does not exploit the manipulation capabilities of the whole-body kinematics to full extent. Moreover, stability constraints are not taken into account.

Zacharias et al. [150] proposed the so-called *capability map* which is a representation of kinematic reachability in the workspace for a robotic arm. The authors use a manipulability measure to evaluate poses in the workspace considering the distance to singularities [148]. Using the capability map, one can determine relative object positions that allow for good manipulation by the robot.

Berenson et al. [9] introduced the constrained bidirectional RRT (rapidly exploring random trees) planner that considers constraints as *task space regions* (TSRs). The authors used TSRs initially for goal pose specification and showed later that more complex constraints can be described by chaining the TSRs together. However, in this approach it is assumed that the robot is already in a good stance position relative to the object to be manipulated. Thus, the task is reduced to the problem of finding an optimal whole-body postures given the robot's current stance location.

The work presented by Vahrenkamp et al. [137] analyzes the workspace capabilities of a manipulator using an extended variant of the manipulability measure, previously introduced by Yoshikawa [148]. By deriving the manipulability measure from an augmented Jacobian, constructed based on penalization terms for joint limits considering a robot's redundancy, the distance between the robot's body parts, as well as the distance to obstacles in the environment, a more accurate representation of the reaching capabilities is obtained. Subsequently, Vahrenkamp et al. [138] used the extended measure to compute base positions for a wheeled mobile manipulator in order to reach different object grasps. Here, the authors precompute a so-called *inverse reachability distribution* (IRD) for the mobile base, which is centered at the robot's gripper. For a specific grasp, the IRD centroid is placed at the corresponding workspace pose and its volume is cut with the floor plane to find valid base positions and orientations. Later, Vahrenkamp and Asfour extended this approach towards bi-manual manipulation tasks [136]. However, loop-closure and stability constraints inherent to legged humanoid platforms are not considered at that stage.

Kaiser et al. [66] proposed an approach for the generation of whole-body locomotion and manipulation actions based on affordance hypotheses, which in turn are determined on the basis of visual and inertial sensing of environmental elements in the scene. In order to identify the most promising locations for the execution of an action, affordances that are out of reach are filtered using precomputed whole-body manipulability and stability maps. Although reachability analysis is not the focus of this work, it nicely demonstrated how perceptional data can be augmented using reachability information.

Recently, Paus et al. [99] presented an approach for determining configuration space trajectories that cover a specified workspace target area with the robot's end-effector. To do so, the authors combined robot placement and coverage path planning that takes constraints like collision avoidance and static stability into account. Since repositioning the robot is associated with significant costs in this work, the use of reachability information is considered to be of great importance. Therefore, a reachability map is build for the humanoid robot ARMAR-III following the approach presented by Vahrenkamp et al. [138] and the one presented in this chapter. Afterwards, this map is used to determine how much of a target surface can be reached from all possible robot placements with sufficient manipulability.

Extending the approach presented in this chapter, Yang et al. [147] recently proposed inverse dynamic reachability maps (iDRM) that allow humanoid robots to find valid and sufficient end-poses in complex and changing environments in real time. Initially, a reachability map is build in an offline step. Here, a full-body IK solver and sequential quadratic programming is used to generate feasible robot configurations, instead of adopting configuration space sampling and sample projection methods. Afterwards, the iDRM is used online and updated according to collisions detected between the iDRM voxels and environment obstacles. According to experiments with the *Valkyrie* humanoid robot, composed of 38 DOF, a valid end-pose is found within 0.1 seconds.

4.11 Conclusion

In this chapter, we presented an approach for efficient stance pose selection for mobile robotic systems elevating the probability of successfully reaching or grasping a desired object. The generated IRM, reflecting the robot's capability of reaching a grasping target from different stance locations is build only once in an offline step and can be subsequently used for arbitrary grasp pose queries. Considering humanoid robots, static stability is additionally taken into account in the process of generating a representation of their reachable workspace. Moreover, our studies revealed that an optimal stance pose selection is especially important if the number of available DOF's in the upper body of a mobile robot is limited and redundancy is not given.

As we have shown in our experiments with a Nao humanoid, equipped only with a 5 DOF arm, the set of potential stance poses providing high manipulability represented within the IRM is relatively small compared to the results for wheeled mobile platforms carrying a redundant manipulator [138]. This emphasizes the demand for intelligent stance pose selection for such platforms as realized by our technique. By considering the IRM for stance pose selection, we substantially increase the probability of successfully reaching a desired object, without the need of repeatedly adapting the stance location and calling an inverse kinematics solver. Note that our approach of building an IRM and its use for stance pose selection is general and can be also applied to legged humanoids and mobile manipulators with a higher number of degrees of freedom to efficiently select an optimal configuration for manipulation.

At the present time, the 3D voxels of the reachability map do not explicitly store the achievable end-effector orientations. Instead, they contain the configurations associated with the poses lying within their extent. In the future, voxels could be extended to 6D incorporating also end-effector orientations.

Furthermore, a static transform between support and swing foot $\mathbf{p}_{SWF}^{SUF}$, placing the feet parallel to each other, has been considered for generating double support configurations. An extension in this context, would be to allow flexible non-parallel planar swing foot placements in the reachability map construction process. Doing so, would enhance the reachability information by adding end-effector poses, which are otherwise not reachable by statically stable double support poses considering only parallel foot placements.

A major subject for future investigations is related to the manipulability measure used to compute a quality index for the configurations entered into the spatial reachability grid. Whereas the classical manipulability measure has been used in our approach to transfer the concept of inverse reachability maps to legged humanoid platforms, several extended variants of the manipulability measure have already been applied to wheeled mobile platforms and could also be adopted for humanoids. These extensions allow, for example, to additionally consider the proximity to joints limits taking into account the kinematic chain's redundancy, the distance to obstacles and among different body parts in the process of evaluating the quality of configurations. Another, humanoid specific, refinement of the manipulability index could be obtained by considering the distance of the CoM projection to the center of the support polygon not only for the evaluation of a configuration's validity, but also as an indicator for its quality. This way, configurations with robust stability are assigned a higher quality index than configurations exhibiting a critical proximity of their CoM projection to the support polygon boundaries. Finally, a more accurate representation of the reaching capabilities of a humanoid could be obtained by considering the torques emerging at the ankle joints in the forward reachability map construction process. By doing so, statically stable configurations with leg actuator torques approaching or exceeding the stall torques, defined according to the hardware specification of the robot, would be excluded from the set of potential stance poses.

Chapter 5

Sampling-Based Motion Planning for Task-Constrained Mobile Manipulation

Mobile manipulators installed in warehouses and factories for conveying goods between working stations need to meet the requirements of time-critical work-flows. Moreover, the systems are expected to deal with changing tasks, cluttered environments and constraints imposed by the goods to be delivered. Selecting an optimal stance pose for the robot, i.e., a terminal configuration for manipulation tasks, as presented in Chapter 4, can be considered in this context only as an initial step. The major challenge, however, lies in the problem of efficiently generating motion transitions safely guiding the robot and the object attached to its end-effector from their current pose to a desired target. In this chapter, we present a planning framework for generating asymptotically-optimal paths for mobile manipulators subject to task constraints. In particular, our approach introduces the Bidirectional Informed RRT* (BI^{2}RRT*) that extends the Informed RRT* [47] towards bidirectional search and satisfaction of arbitrary geometric end-effector task constraints. In various experiments, we demonstrate the efficiency of BI^{2}RRT* for both unconstrained and constrained mobile manipulation planning problems. As the results show, our planning framework finds better solutions than Informed RRT* and Bidirectional RRT* in less planning time.

Mobile manipulators are fast, dexterous robotic service systems with the ability of reliably and repetitively performing pick and place operations in warehouses and production facilities. Here, the time required by such systems to transfer an object from one location to another is a common decision criteria for their deployment. Often, the robotic systems handle pick and place tasks by decomposing them into a sequence of motion planning subproblems, *pick-transition-place* [26]. This decomposition allows to solve a set of independent low-dimensional planning problems, for the manipulator and the mobile base respectively, instead of solving a complex high-dimensional planning problem for the entire task at once. In particular, assuming a safe rest pose for the manipulator above the mobile base, the *transition* planning phase for the mobile manipulation task simplifies to a planar motion planning problem for a rigid box. Several asymptotically-optimal motion planning algorithms have been presented, among which rapidly-exploring random trees (RRT*) and probabilistic roadmaps

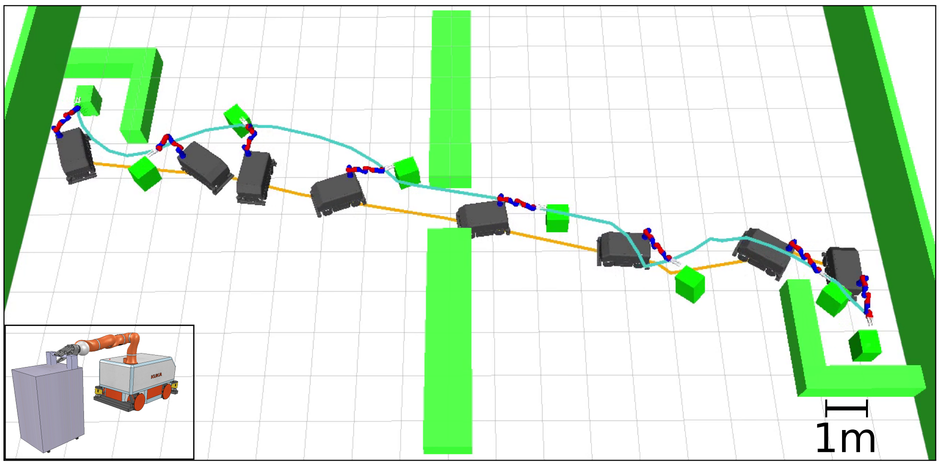


Figure 5.1: Configuration sequence of a solution path generated by our BI^{2}RRT* planning framework for the complex task of maneuvering a cart out of a parking lot and into another. The orange and blue line represent the corresponding trajectory of the mobile base and end-effector.

(PRM*) [69] are the most popular. Advanced variants, such as Informed RRT* [47] achieve an enhanced convergence rate using intelligent sampling techniques. Using one of these planners, optimal solutions can, in principle, be generated for the individual subtask. The global optimal solution, however, does not equal the sum of the local optimal solutions found for the subtasks. In fact, each terminal robot pose resulting from a motion plan generated for a subtask directly affects the set of solutions available for the subsequent motion planning problems. In the worst case, it even leaves this set empty, hence preventing them to find a solution at all. The results are globally suboptimal paths or a high number of unsuccessful motion planning queries. Optimal manipulation planning for the entire task, on the other hand, would result in globally optimal paths and a higher success rate, but is highly computationally demanding. Doing so, requires to plan coordinated base-manipulator motions, which respect the joint limits, avoid self-collisions as well as collisions with obstacles in the environment. Furthermore, constraints imposed by the object to be manipulated must be taken into account, as exemplary illustrated for the task of pulling a cart in Figure 5.1 .

In this chapter, we present a motion planning framework for task-constrained mobile manipulation that unifies asymptotically-optimal sampling-based bidirectional path planning [63] with informed sampling from an ellipsoidal subset [47] and task-constraint satisfaction based on the first-order retraction technique [128], projecting samples onto the constraint manifold [9]. In this way, we make use of the advantages of several existing techniques to achieve an increased rate of convergence also for highly constrained planning problems. The new algorithm, *Bidirectional* Informed RRT* (BI^{2}RRT*), extends the Informed RRT* algorithm [47] towards bidirectional search, informed sampling considering dexterous mobile robotic platforms, and satisfaction of arbitrary geometric task constraints. Experiments with a mobile manipulator show that our approach is capable of generating low-cost solutions paths to complex constrained mobile manipulation tasks. We compared the performance of our planning framework to state-of-the-art path planning algorithms on several planning problems of varying complexity and demonstrate that our method generates low-cost solution paths more reliable and faster than existing methods.

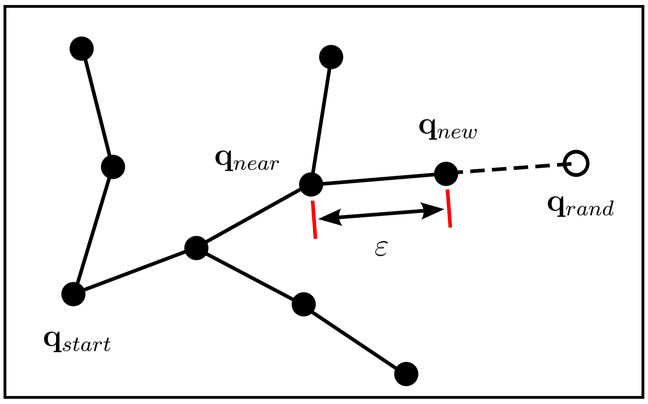


Figure 5.2: Tree expansion step of the Rapidly-Exploring Random Trees (RRT) algorithm.

5.1 Rapidly-Exploring Random Trees

The planning framework presented in this chapter belongs to the family of RRT-based algorithms, which are single-query probabilistic motion planning approaches searching for a path by incrementally expanding trees $\mathcal{T}$ in the configuration space. Introduced by LaValle and Kuffner [80], the RRT algorithm has shown to yield good performance, particularly for planning in high-dimensional configuration spaces. Nowadays, a variety of modified variants exist, that aim to further improve the performance of the original algorithm by adopting different strategies for the configuration space sampling and search tree expansion routine. Others extended the classical algorithm by additional components that permit RRT's to consider differential and task constraints during planning. As our approach is also based on the concept of growing random trees in the configuration space, we will briefly recapitulate the basic idea of RRT's and its descendants claiming the final solution path to be optimal with respect to certain criterions.

As opposed to probabilistic roadmaps (PRM) based algorithms [71], which perform planning on a precomputed graph covering the configuration space on a large scale, RRT's only explore the configuration space to an extent that is sufficient for solving a specific planning problem. Due to the approach of reusing the configuration space graph for every planning query, PRM's are referred to as *multi-query* algorithms in the literature. On the other hand, RRT's are *single-query* algorithms, in the sense that a new $\mathcal{C}$-space tree is grown for every problem instance. The routine to be repeatedly applied for growing the trees proceeds as follows. At the beginning, the search tree $\mathcal{T}$ is initialized with a root configuration. Then, at each iteration of the RRT algorithm a random configuration $\mathbf{q}_{rand}$ is uniformly sampled from the $\mathcal{C}$-space. Afterwards, the closest configuration to $\mathbf{q}_{rand}$ in $\mathcal{T}$, also called the nearest neighbor $\mathbf{q}_{near}$, is found and a new configuration $\mathbf{q}_{new}$ at a distance ε from $\mathbf{q}_{near}$ along the segment connecting $\mathbf{q}_{near}$ and $\mathbf{q}_{rand}$ is generated using a local planner. For better understanding, this step is represented graphically in Figure 5.2. Subsequently, both $\mathbf{q}_{new}$ and the edge connecting it to $\mathbf{q}_{near}$ are checked for collisions. If they are found to be collision-free, the search tree $\mathcal{T}$ is expanded by the new configuration and the associated edge. Note that no collision check needs to be performed for $\mathbf{q}_{rand}$, since its only use is to indicate a direction for the tree expansion. Thus, random samples are discarded by the planner at the end of each iteration.

Algorithm 6: RRT* Algorithm

```
Input  : Start state $\mathbf{x}_s$
Output : Search tree $\mathcal{T}$
$\mathcal{T}.init(\mathbf{x}_s)$
for $i = 1$ to $n$ do
    $\mathbf{x}_{rand} \leftarrow$ sampleConfig()
    $\mathbf{x}_{near} \leftarrow$ nearestConfig($\mathcal{T}$, $\mathbf{x}_{rand}$)
    $\mathbf{x}_{new} \leftarrow$ steer($\mathbf{x}_{near}$, $\mathbf{x}_{rand}$)
    if obstacleFree($\mathbf{x}_{near}$, $\mathbf{x}_{new}$) then
        addNode($\mathcal{T}$, $\mathbf{x}_{new}$)
        $\mathcal{X}_{near} \leftarrow$ nearNodes($\mathcal{T}, \mathbf{x}_{new}, r$)
        $\mathbf{x}_{parent} \leftarrow$ bestParent($\mathcal{T}, \mathcal{X}_{near}, \mathbf{x}_{new}$)
        addEdge($\mathcal{T}$, $\mathbf{x}_{parent}, \mathbf{x}_{new}$)
        rewireTree($\mathcal{T}$, $\mathcal{X}_{near}, \mathbf{x}_{new}$)
    end
end
return $\mathcal{T}$
```

The classical RRT planning algorithm grows a single tree, rooted at a start configuration $\mathbf{q}_{start}$, for a given period of time. The planning attempt is finally considered successful if the resulting tree $\mathcal{T}$ contains a node that is sufficiently close to the desired goal configuration. Alternative implementations occasionally try to connect to the goal configuration during the search by considering it as $\mathbf{q}_{rand}$ in the tree expansion procedure. Using this technique, a planning attempt is found successful when the goal configuration is added to the tree. A faster and more efficient variant is represented by the *bidirectional* RRT. With this planner, two trees, rooted at the start and goal configuration respectively, are grown. Here, the latest configuration added to a tree $\mathcal{T}_a$ is used as $\mathbf{q}_{rand}$ for the expansion of the other tree $\mathcal{T}_b$ in order to minimize the effort required for the search. A further, greedier version of bidirectional RRT, is presented by the *RRT-CONNECT* algorithm [78]. Here, the tree $\mathcal{T}_b$ proceeds expanding towards the latest node added to tree $\mathcal{T}_a$ until the two trees are either *connected* or an invalid configuration is encountered. Adopting this strategy, permits solving motion planning queries with minimal exploration of the configurations space. However, the cost of the best path returned by the aforementioned RRT variants converges to a random variable, thus yields a suboptimal value with probability one. Obtaining low cost solution paths from RRT-based approaches, thus requires to employ more sophisticated strategies in the planning process, as introduced in the RRT* algorithm [69].

5.1.1 Asymptotically-Optimal Motion Planning

The RRT* algorithm, presented by Karaman and Frazzoli [69], is an incremental sampling-based motion planning algorithm that provides an asymptotic optimality guarantee, i.e., almost-sure convergence to optimal solutions. The individual steps of the algorithm, repeatably applied in the planning process, are shown as pseudo code in Alg. 6. As with its suboptimal counterpart, the RRT* algorithm starts with an initialization of the search tree $\mathcal{T}$ (line 1 of

Alg. 6). As before, a start configuration or more general a start state $\mathbf{x}_s$ is used as the root of the tree. Afterwards, the algorithm enters a loop performing a prescribed number of n planning iterations (line 2 of Alg. 6). The first two steps are identical with the classical RRT implementation. First, a state $\mathbf{x}_{rand}$ is uniformly sampled from the state space of the robot (line 3). Afterwards, the nearest neighbor $\mathbf{x}_{near}$ in $\mathcal{T}$ to the random sample is determined and the tree is expanded using the *steer* function to generate a new state $\mathbf{x}_{new}$ (lines 4-5). The *obstacleFree* function checks whether the new state as well as the edge connecting it to $\mathbf{x}_{near}$ are valid (line 6 of Alg. 6). If both were found valid, the standard RRT proceeded by adding the new state and the corresponding edge to the tree. Contrary, RRT* only adds the new state to the tree and searches for the set of nodes $\mathcal{X}_{near}$ that are within a distance r from $\mathbf{x}_{new}$ (lines 7-8 of Alg. 6). Here, the Euclidean norm is typically considered as a metric to evaluate the distance between states. Subsequently, this set is used to find the parent state $\mathbf{x}_{parent}$, allowing to reach the new state along a collision-free path with minimum cost (lines 9-10). Considering the best, instead of the nearest node for the expansion therefore guarantees the tree topology to remain optimal regarding path costs, when new nodes are connected to the existing tree. An additional refinement of the tree structure is achieved by the *rewireTree* function, which tries to reduce the *cost-to-reach* for the near nodes of the current state $\mathbf{x}_{new}$ (line 11 of Alg. 6). Given the new state and its set of near nodes, it creates new edges from $\mathbf{x}_{new}$ to nodes in $\mathcal{X}_{near}$, if the path through $\mathbf{x}_{new}$ has a lower cost than the path through the current parent. In this case, the edge linking the node to its current parent is deleted in order to maintain the tree structure.

For RRT*, as for planning with RRT, a planning query is considered successful if the returned tree $\mathcal{T}$ (line 14 of Alg. 6) contains a node that is within a certain region around the desired goal state $\mathbf{x}_g$ or the goal state itself has been added to the tree during the search by occasionally considering it as $\mathbf{x}_{rand}$. Analogous to RRT, a bidirectional variant of RRT*, growing and simultaneously optimizing two trees rooted at a start and goal state respectively, has been introduced by Jordan and Perez [63].

5.1.2 Task-Constrained Motion Planning

In motion planning for mobile manipulation, configuration space samples are not only required to be collision-free, but also to comply with task constraints imposed on the end-effector by the object to be manipulated. In its simplest form, manipulation planning requires to additionally check for collisions between the object, the robot and environment obstacles during planning. More sophisticated tasks, such as delivering a glass of water, opening a door or pulling a trolley, furthermore claim specific rotational or translational components of the end-effector pose to remain within a certain tolerance interval throughout the motion. In order to permit RRT-based algorithms to efficiently cope with such task constraints during planning, several approaches have been proposed. A first possibility, introduced by Şucan and Chitta [34], is to pre-compute a *constraint approximation graph*, which is subsequently used in the sampling step of the RRT algorithm (e.g., line 3 of Alg. 6). Directly drawing task compliant samples from the graph is highly efficient regarding the runtime performance of the algorithm. The downside is that a new graph needs to be build every time the set of constraints changes, thus yielding low task flexibility of the planning framework. Two alternative solutions, *tangent-space sampling* (TS) and *first-order retraction* (FR), that perform constraint satisfaction during planning have been presented by Stilman [128]. The former method generates configuration samples at a small displacement from $\mathbf{q}_{near}$ and projects them into the null space of the task constraint. However, due to the non-linearity of the constraint manifold these samples are unlikely to be within error tolerance. Therefore, the *randomized gradient descent* (RGD) [81] technique is additionally applied to finally achieve task compliance. The FR method generates in a first step a configuration space sample $\mathbf{q}_{new}$, as depicted in Figure 5.2. Afterwards, a task space error and an associated joint space displacement is iteratively computed to gradually displace the sample towards the constraint manifold. Once the projected sample $\mathbf{q}'_{new}$ is found to be within error tolerance, it is used for extending the tree. In practice, FR is the preferred method, since it is found to be faster and more invariant to expansion step size and error tolerance than the TS and RGD method. Another approach, presented by Berenson et al. [9], introduces the concept of *Task Space Regions* (TSR) used to specify arbitrary constraints for manipulation tasks. Here, task space samples are drawn from TSR's and converted to configuration space samples for the robot using numerical inverse kinematics. Compared to previous approaches, this method has the advantage that multiple constraints can be linked together to form TSR chains. Moreover, using multiple of such chains and a metric for evaluating the distance to them it is possible to determine which chain to satisfy given the robot's current configuration.

5.2 Motion Planning With Bidirectional Informed RRT*

Our system builds upon the Informed RRT* algorithm by Gammell et al. [47], which has already demonstrated the ability to find optimal solutions to planning problems in high-dimensional domains. As the authors concluded, only the addition of further samples from an ellipsoidal subset of the planning domain can lead to an improvement of the current solution path. Thus, directly sampling from this subset can result in a faster rate of convergence. In the following, we describe our bidirectional variant of Informed RRT*, which speeds up the search for a first solution and improves the convergence rate towards the optimal solution also for highly complex planning problems.

5.2.1 The BI^{2}RRT* Algorithm

Informed RRT* achieves an increased rate of convergence once an initial solution is found. Before that, it basically coincides with the classical RRT* planner. Obviously, the earlier an initial solution is found the more planning time is left for informed path refinement. Therefore, we developed a *two-tree* variant of Informed RRT*, which quickly generates a first solution, in turn triggering informed sampling at an earlier planning stage. Furthermore, we consider *two* ellipsoidal subsets, used to perform informed sampling for the translational and rotational component of the configuration space of a mobile manipulator platform. Alg. 7 shows the pseudo code of our BI^{2}RRT* algorithm. For simplicity, we first explain the basic functionality of our planner in the absence of end-effector task constraints as given by $\mathbf{t}_{cc}$ and $\mathbf{B}$. How those constraints are considered is afterwards described in Section 5.2.2 and Section 5.3.

Given a start and goal end-effector pose $\mathbf{p}^e_s$, $\mathbf{p}^e_g$, our algorithm generates a start and goal configuration for the root nodes n_s, n_g of the two trees $\mathcal{T}_a$, $\mathcal{T}_b$, respectively (lines 1-2 in Alg. 7, details in Section 5.2.2). Afterwards, the current solution path cost c_{SP} is initialized, and the algorithm starts growing the search trees until either a maximum number of iterations or time elapsed is reached (lines 3-4). The *sampleRandomConfig* function (line 5, see Alg. 8) returns

Algorithm 7: Bidirectional Informed RRT* Algorithm

```
Input  : Start $\mathbf{p}_s^e$ and goal $\mathbf{p}_g^e$ end-effector pose, vector of constraint task coordinates
         $\mathbf{t}_{cc}$, matrix of task coordinates bounds $\mathbf{B}$, number of near nodes $k_{nv}$
Output : A solution path or FAILURE

$n_s.q \leftarrow$ generateIKsolution($\mathbf{p}_s^e$, $\mathbf{t}_{cc}$, $\mathbf{B}$)
$n_g.q \leftarrow$ generateIKsolution($\mathbf{p}_g^e$, $\mathbf{t}_{cc}$, $\mathbf{B}$, $n_s.q$)
$\mathcal{T}_a.init(n_s)$; $\mathcal{T}_b.init(n_g)$; $c_{SP} \leftarrow \infty$
for $i = 1$ to max_time_iter do
    $\mathrm{q}_{rand} \leftarrow$ sampleRandomConfig($c_{SP}$, $n_s.q$, $n_g.q$)
    $\mathrm{q}_{nn} \leftarrow$ findNearestNeighbor($\mathcal{T}_a$, $\mathrm{q}_{rand}$)
    $\mathcal{L}_{nv}$, $\mathcal{V}_a$, $\mathcal{V}_b \leftarrow \emptyset$
    $ext\_state_{nn}$, $ext\_state_{nv}$, $ext\_state_c \leftarrow$ FAILED
    $ext\_state_{nn}, \mathcal{V}_a \leftarrow$ extendTree($\mathcal{T}_a$, $\mathrm{q}_{nn}$, $\mathrm{q}_{rand}$, $\mathbf{t}_{cc}$, $\mathbf{B}$)
    if ($c_{SP} < \infty$ or $ext\_state_{nn}$ = FAILED) then
        $\mathcal{L}_{nv} \leftarrow$ findNearNodes($\mathcal{T}_a$, $\mathrm{q}_{rand}$)
        $ext\_state_{nv}, \mathcal{V}_a \leftarrow$ bestParentSearch($\mathcal{T}_a$, $\mathrm{q}_{rand}$, $\mathcal{L}_{nv}^{k_{nv}}$)
    end
    if ($ext\_state_{nn} \neq$ FAILED or $ext\_state_{nv} \neq$ FAILED) then
        insertSegments($\mathcal{T}_a$, $\mathcal{V}_a$)
        if $c_{SP} < \infty$ then
            $\mathcal{T}_a \leftarrow$ rewireTree($\mathcal{L}_{nv}^{k_{nv}}$, $\mathcal{T}_a$.last.q)
            $c_{SP} \leftarrow$ recursiveCostUpdate($\mathcal{T}_a$)
        end
        $\mathrm{q}_{nn} \leftarrow$ findNearestNeighbor($\mathcal{T}_b$, $\mathcal{T}_a$.last.q)
        $ext\_state_c, \mathcal{V}_b \leftarrow$ extendTree($\mathcal{T}_b$, $\mathrm{q}_{nn}$, $\mathcal{T}_a$.last.q, $\mathbf{t}_{cc}$, $\mathbf{B}$)
        if $ext\_state_c \neq$ FAILED then
            insertSegments($\mathcal{T}_b$, $\mathcal{V}_b$)
            if $ext\_state_c$ = REACHED then
                $c_{SP,new} \leftarrow$ getSolutionCost($\mathcal{T}_a$.last.q, $\mathcal{T}_b$.last.q)
            end
        end
        if $c_{SP,new} < c_{SP}$ then
            $c_{SP} \leftarrow c_{SP,new}$
        end
    end
    swap($\mathcal{T}_a$, $\mathcal{T}_b$)
end
if $c_{SP} < \infty$ then
    return path($\mathcal{T}_a$, $\mathcal{T}_b$)
else
    return FAILURE
end
```

Algorithm 8: Random Configuration Sampling

```
Input  : Current solution path cost $c_{SP}$, start and goal configuration $\mathbf{q}_s$, $\mathbf{q}_g$
Output : Random configuration $\mathbf{q}_{rand}$
1  conf_valid ← FALSE
2  while conf_valid = FALSE do
3      if $c_{SP} < \infty$ then
4          $\mathbf{q}_{rand,p}$ ← sampleFromEllipse($c_{SP,p}$, $\mathbf{q}_{s,p}$, $\mathbf{q}_{g,p}$)
5          $\mathbf{q}_{rand,r}$ ← sampleFromEllipse($c_{SP,r}$, $\mathbf{q}_{s,r}$, $\mathbf{q}_{g,r}$)
6          $\mathbf{q}_{rand}$ ← $[\mathbf{q}_{rand,p}, \mathbf{q}_{rand,r}]$
7      else
8          $\mathbf{q}_{rand}$ ← getRandomConfig()
9      end
10     conf_valid ← isConfigValid($\mathbf{q}_{rand}$)
11 end
12 return $\mathbf{q}_{rand}$
```

a configuration $\mathbf{q}_{rand}$, randomly sampled from the entire configuration space or the informed ellipsoidal subset (the latter is detailed in Section 5.2.3), depending on whether a solution path is already available or not. Afterwards, *findNearestNeighbor* finds the nearest neighbor $\mathbf{q}_{nn}$ in $\mathcal{T}_a$, to the configuration $\mathbf{q}_{rand}$ (line 6). The *extendTree* function (see Alg. 9) tries to establish a connection between the two configurations by incrementally stepping from $\mathbf{q}_{nn}$ towards $\mathbf{q}_{rand}$ (line 3 in Alg. 9) while performing collision checks for the intermediate configurations (line 9 in Alg. 9). As a result, the *extendTree* operation returns a list of segments $\mathcal{V}$, i.e., pairs of nodes and edges, together with information about the expansion status, *ext_state* (line 9 in Alg. 7). The value of *ext_state* corresponds to either *FAILED*, *PROGRESS*, or *REACHED*, indicating that tree $\mathcal{T}_a$ has not been extended at all, made some progress towards $\mathbf{q}_{rand}$, or has reached $\mathbf{q}_{rand}$, respectively. As opposed to the classical Informed RRT*, our planner performs a best parent search, defined by *bestParentSearch*, not only when a solution is available, but also when the tree expansion from $\mathbf{q}_{nn}$ failed to make some progress (lines 10-12 in Alg. 7). Here, the function *findNearNodes* provides a set $\mathcal{L}_{nv}^{k_{nv}}$ of k_{nv} near nodes, stored in the order of ascending *cost-to-reach*. If either the expansion from $\mathbf{q}_{nn}$ or from one of the near nodes, stored in $\mathcal{L}_{nv}^{k_{nv}}$, has made at least some progress towards $\mathbf{q}_{rand}$, the segments $\mathcal{V}_a$ returned by *extendTree* are added to the tree $\mathcal{T}_a$ (line 15 in Alg. 7). Lines 16-18 in Alg. 7 correspond to the standard rewire operation of RRT*, considering the last node added to the tree $\mathcal{T}_a$.last.q and the k_{nv} near nodes of $\mathcal{L}_{nv}^{k_{nv}}$ with the highest *cost-to-reach*. Note, that the reduction of the *cost-to-reach* for a single node needs to be propagated through the tree to the leaf nodes. By doing so, an improvement of the current solution path c_{SP} may be elicited (line 18 in Alg. 7). The last part of the planner, described in lines 20-27 of Alg. 7, basically corresponds to the classical *connect* step of bidirectional search. First, the nearest configuration $\mathbf{q}_{nn}$ in tree $\mathcal{T}_b$, to the last node added to $\mathcal{T}_a$ is determined (line 20 in Alg. 7). Afterwards, *extendTree* tries to find a new solution path by connecting $\mathbf{q}_{nn}$ to $\mathcal{T}_a$.last.q. If *extendTree* returns *REACHED*, a new solution path is found. Its cost $c_{SP,new}$, however, does not necessarily constitute a better solution. Therefore, we finally compare $c_{SP,new}$ to the cost of the current best solution c_{SP}. If

Algorithm 9: Search Tree Extension

```
Input  : Search tree $\mathcal{T}$, random configuration sample $\mathbf{q}_{rand}$, nearest neighbor $\mathbf{q}_{nn}$,
         vector of constraint task coordinates $\mathbf{t}_{cc}$, matrix of task coordinates bounds $\mathbf{B}$
Output : State of extension $ext\_state$, list of path segments $\mathcal{V}$
1  $ext\_state \leftarrow$ FAILED ; $\mathcal{V} \leftarrow \emptyset$ ; conf_valid $\leftarrow$ TRUE
2  while conf_valid = TRUE do
3      $\mathbf{q}_{ext}, \mathbf{e}_{ext} \leftarrow$ stepTowardsSample($\mathcal{T}$, $\mathbf{q}_{nn}$, $\mathbf{q}_{rand}$)
4      if $\mathbf{t}_{cc}$ = 0 or $\mathbf{q}_{ext}$ = $\mathbf{q}_{rand}$ then
5          $\mathbf{q}_{new}, \mathbf{e}_{new} \leftarrow \mathbf{q}_{ext}, \mathbf{e}_{ext}$
6      else
7          $\mathbf{q}_{new}, \mathbf{e}_{new} \leftarrow$ projectConfigFR($\mathbf{q}_{ext}$, $\mathbf{t}_{cc}$, $\mathbf{B}$)
8      end
9      conf_valid $\leftarrow$ isConfigValid($\mathbf{q}_{new}$)
10     if conf_valid = TRUE then
11         $\mathbf{q}_{nn} \leftarrow \mathbf{q}_{new}$
12         $ext\_state \leftarrow$ PROGRESS
13         $\mathcal{V} \leftarrow$ addSegment($\mathbf{q}_{new}, \mathbf{e}_{new}$)
14         if $\mathbf{q}_{new} = \mathbf{q}_{rand}$ then
15             $ext\_state \leftarrow$ REACHED
16             break
17         end
18     end
19 end
20 return $ext\_state, \mathcal{V}$
```

$c_{SP,new}$ is found to be lower than c_{SP}, c_{SP} is updated, the trees $\mathcal{T}_a$ and $\mathcal{T}_b$ are swapped and the algorithm proceeds with the next iteration. After reaching the stopping criteria, Alg. 7 returns either the final solution path or *FAILURE*.

5.2.2 Tree Initialization

For planning, we assume the initial and desired end-effector poses $\mathbf{p}_s^e$ and $\mathbf{p}_g^e$, as needed for an object manipulation task, to be given. Our algorithm allows for specifying a vector $\mathbf{t}_{cc}$ of constrained task coordinates and a matrix $\mathbf{B}$ of task coordinate bounds (see Section 5.3.1), e.g., to describe task constraints corresponding to carrying objects with an upright orientation. The vector $\mathbf{t}_{cc}$ is simply constituted by binary values indicating whether a task coordinate is constrained. For those coordinates being constrained, the matrix $\mathbf{B}$ defines a lower and upper admissible displacement from $\mathbf{p}_s^e$ and $\mathbf{p}_g^e$, respectively. Considering these constraints, the *generateIKsolution* function (line 1 in Alg. 7) finally runs a damped least-squares controller [24] using random configuration space seeds to generate a valid inverse kinematics solution for the mobile manipulator end-effector pose $\mathbf{p}_s^e$. For the generation of an inverse kinematics solution for $\mathbf{p}_g^e$, we use seeds that are generated by Gaussian sampling around the start configuration $n_s.q$ (line 2 in Alg. 7). This approach yields two advantages. First, it is easier for the planner to find a solution path when the terminal configurations are similar. Second, we avoid initializations of the planner in mobile manipulation tasks with terminal configurations that lie in disjoint regions of the constraint manifold, thus making it impossible to find a valid solution path. Alternatively, it is also possible to select terminal configurations for the specified end-effector poses using a pre-computed inverse reachability map (IRM), as presented in Chapter 4. In the absence of task constraints, i.e., for unconstrained motion planning, it is reasonable to select the pose providing the highest quality measure from the IRM. In fact, using this approach does not affect the prospects of success for the planning phase, since the algorithm is allowed to explore the entire collision-free subspace $\mathcal{C}_{free}$. However, care must be taken when selecting terminal configurations based on IRM's for a constraint manipulation task. Here, a configuration for $\mathbf{p}_s^e$ can be chosen as done before in the unconstrained case. On the other hand, selecting a configuration for $\mathbf{p}_g^e$ would require to additionally filter the IRM for poses lying in the same region of the constraint manifold as the configuration chosen for $\mathbf{p}_s^e$. Naively choosing the configuration with the highest quality measure would very likely result in terminal configurations lying in disjoint regions of the constraint manifold, thus rendering the discovery of a solution path impossible.

5.2.3 Informed Heuristic for Mobile Manipulators

Gammell et al. [47] proposed to directly sample from an ellipsoidal subset of the configuration space to achieve an increased rate of convergence towards the optimal solution, once an initial solution has been found. The subset of configurations that can improve the current solution, is referred to as $\mathcal{C}_f \subseteq \mathcal{C}$. Here, uniformly distributed samples in a hyper-ellipsoid are calculated from

$$\mathbf{q}_{\hat{f}} = \mathbf{R}\mathbf{L}\mathbf{q}_{ball} + \mathbf{q}_{centre}, \tag{5.1}$$

where $\mathbf{q}_{ball}$ are uniformly distributed samples from the unit n-dimensional ball and $\mathbf{q}_{centre}$ is the centre of the hyper-ellipsoid, following from two focal points $\mathbf{q}_g$, $\mathbf{q}_s$ (see Figure 5.3). The matrix $\mathbf{R}$ is given by

$$\mathbf{R} = \mathbf{U}\,\mathsf{diag}\{1, \dots, 1, \det(\mathbf{U})\det(\mathbf{V})\}\,\mathbf{V}^\top, \tag{5.2}$$

where $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ follows from the singular value decomposition of a matrix $\mathbf{M}$, in turn given by the outer product $\mathbf{a}_1\mathbf{I}_1^\top$. Here, $\mathbf{I}_1$ represents the first column of the identity matrix and $\mathbf{a}_1$ is defined as

$$\mathbf{a}_1 = (\mathbf{q}_g - \mathbf{q}_s)/\|\mathbf{q}_g - \mathbf{q}_s\|_2. \tag{5.3}$$

The matrix $\mathbf{L}$ follows from

$$\mathbf{L} = \mathsf{diag}\left\{\frac{c_{SP}}{2}, \frac{\sqrt{c_{SP}^2 - c_{HS}^2}}{2}, \dots, \frac{\sqrt{c_{SP}^2 - c_{HS}^2}}{2}\right\}, \tag{5.4}$$

with c_{SP} and c_{HS} describing the current and hypothetical solution path cost, respectively. Here, we defined c_{HS} as the cost of the linear interpolation between the start and goal configuration, neglecting obstacles and task constraints.

So far, informed sampling has been performed in configuration spaces solely composed of either revolute or prismatic components [47, 48]. In this work, we split the configuration space of the mobile manipulator into these components in order to perform informed sampling

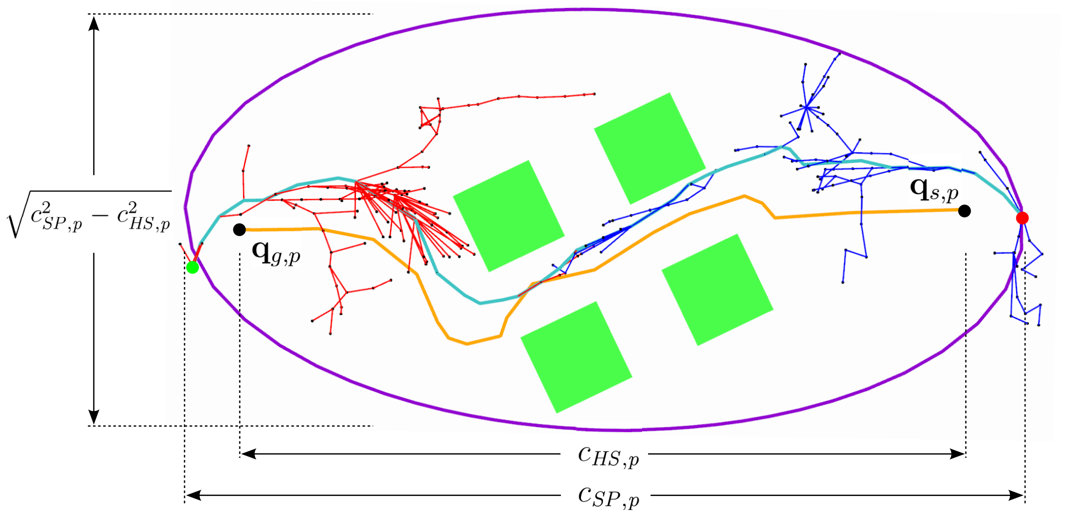


Figure 5.3: Informed subset for the prismatic components of the configuration space (purple ellipse), i.e., the translation of the mobile base. The eccentricity of the ellipse is given by $c_{HS,p}/c_{SP,p}$. The start and goal configuration of the mobile base is given by $\mathbf{q}_{s,p}$ and $\mathbf{q}_{g,p}$. Green blocks constitute obstacles. The orange and blue line represent the current solution path for the mobile base and end-effector. The red and green sphere correspond to the start and desired goal pose of the end-effector.

for each of them. In doing so, we consider in Eq. (5.1)-(5.4) the prismatic and revolute components of the configuration space, i.e., $c_{SP,p}$, $c_{HS,p}$, $\mathbf{q}_{s,p}$, $\mathbf{q}_{g,p}$ and $c_{SP,r}$, $c_{HS,r}$, $\mathbf{q}_{s,r}$, $\mathbf{q}_{g,r}$, respectively. For a better understanding, the quantities used to perform informed sampling for the prismatic components of the configuration space, i.e., the translation of the mobile base (n = 2), are graphically depicted in Figure 5.3. The result are two components $\mathbf{q}_{rand,p}$ and $\mathbf{q}_{rand,r}$ generated from distinct hyper-ellipsoids (lines 4-5 in Alg. 8), which are stacked after sampling (line 6 in Alg. 8) in order to obtain an informed configuration space sample for the entire robot.

5.3 Sample Projection According to Task Constraints

Considering end-effector task constraints in mobile manipulation planning, requires an advanced sample projection method within the *extendTree* function (line 7 in Alg. 9). We therefore apply the *first-order retraction* method [128], which has already shown to be faster and more invariant to expansion step size (used in *stepTowardsSample*, line 3 in Alg. 9) and error tolerance than other projection methods. Additionally, we adopt the idea of constraining task coordinates to remain in bounded intervals instead of considering fixed values [9]. In the following, we briefly recapitulate the definition of task constraints before explaining in more detail how task constraints are satisfied within our planning framework.

5.3.1 Definition of Task Constraints

For the definition of end-effector task constraints, we introduce the notions *task frame* $\mathcal{F}^t$, *coordinate constraint vector* $\mathbf{t}_{cc}$, and *task coordinate bounds* $\mathbf{B}$. The task frame represents

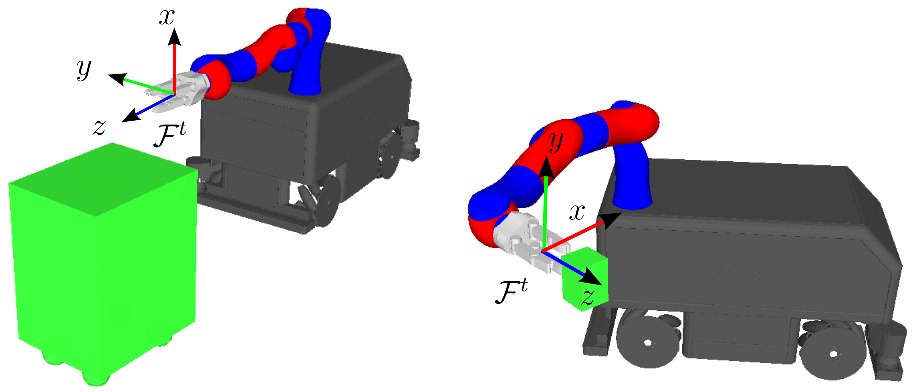


Figure 5.4: Task frame for pulling a cart (left) and transporting liquids (right) used for the specification of end-effector task constraints in the first-order retraction sample projection method.

a reference coordinate system, in which $\mathbf{t}_{cc}$ and $\mathbf{B}$ are expressed (see Figure 5.4). The binary elements of $\mathbf{t}_{cc}$ are used to compose a diagonal constraint selection matrix of allowed translational and rotational displacement from $\mathcal{F}^t$

$$\mathbf{C} = \mathsf{diag}\left\{t_{cc,1}, \ldots, t_{cc,n}\right\}, \tag{5.5}$$

with $t_{cc,i} \in [0, 1]$ and $n = 6$. The matrix $\mathbf{B}$ additionally defines lower and upper bounds for the displacement for those task coordinates constrained by $\mathbf{C}$,

$$\mathbf{B} = \begin{bmatrix} \Delta x_{neg} & \Delta x_{pos} \\ \Delta y_{neg} & \Delta y_{pos} \\ \Delta z_{neg} & \Delta z_{pos} \\ \Delta \varphi_{neg} & \Delta \varphi_{pos} \\ \Delta \theta_{neg} & \Delta \theta_{pos} \\ \Delta \psi_{neg} & \Delta \psi_{pos} \end{bmatrix}, \tag{5.6}$$

where the first and second column refers to the admissible negative and positive deviation of the coordinates from $\mathcal{F}^t$. Here, φ, θ and ψ are used to denote the roll, pitch, and yaw angles. Note that we initialize the task frame $\mathcal{F}^t$ globally with the desired start pose of the end-effector frame $\mathbf{p}_s^e$. For manipulation of articulated objects, a parametrized task frame would be required, instead.

5.3.2 Satisfaction of Task Constraints

After having sampled a random configuration $\mathbf{q}_{rand}$, the *extendTree* function iteratively performs steps towards $\mathbf{q}_{rand}$ from the nearest neighbor $\mathbf{q}_{nn}$ until the random sample is reached or an invalid configuration is encountered (lines 3-9 in Alg. 9). A premise for the extension of a tree considering end-effector task constraints is a successful projection of each intermediate configuration $\mathbf{q}_{ext}$ onto the constraint manifold (line 7 in Alg. 9). To do so, *projectConfigFR* first computes the end-effector frame $\mathcal{F}^e$ by forward kinematics for $\mathbf{q}_{ext}$. The result is given by the transformation $\mathbf{T}_e^0(\mathbf{q}_{ext})$. Afterwards, the displacement of $\mathcal{F}^e$ with respect to the task frame $\mathcal{F}^t$ is found by

$$\mathbf{T}_e^t(\mathbf{q}_{ext}) = \mathbf{T}_0^t\mathbf{T}_e^0(\mathbf{q}_{ext}) = (\mathbf{T}_t^0)^{-1}\mathbf{T}_e^0(\mathbf{q}_{ext}), \tag{5.7}$$

where $\mathbf{T}_t^0$ is assumed to be a fixed transformation between the task frame $\mathcal{F}^t$ and world frame $\mathcal{F}^0$. Next, we need to represent the transform for end-effector displacement with respect to the task frame in task coordinates using the roll, pitch, yaw representation for describing the relative orientation

$$\Delta\mathbf{x} \equiv \mathbf{T}_e^t(\mathbf{q}_{ext}). \tag{5.8}$$

Given the matrix $\mathbf{C}$ from Eq. (5.5), we now select the relevant error components from $\Delta\mathbf{x}$ as follows

$$\Delta\mathbf{x}_{err}^c = (e_1^c, \ldots, e_n^c)^\top = \mathbf{C}\Delta\mathbf{x}, \tag{5.9}$$

$$e_i^c = \begin{cases} 0, & c_i = 0 \\ \Delta\mathbf{x}_i, & c_i = 1 \end{cases} \tag{5.10}$$

where c_i is the i-th element along the diagonal of matrix $\mathbf{C}$. Finally, we consider the admissible negative and possible deflection intervals for the task coordinates in order to obtain the task error components subsequently used in the sample projection method,

$$\Delta\mathbf{x}_{err}^b = (e_1^b, \ldots, e_n^b)^\top, \tag{5.11}$$

$$e_i^b = \begin{cases} 0, & b_{i1} \leq \Delta\mathbf{x}_{err,i}^c \leq b_{i2} \\ \Delta\mathbf{x}_{err,i}^c, & \Delta\mathbf{x}_{err,i}^c < b_{i1} \text{ or } b_{i2} < \Delta\mathbf{x}_{err,i}^c \end{cases} \tag{5.12}$$

where b_{i1} and b_{i2} are the values in the first and second column of the i-th row in the task coordinate bound matrix $\mathbf{B}$, defined according to Eq. (5.6).

Once the task error is identified, we need to find a mapping that generates joint motions, iteratively reducing the error until all its components are within the bounded intervals defined in $\mathbf{B}$. This is done by a Jacobian-based method. The classical Jacobian $\mathbf{J}^0$ is a matrix of partial derivatives relating joint space velocities to end-effector linear and angular velocities expressed in the world frame $\mathcal{F}^0$. As the task space error is defined w.r.t. the frame $\mathcal{F}^t$ in our case, we need to represent the Jacobian in the same frame. The corresponding task frame Jacobian $\mathbf{J}^t$ is obtained using the inverse rotation matrix $\mathbf{R}_t^0$ as follows

$$\mathbf{J}^t = \begin{bmatrix} \mathbf{R}_0^t & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_0^t \end{bmatrix} \mathbf{J}^0. \tag{5.13}$$

The lower three rows of $\mathbf{J}^t$ represent the mapping to angular velocities $\mathbf{w}$ of the end-effector. However, the end-effector angular velocity with respect to the task frame is not given by the rotational velocity of a set of orientation angles. Though, there exists a relationship between the angular velocity and fixed-axis rotational velocities for a given set of orientation angles [128]. For RPY angles, the angular velocity vector is found by summing the fixed-axis velocities, defined according to

$$\mathbf{w} = \mathbf{E}_{rpy}^{-1}(\mathbf{q}_{ext})(\dot{\varphi}, \dot{\theta}, \dot{\psi})^\top. \tag{5.14}$$

By inverting $\mathbf{E}_{rpy}^{-1}$, we obtain $\mathbf{E}_{rpy}$, which represents a matrix mapping angular velocities to fixed-axis velocities. In order to additionally account for the linear velocities, we augment matrix $\mathbf{E}_{rpy}$ with an 3×3 identity matrix yielding

$$\mathbf{E}(\mathbf{q}_{ext}) = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_{rpy} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{3\times3} & \cdots & 0 & \cdots \\ \vdots & c_\psi/c_\theta & s_\psi/c_\theta & 0 \\ 0 & -s_\psi & c_\psi & 0 \\ \vdots & c_\psi s_\theta/c_\theta & s_\psi s_\theta/c_\theta & 1 \end{bmatrix}, \tag{5.15}$$

where s_i and c_i denote the sine and cosine of the rotation angle $i \in \{\varphi, \theta, \psi\}$, respectively. Using $\mathbf{E}(\mathbf{q}_{ext})$, we finally obtain a task Jacobian that considers a mapping to fixed-axis rotational velocities of the end-effector

$$\mathbf{J} = \mathbf{E}(\mathbf{q}_{ext})\mathbf{J}^t(\mathbf{q}_{ext}). \tag{5.16}$$

Subsequently, the pseudo-inverse of the modified task Jacobian $\mathbf{J}^\dagger$ is used to map the task error $\Delta\mathbf{x}_{err}^b$, expressed in frame $\mathcal{F}^t$, to the least-norm velocities in joint space required to correct it:

$$\dot{\mathbf{q}} = \mathbf{J}^\dagger \Delta\mathbf{x}_{err}^b. \tag{5.17}$$

Note, that in *isConfigValid* (line 9 in Alg. 9) a configuration $\mathbf{q}_{new}$ is found invalid not only if it is in collision, but also when it is equal to $\mathbf{q}_{nn}$ (*backprojection*) or has not made progress towards $\mathbf{q}_{rand}$ (*divergence*).

5.4 Implementation Details

Our planner is implemented in the *MoveIt!* framework in ROS [27], uses FCL [97] for collision checks and is available open source[1]. When the planner checks a configuration for validity, the collision mesh model of each robot link is tested for self-collisions and collisions with the environment, considering also objects potentially attached to the end-effector. Computation of the forward kinematics for single configurations is done using the *KDL* library [124]. For computing the pseudo-inverse of the task frame Jacobian in the first-order retraction method, we use the singular value decomposition implemented in the *Eigen* library [53].

5.5 Experimental Results

For the experimental evaluation of our approach, we use the *omniRob* omnidirectional mobile manipulator platform by KUKA Robotics (see Appendix A.2), which is composed of 10 degrees of freedom. Its configuration is given by

$$\mathbf{q} = (\mathbf{q}_{base}, \mathbf{q}_{manip})^\top, \tag{5.18}$$

where

$$\mathbf{q}_{base} = (x, y, \theta)^\top, \tag{5.19}$$

$$\mathbf{q}_{manip} = (q_1, q_2, q_3, q_4, q_5, q_6, q_7)^\top, \tag{5.20}$$

[1]http://github.com/burgetf/mobile_manipulation_planning

denote the configuration of the base platform and manipulator chain, respectively. For planning mobile manipulation tasks, we modeled the planar motion of the base by two prismatic joints representing the translation of the robot in the x, y direction w.r.t. the world frame. For planning, we considered all DOFs of the robot, resulting in a 10-dimensional configuration space. The solution path cost c_{SP} is defined by the sum of the rotational and prismatic path length. Note that we adopted in our comparative experiments anytime variants of the standard RRT, Informed RRT and RRT-CONNECT planning algorithms in order to allow them to exploit the available planning time for finding alternative solution paths of lower cost. We derived these algorithms by selecting different combinations of the features *bidirectional search*, *informed sampling*, *tree optimization* (continuing the search after a first solution is found) in our planning framework. For the following experiments, planning was performed off-board on a single core of a standard desktop CPU (Intel Core i7, 3.4 GHz).

5.5.1 Planning Collision-Free Motions

In a first experiment, we conducted a quantitative comparison between our planning framework and other variants of RRT-based planning algorithms regarding the performance in planning collision-free motions for a complex scenario (Figure 5.5, first column). Here, the robot needed to travel from an initial configuration to a final configuration while avoiding self-collisions and collisions with objects in the environment. Each planner performed 100 runs, each given a maximum planning time of 2 minutes. According to the averaged results, the bidirectional variants generally find better solutions than the unidirectional variants in less time and more often. Moreover, our planning framework finds a first solution faster compared to the Informed RRT* (Figure 5.5, first column, fourth row). Thus, informed sampling is activated at an earlier planning stage and more time is left for the planner to optimize the current solution.

5.5.2 Transportation of Liquids

In the next planning scenario, we evaluated the performance of our planner w.r.t. other RRT-based planning algorithms in the presence of serious end-effector orientation constraints. Here, the robot needed to transport a container of liquid from one location to another (Figure 5.5, second column). To do so, it needed to maneuver the container out of a narrow space, then underneath a ceiling joist, and finally through a narrow gate, while respecting the task constraints. Again, each planner performed 100 runs, this time each given a maximum planning time of 3 minutes. The coordinate constraint vector for this task is defined w.r.t. the task frame, depicted on the right side in Figure 5.4, as follows

$$\mathbf{t}_{cc} = (0, 0, 0, 1, 0, 1)^{\top}. \tag{5.21}$$

Accordingly, we set the admissible negative and positive deflection from the task frame for the roll angle $\Delta\varphi_{neg}$, $\Delta\varphi_{pos}$ and yaw angle $\Delta\psi_{neg}$, $\Delta\psi_{pos}$ in matrix $\mathbf{B}$ to $-10°$ and $10°$, respectively. Note that we discarded the results of RRT and Informed RRT in the figures for this scenario due to their low performance. Regarding the time required to find a first solution, our results show that about 70% of the runs of bidirectional planners provide a solution after 50% of the total planning time of three minutes (Figure 5.5, second column, fourth row).

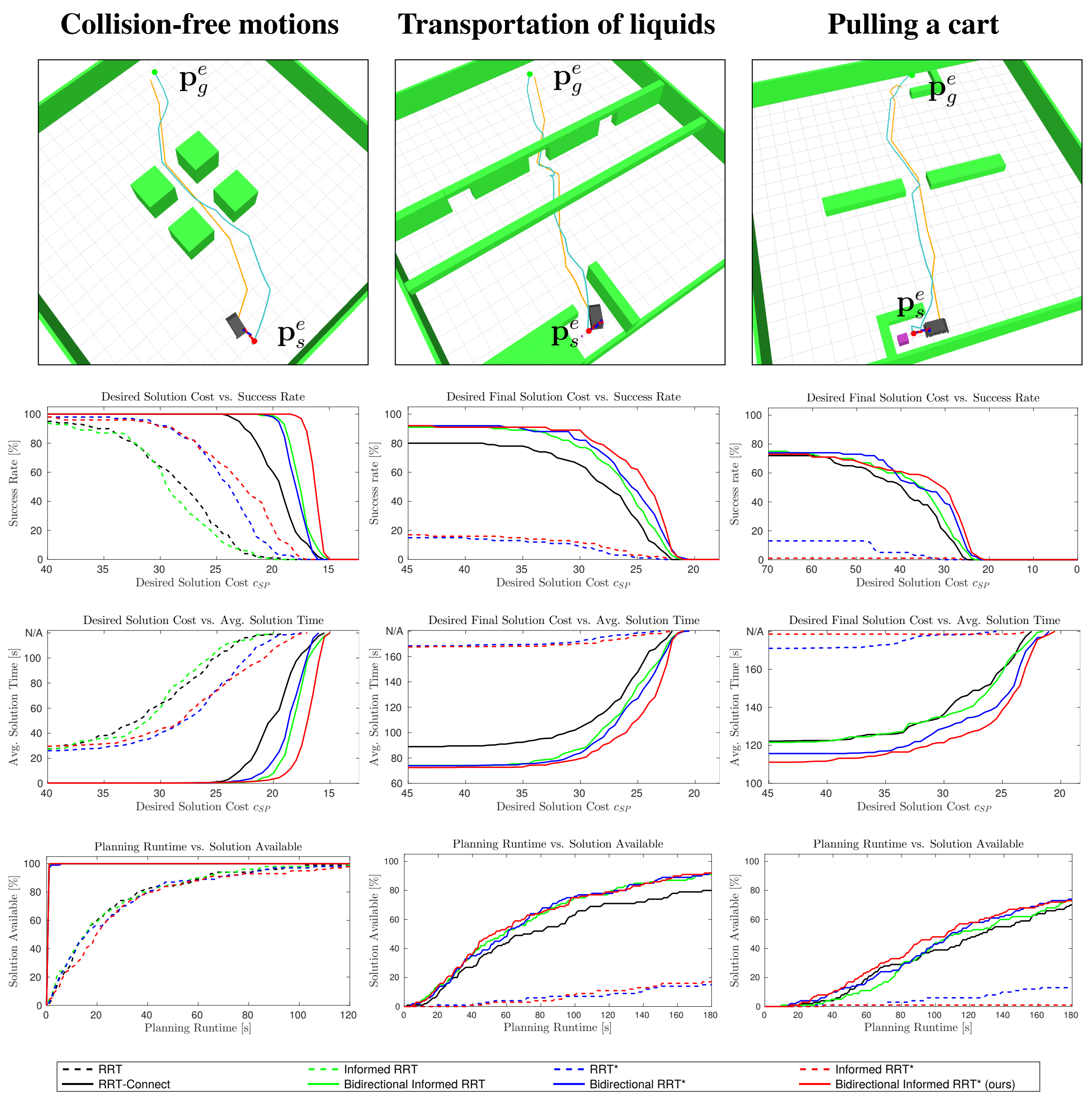


Figure 5.5: Comparison of the performances in planning collision-free motions (2 min planning time, first column), transportation of liquids (3 min planning time, second column) and cart pulling (3 min planning time, third column), averaged over 100 runs. Example solution paths are represented by their corresponding base (orange line) and end-effector (blue line) trajectory. First Row: Planning scenarios. Second Row: Percentage of successful runs w.r.t. different desired solution path costs. Third Row: Average time required to generate solutions of specific costs. Fourth Row: Percentage of runs providing a solution as a function of the planning runtime.

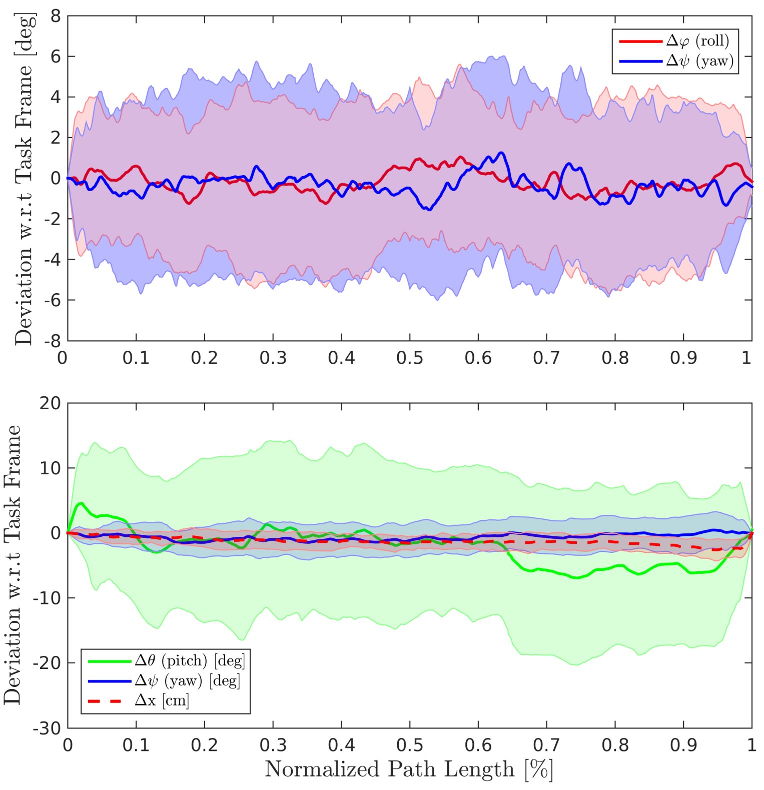


Figure 5.6: Average deviation of the constrained task coordinates from the task frame along the end-effector solution path for 100 runs of the transportation of liquids (top) and cart pulling (bottom) planning scenario.

Furthermore, the combination of the greedy *connect* heuristic with informed sampling in our planning framework has shown to find better solution more often and in less time. The Informed RRT*, on the other hand, needed to dedicate most of the planning time for the discovery of an initial solution, thus having almost no time left for further path improvements (Figure 5.5, second column, second and third row). The mean and standard deviation for the displacement of the constrained orientation angles from the task frame along the solution paths is depicted in Figure 5.6 (top row). As can be seen, both angles stay within the bounds, defined in B.

5.5.3 Pulling a Cart

In a final experiment, we quantitatively evaluated the performance of different planning algorithms considering end-effector position and orientation constraints. Here, the robot needed to maneuver a cart equipped with omnidirectional wheels out of a parking lot and into another, while respecting the aforementioned constraints (Figure 5.5, third column). To compare the results of our planning framework, we resorted to the same set of planning algorithms considered in Section 5.5.2. The coordinate constraint vector for this task is defined w.r.t. the task frame, depicted on the left side in Figure 5.4, as follows

$$\mathbf{t}_{cc} = (1, 0, 0, 0, 1, 1)^{\top}. \tag{5.22}$$

Here, we set the admissible negative and positive deflection from the task frame for the x-direction Δx_{neg}, Δx_{pos} and yaw angle $\Delta\psi_{neg}$, $\Delta\psi_{pos}$ in $\mathbf{B}$ to $\pm$3 cm and $\pm$5°, respectively. The pitch angle, on the other hand, is allowed to rotate the end-effector around the cart handle by $\Delta\theta_{neg} =$-30° and $\Delta\theta_{pos} =$30°. As our results show, the unidirectional variants RRT* and Informed RRT* fail to find a solution in most of the planning runs. In contrast, bidirectional planning algorithms are able to generate a solution in about 70% of the runs (Figure 5.5, third column, fourth row). Moreover, the few solutions generated by the unidirectional planning algorithms are found shortly before the available planning time runs out, whereas their bidirectional counterparts offer solutions of the same cost in a much earlier planning stage (Figure 5.5, third column, third row). Regarding the success rate for different desired solution path costs and the corresponding average time required to generate them, the Bidirectional RRT* algorithm shows a similar performance as our planner. Here, Bidirectional RRT* has a slightly higher success rate for solution paths of cost $c_{SP} > 42$, whereas our planner is superior in generating lower cost solution paths due to the integration of informed sampling (Figure 5.5, third column, second and third row). Furthermore, we evaluated the average deviations of the constrained coordinates from the task frame. Here, we obtain $\Delta\bar{x} =$-1.32 cm$\pm$1.19cm SD for the x-coordinate, $\Delta\bar{\theta} =$-2.16°$\pm$11.73° SD for the pitch angle and $\Delta\bar{\psi} =$-0.61°$\pm$2.39° SD for the yaw angle (see Figure 5.6, bottom row). Note that the resilience of the robot's kinematic structure and the object to be manipulated needs to be taken into account when the task coordinate bounds in $\mathbf{B}$ are defined for real world mobile manipulation tasks.

5.6 Related Work

Sampling-based motion planning approaches from the literature can be classified into techniques focusing on either the generation of optimal solution paths or on finding a solution to planning problems involving motion constraints. Approaches belonging to the former category extend classical motion planning algorithms by allowing the topology of a search tree or graph to improve over time, thus generating minimum cost transitions between their elements. Karaman and Frazzoli introduced PRM* and RRT* [69], which are in comparison to their *probabilistically complete* counterparts also *asymptotically optimal*, guaranteeing that the cost of the returned solution approaches the optimum as the number of samples goes to infinity. These algorithms have the property of improving the solution in the available computation time, however, do not guarantee a high rate of convergence. Furthermore, Karaman et al. [70] presented an extension of the RRT* algorithm toward anytime motion planning, interleaving planning with trajectory execution. The idea is to instantly trigger execution as soon as an initial feasible motion plan has been found whose remaining portions are improved over time.

A two-tree variant of RRT* proposed by Jordan and Perez [63] has shown to very rapidly provide solutions in planning problems with challenging regions such as narrow corridors, or high-dimensional configuration spaces with numerous obstacles.

Janson et al. [62] proposed the FMT* algorithm to increase the efficiency of optimal path search by extending graph-search methods to sampling-based algorithms. This approach generates *resolution-optimal* solutions in an asymptotically manner. A further improvement of the solution, however, requires to restart the search from scratch at a higher resolution.

Alterovitz et al. [4] introduced the rapidly-exploring roadmap algorithm that allows the user to explicitly control the trade-off between free-space exploration and solution path refinement.

The authors show that a careful choice of this parameter enables finding optimal paths more quickly. Similarly, Akgun and Stilman [2] presented a bidirectional variant of RRT* with path biasing that quickly finds an initial solution path. Afterwards the planner predominantly spends the remaining planning time for either refinement of the current path or exploration of other homotopy classes depending on a user-defined parameter.

The claim for optimal solutions usually comes along with a high computational effort. Therefore, several heuristics have been introduced to guide the search in order to achieve a faster convergence of the path cost towards the optimal solution. Nasir et al. [94] developed RRT*-Smart that follows a similar principle as visibility graph techniques. To accelerate the rate of convergence, it generates nodes as close as possible to obstacle vertices instead of adopting a purely random exploration. While the approach quickly reduces the solution cost, it may also cause other homotopy classes to remain undiscovered.

The Cloud RRT* algorithm, proposed by Kim et al. [72], allocates new samples from sampling clouds, initially generated from a generalized *Voronoi* diagram. During planning, these clouds are updated based on the set of configurations constituting better solutions. Additionally new clouds are added to refine the current best solution. However, global sampling remains active throughout the planning phase in order to explore understudied homotopy classes.

Otte and Correll [96] introduced the use of parallel computing for optimal motion planning. Their C-FOREST algorithm grows configuration space trees on multiple CPUs, that propagate improvements found on the current solution among each other. By exchanging this information, the configuration space region from which samples are drawn is collaboratively shrunk to an area of configurations potentially improving the current solution. The Informed RRT* algorithm, proposed by Gammell et al. [47], follows a similar approach of focused asymptotically-optimal motion planning. Here, the current solution is used to define an ellipsoidal subset of the planning domain, which is used to draw only samples that potentially improve the solution. The increased rate of convergence towards the optimal solution achieved in [96] and [47], however, vanishes for planning problems where most of the planning time needs to be dedicated to finding a first solution.

The BIT* algorithm [48] tries to overcome these limitations by combining sampling-based planning with incremental graph search techniques. Instead of shrinking, the ellipsoidal subset is incrementally enlarged and batches of configurations are added, which are subsequently used to update a search graph. Typically, large batch sizes are required to quickly converge to the optimal solution which, in the presence of task constraints, would require to compute a constraint manifold approximation in advance.

Efficient optimal motion planning for mobile manipulation, requires collision-free samples not only to be drawn from particular regions of the configuration space, but also to comply with task constraints imposed on the end-effector by the object to be manipulated. Şucan and Chitta proposed to precompute an approximation of the constraint manifold prior planning [34]. In this way, valid samples can be directly drawn from an approximation graph for tree expansion instead of applying computationally expensive rejection sampling or sample projection techniques. The drawback is that multiple approximation graphs are needed in order to perform planning for different types of constraints.

Stilman presented three modifications to the RRT algorithm for planning with task-space constraints, namely *randomized gradient descent* (RGD), *tangent-space sampling* (TS) and *first-order retraction* (FR) [128]. The results of his comparison indicated that the FR method, which iteratively displaces a sample toward the constraint manifold, is faster and more invariant to expansion step size and error tolerance than the RGD and TS technique.

Berenson et al. [9] described an approach for planning manipulation tasks using Task Space Regions (TSR). These regions define upper and lower bounds for constraint task coordinates and can be linked together in order to obtain more complex constraints. Here, samples are projected onto a region, as opposed to the approach of Stilman [128], where samples are required to be projected onto a single task space point. This leads to a faster and more successful projection of samples, thus generating solutions in a shorter amount of time.

Alternative approaches for task constrained mobile manipulation using graph-search techniques instead of sampling-based motion planning are presented by Chitta et al. and Scholz et al. [25, 115]. These implementations use anytime repairing A* to generate motions for opening a door or pushing a cart. The discretization, based on general motion primitives, leads in many cases to dramatic performance improvements in sampling-based planning algorithms. The main drawback is that these approaches are resolution-complete and provide only resolution-optimal paths.

5.7 Conclusions

In this chapter, we presented a framework for mobile manipulation planning under arbitrary geometric end-effector task constraints. Our BI^{2}RRT* planning algorithm uses the greedy *connect* heuristic to quickly find a first solution. To enable informed sampling for the full configuration space of a mobile manipulator, we propose to use two hyper-ellipsoids, representing subsets for the rotational and prismatic components of the configuration space. In this way, the solution can be improved if time allows. For constraint satisfaction, we adopted the first-order retraction method [128], which has shown to be a fast technique for projecting samples onto the constraint manifold. The experiments reveal that our planner generates solutions to complex mobile manipulation problems that satisfy all the desired constraints, e.g., to deliver a glass of water or a tool trolley. Moreover, we demonstrated in the evaluation on different planning scenarios that our approach is capable of providing low-cost solution paths more reliably and faster than existing state-of-the-art RRT-based algorithms.

A straightforward extension of our framework for future work is to integrate a sophisticated motion model in the local planner, used to perform the tree expansion operation, in order make the algorithm applicable to mobile robotic systems exhibiting differential drive or car-like kinematics [80]. Doing so, feasible motion trajectories are obtained for robotic platforms that do not permit instantaneous motions in directions perpendicular to their sagittal axis.

Moreover, the approach of specifying constraints with respect to a single task frame, also referred to as a *Task Space Region* in the literature [9], has been considered sufficient for the manipulation tasks presented in our experiments. Even though the use of a single TSR is generally admissible also for other tasks such as the manipulation of articulated objects, e.g., doors or drawers, it unnecessarily poses the planning problem overconstrained for such cases. A better solution for such planning scenarios would be obtained by defining constraints with respect to multiple task frames, constituting a TSR chain. Regarding the task of opening a door, for example, this would permit specifying task constraints in a way that the robot's end-effector is allowed to rotate around the door's handle while being constraint to follow the circular trajectory around the door's hinge.

Chapter 6

A Robotic Service Assistant for Users with Limited Communication Skills

As autonomous service robots become more affordable and thus available also for the public, there is a growing need for user friendly interfaces to control the robotic system. Currently available control modalities typically expect users to be able to express their desire through either touch, speech or gesture commands. While this requirement is fulfilled for the majority of users, paralyzed users may not be able to use such systems. In this chapter, we present a novel framework, that allows these users to interact with a robotic service assistant in a closed-loop fashion, using only thoughts. The brain-computer interface (BCI) system is composed of several interacting components, i.e., non-invasive neuronal signal recording and decoding, high-level task planning, motion and manipulation planning as well as environment perception. In various experiments, we demonstrate its applicability and robustness in real world scenarios, considering fetch-and-carry tasks and tasks involving human-robot interaction. As our results demonstrate, our system is capable of adapting to frequent changes in the environment and reliably completing given tasks within a reasonable amount of time. Combined with high-level planning and autonomous robotic systems, interesting new perspectives open up for non-invasive BCI-based human-robot interactions.

Patients with heavily impaired communication capabilities, such as severely paralyzed patients, are forced to constantly rely on the help of human care-takers due to their health condition. Robotic service assistants can re-establish some autonomy for these patients, if they offer adequate interfaces and possess a sufficient level of intelligence. At this time, however, most of these systems are not very flexible regarding the tasks that can be assigned to them. Instead, they are typically dedicated to a specific work area such as cleaning the floor or mowing the grass. Implementing an intelligent system that is capable to cope with a broad range of tasks, generally requires adaptive task and motion planning modules to determine appropriate task plans and motion trajectories for the robot, as well as robust low-level control strategies to realize the tasks in the real world. Moreover, the deployment of a perception component becomes indispensable, e.g., to detect objects of interest or to avoid accidental collisions with obstacles. Typically used interfaces to command the robotic system, such as haptic (buttons),

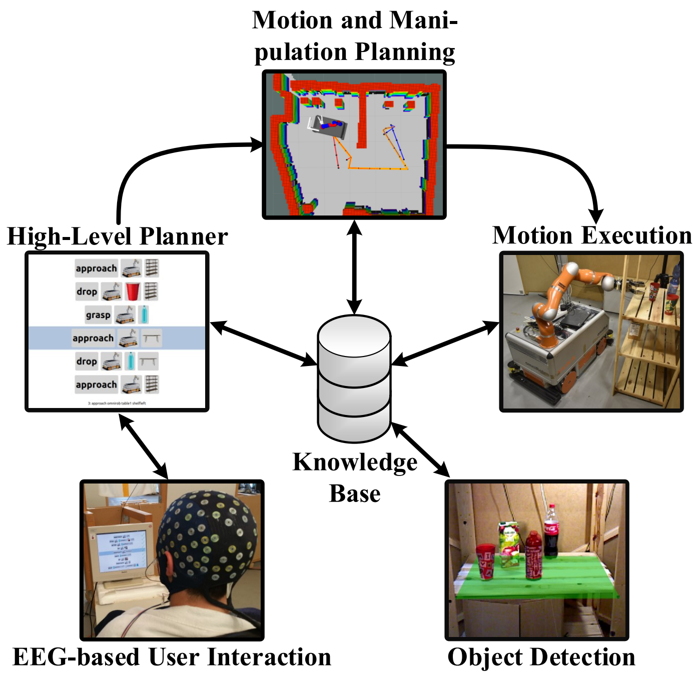


Figure 6.1: Framework unifying decoding of neuronal signals, high-level task planning, low-level motion and manipulation planning and scene perception, with a centralized knowledge base at its core. Intuitive goal selection is provided through a graphical user interface.

audio (speech) or visual (gesture) interfaces, are intuitive and easy options for healthy users, but difficult to impossible to use for paralyzed individuals.

In this chapter, we present a novel framework, schematically depicted in Fig. 6.1, that allows closed-loop interaction between users with minimal communication capabilities and a robotic service assistant. To do so, we record neuronal activity elicited in the human brain, the common origin of all types of communication, with an electroencephalography (EEG) system. Furthermore, we adopt a convolutional neural network (ConvNet) approach for online decoding of neuronal activity, in order to allow users to navigate through a graphical user interface (GUI) provided by a high-level task planner. The set of feasible actions displayed in the GUI, depends in turn on the current state of the world, which is stored in a central knowledge base and continuously updated with information provided by the robot and a camera perception system. Once a task has been selected by the user, it is decomposed into a sequence of atomic actions by the high-level planner. Subsequently, each action is resolved to a motion for the mobile manipulator using low-level motion and manipulation planning techniques, as the one described in Chapter 5. In the following, the individual components shown in Fig. 6.1 will be described in detail. Afterwards, we present a quantitative evaluation of the overall system and its subparts regarding their performance in virtual and real world scenarios.

6.1 Online Decoding of Neuronal Signals

Implementing an EEG-signal based interface to permit severely paralyzed patients to communicate complex tasks to a robotic service assistant represents a major challenge. So far,

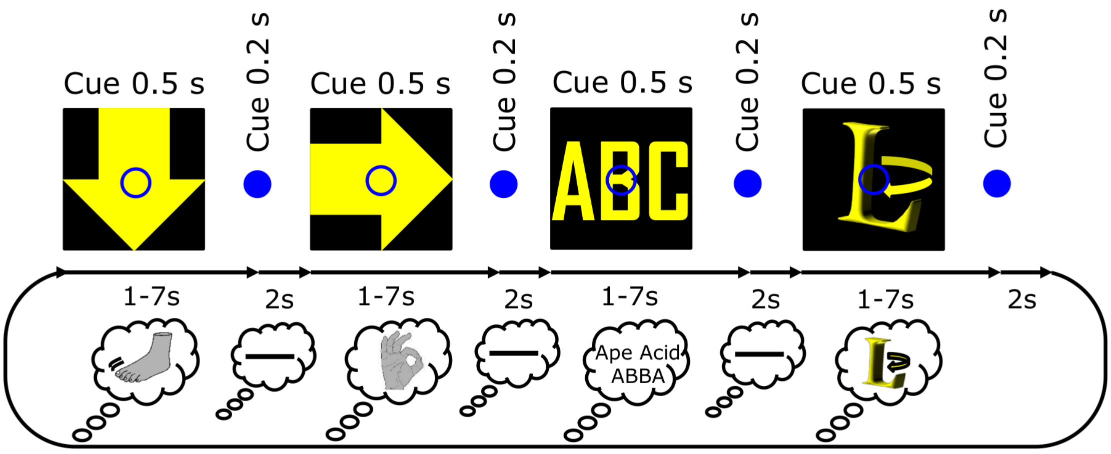


Figure 6.2: Mental task paradigm used to train the ConvNet. Depicted sequence is repeated in random order for 10 min per block. Blue discs are used as cues indicating the end of a mental task. The blue circle represents a fixation point, which needs to be fixated by the user in order to minimize eye movements.

previous work from the literature has considered the deployment of robotic platforms only for assistance in specific tasks, and thus are only of limited use. Contrary, our system involves a robotic service assistant with a rich set of capabilities, including autonomous navigation, manipulation of objects and human-robot interaction. This, in turn, yields a variety of tasks the robot is able to perform and accordingly patients can choose from. To put patients in control of the system, we propose a graphical user interface in this work that displays the currently feasible goals (i.e., tasks) and can be controlled by navigational commands. Finally, we aim to develop a mapping that translates neuronal activity elicited in the human brain to directional commands for the GUI such as *up*, *down*, *confirm selection*, *return*.

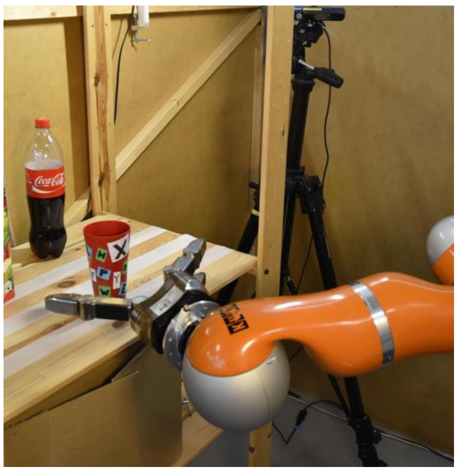
As reliable classification of brain signals into navigation directions can not yet be achieved directly with non-invasive BCIs, we adopt a deep ConvNet approach for decoding of multiple mental tasks from EEG, as proposed by Schirrmeister et al. [114]. This approach introduces a hybrid network, combining a deep ConvNet with a shallower ConvNet architecture. The deep part consists of 4 convolution-pooling blocks using exponential linear units (ELU) [31] and max pooling, whereas the shallow part uses a single convolution-pooling block with squaring non-linearities and mean pooling. The deep as well as the shallow part uses a final convolution with ELU to produce output features. Finally, these features are concatenated and fed to a final classification layer. In our approach, we trained the ConvNet to decode five mental tasks: *right hand finger* and *feet toe* movements, *object rotation*, *word generation* and *rest*. These mental tasks evoke discernible brain patterns and are mapped to directional commands to control the graphical user interface. For *offline* training, we adopted a cropped training strategy using shifted time windows within the trials as input data [114].

Empirical results suggested that it is important to train the BCI decoder and subjects in an environment that is as similar as possible to the environment in which the system is intended to be finally deployed. Otherwise, pronounced performance losses are to be expected in the online decoding phase. Here, reasons for a degraded performance arise from different noise sources. Different ambient light conditions, nearby electrical devices, vibrations as well as background movements, to name just a few, are likely to alter the brain response patterns recorded by the EEG electrodes and therefore deteriorate the ConvNet decoding results. To account for these aspects, we designed a gradual training paradigm within the high-level planner GUI, where the environment, timing and actions are identical to those of the control task subsequently used in our real world applications. The training paradigm is schematically depicted in Figure 6.2 and proceeds as described in the following.

First, each subject is trained *offline* using simulated feedback, i.e., a GUI action is triggered every time a mental task has ended. Here, subjects are aware of not being in control of the graphical user interface. The mental tasks are cued using grayscale images presented for 0.5 s in the center of the display, e.g., a down pointing arrow. Note that the cues illustrated in Figure 6.2 have been colorized in favor of a better visibility. Throughout the training phase, a fixation circle is displayed at the center of the GUI and the subject is instructed to fixate on it to minimize eye movements. Eye movements are considered critical in the context of decoding from EEG data as their associated brain response superposes the mental task related signals in the higher frequency band. After a random time interval of 1-7 s the fixation circle is switched to a disk for 0.2 s, which indicates the end of the mental task. At the same time the GUI action (*go up, go down, select, go back, do nothing*) corresponding to the cued mental task is performed to update the graphical user interface. To keep training realistic, we include a 20 % error rate, i.e., on average every fifth executed action is different from the action actually corresponding to the cued mental task. Additionally, subjects are instructed to count the number of error occurrences to assert their vigilance. Afterwards, this offline data is used to train the individual deep ConvNets. In a next step, we switch to *online* training by letting the subjects perform the decoded mental tasks in the GUI. In a last step, the cueing of mental tasks is ceased completely. In order to evaluate the performance and thus the usability of the BCI control achieved after training, subjects are asked to select instructed tasks using the GUI. These tasks are subsequently decomposed by the high-level planner into a sequence of atomic action and executed by a simulated robot or the real mobile manipulator using the motion and manipulation techniques presented in Section 6.4. Furthermore, subjects have to confirm the execution of every planned action and are also permitted to interrupt the execution of the action sequence at any time. This way, subjects remain in control of the mobile manipulator beyond the task selection process and experience a stronger feeling of agency. Finally, a quantitative assessment of the BCI decoding accuracy for the label-less instructed tasks is obtained by manually rating each decoding based on the prescribed task steps.

6.2 High-Level Goal Formulation Planning

In our work, we use domain-independent planning for the high-level control of our robotic system. Whereas automated planning seeks to find a sequence of actions to successfully complete the task, the goal of the robotic system is determined by the intention of the user. For directly specifying these goals using traditional planning methods, however, users would need to know the internal representation of objects stored in the knowledge base. For example, an expression *ID3185* is used by the underlying planning system to represent the *red cup on the shelf*. Instead, our approach allows users to access objects more easily by referring to them by their *type* and *attributes*, which represent relations to other objects. Doing so, users are given the opportunity to express their desire in a way, similar to when they are



```
(:objects cup01 cup02 - cup
          shelf01 shelf02 - shelf
          omnirob - robot)
(:init (arm-empty omnirob)
       (at omnirob shelf02)
       (position cup01 shelf02)
       (contains cup01 water))
```

Figure 6.3: *Left*: The red cup in the real world, referred to by *cup01*. *Right*: PDDL problem description with object instances as well as their initial state.

communicating with other humans. A detailed description of our automatic goal formulation assistant, incrementally generating references to feasible goals in a menu-driven graphical user interface, will be given in the following.

6.2.1 Domain-Independent Planning

Automated planning is used to transfer a system into a desired goal state by sequentially selecting high-level actions. A planning task consists of a planning domain and a problem description, formulated in the *Planning Domain Definition Language* (PDDL) [89]. The former represents a description of the object types and predicates as well as the preconditions and effects of actions available to manipulate them. The latter models the present objects, their initial state and the desired goal. For example, our domain contains a type hierarchy, where *furniture* and *robot* are of super-type *base*, and *bottle* and *cup* are of super-type *vessel*. Moreover, it defines relations between objects, e.g., *position* is a relation between objects of type *vessel* and type *base*. Finally, it defines the set of actions supported by the robotic platform, such as *grasp*, *drop*, *approach* and *pour*. The problem description, on the other hand, specifies object instances, such as *cup01*, *cup02* of type *cup* and *shelf01* of type *shelf* as well as relations between them, e.g., the *position* of *cup01* is *shelf02*, as illustrated in Figure 6.3.

6.2.2 Goal Formulation with References

The biggest hurdle in making high-level planning systems accessible for non-expert users is to translate and communicate feasible goals in a human understandable manner. This problem is particularly challenging due to limited shared vocabulary between the user and the planning system, whose world is described by a PDDL planning task [89]. For example, the most concise representation of the cup in Figure 6.3 might be *cup01* for the planner, whereas this description is not sufficiently clear to the user if multiple cups exist in the environment. To solve this problem, we provide the goal generation and selection component with a set of *shared references* between planner and user, which can be combined to create *referring* *expressions* to objects or sets of objects in the world [35]. This section briefly recapitulates the relevant concepts in this area, originally introduced by Göbelbecker [50].

In general, a referring expression ϕ is a logical formula with a single free variable. We say that ϕ *refers* to an object o if $\phi(o)$ is valid. For example, the reference $\phi(x) \equiv cup(x) \wedge contains(x, water)$ refers to $cup01$. For our purpose, we restrict ourselves to references that are simple conjunctions of facts, such as the example above. This is not only preferable for computational reasons, but also allows us to incrementally refine references by appending further constraints, such as adding $contains(x, water)$ to $cup(x)$, to restrict the set of all cups to the set of cups containing water. A reference is ***unique*** iff there is exactly one object it refers to. However, it is usually sufficient to create references to sets of objects. If the user, e.g., wants a glass of water it might not be important to refer to a specific glass as long as it contains water.

In the context of high-level planning, it is important to note that the formal concept of an *object* differs from the definition of the notion in the physical world. For example, a $red\ cup$ is an object of type cup, possessing an attribute $color$, which is red. Here, the color red (of type $color$) is considered an object itself by the planner. Instead of attributes, objects are connected by *relations*. For instance $colored(cup01, red)$, describes a binary relation between a cup and a color. To reference objects in planning domains, we need to specify the components that can be used to create references. Here, we distinguish between three fundamental types of references:

- **Individual references**: These objects can be identified by name. Examples include objects such as $water$, $apple\text{-}juice$ (of type $content$) or $omniRob$ (of type $robot$).
- **Typename references**: These objects can be identified by the name of their type. While we cannot refer to the cups and shelves directly in our example, we can refer to an unspecific $shelf$ or cup.
- **Relational references**: Objects can be referred by predicates in which they occur as an argument. In the example depicted in Figure 6.3, we use the $content$ of the cup to clarify which cup is meant.

In practice, these object references can be applied to specific planning problems to generate references to the available goals. For the planning system, it is the most natural way to represent a goal by a conjunction of predicates, e.g., $cup(x) \wedge shelf(y) \wedge position(x, y)$, which translates to *cup is on the shelf* in natural language. For humans, however, this is a rather unnatural way to communicate the goal of putting a cup on a shelf. In our approach, a more comprehensible and natural way to communicate goals to humans has been adopted that makes use of the action achieving the goal instead of the goal itself. Accordingly, the aforementioned goal is represented as *action*$(put, x, y) \wedge cup(x) \wedge shelf(y)$, or *put the cup on the shelf* in natural language.

6.2.3 Adaptive Graphical Planner Interface

Whereas shared references constitute a vocabulary understandable for both, the planning system and human users, it remains to define an appropriate interface permitting them to mutually exchange information. Since graphical user interfaces (GUI) are already widespread

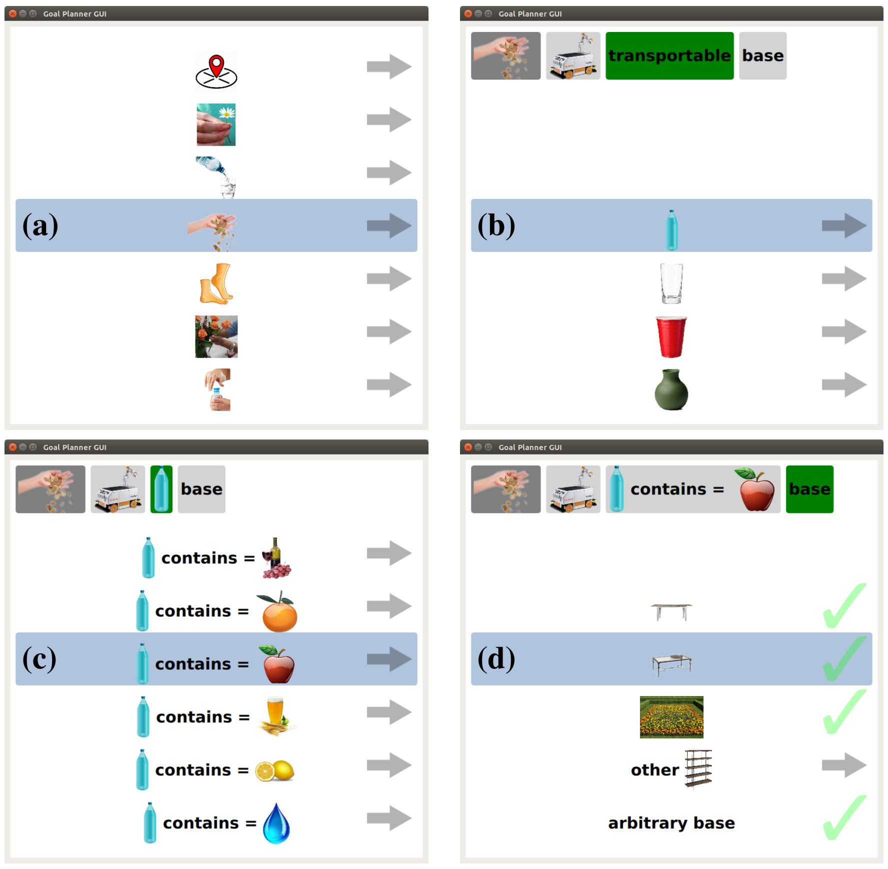


Figure 6.4: Graphical user interface of the high-level planner. **(a)** Selection of a high-level planner action. **(b)** Refinement of the action parameter of type $transportable$. **(c)** Refinement of the argument based on $content$. **(d)** Refinement of the last action parameter of type $base$.

among assistant systems dedicated to disabled people, we aim to provide a similar interface enabling users to set the goal for the robot, or more specifically communicate their desire. More precise, we use the object references to build a dynamic, menu-driven goal selection interface, as depicted in Figure 6.4.

The high-level planner GUI initially displays the set of available actions, e.g., $pick$, $pour$ or $drop$. After selecting one of the goal types, e.g., $drop$ (with three parameters: the robot, the transportable to be dropped and the base to place the object on) in Figure 6.4 (a), objects for all parameters of the goal predicate or action are determined. In the following, we start populating the goal with the most specific reference that still matches all possible arguments, e.g., $omniRob$, $transportable(x)$ and $base(y)$, assuming that $omniRob$ is an individual reference and $transportable$ and $base$ are typename references (Figure 6.4 (b)). The current goal reference is displayed in the top row of the GUI (Figure 6.4 (b-d)). Afterwards, the user is provided with choices to further refine the argument by constraining the previous choice. In our example the first argument $omniRob$ is the only object in the world that fits the parameter type $robot$ so it does not have to be refined any further. Therefore, we start by offering choices for refining the second argument $transportable(x)$, which yields the selections $bottle(x)$, $glass(x)$, $cup(x)$ and $vase(x)$ (see Figure 6.4 (b)). This continues until the argument is either determined uniquely, it is impossible to constrain the argument further or the user declares that any remaining option is acceptable. Considering our example, the refinement of the $bottle$ argument is completed by selecting $apple\text{-}juice$ as the $content$ (see Figure 6.4 (c)) (or in a more relation-centric view, we offer all feasible choices for adding a relation $contains(x, o)$ to the referring expression, where o is an object of type $content$). This procedure is repeated for all parameters of the goal (e.g., selecting $couch\ table$ for the $base$ parameter in Figure 6.4 (d)), which will finally result in a single goal or set of goals (if the references are not unique) that are sent to the planner. The check marks, depicted adjacent to the selections indicate that the goal is fully specified for these choices. Instead, arrows declare that further refinements are available for the respective entry. Some features cannot be used to partition the remaining objects for one parameter, e.g., not all objects have the attribute $color$, in which case an entry for all $other$ objects can be chosen. Additionally, we allow to skip the refinement of the current parameter and use an $arbitrary$ object for it. Finally, we provide an entry to go $back$ to the previous refinement step. To decide which feature to use to refine the current selection, we draw on decision tree learning that maximizes the resulting partition's information gain [101]. This strategy prefers references that require few selections and split referable objects more equally, thus offering the user a meaningful choice at every step. Note that only choices are offered that can result in an achievable goal, as determined by *delete relaxation* of the planning task, an efficient over-approximation of goal reachability [10]. For example, if all bottles were out of reach for the robot, the choice $bottle(x)$ would be removed from the selection depicted in Figure 6.4 (b). This might result in a completely different selection being preferred in Figure 6.4 (c), e.g., one that distinguishes by the transportable's color or position. In case that several objects can satisfy the specified goal, the planner resolves the ambiguity by picking an arbitrary object among them.

6.3 Dynamic Knowledge Base

The knowledge base represents the backbone of our network of planning, execution and perception nodes, as depicted in Figure 6.1. In our work, it is initialized by the previously mentioned domain and problem description, constituting a planning task. The former defines object predicates and available actions together with their preconditions and effects. The later defines which objects are present in the world and sets their corresponding initial state. Once the knowledge base is initialized, it acts as a central database from which all participating network nodes can retrieve information about specific objects and their associated attributes in the world. Dynamic behavior is achieved by additionally introducing a knowledge base server layer that allows nodes to add, remove or update objects as well as their attributes. Moreover, the server actively spreads information about incoming changes in the world, or updates on object attributes across the network. Based on this information, each network node can decide on its own whether the process it is responsible for is affected by the update and thus if actions are required to be initiated.

6.4 Robot Motion Generation

Implementing the task-related action sequence generated by the high-level planning component of our system in the real world requires a robotic service assistant to autonomously plan and execute navigation and manipulation actions. For the purpose of generating motions for the robotic arm and mobile base, respectively, we adopt two motion planning frameworks in our work. Due to the fact, that the workspace of the robotic manipulator is solely defined by its kinematics, we use a multi-query probabilistic road-map (PRM) based task space planner for manipulation tasks. On the contrary, the workspace of the mobile base primarily depends on the size and contour of the environment, which is likely to change as the robot is deployed in different application scenarios. Therefore, the single-query RRT-based *BI^{2}RRT** algorithm, introduced in Chapter 5, is used to realize navigation tasks. In the following, we will describe each of the planners as well as how they interact with the high-level planning component.

6.4.1 Navigation Tasks

Navigation tasks, referred to as the $move$ and $approach$ action by the high-level planner, require the mobile base of the robot to move along a collision-free path to either reach a desired location, e.g., the kitchen or living-room, or to position itself in front of a furniture or human in preparation of a subsequent manipulation action. As we perform bidirectional planning using the *BI^{2}RRT** planner, the start and goal tree need to be initialized with a corresponding configuration for the mobile base. Here, a configuration of the mobile base simply corresponds to a pose in SE(2), i.e., a 2D position and 1D orientation, expressed with respect to a global map frame $\mathcal{F}_{map}$. For precomputing a static 2D grid map representation of the environment, existing *robot mapping* techniques have been adopted [52]. For planning, however, we rely on a virtual representation of the environment, depicted in Figure 6.5, which is obtained by vertical extruding the grid map. The result is a 3D *octomap* [59], that permits our navigation planning component not only to account for collisions between the mobile base and the environment, but also for undesirable contact between the robotic arm and obstacles. In order to acquire the root pose for the start tree, we rely in this work on pose estimates $\hat{\mathbf{p}}^{map}_{start} = (x, y, \theta)^\top$, provided by a particle filter-based localization algorithm [46]. The goal pose, on the other hand, is retrieved from the central knowledge base of our framework using the action parameters provided by the high-level planner. Let us assume, for example, the goal is to navigate to a specific shelf, i.e., $shelf1$ in the notion of the high-level planning system. In this case, the knowledge base is queried to return the $pose$ attribute of $shelf1$, corresponding to a location in front of the shelf offering a high manipulability measure (see Section 4.2.2) for the robotic arm. Subsequently, the response is assigned to $\mathbf{p}^{map}_{goal}$, i.e., the root pose of the goal search tree, depicted in Figure 6.5 (top). Note that the prescribed poses stored in the knowledge base have been selected empirically for the target locations considered in our work, whereas an automatic and optimal selection of target poses could be obtained using the IRM approach presented in Chapter 4.

Finally, given a set of terminal configurations $\hat{\mathbf{p}}^{map}_{start}$ and $\mathbf{p}^{map}_{goal}$, our planner performs a bidirectional search using uniform sampling in the configuration space until an initial, though sub-optimal, solution path is found. Afterwards, the remaining planning time is used for path refinement adopting an informed sampling strategy providing a higher convergence rate towards the optimal solution. An example showing a pair of terminal poses (red and green sphere) and the resulting structure of the start and goal tree (blue and red edges), obtained after termination of the *BI^{2}RRT** algorithm, is shown in Figure 6.5 (top).

For executing the planned paths, as exemplary illustrated by the orange line in Figure 6.5 (bottom), we employ a closed-loop trajectory tracking algorithm using robot localization

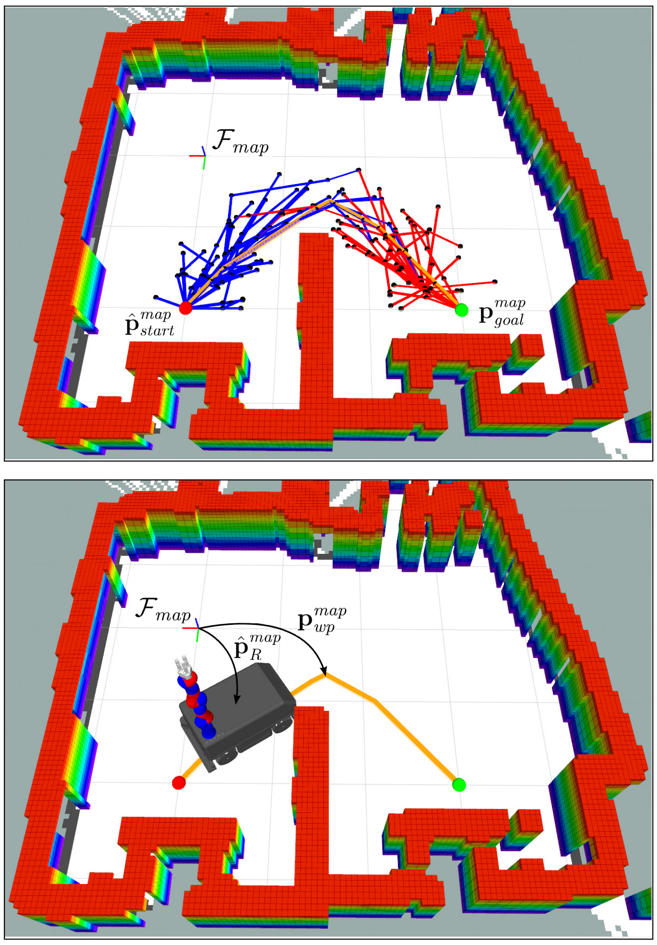

$\mathcal{F}_{map}$
$\hat{\mathbf{p}}_{start}^{map}$
$\mathbf{p}_{goal}^{map}$
$\mathcal{F}_{map}$
$\mathbf{p}_{wp}^{map}$
$\hat{\mathbf{p}}_{R}^{map}$

Figure 6.5: *Top:* Bidirectional motion planning to determine a collision-free path, guiding the mobile base from its current pose (red sphere) to a desired target pose (green sphere). Collision checks are performed in a 3D representation of the environment, obtained by vertically extruding the static 2D grid map. *Bottom:* Closed-loop trajectory tracking using robot localization feedback.

feedback. Given the robot's current pose estimate $\hat{\mathbf{p}}_R^{map}$ in the map frame $\mathcal{F}_{map}$, with $\hat{\mathbf{p}}_R^{map}(0) = \hat{\mathbf{p}}_{start}^{map}$, the algorithm steps forwards in time through the waypoints of the trajectory, while determining the distance between the current robot pose and waypoint, i.e., $\hat{\mathbf{p}}_{wp}^R = \hat{\mathbf{p}}_{map}^R \oplus \mathbf{p}_{wp}^{map}$, expressed with respect to the coordinate system of the mobile base. Once a translational or rotational error component of $\hat{\mathbf{p}}_{wp}^R$, i.e., Δx, Δy, $\Delta\theta$, is found to exceed a certain upper threshold $\epsilon_{lin,max}$, $\epsilon_{rot,max}$, the corresponding waypoint is selected as the new target and velocities $\dot{\mathbf{p}}_{base} = (\dot{x}, \dot{y}, \dot{\theta})^\top$, proportional to the observed error, are send to the mobile base (see Appendix A.2). To compensate for inaccurate motion execution and therefore to reliably reach the waypoint, the vector of base velocities is continuously updated based on pose estimates provided by the localization system. Finally, if all error components of $\hat{\mathbf{p}}_{wp}^R$ fall below a lower threshold $\epsilon_{lin,min}$, $\epsilon_{rot,min}$, the current waypoint is considered as reached and the algorithm proceeds by determining the next waypoint for the mobile base, as described above. Whereas it is admissible to traverse intermediate trajectory waypoints with small deviations, the final waypoint or destination of the mobile base is required to be reached with high accuracy as it affects the outcome of manipulation actions, potentially considered as the subsequent task by the high-level planning system. To account for this aspect, we set the proximity thresholds $\epsilon_{lin,min}$ and $\epsilon_{rot,min}$, indicating the arrival at waypoints, to a sufficiently small value when $\mathbf{p}_{wp}^{map}$ is found to coincide with $\mathbf{p}_{goal}^{map}$.

6.4.2 Manipulation Tasks

Manipulation actions, such as $grasp$, $drop$ or $pour$, require the robotic arm to approach or transfer an object via a collision-free path. As the reachable workspace of the manipulator depends solely on its kinematics, we follow a probabilistic road-maps (PRM) based planning approach [71]. Traditionally, PRMs are adopted for planning planar motions in 2D, i.e., in the (x, y) state space. Instead, in our work we build a 3D Cartesian space road-map of end-effector poses explicitly considering task constraints, e.g., upright or horizontal orientation, in the construction process. The resulting algorithm is referred to as the *probabilistic task-space road-map* planner (PTRM) and will be described in the following.

The PTRMs need to be build only once in an offline step and can be subsequently used for all manipulation task related motion planning queries. To generate the road-maps, end-effector positions with specific orientations are randomly sampled from the spatial hull defined by the manipulator's extent and evaluated for reachability adopting an inverse kinematics solver. If a pose if found to be reachable, it is added as a new node to the road-map or rejected otherwise. This process is continued until the road-map contains a predefined number of nodes. Afterwards, edges between graph nodes are inserted considering neighboring nodes within a certain euclidean distance. The result of a PTRM construction process considering end-effector poses with upright orientation as, for example, required to manipulate a glass of water, is depicted in Figure 6.6 (left). Note that the density of the illustrated road-map (500 samples) has been selected intentionally sparse for the sake of a better visibility of graph nodes and edges. In practice, a much denser map (ca. 5000 samples) is used to achieve reliable planning results.

Solving a specific motion planning query requires to connect the start and desired goal pose to the nearest node contained in the road-map graph, respectively. The start pose always corresponds to the current pose of the end-effector, whereas the goal pose depends on the task. To elevate the prospects of success, we follow in our work the approach of selecting multiple goal poses for planning, which are derived from visual information provided by a depth camera system mounted on the robot (see Figure 6.7). The obtained data includes a pose estimate for the object to be manipulated, expressed with respect to the base link frame of the manipulator. Moreover, an *octomap* representation [59] of the scene is generated from the camera's point cloud data [108] which, in addition to evaluating the validity and quality of goal poses, serves the purpose of performing collision checks during planning. Regarding the $grasp$ action, for example, a set of potential goals is generated by sampling collision-free end-effector poses around the given object pose. For $drop$ actions, on the other hand, feasible goal poses are obtained by determining collision-free object placements on a planar surface using the 3D octomap.

Finally, A* search [56] is iteratively performed on the PTRM to find optimal solution paths to the goal poses, starting with the one being closest to the current end-effector pose. In practice, it has been found useful to initially attempt to connect the start and goal end-effector position by a straight line, thus avoiding unnecessary planning efforts. Once a valid path is

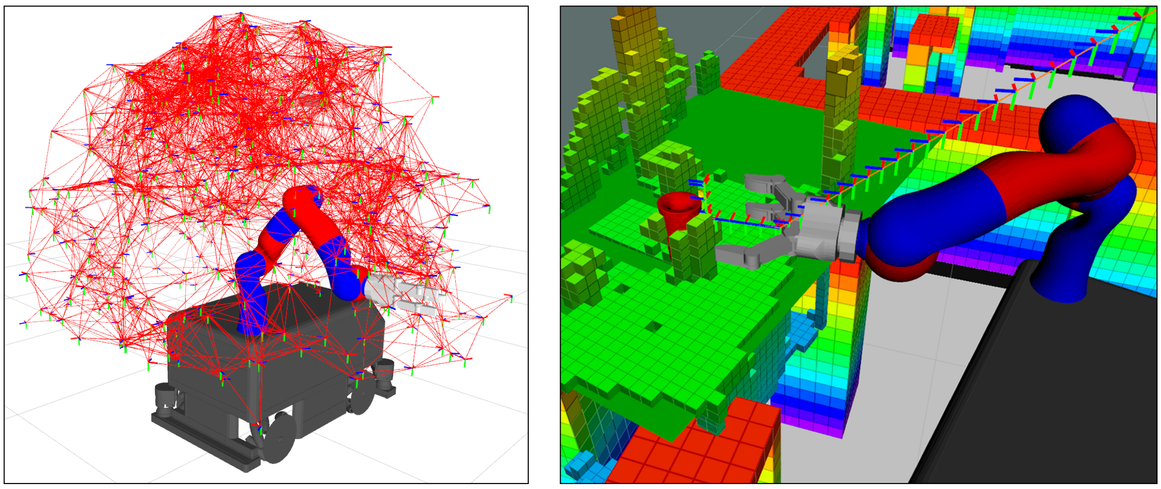


Figure 6.6: *Left:* Probabilistic task-space road-map of upright end-effector poses. *Right:* End-effector path generated by the PTRM planner for the task of grasping a cup (shown as a sequence of frames) and snapshot of the trajectory execution. (Courtesy of Daniel Kuhner)

found, the search for paths towards alternative goal poses is aborted and the corresponding solution is returned. A solution path for grasping a cup is illustrated in Figure 6.6 (right). For planning, the euclidean distance is considered to evaluate the cost of graph node transitions as well as to estimate the expected cost to reach the goal from a specific node, i.e., the heuristic. To ensure that graph transitions comply with the task constraints and are collision-free, a Jacobian-based numerical inverse kinematics solver is used to iteratively generate intermediate robot configurations along the graph edges which are evaluated for validity. For execution of the solution path, we finally employ a task-space motion controller, which maps the desired task-space velocity commands to velocities in joint space.

6.5 Implementation Details

Implementation of our framework in the real world requires several components, such as neuronal signal decoding, scene perception, knowledge base operations as well as symbolic and motion planning, to run in parallel. Therefore, we distributed the computation across a network of 7 computers, communicating among each other via ROS. The decoding of neuronal signals has four components. EEG measurements are performed using a *Waveguard EEG* cap with 64 electrodes and a *NeurOne* amplifier in AC mode. Additionally, vertical and horizontal EOGs (Electrooculography, i.e., measures electrical activity produced by eye movements), EMGs (Electromyography, i.e., measures electrical activity produced by skeletal muscles) of the four extremities and ECGs (Electrocardiography, i.e., measures electrical activity produced by the heart) are recorded. For recording and online preprocessing, we used BCI2000 [112] and Matlab. We then transferred the data to a GPU server, where our deep ConvNet classified the data into five classes.

The high-level planner GUI consists of a back- and front-end. The back-end of the GUI uses the *Fast Downward* planner [58] to iteratively build goal references and to find symbolic plans for the selected goal. As the planning time is not crucial for the performance of our system, we used the A* implementation of *Fast Downward* with a default configuration in our experiments. The front-end of the GUI is implemented in PyQt/QML. Planning task descriptions are provided by *domain* and *problem* PDDL files. The central knowledge base is implemented as a ROS node, which is able to store objects with arbitrary attribute information. All changes in the knowledge base automatically trigger updates of the front-end, unexpected ones interrupt the current motion trajectory execution. For object pose detection and tracking, we used *SimTrack* [100]. Extrinsic calibration of the depth camera system is realized following the approach by Ilonen and Kyrki [60]. For planning collision-free trajectories for the mobile base, we rely on a static map representation of the environment, created using the *Gmapping* algorithm introduced by Grisetti et al. [52]. Finally, we employ the *adaptive Monte Carlo localization* algorithm [46] to continuously acquire estimates of the robot's current pose with respect to a fixed global reference coordinate system.

6.6 Experiments

To evaluate our framework in real world experiments, we consider the environment schematically depicted in Fig. 6.7, containing two shelves and a table as potential locations for manipulation actions. The user sits in a wheelchair in front of a screen, displaying the graphical interface of the high-level planner. The robot used in the experiments is the *omniRob* omnidirectional mobile manipulator platform by KUKA Robotics [79], which is composed of 10 degrees of freedom (DOF), i.e., 3 DOF for the mobile base and 7 DOF for the manipulator. Additionally, the *Dexterous Hand 2.0* by Schunk [117] is attached to the manipulator's flange and used to perform grasping and manipulation actions. A detailed description of the robotic platform can be found in Appendix A.2. The tasks, we considered in our real world experiments required the robotic system to autonomously perform the following actions: drive from one location to another, pick up an object, drop an object (on a shelf or table), pour liquid from a bottle into a cup, supply a user with a drink. Moreover, we use a perception system composed of five RGBD cameras. Three of them are statically mounted at the shelves and the table, in order to observe the scene and to report captured information to the knowledge base. The other two cameras are carried by the robot on-board. The first one is located at the mobile base and used to perform collision checks in manipulation planning. The second camera is mounted at the robot's end-effector and used for tasks involving physical human-robot interaction. Additionally, we evaluated the performance and usability of our goal formulation assistant on virtual environments containing a varying number of objects randomly spawn in different locations. A video demonstrating the capabilities of our framework is available online[1].

6.6.1 Online Decoding of Neuronal Signals

The BCI control setup has been evaluated by four healthy subjects, referred to as S1-S4 in our quantitative results. All of them are right-handed, three are female and the average age is 26.75±5.9 years. The recordings for subject S4 were still in progress during the analysis of the decoding performance achieved by our approach. Therefore, only a few runs are listed

[1]http://www.informatik.uni-freiburg.de/~burgetf/ecmr17

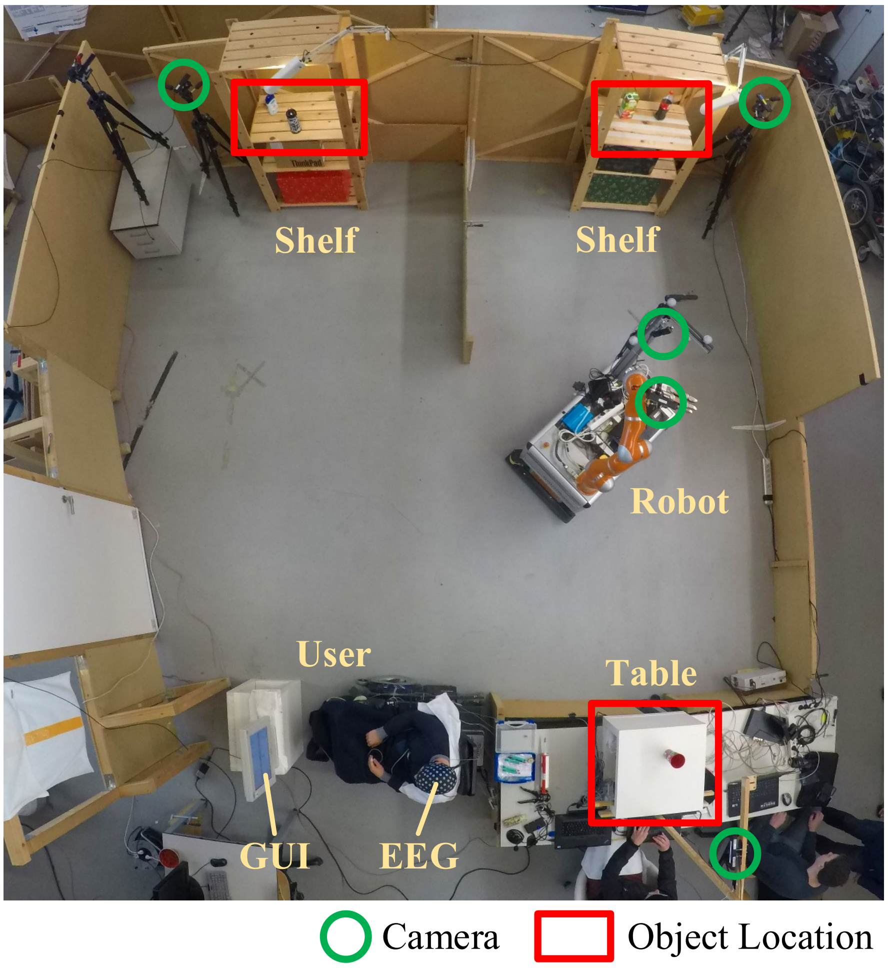


Figure 6.7: Experimental environment: Two shelves and a table can be considered by the robot for performing manipulation actions. A perception system composed of five RGBD cameras is used to observe the environment. A human operator selects a goal from the high-level planner GUI based on BCI control.

for that subject in Table 6.1. In total, 52 runs have been recorded, thereof 20 with the real robot, where the subjects executed various instructed high-level plans. For 32 runs, we used simulated feedback from the GUI, i.e., actions were executed by a virtual robot and assumed to be always completed successfully. By doing so, a significant amount of data is generated and thus meaningful results are obtained. For assessing the BCI decoding performance during these runs, video recordings of interactions within the GUI have been used. Individual GUI actions were rated correct if they corresponded to the instructed path and incorrect otherwise. Actions which are necessary to remediate a previous error are interpreted as correct if the correction is obviously needed in order to proceed. Finally, we rated *rest* actions as correct during the simulated robot executions (no user command required), incorrect if the next robot action had to be initialized (user command *select* required) and ignored them during the phase of navigating through the high-level planner GUI (doing nothing is acceptable). For evaluation, five metrics have been extracted from the video recordings: (i) the accuracy of the control, (ii) the time it took the subjects to execute a high-level plan, (iii) the number of steps taken to execute a high-level plan, (iv) the path optimality, i.e., the ratio between the steps performed

	Runs #	Accuracy* [%]	Time [s]	Steps #	Path Optimality [%]	Time/Step [s]
S1	18	84.1±6.1	125±84	13.0±7.8	70.1±22.3	9±2
S2	14	76.8±14.1	150±32	10.1±2.8	91.3±12.0	9±3
S3	17	82.0±7.4	200±159	17.6±11.4	65.7±28.9	11±4
S4	3	63.8±15.6	176±102	26.3±11.2	34.5±1.2	6±2
	52	76.7±9.1	148±50	16.7±7.1	65.4±23.4	9±2

Table 6.1: Aggregated mean±std results for 52 BCI control runs, * p-value $< 10^{-6}$.

and the minimal possible number of steps required to reach the goal, and (v) the average time per step. The results are summarized in Table 6.1. On average, 76.67 % of the BCI control commands were correct. In order to perform a single step in the GUI it took subjects 9 s on average. For selecting a complete plan users required 148±50 s, performing on average 16.74 steps in the GUI of the high-level planner. The path formed by these steps is on average 34.6 % away from the optimal path, i.e., the minimal sequence of steps required to reach the instructed goal. The decoding accuracy of every subject is significantly above chance (*t-test* result: p $< 10^{-6}$).

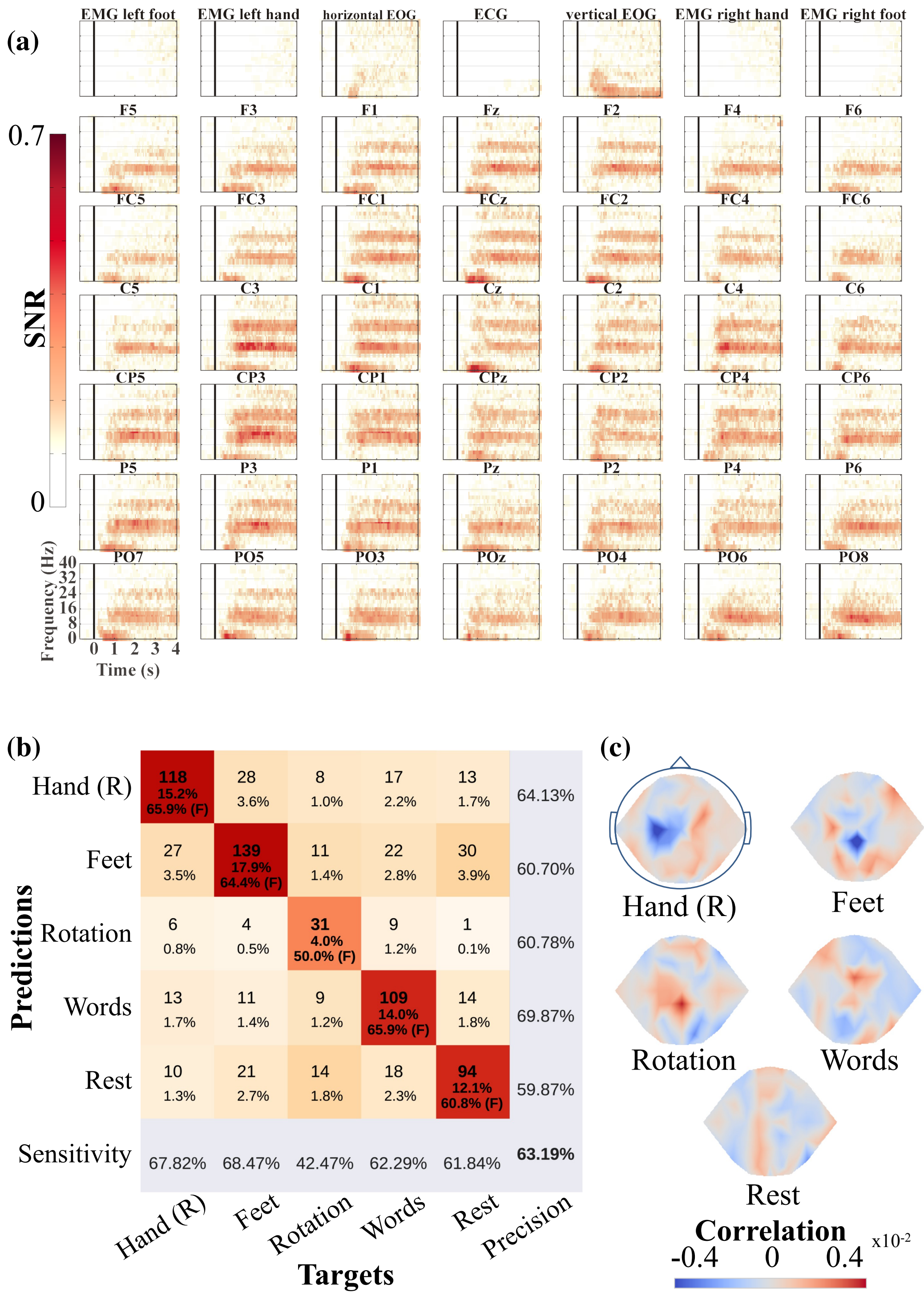
The subject-averaged EEG data used to train the hybrid ConvNets and the decoding results of the train/test transfer are visualized in Fig. 6.8. In Fig. 6.8 (a), the signal-to-noise ratio (SNR) of all 5 classes $\mathcal{C}$ of the labeled datasets, are depicted . The SNR for a given frequency f, time t and electrode e is defined as follows

$$\text{SNR}_{f,t,e} = \frac{\text{IQR}\left(\{\text{median}\left(\mathcal{M}_i\right)\}\right)}{\text{median}\left(\{\text{IQR}\left(\mathcal{M}_i\right)\}\right)} \quad i \in \mathcal{C}, \tag{6.1}$$

where $\mathcal{M}_i$ corresponds to the set of values at position (f, t, e) of the i-th task, with $|\mathcal{M}_i|$ being the number of repetitions. median$(\cdot)$ and IQR$(\cdot)$ is the median and interquartile range (IQR), respectively. The upper part describes the variance of the class medians, i.e., a large variance means more distinguishable class clusters and a higher SNR. The denominator describes the variance of values in each class, i.e., a lower variance of values results in a higher SNR. The low SNR in EMG (Electromyography) and EOG ((Electrooculography) channels shows that the subjects did not move during the tasks, thus proving that navigation actions in the GUI are not triggered by electrical activity produced by skeletal muscles or eye movements. The decoding accuracies achieved on the test data after initial training of the ConvNets are visualized in Fig. 6.8 (b). To further support the neural origin of the BCI control signals, Fig. 6.8 (c) shows physiologically plausible input-perturbation network-prediction correlation results (see Schirrmeister et al. [114] for methods).

6.6.2 Fetch and Carry Task

The first experiment, considering the use of the real robot, evaluates the complete system in fetch-and-carry tasks. The goal was to transfer an object from one location to another, e.g., from a shelf to the table, using the robot (see Figure 6.7). To fulfill such tasks the robot



Predictions \ Targets	Hand (R)	Feet	Rotation	Words	Rest	Precision
Hand (R)	**118** **15.2%** **65.9% (F)**	28 3.6%	8 1.0%	17 2.2%	13 1.7%	64.13%
Feet	27 3.5%	**139** **17.9%** **64.4% (F)**	11 1.4%	22 2.8%	30 3.9%	60.70%
Rotation	6 0.8%	4 0.5%	**31** **4.0%** **50.0% (F)**	9 1.2%	1 0.1%	60.78%
Words	13 1.7%	11 1.4%	9 1.2%	**109** **14.0%** **65.9% (F)**	14 1.8%	69.87%
Rest	10 1.3%	21 2.7%	14 1.8%	18 2.3%	**94** **12.1%** **60.8% (F)**	59.87%
Sensitivity	67.82%	68.47%	42.47%	62.29%	61.84%	**63.19%**

Figure 6.8: EEG data and decoding results. **(a)** SNR of the first 4 s of data used to train the hybrid ConvNet. Highest SNR can be observed in the alpha (7-14 Hz) and lower beta (16-26 Hz) bands. These frequency bands are robust markers of task related mental tasks. Note that the non-EEG channels (top row) were withheld from the ConvNets at any time and are displayed as negative control. Not all channels are displayed because of space constraints. **(b)** Confusion matrix of decoding accuracies for the train/test transfer. Accuracies are well above the theoretical chance level of 20 %. **(c)** Topographically plausible input-perturbation network-prediction correlation maps in the alpha (7-13 Hz) frequency band. For details on the visualization technique, we refer the reader to the work of Schirrmeister et al. [114].

Actions	# Executions (# Scheduled)	Success Executions [%]	Runtime [s] Mean	Std
Grasp	10 (10)	90.0	37.56	4.62
Drop	9 (10)	89.0	34.13	5.75
Approach	19 (20)	100.00	33.05	18.48
Total	38 (40)	94.74	34.42	14.02

Table 6.2: Aggregated results for 10 runs of the fetch-and-carry task.

typically needs to execute four subtasks: *approach* object location, *grasp* object, *approach* other location, *drop* object. The user was instructed to select a predefined goal using the EEG-controlled high-level planner. Moreover, we selected a random initial placement for the objects in each run, in order to cover different environment states. The experiment was repeated ten times by the user. Table 6.2 shows the averaged results for the experiment.

The second column indicates the overall number of desired action calls, as scheduled by the high-level planner, as well as the number of calls actually performed. The third to fifth column represents the success rate, mean and standard deviation for the runtime of actions, respectively. Here, the success rates are given with reference to the number of actually executed planning attempts and not the scheduled ones. Note, that the number of scheduled and actually executed actions might differ for two reasons. A number of executed calls, lower than the scheduled ones, indicates that a previous action step has failed to succeed and plan recovery was not possible. On the other hand, a higher number of executed calls indicates that the user was able to achieve plan recovery by commanding a repetition of the failed action. Moreover, we recorded the largest standard deviation for the *approach* action, which can be attributed to the diverse complexity of the planning problem for the mobile base and the distance to travel between the selected grasp and drop location. In total, our system achieved a success rate of 80% for the entire task. Planning and execution required on average 140.63 ± 36.7 s. Failures were mainly caused by object detection issues, i.e., the perception component of our framework was not able to detect the object or the pose estimate was not sufficiently accurate to complete the corresponding task successfully. The impact of this issue is reflected by a failed drop and grasp action, recorded in Table 6.2 (third column, first two rows).

6.6.3 Drinking Task

Another real world experiment evaluates the direct interaction between a user and the robot. More specifically, we implemented an autonomous robotic drinking assistant. Our approach enables the robot to fill a cup with a liquid, move the robot to the user and finally provide the drink to the user by execution of the corresponding drinking motion in front of the user's mouth. In addition to the techniques adopted in the previous experiment, successful pouring and drinking using a robot requires the detection of the liquid level in the cup and a reliable detection and localization of the user's mouth.

To detect the liquid level while pouring, we follow an approach based on work by Do et al. [41]. The RGBD camera used for this purpose is located on the bottom right in

Actions	**# Executions (# Scheduled)**	**Success Executions [%]**	**Runtime [s] Mean**	**Runtime [s] Std**
Grasp	34 (30)	91.0	40.42	10.31
Drop	30 (30)	97.0	37.59	4.83
Approach	80 (80)	100.0	20.91	7.68
Pour	10 (10)	100.0	62.90	7.19
Drink	13 (10)	77.0	57.10	8.20
Total	167 (160)	95.86	32.46	15.51

Table 6.3: Aggregated results for 10 runs of the drinking task.

Figure 6.7 and provides a top-down view onto the table, where we intend to perform pouring actions. RGBD cameras, such as the *Asus Xtion Pro Live* used in our experimental setup, emit a speckle pattern into the environment and determine the depth from the reflected pattern. For transparent liquids such as water, the light is refracted, which results in an incorrect depth value for the liquid height. When the camera's viewing angle and the liquid's index of refraction are known, the actual liquid height can be determined from the depth measurement using a relationship based on Snell's law [55]. Using this knowledge, we first detect the cup in the depth data and extract depth values for the liquid. This value is converted to an estimated real liquid height. The type of liquid is selected by the user and hence known beforehand. Therefore, the index of refraction is known and the viewing angle can be determined from the depth data. A Kalman filter is then used to track the liquid level and compensate for noise. Once it is detected that the liquid level has exceeded a user defined value, a stop signal is sent to terminate the pouring motion.

For detection and localizing of the user's mouth, we adopt a two step approach. In the first step, we segment the image based on the output of a face detection algorithm in order to extract the image region containing the user's mouth and eyes. Afterwards, we detect the position of the mouth of the user, considering only the obtained image patch. Regarding the mouth orientation, we additionally consider the position of the eyes in order to obtain a more robust estimate of the face orientation, hence compensating for slightly changing angles of the head. The face, mouth and eye detectors are implemented in OpenCV [11] by applying an algorithm that uses *Haar cascades* [84, 139].

Table 6.3 shows the averaged results for the experiment. Here, only 3.75% of the 160 scheduled actions had to be repeated in order to complete the task successfully. In one run, plan recovery was not possible leading to abortion of the task. Thus, our system achieved in total a success rate of 90% for the drinking task. Planning and execution required on average 545.56±67.38 s. For the evaluation of the liquid level detection approach, we specified a desired fill level and executed 10 runs of the pour action. The resulting mean error and standard deviation is 6.9±8.9 mm. In some instances the bottle obstructed the camera view, resulting in poor liquid level detection and a higher error.

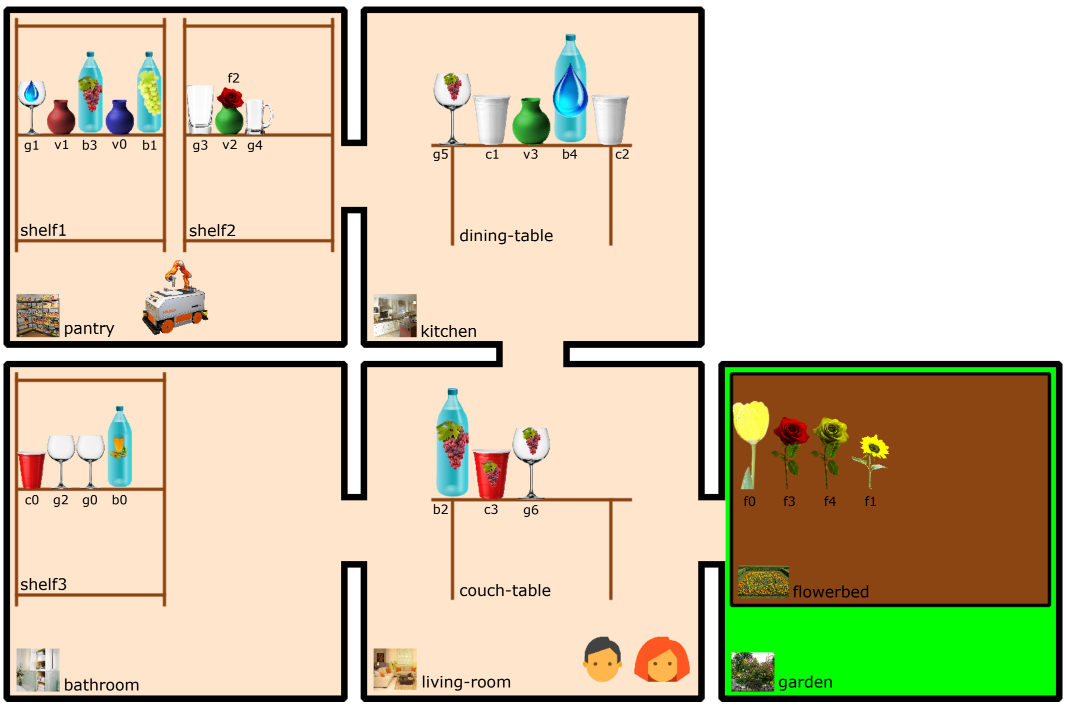


Figure 6.9: Graphical representation of the virtual environment used to evaluate the goal formulation assistant. Scenarios of varying complexity are obtained by populating the environment with a variable number of objects, randomly spawn in different locations.

6.6.4 Goal Formulation Assistant

Due to the workspace limitations in our real world setup depicted in Figure 6.7 and the constraint number and type of objects detectable by our perception component, we conducted an evaluation of the goal formulation assistant considering a virtual environment. The scenario is illustrated in Figure 6.9 and chosen to resemble a domestic domain composed of a kitchen, living-room, pantry, bathroom and garden. Each room, in turn, offers at least one potential locations to perform grasp and drop actions, i.e., a dining table, couch table, shelf or flowerbed, respectively. Bottles, cups, glasses and vases are distributed among the furniture. There are three types of flowers: $roses$, $tulips$ and $sunflowers$. Drinking vessels can contain several drinks: $water$, $lemonade$, $apple$-$juice$, $orange$-$juice$, $beer$, red-$wine$ and $white$-$wine$. Flowers, cups and vases have a color: red, $white$, $yellow$; cups and vases can also be $green$ and $blue$. Finally, glasses can be of shape $cylinder$, $balloon$, $beer$-mug or $white$-$wine$-$glass$. Flowers can be put into vases but may also placed directly on other furniture. A robot ($omniRob$) has the ability to move between the rooms and serve the two persons (me and $friend$). The actions, we consider to be available are: $arrange$ to arrange a flower in a vase, $pick$ to pick a flower out of a vase, $grasp$ to grasp a portable object, $drop$ to drop a portable object on a furniture, $give$ to give a portable object to a human, $pour$ to pour a liquid from one vessel into another, $drink$ to assist a human in drinking, $move$ to move the robot between rooms and $approach$ to approach a furniture or human in preparation of a subsequent manipulation action.

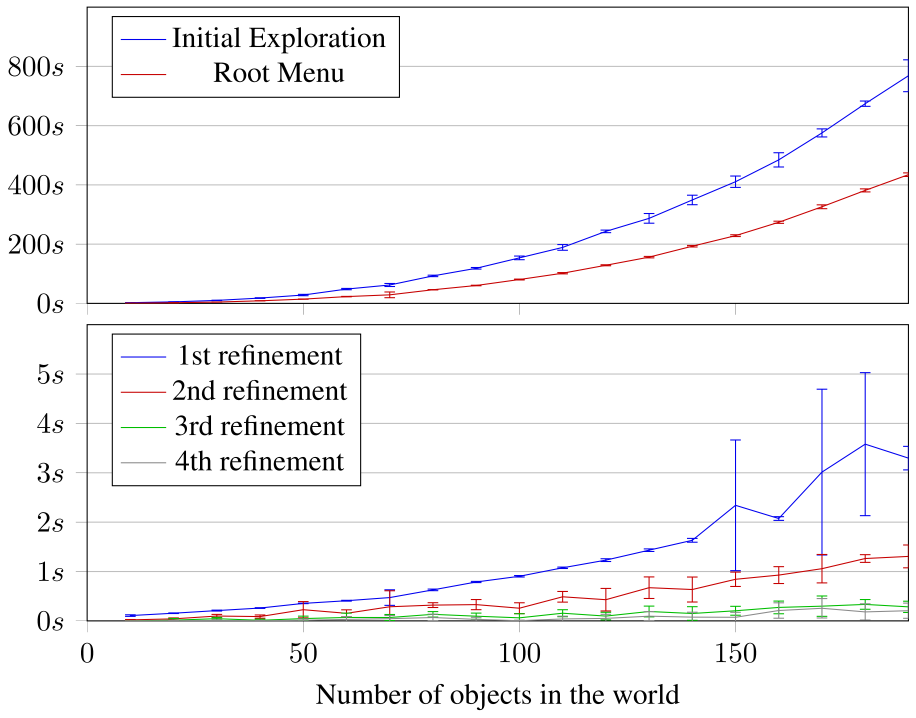


Figure 6.10: Evaluation of the computation time, considering the *arrange* task and an increasing numbers of objects in the environment. *Top:* Times for initial object exploration and root menu creation, required to build the menu structure. *Bottom:* Times required to refine references on different depths in the GUI.

Performance Evaluation

In a first experiment, we evaluated the scaling behavior of the goal formulation framework. To do so, we deploy a random scenario generator specifically designed to create planning problem instances with an increasing number of world objects. One of such random planning problems is graphically depicted in Figure 6.9. In order to quantitatively compare the performance of the goal formulation assistant on these scenarios, we consider the time required to start the application and to select the parameters of the *arrange* action (arranges flowers in a vase). Moreover, as the performance of the high-level planner GUI does not depend on the control paradigm adopted, the keyboard is used to send navigation commands for simplicity. The experiment was repeated ten times and averaged to obtain meaningful results.

Figure 6.10 illustrates the times needed for several operations as a function of the number of objects present in the world. The most time-consuming component is given by the initial object exploration of the goal formulation GUI, where potentially reachable goals are determined based on relaxed exploration. Another computational expensive operation is the root menu generation, where initial partitions are chosen for all actions (see Figure 6.10, top). The reference refinements for the current parameter of an action, on the other hand, requires less than 10 s even for scenarios containing numerous objects (see Figure 6.10, bottom). Thus, delays are reasonable for scenarios with up to 200 objects. However, this assertion only holds as long as the world and thus the references do not change. Considering dynamic environments, changes of the world are frequently triggered by actions taken by the robotic service assistant or other robotic and human agents. For example, when the robot has grasped a cup, the system should no longer refer to the cup as *the cup on the table*. Instead, the reference must be rebuilt given the updated environment state yielding *the cup at the robot's gripper*. For simplicity, our approach rebuilds all object references when an environment change has been detected. In the future, only obsolete references should be recomputed in order to scale well on larger scenarios.

Usability Study

In a further experiment, we evaluated the user-friendliness and intuitiveness of our goal formulation assistant and investigated how humans refer to objects in their surrounding. To do so, we conducted a user study with healthy participants in which we advised the users to reach a specific goal state in five different scenarios of increasing complexity:

- **S1**: Move the robot to the garden.
- **S2**: Drink beer using a beer-mug.
- **S3**: Arrange a red flower in a red vase.
- **S4**: Place a red rose on the couch table.
- **S5**: Give a red wine glass with red wine to your friend.

In total, we gathered data from 20 participants, 3 female and 17 male whose age ranges from 25 to 45 years. The participants were students in computer science and administrative employees of the university. This user study has to be seen as a first evaluation of the intuitiveness and user-friendliness of the system. Further investigations considering a larger number of users as well as paralyzed patients are required to assess these properties in detail. All participants gave their informed consent and their data was pseudonomized at study inclusion. As for the previous experiment, we used the virtual environment depicted in Figure 6.9 as the basis of our investigations and permitted the subjects to use the keyboard for navigation in the GUI. The five scenarios, i.e., S1-5, were generated beforehand to get comparable results for all participants. In particular, care has been taken in the selection of the scenarios to ensure that the corresponding goals are not trivially reachable.

For conducting the experiment, we defined the following protocol. First, the user interface was introduced to the inexperienced and untrained participants (i.e, they used the system the first time) by showing them its individual elements and components. Secondly, participants were asked to use the navigation tools of the interface in order to achieve the respective goal states for S1-S5. Meanwhile, we recorded the number of steps performed by them in the GUI to reach the desired goal. Since there were no time constraints and sub-optimal navigation strategies were allowed, all users managed to reach the predefined goal states. After finishing each of the scenarios the participants had to evaluate the system in a questionnaire. First, they were asked to rate on a scale of 1 (unreasonable) to 5 (fully comply) how well the displayed control options in the GUI complied with their expectations. Additionally, they had to rate the overall *intuitiveness* of the user interface in the range of 1 (not intuitive) to 5 (excellent). Moreover, we asked the participants whether they prefer using references or internal names (e.g., $b2$ in Figure 6.9) to describe an object.

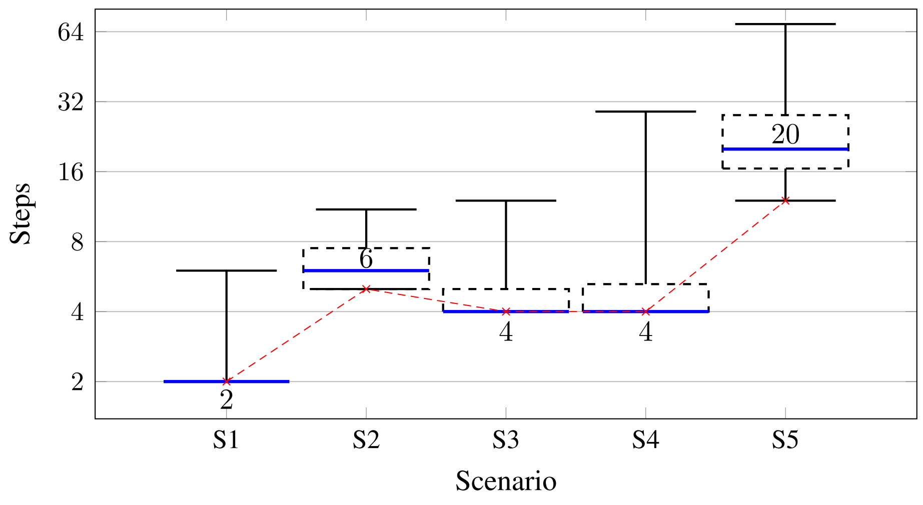


Figure 6.11: Box plot illustrating the number of steps required by the participants in the user study to achieve a given goal for the five scenarios S1-S5. The dashed red line indicates the minimum number of steps required to reach the goals.

Figure 6.11 shows the quantitative result of the user study. Here, we evaluated the number of steps performed by each of the participants in the GUI to achieve the predefined goals successfully. In this context, a *step* corresponds either to a refinement of an attribute or the selection of the *back* entry in the GUI. The figure illustrates the maximum, minimum, mean, upper and lower quantile regarding the performed number of steps for each scenario, respectively. Additionally, the minimal possible (or optimal) number of steps required to successfully achieve the goal, is depicted. Most of the participants were able to find a near-optimal strategy to solve the tasks S1-4. The outliers in the first four scenarios are mainly caused by the user exploring the possibilities of the user interface. In the last scenario (S5) an increased number of steps has been observed. This increase can be traced back to the following reasons. First, contrary to S1-4, this scenario required to execute two consecutive actions to achieve the goal, i.e., fill a balloon shaped glass with red wine and give this glass to the friend. Many users were not able to discover that two actions are required to achieve the goal state, which led to confusion. Therefore, participants often had to correct their decisions which resulted in a higher number of steps recorded for the fifth scenario. Secondly, the pour action required to specify the type of liquid to be poured. To keep the pour action as flexible as possible, we decided to permit pouring arbitrary liquids from all vessels. Thus, it is possible for users to express the desire of pouring wine from a bottle containing water, which resolves into an action sequence for the robot initially emptying the water bottle and subsequently pouring the wine into that bottle. However, after selecting a bottle containing a specific liquid, e.g., water, for the pour action, the redundant refinement of the $\mathit{content}$ is not intuitive to the users (water needs to be selected once again). Finally, we split a goal partition based on its information gain to reduce the number of remaining refinements required to fully specify the goal. Whereas this strategy is reasonable regarding computational effort, it does not necessarily reflect the expectations of users. Regarding the grasp action, for example, humans usually prefer to select among bottles based on their $\mathit{content}$ attribute first, instead of being initially prompted to select the location of the bottle.

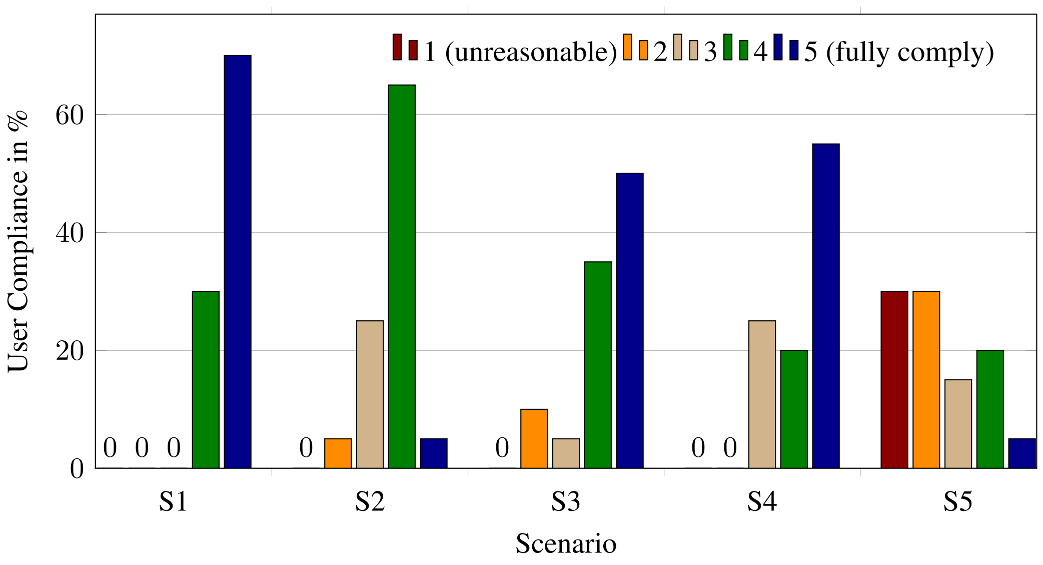


Figure 6.12: Compliance of the choices offered in the GUI with the user's expectation for the five different scenarios S1-S5. The participants had to select compliance levels from 1 (unreasonable) to 5 (fully comply).

Figure 6.12 shows the results on how well the choices offered by the high-level planning GUI actually comply with the expectations of users. A large percentage of the users comply with the refinements of the GUI in scenarios S1 to S4. Due to the previously mentioned issues, however, S5 received a much lower rating compared to the other four scenarios. This observation suggests that an extended training period is probably needed for users in order to get more familiar with the capabilities and properties of the provided interface.

In total, 80% of the participants rated the overall intuitiveness of the GUI as *good*, i.e., according to the aforementioned metric they rated the intuitiveness with at least 3 (acceptable) in a range of 1 (not intuitive) to 5 (excellent). Moreover, 85% of the participants preferred the proposed referencing of objects over the alternative of representing them by internal names, e.g., *green vase on the couch table* vs. $v1$.

In a final experiment, we evaluated the subjective quality of object references. To do so, we proposed four references to objects depicted in Figure 6.9 and let the users rate how well each of those references describe the corresponding object. Moreover, subjects were asked to generate references to these objects in natural language themselves. For the purpose of this study, we considered the green vase with the red rose located in the pantry ($v2$) and the glass, filled with red wine ($g6$), located on the couch table in the living-room. The proposed references ranged from under-determined to over-determined descriptions, e.g., *the green vase* vs. *the green vase located in the right shelf in the pantry which contains a red rose*. According to our results, the preferred references highly depend on the location of the user and robot and particularly on whether they share the same location. Interestingly, the capability of users to impersonate the robot has also an strong influence on the references preferred. For referring to $v2$, for example, many users described the object as *the vase in the right shelf* or *the vase containing a rose*, knowing that the robot is already located in the pantry and therefore does not need further information to identify the object. In contrast, a second group of users described the same object by additionally specifying the room and content of the vase, assuming that the robotic service assistant is also located in the living-room and therefore requires a more detailed object description. These observations were confirmed by analyzing the preferred user references regarding object $g6$. A first group of users referred to this object as *the red wine glass on the couch table*, whereas the other group specified the living-room as additional information for the robot assumed to be currently located in the pantry. Although detailed references, that uniquely identify each of the two objects were proposed, most users preferred shorter descriptions assuming to have full knowledge about the current state of the world. In summary, our evaluation revealed that unique object references are precise descriptions, however, lack the common sense assumptions expected by human users. Therefore, references to an object should not only be adapted if the object attributes itself have changed, but also according to updated information on other entities of the world and actions previously taken by the robot.

6.7 Related Work

The robotic service assistant framework introduced in this chapter is composed of several components, whereby each of them can be assigned to a separate field of research. In the following, we will describe a collection of approaches addressing the problem of implementing a robotic service assistant, decoding EEG signals and combined task and motion planning, respectively. Regarding the field of motion and manipulation planning, we refer the reader to the related work in Section 5.6.

Robotic Service Assistants Park et al. [98] proposed a system that allows paralyzed patients to command a robotic system for autonomous feeding tasks. To detect task relevant targets, e.g., the spoon, a bowl and the user's mouth, a marker-based vision system is used. Control of the system, however, requires the operator to be able to control a graphical user interface by hand, which limits the scope of patients that are capable of using it. The system presented by Chung et al. [29], uses a wheelchair-mounted robotic manipulator to assist a user in eating and drinking tasks. To do so, the robot needs to locate and pick up the drink, conveying it to the user's mouth and finally placing it back on the table. The pick up action is realized adopting a position-based visual servoing (PBVS) approach. Object manipulation is implemented using the Contrained Bidirectional RRT planning algorithm. The work, however, does not consider user commands nor feedback. The work of Achic et al. [1] proposes an integral system combining a hybrid BCI interface and shared control system for navigation and manipulation application using a wheelchair mounted manipulator. Here, users are permitted to select control tasks from a GUI through EEG signals based on steady state visual evoked potentials (SSVEP). The wheelchair is controlled considering head movements obtained by inertial sensors.

The aforementioned approach relies on the response of the human brain evoked by visual cues provided by the system, as commonly used also in other approaches. Generally, these control paradigms can not be considered as pure BCI control, since users are not actively communicating their intent, but rather wait for a specific visual event to occur. In contrast, Wang et al. [144] used a motor imagery BCI with three classes to achieve low-level control of a robotic arm. The classes correspond to left, right hand and foot motor imageries, which are actively evoked by users, and subsequently used to navigate through a GUI providing eight different instructions for the robot arm's end-effector, e.g., *left*, *right*, *up* and *down*. Using their approach, users can steer the end-effector in Cartesian space. For complex manipulation actions, however, the low-level control of the end-effector constitutes a tedious task, demanding a high mental effort from users.

Schröer et al. [116] developed a system which allows users to command an autonomous robotic manipulator based on EEG signal recordings. The aim of their work is to assist users in the tasks of drinking from a cup. Here, the task is represented by a fixed sequence of actions, which is processed according to *go*-signals received from the BCI interface. The disadvantage of their approach is that it is very task specific and therefore not capable of offering users a wider range of tasks. Moreover, due to the use of a fixed-base manipulator the scope of potential tasks to be completed by the robotic assistant is strictly limited.

Recently, Muelling et al. [90] presented a shared-control assistance framework using an implanted intracortical brain-computer interface for controlling a seven degree-of-freedom robotic manipulator, albeit as a prosthetic instead of an autonomous robotic service assistant. In a first step, user commanded end-effector velocities are extracted from EEG signal recordings. Afterwards, the user's intent is inferred by entropy maximization considering multiple manipulation targets. The obtained goal probabilities are subsequently used to decide which of the objects is the most likely target of the user. The motion of the robotic arm is adapted accordingly. Compared to the work of Wang et al. [144], this demands a lower mental effort, but still requires users to stay focused throughout the task execution.

EEG Signal Decoding Robust decoding of neuronal signals is a prerequisite for the above described applications. Inspired by the successes of deep convolutional neural networks (ConvNets) in computer vision [57, 77] and speech recognition [111, 118], deep ConvNets have recently been applied more frequently to EEG brain signal decoding. Deep ConvNets were already applied to decoding tasks useful for building brain-computer interfaces. Lawhern et al. [83] presented a deep ConvNet to decode P300 oddball signals, feedback error-related negativity and two movement-related tasks. When evaluated cross-subject, i.e., trained on some subjects and evaluated on other subjects, their ConvNet yields competitive accuracies compared with widely-used traditional brain-signal decoding algorithms.

Tabar and Halici [132] proposed an approach combining a ConvNet with a convolutional stacked auto-encoder in order to decode motor imagery within-subject, yielding better accuracies than several non-ConvNet decoding algorithms.

The work of Schirrmeister et al. [114] deploys a shallow and a deep ConvNet to decode both motor imagery and motor execution within-subject, reaching or slightly surpassing the accuracies of the widely used EEG motor-decoding algorithm *filter bank common spatial patterns* [5]. Moreover, Bashivan et al. [8] used a ConvNet trained on fourier-transformed inputs to estimate mental workload. In addition to approaches evaluating the accuracies of ConvNet decoding techniques, ConvNet visualization methods have been developed that allow to get a sense of what brain-signal features the network is actually using [8, 114, 129].

In summary, these advances make deep ConvNets a viable alternative for brain-signal decoding in brain-computer interfaces. Still, to the best of our knowledge, the work presented in this chapter represents the first approach performing online control with deep ConvNets using an EEG-based brain-computer interface.

Task and Motion Planning For tasks requiring only a single action or a predefined action sequence to be executed it is sufficient to rely on existing motion and manipulation techniques. On the other hand, high-level planning is essential if the tasks become more complex and are not specified beforehand. In our framework, it is a user to specify a desired goal and the task of a high-level planner instance to determine an appropriate combination and order of actions implementing the goal state in the world. To do so, knowledge about the current state of the world and the preconditions and effects of actions is important. The available actions in high-level planning, such as $drop$ or $grasp$, are typically presented in a human readable fashion, whereas objects are assigned an internal representation, e.g., *ID3185*, which is hard to relate to the associated entity of the environment, e.g., *red cup on the shelf*, by the user.

In this work, we adopted the approach of Göbelbecker [50], proposing an automatic goal formulation assistant, which enables to reference objects more naturally by their type and attributes. Following this approach, human operators can express desires such as *bring me a glass of water* to the task planner without the need of knowing the underlying system. The generation of referring expressions has already been subject to computational linguistics research for years [76]. With recent advances in natural language processing, computer vision and the rise of neuronal networks, it is nowadays possible to identify objects in images by building referring expressions generated from features [119]. Spatial references can be used to discriminate similar objects [149]. Robotic systems can furthermore not only be controlled by natural language, they can even learn more complex tasks composed of several primitive actions [86]. However, such systems usually lack knowledge about the actions that can be executed by the robot, and the objects in the world that can be manipulated by the desired action. Instead of natural language, we permitted users to communicate goals in a menu-driven goal selection interface, where only applicable actions and feasible object parameters are offered to the user.

Task and motion planning (TAMP) is an important component for realizing an autonomous service assistant, which ensures that high-level actions, i.e., tasks, can be refined to motion trajectories for the robot. Common to most approaches is a hierarchical decomposition of the problem into task and motion planning layers. Due to the high dimensionality of the TAMP problem the decomposition can be understood as a way to guide the low-level planners based on the high-level plan solution and vice versa.

For example, Kaelbling and Lozano-Pérez [64, 65] propose an aggressively hierarchical planning method, which operates on detailed, continuous geometric representations and partial symbolic descriptions. Once it made choices, it commits to them in a top-down fashion in order to limit the length of plans that need to be constructed. Moreover, their approach limits the need to project the effect of actions into the far future. Thus, using such a hierarchical decomposition permits to handle planning problems with long horizons in an efficient way.

De Silva et al. [40] presented an approach that combines the *Hierarchical Task Network* (HTN) planning formalism, which reasons on abstract tasks, with a *Geometric Task Planner* (GTP). The result is a planner that is able to perform geometric reasoning on abstract entities, i.e. tasks, relying on a discrete space of precomputed grasp, drop and object positions.

Recently, Dantam et al. [36] introduced the probabilistically-complete *Iteratively Deepened Task and Motion Planning* (IDTMP) algorithm, which uses a constrained-based task planner to create tentative task plans and sampling-based motion planning to perform feasibility tests.

Srivastava et al. [126] propose a planner-independent interface layer that allows combining off-the-shelf task and motion planning algorithms. Thus, future research can be dedicated to advance the techniques in either field instead of further pursue the development of special purpose, integrated implementations of task and motion planning algorithms.

The approach introduced by Lozano-Pérez and Kaelbling [87] postpones the decision on motion plans to avoid expensive backtracking due to restrictions which might happen, if the low-level planner is queried too early. Instead, they generate a *skeleton* high-level plan and a set of constraints, which need to be satisfied to achieve the goals of the high-level planner.

Dornhege et al. [43] integrated task and motion planning by extending the Temporal Fast Downward (TFD) task planner, introduced by Eyerich et al. [44], with semantic attachments. These attachments represent external procedural reasoning modules that can compute the valuations of state variables at planner run-time. The symbolic planner remains mostly unaffected by the extension, since only table look ups are substituted with function calls to obtain the necessary information. Integration of such modules, permit to evaluate the feasibility of motion plans on demand, thus ensuring that high-level plans can be refined to a valid motion sequence.

6.8 Conclusion

In this chapter, we presented an interdisciplinary framework that permits severely paralyzed patients to communicate their desire to a mobile robotic service assistant solely using thoughts. By integrating perception components and autonomous motion and manipulation planning capabilities, the robotic assistant is able to successfully perform complex tasks, including close range interaction with the user, in a frequently changing environment. Moreover, users are provided with a large set of selectable actions, reflecting what is actually feasible given the current state of the environment. A crucial component of the framework is represented by the high-level planner and its graphical user interface, which serves as an intermediate layer between user and autonomous robotic service assistant. In particular, relying on such a system permits to overcome the curse of dimensionality typically encountered in non-invasive BCI control schemes, thus opening up new perspectives for human-robot interaction scenarios.

As the framework is composed of several approaches belonging to different research fields, aspects for potential future improvements are manifold. In the following, we will address only the most relevant ones for each area, respectively.

EEG Signal Decoding Decoding EEG signals with sufficient accuracy is a basic prerequisite for the deployment of the proposed robotic service assistant system. Without this capability, paralyzed patients may experience the control of the user interface as inconvenient and are likely to refuse its use due to the additional mental load required to revise erroneous GUI navigation actions. The decoding accuracies achieved by our approach are promising (see Table 6.1), however, require recording labeled data for each subject for several days. In order to obtain reliable decoding results within less time, the training phase of the ConvNet could be shortened using *data augmentation* techniques. To do so, time windows of smaller size could be incrementally shifted along the data blocks recorded while a visual cue has been displayed in the high-level planner GUI, e.g., the down pointing arrow in Figure 6.2. Thus, the amount of data available for training the ConvNet is many times greater than the size of the original dataset. Another interesting approach to speed-up the EEG signal decoding setup is represented by *transfer learning*, i.e., the neural network is trained on some subjects and evaluated on other subjects. First steps towards this direction have been already presented in the work of Völker et al. [140] and need to be evaluated in the context of our application. In the same work, error-related potentials (ERP) have been investigated, which represent neuronal activity elicited in the human brain when a user observes an erroneous action. Integrating such error decoding capabilities in our framework, may allow users to avoid or abort erroneous actions at an earlier stage, thus improving the responsiveness of the graphical user interface.

Providing subjects with continuous feedback during action execution instead of communicating only the outcome of actions via the high-level planner GUI, represents another point for improvement. Considerations regarding this issue include the idea of providing users with the live-stream of the robot's on-board camera during action execution. The solution would be easy to implement in our framework, however, goes along with an increased electrical activity recorded in EOG (eye movements). Moreover, as the EOG signals superimpose with the EEG signals in the frequency bands relevant for decoding the five mental tasks (see Figure 6.8), this approach may in turn deteriorate the decoding accuracies. A less intrusive alternative to realize continuous feedback could be obtained by displaying a progress bar in the GUI during action execution.

Goal Formulation Assistant Representing available goals by referring expressions and permitting users to select among them by navigating through a graphical user interface has been found an appropriate strategy to establish a communication between a human and a robotic service assistant. However, the order of attribute refinements for actions following from the adopted information gain sometimes does not fully comply with the users expectations. A possibility to further improve the user friendliness of the interface would be to record user feedback during the GUI control, which can be subsequently used to fine-tune the information gain. This way, the system could propose refinements in an order which complies with the user's preferences, e.g., inquiring the content of the bottle before asking for its position in the process of specifying the $grasp$ action parameters. Another, though challenging, extension of our framework is to integrate *deictic references*, which permit creating references to the position of entities and events by pointing to a time, place or situation. By maintaining the history of previously selected actions, for example, this would enable users to refer to the cup the robot just grasped in the previous action for defining the subsequent $pour$ action, i.e., *pour from the cup just grasped*, instead of requiring the user to fully specify which cup is meant once again. In the experiment described in Section 6.6.4, we referred to this aspect also by the notion of *common sense*.

Finally, our current implementation rebuilds all object references when an environment change has been detected, which poses a computationally expensive step considering scenarios containing numerous objects. In the future, only obsolete or new references should be recomputed/added in order to scale well on real world domestic environments typically containing objects numbering in the hundreds.

Motion and Manipulation Planning The probabilistic planning techniques adopted in our framework have shown to reliably generate and execute solution trajectories for the mobile base and robotic arm, whereas we assumed the environment to remain static throughout trajectory execution. For the deployment of an autonomous robotic service assistant in the real world, however, it is to be expected that there are other moving entities sharing the workspace with the robot. Therefore, dynamic collision avoidance becomes an important ability to be implemented in the PTRM and BI^{2}RRT* planning system. A solution regarding manipulation tasks, is to substitute the A* algorithm with its anytime dynamic variant AD* [85]. Here, the *octomap* representation of the scene, already employed to perform collision checks during planning, could be used to continuously monitor the validity of the current solution. In navigation tasks, the measurement of a pair of 2D laser range finders, mounted at the mobile base, are currently used to localize the robot in the environment. Additionally, these measurements could be used to locally adapt the planned path and thus the motions of the mobile base if needed. An implementation of this approach is represented by the *elastic bands* algorithm, proposed by Quinlan and Khatib [102], connecting motion planning and control. In this context, the solution generated by BI^{2}RRT* would serve as a global reference path, whereas the control commands for the mobile base are handled by the elastic bands algorithm.

So far, offline planning using the PTRM planner has been adopted to realize physical human-robot interactions. Whereas this approach is appropriate to reach the human through a collision-free trajectory, it assumes the perception component to provide highly accurate pose estimates, e.g., for the user's mouth, and that users do not move during motion execution. A solution, improving the safety for such interactions, could be obtained by switching to a control-based approach for the execution of the last stretch of the planned end-effector trajectory. Doing so, would allow to explicitly control the exchange of contact forces between the end-effector (or cup regarding the drinking task) and human, as already proposed in the literature by Magrini et al. [88].

Finally, further primitive actions, such as *open* or *close*, requiring to plan coordinated base-arm motions could be included in our framework by exploiting the full capabilities of the BI^{2}RRT* planning system. This way, the robot could, for example, open doors to navigate to different rooms or open a closet/drawer to examine its contents. Manipulation of articulated objects, however, generally requires the object's handle pose, articulation model and range of motion, to be given. Whereas it is admissible to specify the corresponding information for the latter beforehand (assuming furniture to be static), the former requires the deployment of advanced visual perception techniques, such as the one presented by Rusu et al. [109].

Chapter 7

Conclusions

7.1 Summary

In the scope of this thesis, we introduced several novel contributions to the field of motion imitation and generation for mobile robotic systems. We hereby considered different levels of autonomy for the robot, initially relying on a human operator to provide the knowledge required to complete a task successfully towards a robotic service assistant capable of autonomously planning and executing mobile manipulation actions. Moreover, we incorporated motion imitation techniques to investigate what makes human movements particularly natural by comparing motion demonstrations of healthy subjects with the ones of patients exhibiting motor control deficits. Motion imitation and generation are both valuable approaches, as each of them offers its own individual advantages. Therefore, preference to the appropriate technique should be given depending on the intended field of application. For all proposed approaches, we presented the relevant theoretical background, discussed related works and conducted various experiments.

We first introduced an approach that permits humanoid robots to imitate whole-body motions from a human operator in real time. Here, we used data from an inertial sensor-based whole-body motion capture suit to infer trajectories for the human extremities. To transfer motion samples to the humanoid, we applied a motion mapping method establishing similarity between the robot and demonstrated human posture in a first step. In a second step, we performed posture stabilization which adapts the posture in order to make it compliant with the stability constraint before sending it to the real robot platform. Using the developed system, the humanoid is able to imitate complex motion sequences, including extended periods of time in single support and teleoperated mobile manipulation tasks. The actual value of the prescribed motion significantly affects the performance of the presented system. At this stage, we assumed that the motion samples are representative for a desirable, natural human kinesic behavior that should be adopted by the robotic platform without putting them into question.

Evaluating the value of human motion demonstrations requires to quantitatively compare motion recordings of different human subjects. For our study, we examined in this work the effect of neurological impairments, which are well known to induce motor control deficits, on the human musculoskeletal system. In particular, we analyzed the differences in motor control behavior between healthy subjects and Parkinson's disease (PD) patients in a hand coordination task. To do so, we proposed an algorithm that iteratively learns the joint weights of a Jacobian-based controller, required to reflect the demonstrated human motion as closely as possible. On the basis of the resulting joint weights learned for each group, we subsequently conducted a motor control strategy analysis, suggesting a distinction of a *proximal* and *distributed* motion strategy for healthy and PD subjects, respectively.

When the exchange of information between a human operator and a robotic system becomes prohibited or undesirable, a robot needs to acquire various autonomous navigation and manipulation capabilities. A prerequisite for achieving this level of autonomy is that the robot has sufficient knowledge about its own capabilities and the environment in which it operates. Accordingly, we presented in this thesis an approach that permits mobile robotic systems to autonomously select a stance pose relative to a given grasping target in preparation of a subsequent manipulation action. To represent the robot's reaching capabilities, we precomputed so-called reachability maps, which represent a spatial grid of workspace voxels reachable by the robot's end-effector. Additionally, manipulability measurements have been considered in the map construction process that permit evaluation and comparison of the quality of different reachability map voxels. A representation of potential stance poses relative to a specific grasping target is obtained by inverting the reachability information and subsequently filtering the so-called inverse reachability map considering the relative location and orientation of the support surface and base (or feet), respectively. As demonstrated by our results for the Nao humanoid robot, the IRM-based stance pose selection leads to a substantially increased success rate in reaching grasping targets compared to other meaningful foot placements within the vicinity of the desired grasp.

Determining a terminal configuration for mobile manipulation tasks can be considered as an initial step, whereas it remains to equip a robot with the ability to autonomously plan a collision-free path connecting its current configuration to the desired target configuration. To implement this capability for mobile manipulator platforms, we presented in this thesis the Bidirectional Informed RRT* algorithm, which extends Informed RRT* towards bidirectional search and satisfaction of arbitrary geometric end-effector task constraints. Given a pair of terminal configurations, our planner performs a bidirectional search using uniform sampling in the configuration space until an initial, though sub-optimal, solution path is found. This path is subsequently refined for the remaining planning time, adopting an informed sampling strategy, which yields a higher rate of convergence towards the optimal solution. We thoroughly evaluated our approach by comparing the performance of the proposed planning framework to state-of-the-art path planning algorithms on several planning problems of varying complexity. As the results show, our approach is capable of generating low-cost solution paths more reliable and faster than existing RRT-based planning methods.

Finally, we introduced a robotic service assistant framework dedicated to user with limited communication skills. The brain-computer interface (BCI) system is composed of several interacting components, i.e., non-invasive neuronal signal recording and decoding, high-level task planning, motion and manipulation planning as well as environment perception. A dynamic knowledge base represents the backbone of this network and permits the involved components to continuously exchange information about the current state of the world. A graphical user interface provided by the high-level planner displays the currently available actions, as determined from the knowledge base whose contents are frequently updated based on information reported by a camera system observing the environment. In turn, navigation in the GUI, action selection and parameter refinement is realized by decoding EEG data using a convolutional neural network approach. In order to implement the desired tasks in the real word, we adopted the BI^{2}RRT* and PTRM planning algorithms for navigation and manipulation actions, respectively. In various experiments, we demonstrated the applicability and robustness of our framework in real world scenarios, considering fetch-and-carry tasks and tasks involving human-robot interaction. Furthermore, we demonstrated on virtual environments that the performance of the system remains reasonable considering up to a few hundred objects. Moreover, our user study revealed that the proposed goal formulation assistant is perceived as convenient for the majority of users, even though no prior training on the GUI has been conducted with the participant.

Summarizing, we developed in this thesis solutions for the following questions:

- How can we capture and transfer whole-body human motions to a humanoid robot in real time?
- How can we quantify motor control deficits in human subjects induced by neurological impairments, e.g., arising from Parkinson's disease, using motion imitation techniques?
- How do humans position themselves relative to an object in preparation for a subsequent manipulation action and how can that strategy be transferred to mobile robotic systems?
- How can we efficiently plan for challenging reaching and delivery tasks for mobile robots, thereby avoiding collisions and respecting geometric constraints imposed by the objects to be manipulated?
- What are the requirements and challenges to build a robotic service assistant system for users with limited communication skills?

All of our approaches were thoroughly evaluated using the *Nao* humanoid platform and the *omniRob* mobile manipulator. According to the presented results, we believe that the approaches developed in this work are valuable contributions for the advancement of motion imitation and generation techniques and will be incorporated in the future by other researchers.

7.2 Outlook

The thesis at hand presented important contributions and promising results in the area of motion imitation and generation for mobile robotic systems. Furthermore, it demonstrated that motion imitation techniques are useful beyond the field of robotics by developing new measures that can be used to evaluate the effectiveness of therapeutic interventions in the medical field. As the research questions addressed in the scope of this thesis are manifold, there exist several potential extensions to be elaborated on in future research.

The system proposed for the imitation of complex human whole-body motions by a humanoid robot currently considers only the position for the upper extremities. A possible extension of our implementation could consider also the orientation of the hands. Doing so, would facilitate the performance of teleoperated object manipulation tasks for the human demonstrator. Moreover, human head movements could be included in the imitation process to let the operator actively control the alignment of the camera, which is typically integrated in the robot's head for humanoid platforms. This would permit the operator to explore the robot's surrounding as well as to receive visual feedback in delicate manipulation tasks. An approach integrating these components in a whole-body motion imitation framework has already been presented by Wang et al. [143]. Ultimately, a full virtual reality experience could be realized using a head-mounted display, e.g. the *Oculus Rift* system [82], showing the robot's on-board camera view, in combination with an omnidirectional treadmill for teleoperating navigation actions.

For analyzing the motion strategy adopted by healthy and Parkinson's disease patients, we considered a hand coordination task and thus only motor control parameters related to the upper body limbs. In future research an equivalent artificial kinematic model for the lower limbs could be used to extend the analysis towards further tasks, such as walking or climbing stairs. The ultimate objective would be to consider whole-body motions to obtain a comprehensive analysis of human motor control behavior on the joint activity level. Other possible extension relate to the data acquisition process and the statistical tools used to evaluate the joint weight means learned for each group. Regarding the former, it would be beneficial to increase the amount of motion samples in order to further reduce the intragroup variability for the joint weight means. In doing so, particular care must be taken to ensure that the number of male and female, mean age and handiness for the healthy subjects group matches the one of the PD group. Furthermore, participating patients should have the same pathological side in order to avoid bias of the results. For the latter, one could consider performing an *analysis of variance* (ANOVA) test to evaluate whether there is a significant difference between the means of multiple joint weights. Finally, it would be interesting to investigate whether there is a relationship between motor symptom severity, i.e., UPDRS scores, and joint weight means. This may provide some insight into whether there is a gradual shift in the motor control strategy as the motor symptoms of the Parkinson's disease worsen.

In order to add quality information to the voxels of the spatial grid representing the extent of the workspace reachable by the end-effector of a mobile manipulator, we used the classical manipulability measure introduced by Yoshikawa [148]. A more accurate description of the reaching capabilities, however, could be obtained by adopting an extended variant of the manipulability measure, as already proposed by Vahrenkamp et al. [137]. Doing so, would permit consideration of the proximity to joint limits taking into account the kinematic chain's redundancy, the distance to obstacles and between different links of the robot in evaluating the quality of configurations. Another extension is to combine our approach with a footstep or motion planner to actually reach the desired stance pose, given the robot's current configuration. In this regard, a solution for humanoid platforms is provided by Garimort et al. [49], whereas the planning framework proposed in Chapter 5 could be used for wheeled robotic platforms. Assuming perfect navigation capabilities of the robot, the whole-body configuration associated with the selected stance pose could be used as a goal configuration for a bidirectional planning algorithm implementing a subsequent manipulation action. Considering wheeled mobile manipulators, the BI^{2}RRT* or PTRM planning framework could be deployed for this purpose. As a counterpart for humanoid robots, the extended RRT-Connect planner [13] could be used.

The bidirectional informed RRT* (BI^{2}RRT*) planner presented in this work permits the definition of arbitrary geometric end-effector constraints for mobile manipulation tasks. However, currently it is not possible to specify these constraints with respect to different coordinate systems, also referred to as *task frames* in this context. A solution that permits defining constraints with respect to multiple task frames using the concept of *Task Space Region* chains, has already been proposed by Berenson et al. [9] and could be easily integrated into our framework. So far, the local planner used to expand the search trees during planning assumes the robotic platform possesses omnidirectional navigation capabilities. In order to make our planner applicable for mobile platforms exhibiting differential drive or car-like kinematics [80], other sophisticated motion models need to be made available for the local planner component. Consequently, a quantitative comparison of the performance of our planning framework with respect to state-of-the-art algorithms need to be conducted for these motion models, respectively.

Regarding the presented robotic service assistant framework, several extensions are possible in the field of EEG signal decoding, high-level task planning and motion and manipulation planning. Formulating goals based on referring expressions has been experienced as convenient for the majority of users in our study. A possible extension in the context of assisted goal formulation would be to include *deictic references*, which permit creating references to the position of entities and events by pointing to a time, place or situation. In order to speed up the training of the convolutional neural network used to decode EEG data from various mental tasks, we may apply *transfer learning* techniques. Furthermore, error-related potentials (ERP), elicited in the human brain when an erroneous action is observed, could be used for early abortion of actions. Using the graphical user interface of our high-level planning component, these aspects have already been investigated in the work of Völker et al. [140] and are to be integrated into our framework in a future revision. Lifting the static world assumption for autonomous execution of navigation and manipulation actions implies the requirement for the BI^{2}RRT* and PTRM planning framework to possess dynamic collision avoidance capabilities. This could be realized using the *elastic band* approach proposed by Quinlan and Khatib [102] and the *anytime dynamic* A* planner presented by Likhachev et al. [85] for navigation and manipulation tasks, respectively. Finally, BI^{2}RRT* planning could be deployed to make further actions available to the user, e.g., *open the door*, which requires the robot to perform a coordinated base-arm motion. Execution of such tasks using the real robot platform, however, would pose additional challenges regarding the synchronization of the base and arm controller.

Appendix A
Mobile Robot Platforms

A.1 The Humanoid Robot NAO

For the experimental evaluation of our humanoid-related approaches, we used the Nao (V4) robotic platform [125], also referred to as *Nao Next Gen*, by SoftBank Robotics, illustrated in Figure A.1 (left).

The robot is 58 cm tall, weighs 4.8 kg and has 25 DOF: two in the neck, six in each arm (including one to open and close the hand), and five in each leg. In addition, the legs share a common (linked) hip joint that cannot be controlled independently. The overall kinematic model is illustrated in Figure A.1 (right). Inertia, mass, and CoM of each link are known from CAD models. To perceive its own state and the environment, the robot is equipped with a large network of proprioceptive and exteroceptive sensors, including 2 cameras located in the head, 4 microphones, a sonar rangefinder, 2 infrared transmitters and receivers, an inertial board (IMU), several joint encoders, 9 tactile and 8 pressure sensors. The head of the robot contains an Intel Atom 1.6 GHz CPU with 1 GB RAM, running a Linux kernel. In this thesis, most of the computations have been performed off-board on a standard desktop PC, connected to the robot wireless or via a wired interface.

In order to measure the configuration of the robot and to correct for backlash of the gears while executing joint angle trajectories, each joint of Nao is equipped with a magnetic rotary encoder exploiting the Hall effect. According to the hardware specification, these encoders possess a precision of $0.1°$ and remain active even if the motors of the robot are disabled. Thus, pose estimates can be also acquired when the robot is actuated manually, which represents a convenient feature for analyzing the kinematic capabilities of the robot.

The inertial measurement unit (IMU), located in the robot's chest, is composed of a two-axis gyroscope and a three-axis accelerometer and continuously provides an estimate of the robot's orientation. In order to protect the robot from damages, IMU measurements have been used in Chapter 2 to initiate a fall-back motion strategy when falling is detected inevitable.

The two cameras in the robot's head are vertically aligned and provide output images with a resolution of 640×480 pixel at 30 Hz. The vertical and horizontal field of view is $47.6°$ and $60.9°$, respectively. As the lower camera is horizontally aligned with the *HeadPitch* joint (see Figure A.1, right) and slightly points downwards observing the components of the scene proximal to the robot, it has been used for the tele-operation experiment described in Section 2.7.4.

By default, the robot comes with the $NAOqi$ software, an API that allows communicating with its sensors and actuators. Regarding motion execution, the API can be used to obtain a smooth end-effector trajectory from a sequence of timed joint angles for several motors,

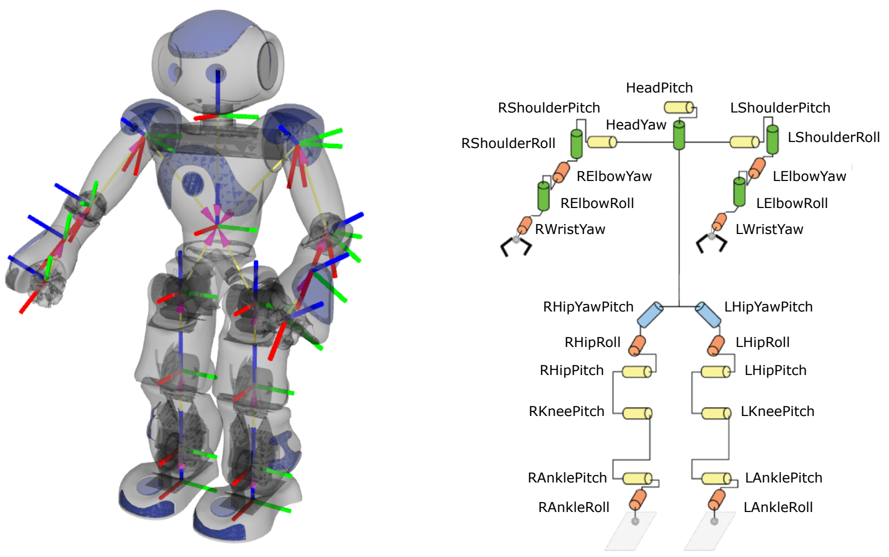


Figure A.1: *Left:* The Nao (V4) humanoid robot platform. *Right:* Kinematic model of the humanoid robot (Source: [51]).

given as input. Here, the configuration samples are considered as control points for Bezier interpolation. Moreover, the API supports dynamic omnidirectional walking adopting the *linear inverse pendulum* approach introduced by Kajita and Tani [67] as well as the generation of a sequence of footsteps for a given navigation target, as needed, for example, to reach the stance poses generated by the approach presented in Chapter 4.

A.2 The Mobile Manipulator OmniRob

In this part, we describe the *omniRob* robotic platform, illustrated in Figure A.2, used throughout our experiments involving navigation and manipulation tasks. Originally, the robot has been developed by KUKA Robotics as a technology platform to integrate and evaluate technologies required for mobile manipulation [79].

The omniRob mobile manipulator consists of an omnidirectional base and a 7 DOF robot arm. The mobile base has a dimension of 119.6×67.2×64.5 cm (L×W×H), weighs 270 kg and can travel at a maximum speed of 1.4 m/s. The maximum admissible inclination/declination for navigation is 0.5%. The base is driven by four independently powered mecanum wheels, which allows commanding arbitrary combinations of translational and rotational motions. The angular wheel velocities $\dot{\psi}$ required to realize a desired base velocity $\dot{\mathbf{p}}_{base} = (\dot{x}, \dot{y}, \dot{\theta})^\top$ follow from the equations of motion, defined as

$$\begin{pmatrix} \dot{\psi}_1 \\ \dot{\psi}_2 \\ \dot{\psi}_3 \\ \dot{\psi}_4 \end{pmatrix} = \frac{1}{r} \begin{pmatrix} 1 & 1 & (a+b) \\ 1 & -1 & -(a+b) \\ 1 & 1 & -(a+b) \\ 1 & -1 & (a+b) \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix}, \tag{A.1}$$

where r is the wheel radius and a and b are half the track and wheelbase, respectively. For a better understanding, these quantities are schematically depicted in Figure A.2 (right).

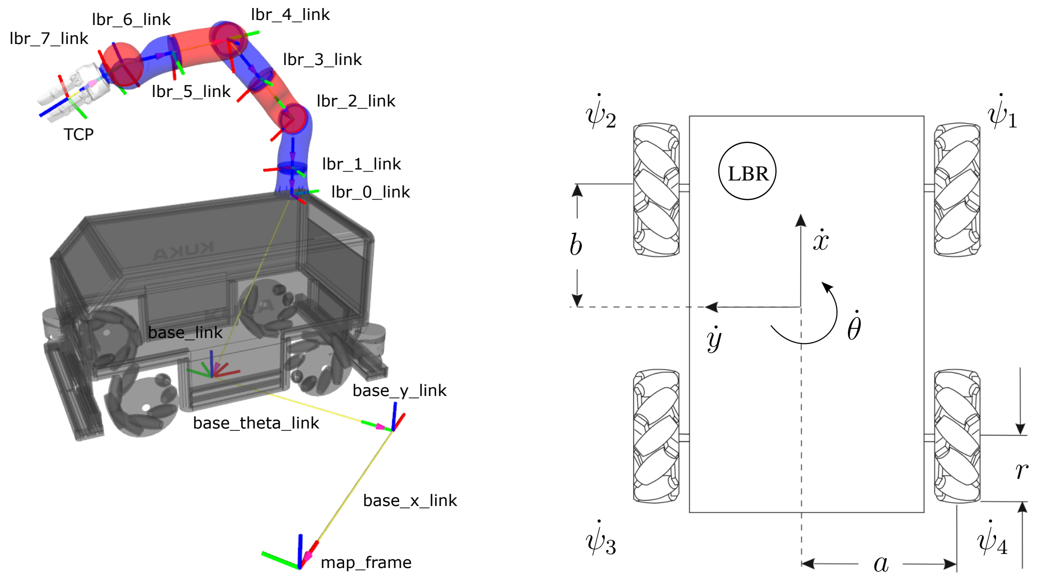


Figure A.2: *Left:* Kinematic model of the KUKA omniRob mobile manipulator platform with virtual planar joints. *Right:* Motion model of the omnidirectional mecanum wheeled mobile base (Source: [106]).

Due to the omnidirectional navigation capabilities of the robot, we considered for motion planning a simplified kinematic model, representing the translational and rotational motion components of the base by two prismatic and a revolute joint, respectively. Additionally, two bumpers are installed at the front and rear to protect the two *Sick S300 Professional CMS* laser scanners mounted at the corners of the base. The laser scanners provide four switchable protective/warning fields and posses a scan angle of 270°, thus covering the surrounding of the mobile base at a height of 13.9 cm above the ground.

The robotic arm mounted on the base is a KUKA *Lightweight Robot* (LWR) manipulator, also referred to as *Leichtbauroboter* (LBR) in german. The arm has one degree of redundancy, weighs 15 kg and is designed for a maximum payload of 7 kg.

The gripper mounted at the end-effector flange of the LBR manipulator is a 3-finger dexterous hand (SDH) by Schunk [117]. The hand weighs 1.95 kg and has 7 DOF, two for each finger and a revolute joint for rotating two of them. Additionally, each finger is equipped with two pressure sensors for tactile sensing. This permits to monitor the manipulation process as well as to adapt the finger configurations if needed to achieve an optimal grasp.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] F. Achic, J. Montero, C. Penaloza, and F. Cuellar. Hybrid BCI system to operate an electric wheelchair and a robotic arm for navigation and manipulation tasks. In *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 249–254, 2016.

[2] B. Akgun and M. Stilman. Sampling heuristics for optimal motion planning in high dimensions. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2640–2645, 2011.

[3] S. Albrecht, K. Ramirez-Amaro, F. Ruiz-Ugalde, D. Weikersdorfer, M. Leibold, M. Ulbrich, and M. Beetz. Imitating human reaching motions using physically inspired optimization principles. In *Proc. of IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2011. doi: 10.1109/Humanoids.2011.6100856.

[4] R. Alterovitz, S. Patil, and A. Derbakova. Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3706–3712, 2011.

[5] K. K. Ang, Z. Y. Chin, H. Zhang, and C. Guan. Filter Bank Common Spatial Pattern (FBCSP) in Brain-Computer Interface. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 2390–2397, 2008. doi: 10.1109/IJCNN.2008.4634130.

[6] A. Aristidou and Y. Chrysanthou. Feature extraction for human motion indexing of acted dance performances. In *Proc. of the Int. Conf. on Computer Graphics Theory and Applications*, 2014.

[7] J. Barbic, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard. Segmenting motion capture data into distinct behaviors. In *Proc. of the Int. Conf. on Graphics Interface*, 2004.

[8] P. Bashivan, I. Rish, M. Yeasin, and N. Codella. Learning Representations from EEG with Deep Recurrent-Convolutional Neural Networks. In *arXiv: 1511.06448*, 2016.

[9] D. Berenson, J. Chestnutt, S. Srinivasa, J. Kuffner, and S. Kagami. Pose-constrained whole-body planning using task space region chains. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2009.

[10] B. Bonet, G. Loerincs, and H. Geffner. A Robust and Fast Action Selection Mechanism for Planning. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI / IAAI)*, pages 714–719, July 27–31 1997.

[11] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[12] F. Burget and M. Bennewitz. Stance selection for humanoid grasping tasks by inverse reachability maps. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, pages 5669–5674, Seattle, USA, 2015.

[13] F. Burget, A. Hornung, and M. Bennewitz. Whole-body motion planning for manipulation of articulated objects. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.

[14] F. Burget, M. Cenciarini, B. Meier, H. Bast, M. Bennewitz, W. Burgard, and C. Maurer. A closed-loop system for real-time calibration of neural stimulation parameters using motion data. *In Proc. of the ICRA Workshop on Wearable Robotics for Motion Assistance and Rehabilitation - RoboAssist*, 2014.

[15] F. Burget, C. Maurer, W. Burgard, and M. Bennewitz. Learning motor control parameters for motion strategy analysis of parkinson's disease patients. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5019–5025, Hamburg, Germany, 2015.

[16] F. Burget, M. Bennewitz, and W. Burgard. BI^{2}RRT*: An efficient sampling-based path planning framework for task-constrained mobile manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, 2016. doi: 10.1109/IROS.2016.7759547.

[17] F. Burget, L. D. J. Fiederer, D. Kuhner, M. Völker, J. Aldinger, R. T. Schirrmeister, C. Do, J. Boedecker, B. Nebel, T. Ball, and W. Burgard. Acting thoughts: Towards a mobile robotic service assistant for users with limited communication skills. In *European Conference on Mobile Robots (ECMR)*, pages 1–6, Sept 2017. doi: 10.1109/ECMR.2017.8098658.

[18] F. Campos and J. Calado. Approaches to human arm movement control—a review. *Annual Reviews in Control*, 33(1), 2009. ISSN 1367-5788. doi: http://dx.doi.org/10.1016/j.arcontrol.2009.03.001.

[19] A. Cela, J. J. Yebes, R. Arroyo, L. M. Bergasa, R. Barea, and E. Lopez. Complete low-cost implementation of a teleoperated control system for a humanoid robot. *Sensors*, 13(2), 2013.

[20] R. Chalodhorn, D. B. Grimes, K. Grochow, and R. P. N. Rao. Learning to walk through imitation. In *Int. Conf. on Artificial Intelligence (IJCAI)*, 2007.

[21] T. F. Chang and R. V. Dubey. A Weighted Least-Norm Solution Based Scheme for Avoiding Joints Limits for Redundant Manipulators. *IEEE Trans. on Robotics and Automation*, 11(2), Apr. 1995.

[22] F. Chaumette and E. Marchand. A redundancy-based iterative approach for avoiding joint limits: application to visual servoing. *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 17(5), 2001. ISSN 1042-296X. doi: 10.1109/70.964671.

[23] S. Chiaverini, O. Egeland, and R. Kanestrom. Achieving user-defined accuracy with damped least-squares inverse kinematics. In *Int. Conf. on Advanced Robotics*, 1991. doi: 10.1109/ICAR.1991.240676.

[24] S. Chiaverini, B. Siciliano, and O. Egeland. Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator. *IEEE Transactions on Control Systems Technology*, 2(2):123–134, 1994.

[25] S. Chitta, B. Cohen, and M. Likhachev. Planning for autonomous door opening with a mobile manipulator. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[26] S. Chitta, E. G. Jones, M. Ciocarlie, and K. Hsiao. Perception, planning, and execution for mobile manipulation in unstructured environments. *IEEE Robotics and Automation Magazine, Special Issue on Mobile Manipulation*, 19, 2012.

[27] S. Chitta, I. Sucan, and S. Cousins. MoveIt! [ROS topics]. *IEEE Robotics Automation Magazine*, 19(1):18 –19, 2012.

[28] S. L. Chiu. Task compatibility of manipulator postures. *Int. J. Rob. Res.*, 7(5):13–21, Oct. 1988. ISSN 0278-3649. doi: 10.1177/027836498800700502. URL `http://dx.doi.org/10.1177/027836498800700502`.

[29] C.-S. Chung, H. Wang, and R. A. Cooper. Autonomous function of wheelchair-mounted robotic manipulators to perform daily activities. In *IEEE International Conference on Rehabilitation Robotics (ICORR)*, pages 1–6, 2013.

[30] G. Cimen, H. Ilhan, T. Capin, and H. Gurcay. Classification of human motion based on affective state descriptors. *Computer Animation and Virtual Worlds*, 24(3-4):355–363, 2013.

[31] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *ArXiv e-prints*, volume 1511, page arXiv:1511.07289, 2016.

[32] J. Cortés. *Motion Planning Algorithms for General Closed-Chain Mechanisms*. PhD thesis, Institut National Polytechnique de Toulouse, Toulouse, France, 2003.

[33] S. Cotton, A. Murray, and P. Fraisse. Statically equivalent serial chains for modeling the center of mass of humanoid robots. In *Humanoids*, pages 138–144, 2008.

[34] I. A. Şucan and S. Chitta. Motion planning with constraints using configuration space approximations. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[35] R. Dale and E. Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive science*, 19(2):233–263, 1995.

[36] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki. Incremental task and motion planning: A constraint-based approach. In *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, 2016. doi: 10.15607/RSS.2016.XII.002.

[37] B. Dariush, M. Gienger, A. Arumbakkam, Y. Zhu, B. Jian, K. FujiMura, and C. Goerick. Online transfer of human motion to humanoids. *Int. Journal of Humanoid Robotics (IJHR)*, 6(2), 2009.

[38] S. Das, L. Trutoiu, A. Murai, D. Alcindor, M. Oh, O. D. L. Torre, and J. Hodgins. Quantitative measurement of motor symptoms in parkinson's disease: A study with full-body motion capture data. In *Proc. of the Int. Conf. of the IEEE Engineering in Medicine and Biology Society*, 2011.

[39] B. Day, J. Dick, and C. Marsden. Patients with parkinson's disease can employ a predictive motor strategy. *Journal of Neurology, Neurosurgery & Psychiatry*, 47(12): 1299–1306, 1984.

[40] L. De Silva, A. K. Pandey, M. Gharbi, and R. Alami. Towards combining HTN planning and geometric task planning. *CoRR*, abs/1307.1482, 2013.

[41] C. Do, T. Schubert, and W. Burgard. A probabilistic approach to liquid level detection in cups using an RGB-D camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, 2016.

[42] M. Do, P. Azad, T. Asfour, and R. Dillmann. Imitation of human motion on a humanoid robot using non-linear optimization. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2008.

[43] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel. Semantic Attachments for Domain-independent Planning Systems. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, 2009.

[44] P. Eyerich, R. Mattmüller, and G. Röger. Using the Context-enhanced Additive Heuristic for Temporal and Numeric Planning. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 130–137, 2009.

[45] T. Flash and N. Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *The Journal of Neuroscience*, 5(7):1688–1703, 1985.

[46] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999(343-349):2–2, 1999.

[47] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. *arXiv preprint arXiv:1404.2334*, 2014.

[48] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3067–3074, 2015.

[49] J. Garimort, A. Hornung, and M. Bennewitz. Humanoid navigation with dynamic footstep plans. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[50] M. Göbelbecker. Assisting with Goal Formulation for Domain Independent Planning. In *KI 2015: Advances in Artificial Intelligence*, pages 87–99. Springer, 2015. ISBN 9783319244891.

[51] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier. Mechatronic design of nao humanoid. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 769–774, 2009.

[52] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[53] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[54] L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 233–246. A.K. Peters, Wellesley, MA, 2001.

[55] Y. Hara, F. Honda, T. Tsubouchi, and A. Ohya. Detection of Liquids in Cups Based on the Refraction of Light with a Depth Camera Using Triangulation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[56] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107, 1968.

[57] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385*, 2015.

[58] M. Helmert. The Fast Downward Planning System. *Journal of Artificial Intelligence Research 26 (JAIR)*, pages 191–246, 2006.

[59] 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(3):189–206, 2013.

[60] J. Ilonen and V. Kyrki. Robust robot-camera calibration. In *15th International Conference on Advanced Robotics (ICAR)*, pages 67–74, 2011.

[61] L. Jamone, L. Natale, K. Hashimoto, G. Sandini, and A. Takanishi. Learning the reachable space of a humanoid robot: A bio-inspired approach. *IEEE RAS and EMBS Int. Conf. on Biomedical Robotics and Biomechatronics (BioRob)*, 2012. doi: 10.1109/biorob.2012.6290729.

[62] L. Janson, E. Schmerling, A. Clark, and M. Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research*, 34(7):883–921, 2015.

[63] M. Jordan and A. Perez. Optimal bidirectional rapidly-exploring random trees. Technical Report MIT-CSAIL-TR-2013-021, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, August 2013.

[64] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1470–1477, 2011.

[65] L. P. Kaelbling and T. Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013. doi: 10.1177/0278364913484072.

[66] P. Kaiser, D. Gonzalez-Aguirre, F. Schültje, J. B. Sol, N. Vahrenkamp, and T. Asfour. Extracting whole-body affordances from multimodal exploration. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2014.

[67] S. Kajita and K. Tani. Experimental study of biped dynamic walking in the linear inverted pendulum mode. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2885–2891, 1995.

[68] F. Kanehiro, E. Yoshida, and K. Yokoi. Efficient reaching motion planning and execution for exploration by humanoid robots. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[69] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[70] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the RRT*. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1478–1483, 2011.

[71] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[72] D. Kim, J. Lee, and S.-e. Yoon. Cloud RRT*: Sampling cloud based RRT*. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2519–2526, 2014.

[73] S. Kim, C. Kim, B. You, and S. Oh. Stable whole-body motion generation for humanoid robots to imitate human motions. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[74] J. Koenemann and M. Bennewitz. Whole-body imitation of human motions with a nao humanoid. In *Video Abstract Proc. of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2012.

[75] J. Koenemann, F. Burget, and M. Bennewitz. Real-time imitation of human whole-body motions by humanoids. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2806–2812, Hong Kong, China, 2014.

[76] E. Krahmer and K. Van Deemter. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218, 2012.

[77] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[78] J. J. Kuffner and S. M. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, volume 2, pages 995–1001, 2000.

[79] KUKA AG. The omniRob mobile manipulator, 2017. URL `http://www.kuka.com/en-gb`.

[80] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[81] S. M. LaValle, J. H. Yakey, and L. E. Kavraki. A probabilistic roadmap approach for systems with closed kinematic chains. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, volume 3, pages 1671–1676, 1999.

[82] S. M. LaValle, A. Yershova, M. Katsev, and M. Antonov. Head tracking for the Oculus Rift. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 187–194, 2014.

[83] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance. EEGNet: A compact convolutional network for EEG-based brain-computer interfaces. *arXiv preprint arXiv:1611.08024*, 2016.

[84] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Proceedings of the International Conference on Image Processing*, volume 1, pages I–I, 2002.

[85] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun. Anytime Dynamic A*: An Anytime, Replanning Algorithm. In *ICAPS*, pages 262–271, 2005.

[86] P. Lindes, A. Mininger, J. R. Kirk, and J. E. Laird. Grounding language for interactive task learning. In *Proceedings of the First Workshop on Language Grounding for Robotics*, pages 1–9, 2017.

[87] T. Lozano-Pérez and L. P. Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3684–3691, 2014.

[88] E. Magrini, F. Flacco, and A. De Luca. Control of generalized contact motion and force in physical human-robot interaction. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2298–2304, 2015.

[89] D. McDermott. `PDDL` - The Planning Domain Definition Language. Technical report, AIPS-98 Planning Competition Committee, 1998.

[90] K. Muelling, A. Venkatraman, J.-S. Valois, J. E. Downey, J. Weiss, S. Javdani, M. Hebert, A. B. Schwartz, J. L. Collinger, and J. A. Bagnell. Autonomy infused teleoperation with application to brain computer interface controlled manipulation. *Autonomous Robots*, pages 1–22, 2017. ISSN 1573-7527. doi: 10.1007/s10514-017-9622-4.

[91] J. Müller, U. Frese, and T. Röfer. Grab a mug - Object detection and grasp motion planning with the Nao robot. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2012.

[92] S. Nakaoka, A. Nakazawa, K. Yokoi, H. Hirukawa, and K. Ikeuchi. Generating whole body motions for a biped humanoid robot from captured human dances. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2003.

[93] S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa, H. Hirukawa, and K. Ikeuchi. Learning from observation paradigm: Leg task models for enabling a biped humanoid robot to imitate human dances. *Int. Journal of Robotics Research (IJRR)*, 26 (8), 2007.

[94] J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, and M. S. Muhammad. RRT*-SMART: A rapid convergence implementation of RRT*. *International Journal of Advanced Robotic Systems*, 10, 2013.

[95] C. Ott, D. Lee, and Y. Nakamura. Motion capture based human motion recognition and imitation by direct marker control. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2008.

[96] M. Otte and N. Correll. C-FOREST: Parallel shortest path planning with superlinear speedup. *IEEE Transactions on Robotics*, 29(3):798–806, 2013.

[97] 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 & Automation (ICRA)*, 2012.

[98] D. Park, Y. K. Kim, Z. M. Erickson, and C. C. Kemp. Towards assistive feeding with a general-purpose mobile manipulator. *arXiv:1605.07996*, 2016.

[99] F. Paus, P. Kaiser, N. Vahrenkamp, and T. Asfour. A combined approach for robot placement and coverage path planning for mobile manipulation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[100] K. Pauwels and D. Kragic. Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1300–1307, 2015.

[101] J. Quinlan. Induction of decision trees. *Machine Learning 1*, pages 81–106, 1986.

[102] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 802–807, 1993.

[103] F. Rahimi, C. DuVal, M. Jog, C. Bee, A. South, M. Jog, R. Edwards, and P. Boissy. Capturing whole-body mobility of patients with parkinson disease using inertial motion sensors: Expected challenges and rewards. In *Prof. of the Int. Conf. of the IEEE Engineering in Medicine and Biology Society*, 2011. doi: 10.1109/IEMBS.2011.6091443.

[104] M. Riley, A. Ude, K. Wade, and C. G. Atkeson. Enabling real-time full body imitation: A natural way of transferring human movements to humanoids. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2003.

[105] D. Roetenberg, H. Luinge, and P. Slycke. Xsens MVN: Full 6DOF human motion tracking using miniature inertial sensors. Technical report, xsens, 2009.

[106] C. Röhrig, D. Heß, and F. Künemund. Motion controller design for a mecanum wheeled mobile manipulator. In *IEEE Conference on Control Technology and Applications (CCTA)*, pages 444–449, 2017.

[107] T. Rühr, J. Sturm, D. Pangercic, M. Beetz, and D. Cremers. A generalized framework for opening doors and drawers in kitchen environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[108] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[109] R. B. Rusu, W. Meeussen, S. Chitta, and M. Beetz. Laser-based perception for door and handle identification. In *International Conference on Advanced Robotics (ICAR)*, pages 1–8, 2009.

[110] A. Safonova, N. Pollard, and J. K. Hodgins. Optimizing human motion for the control of a humanoid robot. In *Int. Symp. on Adaptive Motion of Animals and Machines (AMAM)*, 2003.

[111] T. N. Sainath, B. Kingsbury, G. Saon, H. Soltau, A.-r. Mohamed, G. Dahl, and B. Ramabhadran. Deep Convolutional Neural Networks for Large-scale Speech Tasks. *Neural Networks*, 64:39–48, 2015. ISSN 0893-6080. doi: 10.1016/j.neunet.2014.08.005.

[112] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw. BCI2000: A general-purpose brain-computer interface (BCI) system. *IEEE Transactions on biomedical engineering*, 51(6):1034–1043, 2004.

[113] D. Schinstock, T. Faddis, and R. Greenway. Robust inverse kinematics using damped least squares with dynamic weighting. Technical report, NASA, 1994.

[114] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball. Deep learning with convolutional neural networks for brain mapping and decoding of movement-related information from the human eeg. *arXiv:1703.05051*, 2017.

[115] J. Scholz, S. Chitta, B. Marthi, and M. Likhachev. Cart pushing with a mobile manipulation system: Towards navigation with moveable objects. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 6115–6120, 2011.

[116] S. Schröer, I. Killmann, B. Frank, M. Völker, L. D. J. Fiederer, T. Ball, and W. Burgard. An autonomous robotic assistant for drinking. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6482–6487, 2015.

[117] SCHUNK GmbH & Co. KG. Servo-electric 3-finger gripper hand SDH, 2017. URL `http://schunk.com/us_en/gripping-systems`.

[118] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. LeCun. Very deep multilingual convolutional neural networks for LVCSR. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4955–4959, 2016. doi: 10.1109/ICASSP.2016.7472620.

[119] M. Shridhar and D. Hsu. Grounding spatio-semantic referring expressions for human-robot interaction. *CoRR*, abs/1707.05720, 2017.

[120] B. Siciliano and O. Khatib, editors. *Springer Handbook of Robotics*. Springer, Berlin, Heidelberg, 2008. ISBN 978-3-540-23957-4. URL `http://dx.doi.org/10.1007/978-3-540-30301-5`.

[121] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition, 2008. ISBN 1846286417, 9781846286414.

[122] R. Smith. Open dynamics engine, 2008. URL `http://www.ode.org/`. http://www.ode.org/.

[123] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

[124] R. Smits. KDL: Kinematics and Dynamics Library. `http://www.orocos.org/kdl`.

[125] SoftBank Robotics. Discover Nao, the little humanoid robot from Aldebaran. Accessed 4 August 2016.

[126] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 639–646, 2014.

[127] C. Stanton, A. Bogdanovych, and E. Ratanasen. Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning. In *Proc. of the Australasian Conf. on Robotics and Automation (ACRA)*, 2012.

[128] M. Stilman. Global manipulation planning in robot joint space with task constraints. *IEEE Transactions on Robotics*, 26(3):576–584, 2010.

[129] S. Stober. Learning Discriminative Features from Electroencephalography Recordings by Encoding Similarity Constraints. In *Bernstein Conference 2016*, 2016. doi: 10.12751/nncn.bc2016.0223.

[130] F. Stulp, A. Fedrizzi, L. Mösenlechner, and M. Beetz. Learning and reasoning with action-related places for robust mobile manipulation. *Journal of Artificial Intelligence Research (JAIR)*, 43:1–42, 2012.

[131] W. Suleiman, E. Yoshida, F. Kanehiro, J.-P. Laumond, and A. Monin. On human motion imitation by humanoid robot. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008.

[132] Y. R. Tabar and U. Halici. A novel deep learning approach for classification of EEG motor imagery signals. *Journal of Neural Engineering*, 14(1):016003, 2017. ISSN 1741-2552. doi: 10.1088/1741-2560/14/1/016003.

[133] E. Todorov and M. I. Jordan. Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements. *Journal of Neurophysiology*, 80(2):696–714, 1998.

[134] A. Ude, C. Atkeson, and M. Riley. Programming full-body move- ments for humanoid robots by observation. In *Robotics and Autonomous Systems*, 2004.

[135] University of North Carolina. PQP: A proximity query package. GAMMA Research Group, Available from http://www.cs.unc.edu/∼geom/SSV/, 2005.

[136] N. Vahrenkamp and T. Asfour. Representing the robot's workspace through constrained manipulability analysis. *Autonomous Robots*, 38(1), 2015.

[137] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini, and R. Dillmann. Manipulability analysis. In *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 568–573, 2012.

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

[139] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2001.

[140] M. Völker, R. T. Schirrmeister, L. D. J. Fiederer, W. Burgard, and T. Ball. Deep transfer learning for error decoding from non-invasive EEG. In *6th International Winter Conference on Brain-Computer Interfaces*, 2018.

[141] R. Vuga, M. Ogrinc, A. Gams, T. Petric, N. Sugimoto, A. Ude, and J. Morimoto. Motion capture and reinforcement learning of dynamically stable humanoid movement primitives. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[142] M. Vukobratovic and B. Borovac. Zero-moment point – thirty five years of its life. *Int. Journal of Humanoid Robots*, 1, 2004.

[143] A. Wang, J. Ramos, J. Mayo, W. Ubellacker, J. Cheung, and S. Kim. The HERMES humanoid system: A platform for full-body teleoperation with balance feedback. In *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 730–737, 2015.

[144] C. Wang, B. Xia, J. Li, W. Yang, D. Xiao, A. C. Velez, and H. Yang. Motor imagery BCI-based robot arm system. In *Seventh International Conference on Natural Computation (ICNC)*, volume 1, pages 181–184, 2011.

[145] K. Yamane and J. Hodgins. Controlling humanoid robots with human motion data: Experimental validation. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2010.

[146] K. Yamane and J. Hodgins. Control-aware mapping of human motion data with stepping for humanoid robots. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[147] Y. Yang, V. Ivan, Z. Li, M. Fallon, and S. Vijayakumar. iDRM: Humanoid motion planning with realtime end-pose selection in complex environments. In *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 271–278, 2016.

[148] T. Yoshikawa. Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, 4(2):3–9, 1985.

[149] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg. Modeling context in referring expressions. *CoRR*, 2016.

[150] F. Zacharias, C. Borst, and G.Hirzinger. Capturing robot workspace structure: representing robot capabilities. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[151] F. Zhou, F. De la Torre Frade, and J. K. Hodgins . Aligned cluster analysis for temporal segmentation of human motion. In *Proc. of the IEEE Conf. on Automatic Face and Gestures Recognition*, 2008.

[152] F. Zhou, F. De la Torre Frade, and J. K. Hodgins . Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(3):582–596, 2013.