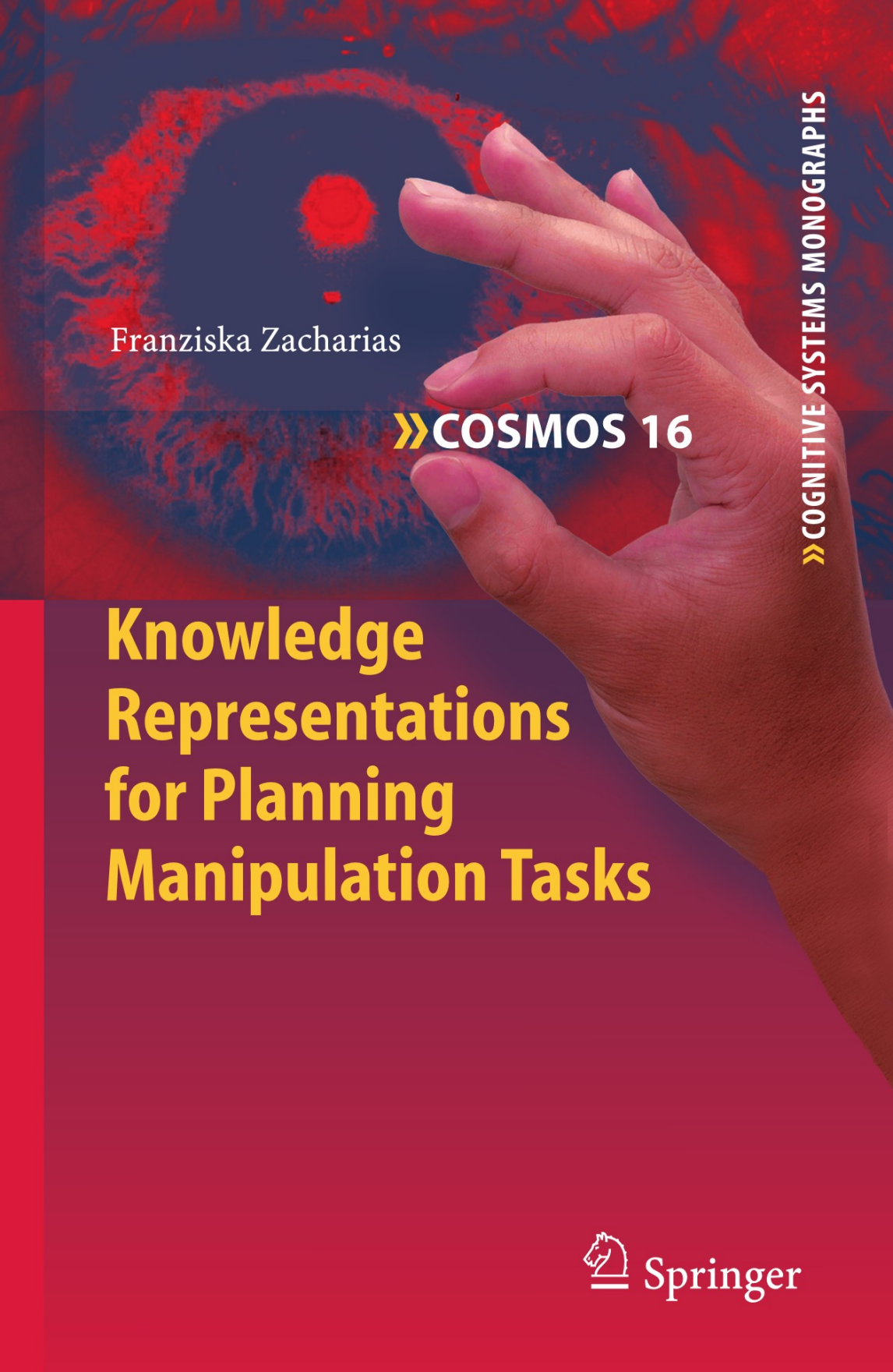


Franziska Zacharias

»COSMOS 16

»COGNITIVE SYSTEMS MONOGRAPHS

Knowledge Representations for Planning Manipulation Tasks

Springer

Cognitive Systems Monographs

Volume 16

Franziska Zacharias

Knowledge Representations for Planning Manipulation Tasks

Springer

Rüdiger Dillmann, University of Karlsruhe, Faculty of Informatics, Institute of Anthropomatics, Humanoids and Intelligence Systems Laboratories, Kaiserstr. 12, 76131 Karlsruhe, Germany

Yoshihiko Nakamura, Tokyo University Fac. Engineering, Dept. Mechano-Informatics, 7-3-1 Hongo, Bukyo-ku Tokyo, 113-8656, Japan

Stefan Schaal, University of Southern California, Department Computer Science, Computational Learning & Motor Control Lab., Los Angeles, CA 90089-2905, USA

David Vernon, Khalifa University Department of Computer Engineering, PO Box 573, Sharjah, United Arab Emirates

Author

Dr. Franziska Zacharias

DLR German Aerospace Center
Institute of Robotics and Mechatronics
P.O. Box 1116
82230 Wessling
Germany
E-Mail: Franziska.Zacharias@dlr.de

ISBN 978-3-642-25181-8 e-ISBN 978-3-642-25182-5

DOI 10.1007/978-3-642-25182-5

Cognitive Systems Monographs ISSN 1867-4925

Library of Congress Control Number: 2011940775

Typeset by Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed in acid-free paper

5 4 3 2 1 0

springer.com

Abstract

A complex manipulation task for a robot has to be planned on different levels of abstraction before it can be executed on a real system. At a high-level of abstraction, a task planner generates a sequence of subtasks from a given abstract command, its pre- and postconditions and a current world model. To restrict the already huge search space of the task planner, aspects like geometric information are not contained in the world model at this level. The task planner then assigns the subtasks to planners on a lower level of abstraction like path planners or grasp planners. These planners rely on geometric information to compute their solutions. However, without considering geometric information the task planner cannot determine good parameters for these low-level planners. Therefore, the derived plans may not be valid for a given situation and may require backtracking.

Knowledge representations can bridge the gap between these planning levels and thereby enable scene reasoning. The kinematic capabilities of a robot are one aspect the task planner has to be aware of. Therefore this book introduces a novel general representation of the kinematic capabilities of a robot arm. The *versatile workspace* is defined to describe in which orientations the end effector attached to a robot arm can reach a position. The *capability map* is a calculable representation of the versatile workspace that allows to determine how well regions of the workspace are reachable. It accurately represents the versatile workspace of arbitrary arm kinematics and enables autonomous task-specific reasoning. Furthermore, its visualization scheme is intuitively comprehensible for the human.

The versatile applicability of the capability map is shown by examples from several distinct application domains. The workspace of several robot arms is visualized and analyzed. It is shown how the capability map can be used to compare the abilities of different robot arms. In human-robot interaction, the capability map is used to objectively evaluate a bi-manual interface for tele-operation. The results can also supplement the design process for humanoid robot design. In low-level geometric planning, the capability map is used to reduce the search space and to parameterize path planners and grasp planners. Thus, in a manipulation task for a humanoid robot more human-like motion is planned while simultaneously the computation time is reduced. In high-level task reasoning, the capability map is used to evaluate how

well a robot is suited for a task. In an example a humanoid robot has to perform a task involving 3D trajectories. Regions are extracted from the capability map that permit the task execution and the suitability of the robot is inferred.

Acknowledgments

This book was written during my employment at the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR) in Oberpfaffenhofen, Germany. First of all, I would like to thank the Head of the Institute, Prof. Gerd Hirzinger, for his support and for his great efforts in providing the best possible research conditions. I would like to thank Prof. Michael Beetz from the Intelligent Autonomous Systems (IAS) group at the Technical University of Munich for his continuous support and advice. My special thanks go to my colleagues at the DLR for providing such a nice working atmosphere.

The research covered by this book has been partially funded by the EC Seventh Framework Programme (FP7) under grant agreement no. 216239 as part of the IP DEXMART.

Wessling, September 2011 Franziska Zacharias

Contents

Chapter 1
Introduction

Current humanoid robots are mostly deployed in laboratory environments. However they are envisioned as helpful assistants in future households that e.g. enable elderly citizens to live independently in their homes by supporting them in their daily life. Possible service tasks are clearing the dish washer or setting the table. In this book a model of a robot arm's workspace is developed. It is used to analyze the scene and a robot's capabilities. In planning processes it serves as a source of information that supports the decision process.

1.1 Problem Statement

A human learns during his life to use his arms as well as to grasp and use tools and objects. He relies on knowledge about the world and about himself to decide which regions are reachable, how to approach objects, which places or objects are easily reached and which are difficult to approach. In the survey by Kawato [50] the existence of internal models for human motor control is reported. It can be assumed that the human also relies on knowledge representations in a variety of other contexts, in particular when determining the next action to be performed. In computer science, knowledge representations model information for the use in knowledge-based systems or planning systems. A model is an image of the reality. However, the reality is not represented in every detail. Rather, a compact representation of the information is desired.

In the household, humanoid robots are envisioned to relieve the human of repetitive or annoying tasks. Therefore, humanoid robots have to be able to perform a variety of tasks autonomously. This requires that the knowledge of how to perform a certain task is provided to the robot. The robot can learn from the human [122], [18]. The human demonstrates the task and the robot observes the execution. The robot extracts the important aspects and imitates the human. However, determining the essential aspects that enable accomplishing a task and segmenting the task is difficult. Furthermore, the robot and the human kinematics differ. Therefore the observed task cannot be directly mapped onto the robot. Due to these issues gener-

F. Zacharias: Knowledge Representations for Planning Manipulation Tasks, COSMOS 16, pp. 1–6.
springerlink.com

alization to previously unseen tasks and objects is complicated. However, it is important that a robot can cope with new situations. Therefore, an alternative approach is possible that involves various planner types. The planners determine grasps or provide collision-free movement for the robot. The different planners have to be coordinated to successfully realize a given service task.

In general a service task like setting the table can be further resolved into several subtasks. When objects have to be grasped, choosing grasps and approach directions are problems that have to be solved. In Figure 1.1 several service tasks and potential challenges are illustrated. Objects have to be moved from one location to another. If a plate should be positioned on the table, the question is at which exact position to place it (Figure 1.1 (a)). A number possibilities for grasping a glass are shown in Figure 1.1 (b) by three different arm and hand configurations. Closets, a refrigerator or a microwave have to be opened to be able to insert or extract an object (Figure 1.1 (c)). For these tasks, the robot has to decide where to place itself to be able to reach

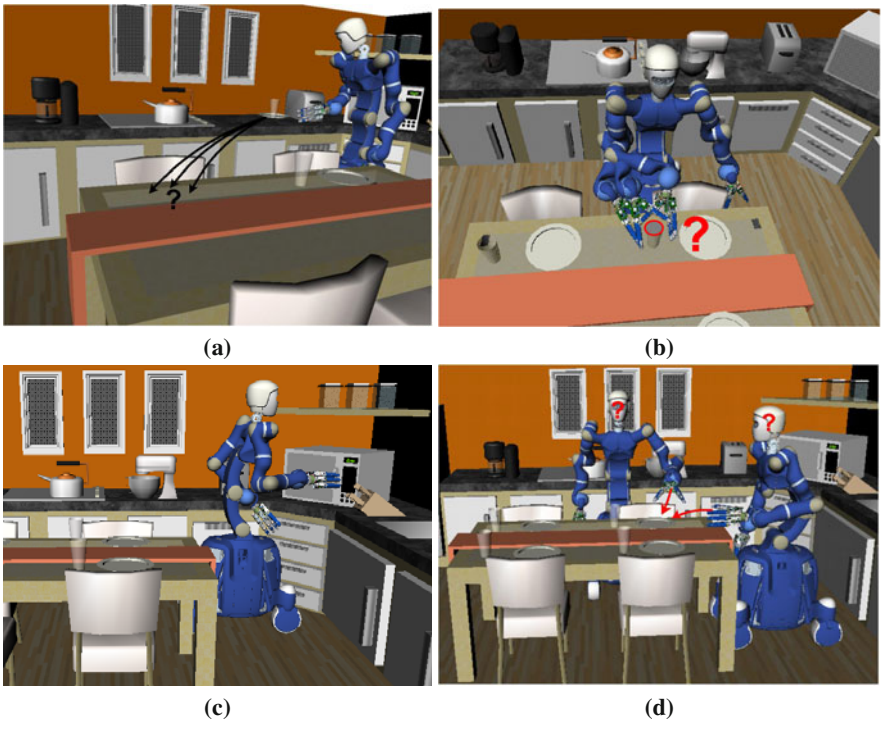


Fig. 1.1 Illustration of several challenges during the planning of a service task. A humanoid robot is shown in a kitchen performing several tasks. **(a)** The position where the plate and the glass are to be placed is not exactly defined. **(b)** With a multi-fingered hand, a glass can be grasped or approached in different ways. **(c)** The robot has placed itself to open the microwave. **(d)** Two robot placements are shown for grasping the plate on the table. (Kitchen model source [32])

an object. However, the number of robot placements is infinite. Figure 1.1 (d) shows two possible placements of a robot for grasping a plate. These issues have to be addressed by a task planner when planning a task.

Logical planners (also called task planners) are expected to divide a task into a set of subtasks, e.g. first the closet is opened, then the dishes are taken out of the closet, the closet is closed and the dishes are transported to the table. Path planners are used for moving the robotic arm without collisions between two positions. A grasp planner provides a good grasp for handling an object. A robot placement planner positions a robot for performing a grasp or a trajectory. The logical planner has to trigger the execution of the subtasks by appropriately using the low-level planners, i.e. the path planner, the grasp planner and the robot placement planner. The mapping from the high-level abstraction layer of the task planner onto the low-level planners is illustrated in Figure 1.2. Depending on the parameterization, e.g. the start and the goal robot arm configuration, a low-level planning problem is not always solvable. Furthermore, a solution is possibly unavailable because no collision-free path is found or because the object cannot be grasped from the queried direction.

A logical planner, however, has no knowledge about the geometry of the scene and works with an abstract scene model where objects are represented by labels. It does not know e.g. from which direction an object can be approached best. Therefore a chosen subtask may not be executable. Furthermore, a low-level planner cannot provide information concerning the reason for its failure. For high-dimensional

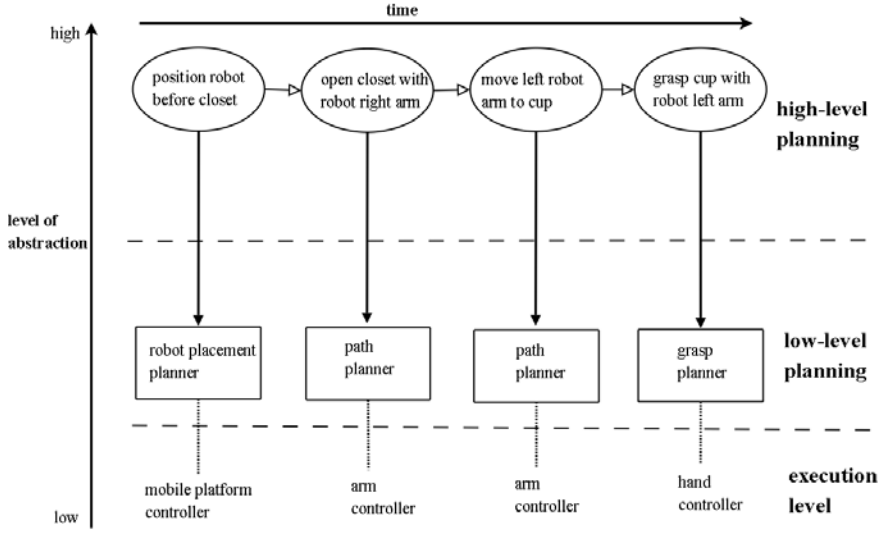


Fig. 1.2 Different levels of abstraction during the planning and execution of a service task are shown. The task of fetching a cup from a closet is used as an example. High-level planning is equivalent with task planning. Possibly parallel running actions are mapped to low-level planners. The planning results are then executed by divers robot controllers.

planning problems determining whether a solution exists is computationally too expensive. A brute force approach lets the logical planner propose a plan and test whether the plan is valid. In this process the low-level planners try to find a solution for assigned subtasks. This is repeated until a solution is found or a termination criterion is met. Since each of the low-level planning problems is already very complex, even simple tasks can take a long time to be solved.

Furthermore, for objects like a coffee mug, directions exist from which the object is better graspable. To open a closet door or a dish washer not every placement of the humanoid robot results in successful task execution. The humanoid robot may place itself so that it can grasp the door handle. However, from that position the trajectory for opening the door is not possible.

In the reachable workspace volume of the robot arm, positions can be reached in *at least one* orientation. In the dexterous workspace volume positions can be reached in *all* orientations [19]. However, in general seldom all orientations are needed. In this work, the so-called *versatile workspace* of a robot indicates with which orientations a position can be reached. A representation of the versatile workspace for a robot arm can be exploited by high-level and low-level planner types. A suitable representation of the versatile workspace enables a task planner to predict whether an object is graspable or whether a certain trajectory is executable for the robot. This information helps to estimate whether the proposed plan is executable without triggering the low-level planners. Given a set of grasps for an object and a scene description, the representation can be used to estimate the difficulty of the planning problem. For instance, if a lot of grasps are unreachable it is possible that the scene is very crowded and the target object is difficult to reach. This information can be used by the task planner to e.g. consider a rearrangement of the scene to make the planning problem easier, or to reposition the robot in the scene.

Therefore this book provides a representation of the versatile workspace of a robot arm that contains information that can be used on different planning levels. Using the knowledge representation the search is focused on promising regions, good parameterizations for planners are determined or the search space is reduced. The representation of the versatile workspace of robotic arms describes how well regions of the workspace are reachable. A visualization of the versatile workspace facilitates analysis and interactive planning. The representation is used to guide planning processes, make reliable predictions about the feasibility of tasks and avoid unsuccessful planning runs. The developed model is used to parameterize path and grasp planners and enable them to solve a problem more quickly. The generation of the model is performed offline. However, fast access to the data saved in the representation allows usage in online algorithms. In Figure 1.3, a knowledge base contains the representation of the versatile workspace and information about the environment. In future robot operating systems a knowledge base could be used by task planners, path planners and grasp planners. Furthermore, the low-level planners could be used by the high-level planner.

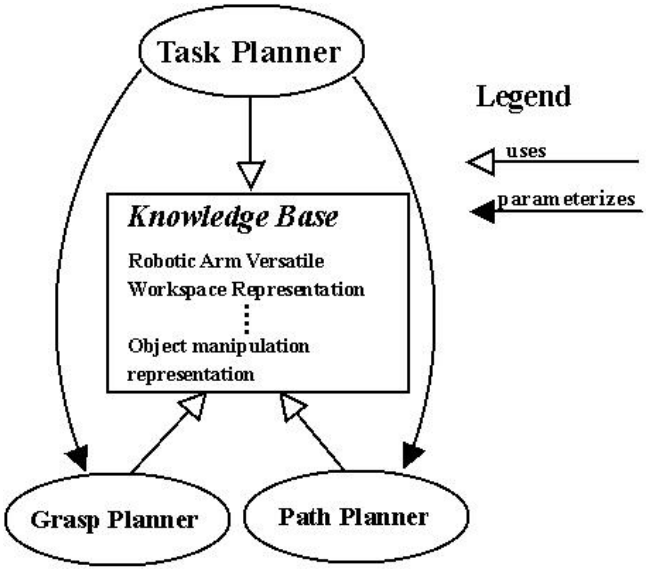


Fig. 1.3 Knowledge about the world and about the robot is used to guide and parameterize planning processes at different levels of abstraction. A knowledge base is used by task planners, path planners and grasp planners. Furthermore, the low-level planners are used by the high-level planner.

1.2 Contributions of the Book

In this book, the capability map, a model of the versatile robot arm workspace is developed. The focus is on manipulators with serial kinematics but the model is not limited to these manipulator types. This book shows how knowledge about the versatile workspace of robot arms is exploited in planning and evaluation processes. The use of the capability map makes search processes much more effective. It furthermore helps to make reliable predictions about the feasibility of a task.

The contributions of this book are summarized in the following:

1. Multiple models for representing the workspace are compared. Strengths and weaknesses of these models are discussed. Their suitability for the representation of a robot arm's versatile workspace is evaluated.
2. The **capability map**, a representation of the versatile robot arm workspace, is developed. In contrast to current state of the art workspace representations, the capability map represents position and orientation information.
3. An intuitive visualization scheme is provided for the capability map. It allows the detailed visual inspection and analysis of the kinematic capabilities of the robot arm in the workspace.

4. The capability map permits the functional comparison of arm kinematics. To evaluate different setups for bi-manual human-robot interaction, the workspaces of the human arm and a robot arm are compared.
5. The capability map is used in planning tasks such as path planning and robot positioning for the humanoid robot *Justin* and other systems. In robot placement planning, the search is focused and solution regions containing valid robot positions are extracted. In path planning, the planned robot motion appears more human-like and computation time is reduced.

1.3 Outline of the Book

First, current research results with respect to the presented problem formulation are examined in the **state of the art** (Chapter 2). A focus is placed on whether and how knowledge encapsulated in models is currently used in planning. **Criteria and performance indices** used in robot design are analyzed in Chapter 3. It is examined whether they can be used to derive a model of the robot arm's versatile workspace and thus describe the robot arm's capabilities. As a conclusion of this chapter, requirements for a model of the kinematic capabilities of a robot arm are derived. The **capability map** proposed in this book is developed in Chapter 4 using the identified requirements. Information that planning processes need is ensured to be contained in the model. Information access is fast. A concept for visualizing the capability map is also provided. The dependencies of the capability map on parameters and their choice is discussed. The **usability** of the model **for visualization and setup evaluation** is presented in Chapter 5. In Chapter 6 the capability map is **applied in planning tasks**. This book is concluded with Chapter 7. An outlook on future research directions is presented.

Chapter 2
Review of the Literature

In this chapter, the technical terms relevant for this work are introduced. Furthermore, on different levels of abstraction, components are described that are necessary for a service robot to solve manipulation tasks. The state of the art in high-level logical planning, also called task planning, is analyzed. Here the focus is on robotic manipulation problems. In the subsequent sections, it is analyzed how low-level planners like path planners, grasp planners and robot placement planners are used to solve the subtasks involved in manipulation, e.g. moving to an object, grasping it, and transporting it to a different position. It is examined whether knowledge representations are used to speed up or facilitate planning processes. Furthermore, it is outlined how the high-level and low-level planning can benefit from the use of knowledge representations.

2.1 Robotics Technical Terms

Since the 1980's, a robot is understood to be a system that is able to move in the environment and/or a system able to manipulate objects in the environment. This book is mainly concerned with robot arms. The terms robot arms and robots are used synonymously. There are robots with parallel or serial kinematics. Some robots are mobile. Robot arms that are mounted on a mobile platform are called mobile manipulator. Actuators move the mechanical components of the robots. Using sensors, the environment and state of the robot itself is perceived [100]. Today, the most widely used robots are industrial manipulators (Figure 2.1 (a)), also called robotic manipulator arms, or just manipulators. In this book the focus lies on manipulators with serial kinematics, i.e. with a serial chain of links. The maximum and minimum positions that are possible for a link are defined by its link limits. The number of links a manipulator has, defines its degrees of freedom (DOF). Furthermore, each link spans one axis of a multi-dimensional space, the configuration space or C-Space of the robot [61]. Together the links move a tool attached to the end of the robot, the so-called end-effector. The reference coordinate system attached to the tool is the tool center point coordinate system, or TCP in short. In general, a coordinate

F. Zacharias: Knowledge Representations for Planning Manipulation Tasks, COSMOS 16, pp. 7–26.
springerlink.com

system is also referred to as a frame. The TCP is defined as the coordinate system T_{TCP}^{Base}, where *Base* defines the robot arm base coordinate system. It is reached by a homogeneous transformation T_{TCP}^{n} (Appendix Chapter B.1) multiplied onto the last link's (n-th) coordinate system. It is the reference frame for manipulation tasks. For a robot with n DOF in a configuration q, the pose of the TCP is computed by the forward kinematics $K(q)$ [19].

$$K(q) = T_{TCP}^{Base} = T_0^{Base} \cdot T_1^0(q_0) \cdot T_2^1(q_1) \cdot T_3^2(q_2) ... T_n^{n-1}(q(n-1)) \cdot T_{TCP}^n \quad (2.1)$$

Using the link's DH-parameters [19] the individual link transformation matrices T_i^{i-1} can be computed. The link transformations are multiplied to compute the transformation T_{TCP}^{Base} that relates the TCP frame to the base coordinate system of the robot. Thus, the frame T_{TCP}^{Base} describes the TCP pose in the base coordinate system. If the TCP should be positioned at a pose T_{TCP}^{Base}, the inverse kinematics determines whether a configuration q exists to reach the pose.

Currently, robots are mostly used in industrial applications to automate factory processes. In the future, service robots may become part of every day life. Service robots could work as the human's assistant that performs repetitive tasks in the household.

2.2 Robots in Industrial Applications

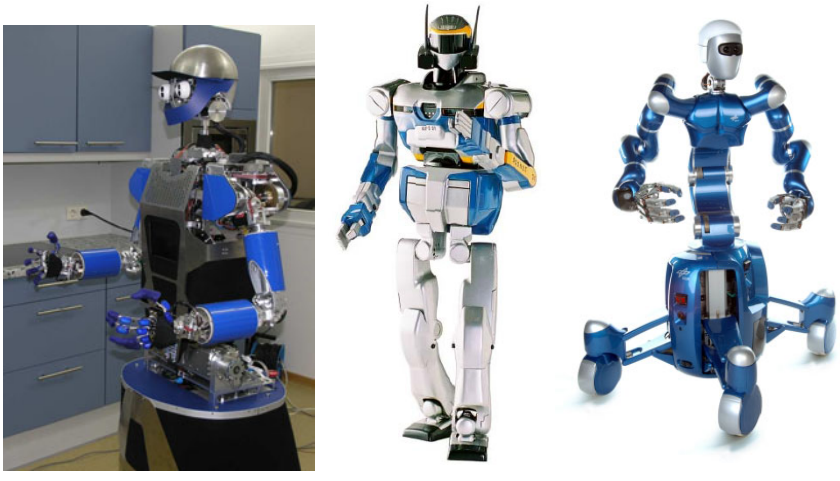
In an industrial application, a task is a sequence of steps the manipulator has to perform to obtain a specific result, e.g. two pieces of metal have to be welded together. An industrial task is precisely defined. For the robot arm, each step in the task is specified in terms of exact positions or velocities. A trajectory is a sequence of steps that describe the movement of the end-effector of the manipulator. The human operator uses a teach-in panel to define each step of the task [83] by controlling the Cartesian pose of the TCP or by controlling an individual axis of the robot. The strength of industrial manipulators lies in repetitive, fast and accurate task execution. Special tools are attached to the robot manipulator to perform welding or gluing tasks. In general, the tool is chosen according to the task and the manipulator is responsible for the execution of the given trajectory.

2.3 Service Robots

Increasingly, mobile systems like the Kärcher Saugbot (Figure 2.1 (b)), an autonomous vacuum cleaner, or the mobile manipulator youbot, a manipulator attached to a mobile platform (Figure 2.1 (c)), become commercially available and begin to find their way into everyday life or industrial applications. However in the longterm, humanoid robots (Figure 2.2) are envisioned as the human's helpful companion. Humanoid robots are designed after the model of the human. They resemble a human in their build, appearance and/or their abilities. A complete humanoid robot is equipped with two legs, two arms with hands or grippers, an upper

(a) **(b)** **(c)**

Fig. 2.1 Several robots are shown that are commercially available. **(a)** The Kuka KR 16 industrial manipulator [59], **(b)** the saugBot by Kärcher [48], **(c)** the youbot by Kuka [58].



(a) **(b)** **(c)**

Fig. 2.2 Several humanoid robots are shown. **(a)** Armar III by KIT [5], **(b)** HRP2 by Kawada Industries [46], **(c)** Rollin' Justin by DLR [29].

body and a head. A humanoid robot could serve as as a household assistant [7], that e.g. supports elderly citizens in their homes [93]. Compared to tasks in an industrial scenarios, tasks in the household are not precisely defined and are often given as a natural language description like *put the dishes on the table*. In the task of setting

the table, it is not defined which dishes are to be put at which exact position on the table or in which time and with what velocity the task should be performed. The humanoid robot has a lot of possibilities concerning how a task can be performed. And exactly that is the challenge when a humanoid robot has to perform tasks in the household autonomously.

2.4 Solving Complex Tasks Autonomously

A complex task is considered to be a task that consists of a number of subtasks which possibly are further decomposable into another set of subtasks. As Figure 1.2 shows several levels of abstraction are involved in planning a manipulation task. The subtasks need different planner types to be successfully fulfilled. In Figure 2.3, the example of setting the table is used to illustrate possible components of a complex task. Dishes have to be extracted from closets, transported to the table and placed in their designated locations.

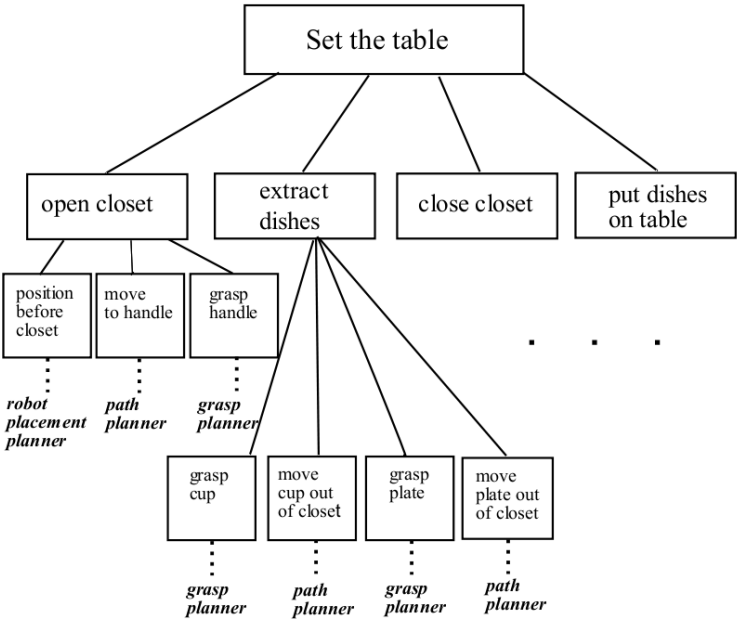


Fig. 2.3 A complex task and its subdivision into subtasks enclosed by rectangles. The task is decomposed by a task planner until elementary subtasks can be defined. The elementary subtasks are mapped to low-level planners shown in italic. The low-level planners compute trajectories or poses for a robotic arm or hand.

2.4.1 Overview of Planner Types

The task planner decomposes a high-level task into a set of possibly parallel executed subtasks that achieve the goal situation, e.g. to set the table. A task planner is given a description of the initial situation, e.g that the table is empty and all dishes are in the closet. It is also given a number of goal specifications describing the goal state to be achieved, e.g. the table is set for a number of persons. The task planner has knowledge about the structure of the problem domain, i.e. the prerequisites and results of actions [30]. It has however no knowledge about the scene geometry, i.e. whether objects are reachable. To solve the individual subtasks, the task planner triggers low-level planners. A path planner computes collision free paths. A grasp planner provides grasps for handling objects. And a robot placement planner positions a robot for a task. These low-level planners use the exact geometric information of the scene to solve the task, e.g. to grasp a cup, or to move a cup to a different location. It is a challenge to combine the different planner types which work on different levels of abstraction to achieve the task goal.

2.4.2 Planning Benchmarks

The domains of task planning, path planning and grasp planning each represent an area of active research and therefore have separate benchmarks. The task planners measure their quality e.g. in planning and scheduling competitions. The focus is on knowledge engineering technology and not on solving robotics problems. Tools, translators and planning techniques are evaluated. They are provided with a model in a domain independent language and output a solver-ready problem-specific domain model [12] for the underlying automated task planning system. The main focus lies on the improvement of input and output languages, planning and search algorithms, and heuristic interfaces.

The path planners especially strive to excel at solving narrow passage problems like the Alpha puzzle problem proposed by Amato et al. [4] or the piano mover's problem proposed by Reif [91]. In both cases a single 3D rigid object can rotate and translate in a 3D environment. The task of the path planner is to compute a path that brings the object from its start to its goal pose successfully navigating tight spaces, the narrow passages.

2.4.3 Household Tasks

In contrast a humanoid robot performs household tasks in a real physical world. Manipulation problems on the table rarely encompass narrow passages where the robot arm has to move through tight spaces to get to its destination [116]. Thus the benchmarks in the individual planning research areas do not address the specific problems one faces in the realization of an autonomous humanoid robot working in the household. Here, one of the main issues is the variability of the problem specification, the possible solutions and the associated huge search space for the

different planner types. To solve tasks like setting the table all planner types are needed and they must interact. Therefore, the state of the art in high-level and low-level planning is presented in the next sections.

2.5 High-Level Planning

In this section it is examined how task planners work in general and how they are currently used for solving manipulation problems. State of the art task planning is presented mainly with respect to its input, its output and the information it needs in the decision process. Since the logical deduction mechanisms and languages involved in deriving a plan are not relevant for this book, they are only roughly outlined.

2.5.1 Classical Task Planning

For solving a particular problem, a task planner takes as input the problem specification and knowledge about its domain. Most planning systems rely on the model of state-transition systems [30]. Such a system is denoted by $\Sigma = (S, A, \gamma)$. States $S = \{s_1, s_2, ..\}$ describe the current situation. Through actions $A = \{a_1, a_2, ..\}$ the transition to new states is triggered. Transitions are described by a state-transition function $\gamma : S \times A \rightarrow 2^S$. The action a is applicable in state s, if a is an action and $\gamma(s, a)$ is not empty. In Figure 2.4, a sequence of transitions is illustrated for the task of extracting a cup from a closet. The ellipses represent states of the world. The arrows are labeled with actions which cause transitions between states.

The situation calculus [75] is a predicate calculus formalization of states, actions and the effect of actions on states. It is a planning language for representing a

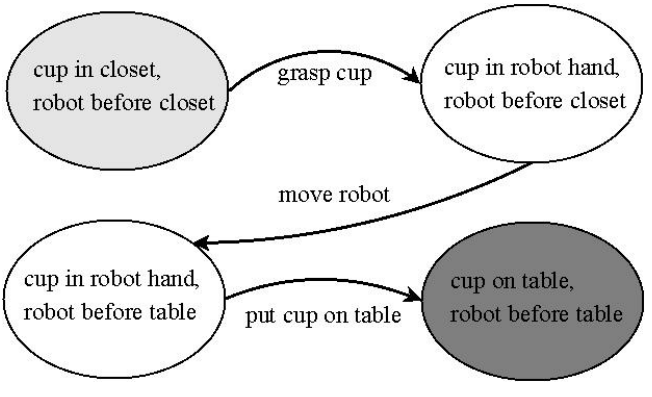


Fig. 2.4 The task of extracting a cup from a closet and putting it on the table is illustrated. The ellipses represent states of the world. The arrows are labeled with actions which cause transitions between states. The initial state is light gray. The final state is dark gray.

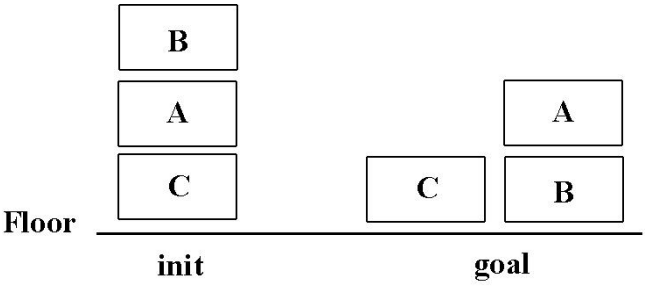


Fig. 2.5 Labeled blocks are shown at two positions on the floor. The task planner has to determine an action sequence that transform the initial state into the goal state.

domain and reason in this domain. Resolution theorem proving is applied to solve the planning problem. This means that iteratively resolution rules are applied to determine whether a propositional formula is satisfiable, i.e. resolves to true or false. If a proof can be found, a plan can be derived thereof. In Figure 2.5, the initial and the goal states of a block stacking task are shown. In Equation 2.2, the initial state of the block stacking task is specified using situation calculus. $On(B,A,s_0)$ describes that in state s_0 block B is on block A and $Clear(B,s_0)$ means that the place on block B is free.

$$\begin{aligned} &On(B,A,s_0) \wedge On(A,C,s_0) \wedge On(C,Floor,s_0) \\ &\wedge Clear(B,s_0) \wedge Clear(Floor,s_0) \end{aligned} \tag{2.2}$$

A well known planning system that relies on deduction is the STRIPS system [27]. In STRIPS, the initial state of the world (Equation 2.3) and the final state of the world (Equation 2.4) are given.

$$s_0 = On(B,A), On(A,C), On(C,Floor), Clear(B), Clear(Floor) \tag{2.3}$$

$$g = On(C,Floor), On(B,Floor), On(A,B) \tag{2.4}$$

$$\begin{aligned} &move(x,y,z): \\ &\quad pre: On(x,y) \wedge Clear(x) \wedge Clear(z) \\ &\quad del: On(x,y), Clear(z) \\ &\quad add: On(x,z), Clear(y) \end{aligned} \tag{2.5}$$

A set of operators is given to change the state of the world. In Equation 2.5, the operator $move(x,y,z)$ is given for moving an object x stacked on an object y onto an object z. Its preconditions are specified *pre*. The changes invoked in the environment are given in the *del* and *add* lists. States and planning operators are described with First-Order logic. A resolution theorem prover is used for verifications of preconditions and to direct the search [65]. In current task planning competitions the planning domain definition language (PDDL) [76] is used to standardize

planning domain and problem description languages. PDDL is based on the STRIPS formalism.

The search for a possible plan can also be formulated as a graph search problem as done in Graphplan [13]. A graph is iteratively expanded level by level until a solution is found or a termination condition is met.

In classical task planning, a lot of aspects of the environment are not represented to restrict the search space which is already large. Thus derived plans are not adapted to a given situation and it is difficult to obtain an efficient plan. Furthermore, for a derived plan all steps are assumed to be possible. It is not taken into account that a subtask may fail. Also, the quality of plans can not be determined. A plan is valid or not. However, a valid plan may be very inefficient [80]. Furthermore, even restricted planning domains and associated problems can be intractable due to combinatorial complexity. Planning a complex task like setting the table only with the afore mentioned classical planning methods is too time consuming for online application where immediate system responses are expected. The planning system has to operate on a huge search space and one of its biggest challenges is to efficiently search this space. However, planning can also be seen as a search problem in which steps include refinement, branching and pruning [30]. If a node-selection heuristic ranks a set of nodes in the order of their desirability, the search is accelerated because the search space is reduced. In the Fast-Forward Planning system [40], the forward search is guided by an heuristic that estimates goal distances. Only through the introduction of heuristic search it is possible to apply methods from automated planning to a robotic task planning problem.

2.5.2 Planner Integration

In the context of service tasks the robot must be able to open closets, grasp objects and transport them to the target positions. To solve these real world robotics problems, task planners and low-level planners have to be combined. This can be done in a hierarchical manner that preserves the independence of the individual planning modules, or the diverse planners can be tightly interleaved.

Using a hierarchical top-down approach, a plan is generated first and then mapped onto low-level planners assuming that the symbolic abstraction, i.e. the world model and selected actions, are correct [24]. The problem with this approach is that a particular choice on the task planner level can cause the low-level planner to fail, e.g. an object cannot be grasped with the robot at its current position.

Using a hierarchical bottom up approach, all information that is possibly relevant to the task planner is precomputed by the low-level planners, e.g. all possible paths are determined and provided to the task planner which then determines a set of valid actions [24]. This is unfeasible for planning problems involving articulated manipulators.

A further possibility to combine task planners and low-level planners is to tightly integrate them. The tight integration of different planner types results in a monolithic and possibly inflexible system. While the planners may be able to work well together for a specific problem, exchanging a module for a newer development is difficult.

2.5.3 Task Planning for Robotics Applications

Current research approaches deal with the problem of combining different planners to achieve a robotic manipulation task as described in the following. Task planning is also called symbolic planning. Gravot et al. tightly integrate path and task planning in the planning system Asymov [33] in a non hierarchical manner. Symbolic representations and predicates are defined to link the symbolic and the geometrical parts. The symbolic representations describe the state of the environment. Heuristics guide the alternation between task planning and exploration of the path planners. Gravot et al. do not try to identify a general interface between task planning and path planning, but provide a very specialized solution to their planning task.

Dornhege et al. take first steps towards making symbolic planners applicable to real-world problems [24], [25]. A robotic manipulation planning problem is decomposed into a symbolic and a geometric part. The symbolic view is formulated as a representation that can easily be solved by classical symbolic planners. The manipulation planning problem itself is addressed by hierarchically integrating symbolic and geometric reasoning. The low-level geometric planners provide information concerning the success of their assigned tasks to the high-level symbolic planners during the planning process through so-called semantic attachments. A semantic attachment works like a callback function. Through a semantic attachment, e.g. a path planner is requested to solve a problem and return true or false upon completion. By trying out plans, different plans and parameterizations are tested till a successful one is found.

Kaebling et al. [45] hierarchically integrate a path planner. To save time and reduce backtracking, they use a simplified path planner to simulate a path planning task and predict its success. For the execution of the plan a state of the art path planner is used.

Beetz et al. [8] propose to use task planning only for unknown situations. They use the flexibility of the reactive planning language (RPL) to deal with failures e.g. of the path planner or the navigation system, and schedule recovery actions.

To adapt plans to a situation during execution, Ziparo et al. [121] use a probabilistic prediction of the expected action duration.

Hauser et al. [39] tightly integrate task and path planning. The transition space denotes the configurations that connect two subtasks. The feasible space is the part of the configuration space used for solving a subtask. Subtasks are selected and a subtask graph is built in the feasible space. Subtasks are connected by drawing samples in transition space.

Jain et al. [43] couple a task planner with a probabilistic knowledge representation system. Using the example of setting the table, a knowledge representation is used to infer who sits where and requires which dishes. It is shown that the use of knowledge representations facilitates the planning process by constraining the search space and providing valid parameterizations. The derived plan is performed in simulation on the robot. Conceptually, task planner and low-level planners are hierarchically integrated. The low-level planners reside on the lower levels of the

hierarchy. However, low-level planners are not queried and the simulation is only of symbolic nature.

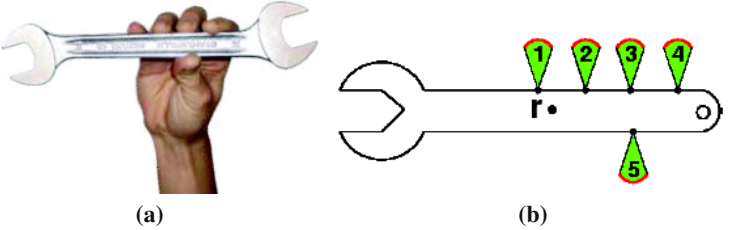
2.5.4 Summary

Path planners either find a plan or terminate upon meeting their termination condition without finding a path. They can not determine whether a path planning problem is difficult, or how far they are from finding a solution. However, a task planner needs to determine whether a path can be found.

For difficult task planning problems the current manner of integrating task planning and low-level planning without the use of knowledge representations can result in an inefficient time-consuming trial and error process where new plans are deduced and the path planner is constantly triggered with different parameter sets. Therefore, fast heuristics are needed to restrict the search space. They need to be able to reliably predict the solvability and difficulty of a planning problem. These heuristics should use scene analysis methods and knowledge provided a-priori. They would enable task planners to choose good parameters, or start and target configurations for the low-level planner. In general, they would enable the task planner to predict the success of a subtask, or low-level planning problem. This would significantly reduce the amount of backtracking or error recovery and speed up the planning process. Furthermore, the quality of plan could be improved.

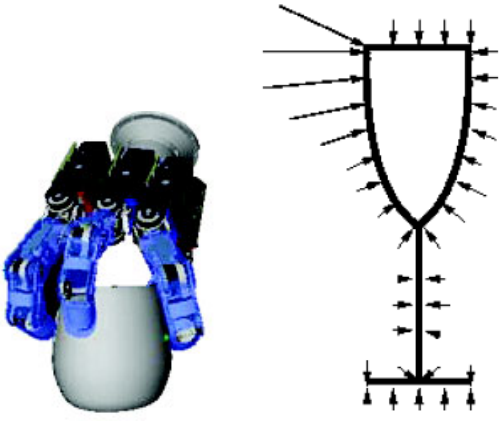
2.6 Grasp Planning

The main objective of a planned grasp is to hold an object firmly and safely, also in the presence of disturbances acting on the object. Power grasps and precision grasps are distinguished [20]. A precision grasp is a grasp where only the finger tips are in contact with the object. It is described by a set of point contacts on an object and the pose of the hand base. In Figure 2.6 an example of a precision grasp is shown. Given the kinematics of a hand, the configurations of the individual fingers can be computed. Precision grasps are used for fine manipulation and for grasping small



(a) (b)

Fig. 2.6 Precision grasping is illustrated. **(a)** A human hand grasps a wrench with all five fingers. **(b)** Five point contacts on a wrench are shown [16].



(a) (b)

Fig. 2.7 **(a)** A precision grasp with a four finger hand is shown for a coffee cup. **(b)** Potential disturbances acting on an object are shown. The arrows symbolize the direction and size of the disturbing force [16].

objects. If a grasp has the force closure property, a set of valid contact forces exists that allows a grasp to balance any occurring disturbance forces or torques [71]. In Figure 2.7 (right) forces acting on a glass are displayed as arrows. Precision grasps are often used in manipulations tasks.

For a power grasp, the human hand molds itself to fit the object. The whole hand is used to grasp the object. The finger links and the palm are in contact with the object. The aim of a good power grasp is to maximize the number of contacts or contact areas.

Most existing robotic hands have inflexible palms. Imitating human power grasps with them is difficult because the components of the hand can not fit themselves to the shape of the object. The hands are more fit for precision grasps than for realistically executing a power grasp. Therefore, in this book the focus lies on the state of the art in precision grasping. A grasp planner computes a valid grasp for a given object automatically.

In humanoid robot manipulation tasks, grasp planners are seldom used online. Instead often fixed manipulation points on the objects [84], [102] are defined to be able to move objects around and avoid grasp planning. Further, predefined grasps [53] or simple rules for grasp generation are used [9]. Miller et al. presented the grasp synthesis and analysis tool GraspIt! [77]. With this system, grasps can be evaluated. It can be used to build a set of valid grasps.

In automated grasp planning two directions are pursued. One direction tries to generate optimal grasps. The other directions generates grasps that are good enough to accomplish the task. Finding an optimal grasp is e.g. reported by Watanabe et al. [110]. Mishra [79] shows that in a planar case an optimal three finger grasp can can be computed efficiently. However, for the 3D case yet no general and efficient

algorithm is known. In general, methods for finding optimal grasps have the disadvantage of high processing times and require a perfect geometrical object model. They are therefore not eligible for an online grasp planner. Computing optimal grasps that are in addition reachable for an attached robot arm is even more complex.

Given an object's 3D model the grasp planner by Borst et al. [15] computes force-closure precision grasps for multi-fingered hands. The computation time is very low

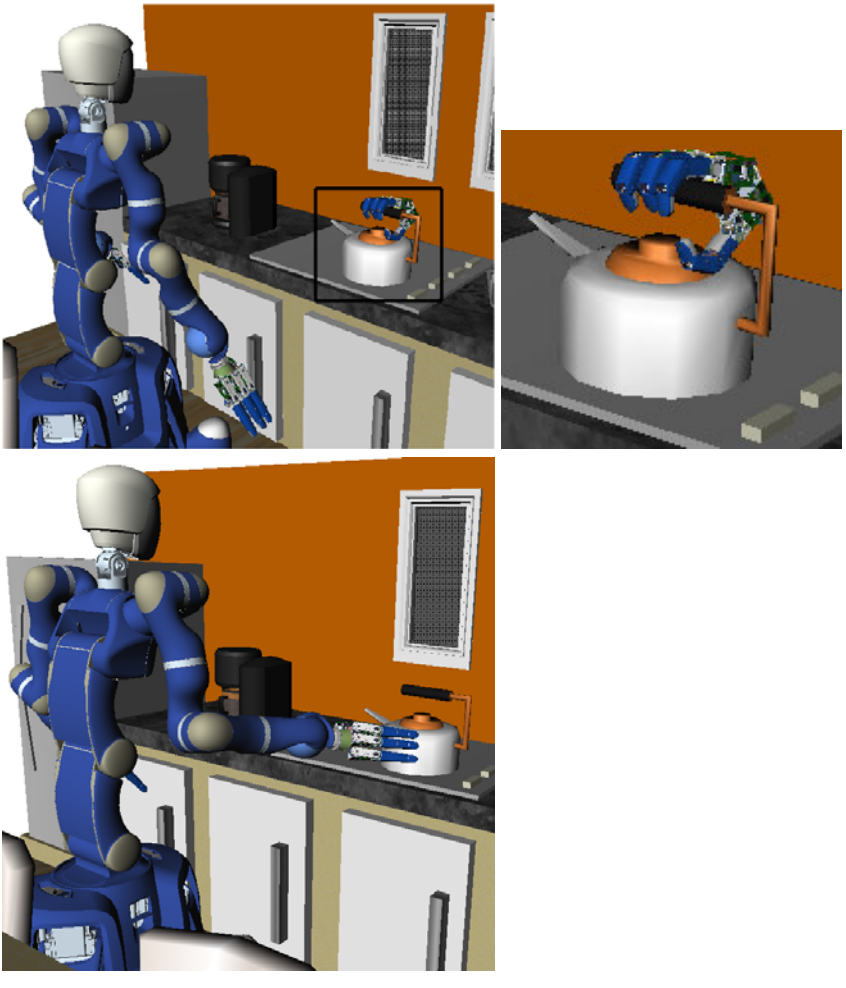


Fig. 2.8 Two challenges with respect to grasp planning for object manipulation. **(top)** The grasp that is returned by the grasp planner grasps the tea kettle handle with the hand base facing the wall. This grasp is unreachable for the robot. A zoomed view is shown in the right column. **(bottom)** The robot determined that to approach the tea kettle from the front is the best way. However a grasp from this direction has poor quality and the kettle might slip from the robot's hand.

for simple objects. The returned grasp is a valid force-closure grasp, but not optimal. It is not considered whether the object is reachable for a robot arm.

Comparable state of the art grasp planners are proposed by Lopez-Damian et al. [66], Miller et al. [78], or Harada et al. [37]. Grasps computed by these systems are reachable by the robot arm and also consider additional obstacles. However some manual segmentation of the objects to be grasped is needed [37] or manually providing a set of grasp starting positions and pregrasp shapes [78] is required before grasps can be generated.

2.6.1 Summary

Grasp planners are either given a TCP pose and return a valid reachable grasp or compute grasps for an isolated robotic hand that is not attached to a robotic arm. In Figure 2.8, two challenges are shown for grasp planning in the context of object manipulation. An object is not graspable equally well from all directions. In Figure 2.8 (bottom) grasps for the requested approach direction may be unable to hold the object firmly. In Figure 2.8 (top) the grasp itself can hold the tea kettle but is not reachable for the robotic arm.

A task planner needs to be able to request a grasp for a given TCP pose with respect to the object. Therefore, a representation is needed that provides information concerning where good grasps on an object are found and from which directions an object is approachable. This enables the task planner to correctly parameterize a grasp planner and avoid unnecessary plan iterations or modifications.

2.7 Path Planning

In the context of humanoid robots and manipulation planning, path planners determine collision-free movement from a given start configuration to a target configuration [3]. Since this book is focused on the representation of the workspace of a robot arm, the state of the art in path planning is presented with a focus on robot arms. However the same algorithms can be used to plan paths for mobile robots.

2.7.1 Planning Algorithms

Paths are a sequence of points in the configuration space of the robot arm [61]. Most often, the points are connected by straight line segments. Two classes of path planners can be distinguished, local path planners and global path planners.

Local path planners [111, 6] locally distort the direct path connecting a given start and goal configuration of a robot arm. The local distortions are used to avoid obstacles. The potential field method is a popular example of a local planner [61]. Obstacles have a repulsive potential, the goal configuration has an attractive potential. A path is found by avoiding the repulsive forces and moving towards the attractive force. Local planners can work well in scenarios composed of only few objects and

often generate smoother paths than global planners. However, in connection with robot arms they are seldom used because it is difficult to define potentials for each object whose combination does not have local minima. In general, local planners cannot guarantee to find a solution even if one should exist.

Global path planners search the whole configuration space of the robot arm for valid paths. The most widely used planners of this category are probabilistic path planners. A robot arm configuration is randomly sampled from configuration space. It is valid if it is collision-free. Most often, it is connected to other valid configurations by using collision-free straight line segments in configuration space. The rapidly-exploring random tree (RRT) planner by Lavalle et al. [62] and the probabilistic roadmap (PRM) planner by Kavraki et al. [49] are the most widely used probabilistic path planners in the state of the art. In searching for a path, the RRT planner iteratively grows a tree in configuration space. The search is successful when the goal configuration q_{goal} is reached. In Figure 2.9 (left) two trees are grown, one from the start and one from the goal configuration. Once the two trees are connected, a path is found.

The PRM planner has an exploration and a planning phase. In the exploration phase, the PRM planner first tries to capture the connectivity of the configuration space in a graph structure, the PRM. In Figure 2.9 (right) a PRM is shown for a simple 2D planning problem. Given a start and a goal configuration, a path is searched in the PRM using graph search methods. Both, PRM and RRT path planners are probabilistically complete, i.e. guarantee to find a solution whenever one exists provided enough samples and time. RRT-planners are best used for single query problems where the environment changes between two planning requests. Since PRM planners invest much effort in capturing the connectivity of the C-space, they are best used for multi query problems. Here, the environment does not change and the investment pays off. Only the start and goal configurations for the robot change.

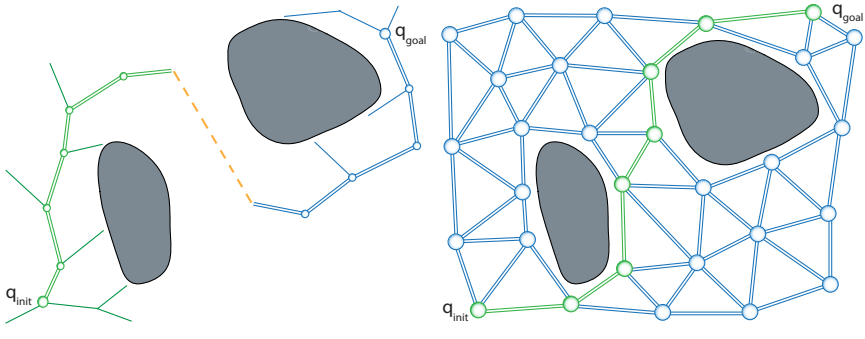


Fig. 2.9 (left) Illustration of a bidirectional RRT-planning run. A tree is grown from the start configuration q_{init} and from the goal configuration q_{goal}. When the two trees are connected a path is found. **(right)** Illustration of an PRM-planning run. In the exploration phase the graph (PRM) is built to represent the connectivity of the configurations space. Graph search methods are used to find a path from q_{init} to q_{goal}. (Source [63])

In manipulation tasks, the robot constantly changes its environment by removing and placing objects. Therefore, the RRT path planners are the best choice for manipulation planning.

2.7.2 Path Planner Inputs

At first glance it seems as if path planners are ready for use by the task planners. The problem of approaching and grasping an object seems solved by using a state of the art path planner to get the arm to the desired grasp configuration. Every path planner needs collision-free, reachable start and goal configurations of the robot arm to be able to plan a geometric trajectory. However, the configurations have to be determined and provided to the path planner before the start of planning. When grasping an object, there is a huge number of possibilities where and how to grasp it. Thus there is an undetermined number of goal configurations, one of which has to be chosen for use by the path planner. Using the best grasp for an object might lead to an unreachable arm configuration (Figure 2.8 top) while defining a certain arm configuration might lead to bad grasps (Figure 2.8 bottom). Therefore the choice of start and goal configurations for a path planner has to be addressed.

2.7.3 Selection of Goal Configurations

Lozano-Perez et al. [67] recognize that to solve pick-and-place tasks, choosing how to grasp an object is crucial for the success of a task. They state that the selection of a good target position depends on the environment. The hand position and resulting arm configuration have to be free of collision and reachable. In principle, also the availability of a path between start and goal configuration should be checked. However there are a huge number of goal configurations, a path planner's worst-case time-complexity is exponential and the path planning problem is PSpace-hard [92]. Therefore checking the availability of a path for each possible goal configuration is computationally too expensive. Lozano-Perez et al. [67] grasp objects with a parallel jaw gripper. The set of possible grasps for an object is limited. Lozano-Perez et al. pick a grasp location and try to plan a path, repeating the process if no path can be planned to the grasping location. For redundant robots with multi-fingered hands, the set of grasps for an object is infinite. Therefore, a brute force method of iterating over the whole set of grasps is not feasible.

Wösch et al. [111] use an 8-DOF robot arm and a parallel jaw gripper. They define by hand a set of approach TCPs and sample the TCP orientation. Performing inverse kinematics computations it is determined whether there is one among them that results in a collision-free robot arm configuration. Depending on the number of poses, they report that the determination of an optimal approach TCP in this brute force manner takes longer than the actual path planning. Thus a feasible solution is needed to close this gap.

The approach presented by Terasaki et al. [108] again operates with a parallel jaw gripper on objects taken from an assembly scenario. To ensure the availability

of a path, the target object is analyzed to obtain a set of candidates for grasping positions. The space around each grasping position is described by a so-called open space characterization. It represents approach directions by pyramidal slices labeled with a measure of the available free space in that direction. This characterization is used to select among the grasp candidates. If no grasp could be found, the method stops.

Diankov et al. [23] use a set of grasps and incorporate the choice of the grasp into an RRT path planner. Multiple goal trees are grown, one for each grasp. A path is found if one of the goal trees is connected to the tree grown from the start configuration. This approach is extended by Berenson et al. [11]. Workspace goal regions describe where an object can be grasped. From these goal regions grasps are sampled and used to seed goal trees in a RRT path planner. Therefore, the selection of the grasp used in a task is performed by the path planner and not by a task planner.

2.7.4 Summary

In summary, from where to grasp an object is either decided by testing individual grasps or by integrating the choice of grasp into the path planner. Knowledge representations and scene analysis are not used to determine good start and goal configurations. Diankov et al. [23] even incorporate the choice of which object to move first into the path planners. However, if the choice of the grasp or which object to move is incorporated into the probabilistic path planner, a huge inflexible monolithic system emerges. Its decisions are not deterministic or comprehensible. It has no information about the structure of the task (i.e. that the cup should be placed on the saucer). It is difficult to combine with a task planner because the system already assumes some of the duties of a task planner, i.e. the selection of the grasp to use.

2.8 Robot Placement

For tasks such as setting the table, cleaning the kitchen or filling the dishwasher, different manipulation tasks have to be executed. In this context, one of the main challenges to be addressed is where the robot should place itself to be able to execute a given task. In Figure 2.10 (top) a bad placement is shown for grasping a glass on the table. For opening doors or drawers this question is even more difficult to answer (Figure 2.10 (bottom)). Here, the motion of the end-effector and its attached TCP is constrained to follow the motion of the closet door or the drawer. The trajectory of the TCP is constrained.

2.8.1 Selection of Fixed Base Positions

Stulp et al. [105] learn where the robot can place itself to grasp an object using a support vector machine approach. However only one orientation of the cup is used. Therefore, it may be difficult to generalize the results to other objects, or objects

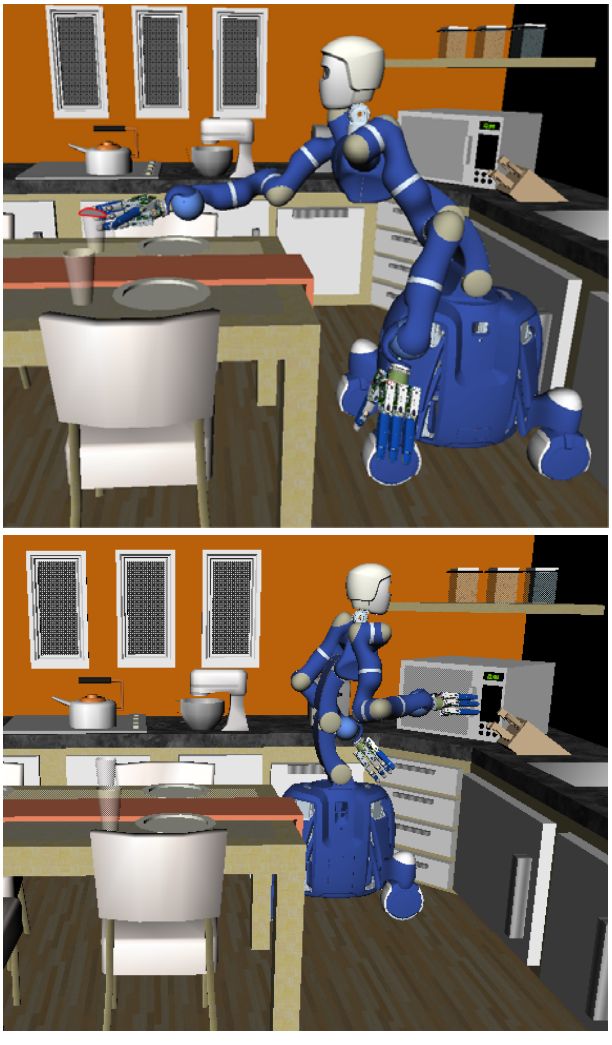


Fig. 2.10 Examples for robot placement are shown. **(top)** The robot has placed itself badly. It cannot reach the glass it wanted to grasp. **(bottom)** The robot placed itself before the microwave. To open it the hand is constrained to follow the circular trajectory of its door.

placed not on the table but in the closet. The approach is also not yet expanded to constrained trajectories. With respect to choosing a fixed mobile base position to execute constrained trajectories, few approaches are known. Neo et al. [82] present an interactive operation system for commanding an HRP-2 humanoid robot to open a fridge. To choose a standing position for HRP-2 from which it is able to execute the task, a heuristic approach is used. Yoshida et al. [114] reach for target end-effector positions with an HRP-2 humanoid robot. An inverse kinematics for the

complete humanoid is combined with an heuristic approach for support polygon reshaping and stepping. Konietschke [54] uses genetic algorithms and gradient-based optimization to position a medical robot for operation in a given region of interest. However, only one solution is computed. Thus it cannot be deduced how well the robot is suited for a task.

2.8.2 Base Movement during Task

Robot placement planning for constrained trajectories determines positions for a mobile manipulator to execute the task. Most approaches combine the positioning of the robot with the search for feasible trajectories for the robot arm in the configuration space (C-space). Optimization and path planning techniques are used.

Optimization techniques are applied to the whole kinematic chain. When using multi-criteria optimization for positioning a mobile manipulator to reach a point, choosing criteria weights, competing criteria and the resulting local minima are a great challenge [89]. Multi-criteria optimization can get stuck in local minima due to unfavorable start positions and fail to find a solution.

Existing path planning techniques, e.g. the PRM approach or the RRT approach are adapted to search for constrained trajectories. Given a start configuration, a goal frame and task constraints, Yao et al. [113] search a collision-free path for a fixed base robot arm by alternating Cartesian space and configuration space exploration. Valid path segments in Cartesian space are tracked in C-space using trajectory tracking methods from control theory, e.g. Jacobian-based methods. If tracking is not possible due to the kinematics and link limits, the path segment is discarded and another segment is tested. Some experiments with a 3 DOF platform are also reported.

Oriolo et al. [85] label the n DOF of the mobile manipulator as redundant ($n-6$ DOF) or non-redundant (6 DOF). Random samples are drawn from the C-space of the redundant DOF, i.e. from the 3 DOF C-space of the mobile base. The RRT path planning approach is adapted and combined with inverse kinematics algorithms. The resulting path is a combination of motion of the redundant and the non-redundant joints, i.e. the mobile base and the robot arm. Since no knowledge is used about which regions are reachable from what direction, many platform positions are examined from which executing the trajectory is impossible.

Yang et al. [112] adapt PRMs to plan task-consistent collision-free motion for mobile manipulators with a holonomic base in dynamic environments. This approach is able to plan the mobile manipulator motion but always moves the mobile base.

Stilman [101] and Berenson et al. [10] plan task-constrained, collision-free motion for a mobile manipulator. A task formalism is proposed to indicate constrained motion directions in Cartesian space. The Jacobian and its pseudo-inverse are used in the extension of an RRT path planner to find joint space displacements that resolve Cartesian space errors and then use these to generate samples for a RRT path planner in C-space. A Cartesian start and end frame of the trajectory is known as well as an initial grasping configuration of the robot arm. However, the system

always moves the mobile base to accomplish a task. For simple tasks this may not always be necessary.

2.8.3 Summary

In summary, few approaches deal with placing the robot for grasping tasks. Concerning placement for constrained trajectories, two directions are pursued. An inverse kinematics is applied to the whole kinematic chain. It is assumed that the task can be completed if the redundancy is adequately exploited. Or a start configuration of the mobile manipulator is given from which the task is feasible and path planning techniques are adapted to use the complete kinematic chain. In both cases the redundancy provided by the mobile base is used. However this may complicate the task execution, since the use of the mobile base may not be necessary to achieve the task.

Using current approaches, a task planner can only try out whether a chosen placement allows the execution of the trajectory or trust an extended path planner that includes the mobile base to choose correct positions. In the latter case, the low-level planner changes the states of the environments without the knowledge of the task planner. Preconditions of subsequent actions planned by the task planner can be invalidated. Therefore, this method is difficult to combine with a task planner.

2.9 Motion Primitives

Motion primitives describe stereotypical trajectories that a human performs during a specific task. The trajectories can be compactly and efficiently represented e.g. by Dynamic Movement Primitives proposed by Schaal et al. [97]. They represent parametric, standard models of behavior or movement that can be adapted to an altered task context. Stulp et al. [106] learn dynamic movement primitives for human reaching motions with and without obstacles. Through the use of the model knowledge represented by the dynamic movement primitives Stulp et al. are able to produce predictable and human-like reaching motions for a humanoid robot. The motor primitives by Matarić [72] are another example of motion primitives. Motor primitives are modular, computationally efficient representations of behaviors. They can be used to generate and classify movement as reported by Drumwright et al. [26].

Hauser et al. [38] use motion primitives that represent walking motions to guide the sampling strategy of a PRM path planner and thus reduce the search space. The use of motion primitives results in a reduction in planning time and an increase in motion quality especially when the robot walks over uneven terrain. In summary, model-based knowledge can support movement generation and planning for robots.

2.10 Robot-Specific Knowledge

Robot-specific knowledge describes characteristics and capabilities of a specific robot, e.g. what weights it can lift in which regions of the workspace. It can be encapsulated in knowledge representations and be used to formulate heuristics to guide planners, facilitate decision processes, and significantly speed up planning processes in general. The use of models encapsulating robot-specific knowledge is recently taken up by several research groups. Pettré et al. [88] make the animation of a digital actor more efficient by dividing the large number of degrees of freedoms of a humanoid into functional units providing the locomotion and the manipulation capabilities. Diankov et al. [23] use a similar functional structure for a humanoid robot to plan a path from a given start position to an object to be manipulated. In the process they furthermore consider a model of the reachable workspace of the robot arm to decide where the robot may stand to grasp an object and thus focus the search. Gienger et al. [31] use an object-specific model of the grasping capabilities of their humanoid robot to optimize the whole body motion to reach and grasp an object. Stulp [104] et al. show that significantly fewer learning examples are needed if knowledge encapsulated in models is used to guide the learning process.

In general, it is reported that exploitation of prior knowledge facilitates and speeds up planning.

2.11 Summary

The current manner of combining high-level and low-level planners is inefficient. In general, robot-specific knowledge is not used in planning. To bridge the gap between high-level and low-level planning, methods are needed that reduce the search space based on knowledge encapsulated in models. These methods could help predict the success of subtasks or find parameters for planners that permit them to solve a task faster.

This book emphasizes the speed up and flexibility in decision making gained through the use of robot-specific knowledge encapsulated in a model. If the robot has a model of the workspace of its arms, that describes which region is reachable from which direction, a task planner can use this information. It can determine from which direction an object can be grasped or how difficult it is to grasp it. Thus predictions about the success of a subtask can be made.

Chapter 3
Robot Performance Indices

Robot performance indices evaluate how well a robot can apply forces or move during a specific task or throughout the whole workspace. Hence, they potentially contribute to a general description of the versatile workspace that is the focus of this book. In this chapter, criteria used in robot arm design are compared and evaluated with respect to their objectives. It is identified whether these criteria provide measures to evaluate a robotic arm's kinematic capabilities with respect to reachability and manipulation of objects throughout the workspace. It is determined whether they can be used to represent the robot arm workspace. Furthermore state of the art approaches to model the workspace are analyzed and evaluated.

3.1 Robot Design Criteria

In the design stage, a robot manipulator is optimized with respect to kinematic and dynamic criteria. In this book, the focus is on the kinematic aspects since the methods to be developed are intended to support static planning methods. The kinematic design can be divided into task-oriented robot design criteria and general purpose design criteria.

3.1.1 Task-Oriented Design

For some application areas, a robot needs to guarantee a certain set of crucial tasks without having to perform many other kinds of tasks. Such areas include e.g. medical applications. In task-oriented design, the robot arm is designed for a set of specified tasks. A number of task points [87] or a task volume [56] is given that has to be reached by the robot. Park et al. [87] present a task-oriented design method based on an optimization method used in heat transfer/fluid analysis. Multiple objective optimization of the kinematic parameters of the robot manipulator is performed. The kinematic parameters determine the pose of the link axes. The method receives a number of task points, the number of degrees of freedom and the position of the robot base. It strongly depends on the given initial solution and the chosen weights.

F. Zacharias: Knowledge Representations for Planning Manipulation Tasks, COSMOS 16, pp. 27–35.
springerlink.com

Konietschke et al. [56] present a method for optimal design of a medical robot for minimal invasive surgery. Optimization is carried out using genetic algorithms and a subsequent gradient-based optimization step. Several optimization criteria are added as constraints on the optimization process, such as minimum positioning accuracy to be achieved and the avoidance of singularities. Optimization is carried out for a number of small workspace volumes. These are discretized and the robot setup is evaluated at each discretization step.

Both methods are constructive and aim at comparing different robot designs for very restricted regions of the workspace. The aim of this book, however, is to characterize the performance of the robot in discretized regions of the whole workspace. The two methods are computationally too expensive to use them for the whole workspace. Even if each region of the workspace is evaluated, only single evaluation values are reported, that represent the quality of the region. The information from which direction a region is reachable is lost.

3.1.2 General Purpose Robot Design

In general, the robot kinematics are optimized to maximize the workspace volume or to maximize various dexterity indices at specific positions or with respect to the entire workspace. A dexterity index evaluates how well a manipulator is able to move and apply forces in arbitrary directions.

Lenarčič et al. [64] use the workspace volume and the ”compactness” of the workspace as design criteria. The dispersion of the reachable workspace is the average quadratic distance between elements of the workspace volume and its arithmetical center. A spherical workspace is the most compact workspace. Therefore, the workspace compactness is defined as the ratio between the dispersion of the reachable workspace elements and the dispersion of a sphere with the same volume as the robot workspace.

Park et al. [86] introduce general performance criteria for workspace volume and dexterity using differential geometry. Their aim is to derive performance criteria that are independent of a particular coordinate transformation. Global indices are obtained through integration of local criteria. These global indices are used to compare different robot arm designs or to derive optimal parameters e.g. for link lengths or actuator dimensioning. However, they are not aimed at characterizing individual subspaces of the robot workspace.

Sturges et al. [107] define a dexterity measure that relates the difficulty of an assembly task to the capabilities of a planar robot arm and gripper. It is proposed that the effective dexterous ability of a system should be mapped as a function of position of the end-effector within the workspace. A two-dimensional peg-in-hole assembly task is used to illustrate task-dependent difficulty measures. A task index of difficulty and an effector index of difficulty are defined. To be able to complete the task the effector index of difficulty has to equal or exceed the task index of difficulty. However, for general tasks and spatial robots, these indices are difficult to derive. Especially if a redundant robot arm and a multi-fingered hand are used, many factors influence the success of a task. The robot has to be positioned for the task,

or the scene geometry has to allow the execution of the task. Often there are also interactions between the factors that enable the task success. Therefore, the method by Sturges et al. cannot be used for service robot tasks.

3.1.2.1 Analysis of the Jacobian Matrix

A popular means used in robot arm design is the analysis of the Jacobian matrix. The Jacobian matrix J at a configuration $\mathbf{q}$ relates the joint velocities $\dot{\mathbf{q}}$ with the total end-effector velocity in Cartesian space (consisting of angular velocity ω_E and translational velocity $\dot{\mathbf{p}}_E$).

$$\begin{pmatrix} \dot{\mathbf{p}}_E \\ \omega_E \end{pmatrix} = J(\mathbf{q})\dot{\mathbf{q}} \tag{3.1}$$

$$J = U\Sigma V^T \tag{3.2}$$

The principal axes and singular values $\sigma_\mathbf{i}$ $(i = 1..n)$ of the Jacobian matrix are obtained by singular-value decomposition [90] of the Jacobian matrix (Equation 3.2). Here n is the number of links of the robot arm. U and V^T are orthogonal matrices, and Σ is a diagonal matrix containing the singular values σ_i of J. σ_1 marks the largest singular value and σ_n marks the smallest. In the following, several indices are described that have been derived thereof. Klein et al. [51] examine dexterity measures with the aim to optimize the robot arm configuration for a given end-effector position. They analyze the relationship of the determinant, the condition number and the smallest singular value of the Jacobian. A determinant equal to zero marks the presence of a singularity. The condition number of the Jacobian

$$cond(J) = \frac{\sigma_1}{\sigma_n} \tag{3.3}$$

evaluates the isotropy of movement of the TCP. If its value is one, the TCP can move in all directions of the principle axes equally well.

With the goal to obtain a global measure for the isotropy movement, Stocco et al. [103] optimized the ratio of the global maximum and the global minimum singular value of the Jacobian in the entire workspace to obtain a global version of the condition number.

However, the smallest singular value varies radically in the presence of singularities thus dominating the behavior of the condition number and the determinant. Each presented measure only allows conclusions for infinitesimal small environments. Non of the examined dexterity measures is superior over the others as each has a slightly different purpose. All have in common however to promote a robot design that can move the TCP in all directions equally well. This book is interested in capturing directional preferences rather than in finding configurations where directionally uniform movement is possible. The manipulability ellipsoid and derived measures proposed by Yoshikawa [115] are often examined to evaluate directional preferences. They receive a closer inspection in the following sections.

3.1.2.2 The Manipulability Measure

The manipulability ellipsoid in the n-dimensional Euclidean space introduced by Yoshikawa [115] is intended to quantify the ease of arbitrarily changing the position and orientation of the end-effector. It is derived by analyzing the Jacobian matrix of a manipulator (Equation 3.1) and its singular value decomposition (Equation 3.2). The principal axes and singular values σ_i are taken to define the orientation and the shape of the so-called manipulability ellipsoid. The size of the ellipsoid and its major and minor axes are assumed to represent an ability of manipulation at a certain configuration. In the direction of its major axis the TCP can move at high speed. In the direction of its minor axis, the TCP can only move at low speed. A singular value σ_i is interpreted as the radius of the ellipsoid in the direction of the corresponding principal axis. The ratio of the minimum and maximum singular value of the ellipsoid (Equation 3.3) can be used to describe the directional uniformity of the ellipsoid and thus the directional uniformity of possible movements at the considered configuration. The volume of the ellipsoid is known as the *manipulability measure* w (Equation 3.4)

$$w = \sigma_1 \cdot \sigma_2 \cdots \sigma_n \tag{3.4}$$

and can be interpreted as the distance of the manipulator from a singular configuration. In Figure 3.1 the distribution of the minimum value of the manipulability measure is shown across the workspace of the DLR LWR arm. A cut through the workspace is shown. The TCP position for a randomly sampled robot arm configuration is mapped to a voxel of the discretized workspace. The manipulability measure for the configuration is compared to the current minimum of the voxel and the new minimum is determined. Red voxels indicate regions reached by singular configurations. In blue voxels the distance to singular configurations is high according to the manipulability measure. Thus the worst-case manipulability is shown in Figure 3.1 for voxels of the workspace.

3.1.2.3 Characteristics of the Manipulability Measure

Since the Jacobian (Equation 3.1) relates link velocities to Cartesian translational and rotational velocities, all manipulability criteria for the Cartesian space are unit dependent. Ma et al. [70] propose to divide the translational components by a so-called characteristic link length to remove this inhomogeneity. Still, this remains a heuristic solution to the problem of inhomogeneity and only defines a scaling between the rotation and translation.

Furthermore the principal axes of the manipulability ellipsoid mix rotational and translational components because they are derived from the Jacobian. This reduces the interpretability of the ellipsoid and the manipulability measure.

Since manipulability values are derived from the Jacobian matrix at a given configuration they are purely local measurements valid only for an infinitesimal small neighborhood. Furthermore, link limits are not taken into account. Thus, attested good movability at a configuration may not be possible in the desired direction due to link limits. Abdel-Malek et al. [2] augmented the Jacobian with joint limit

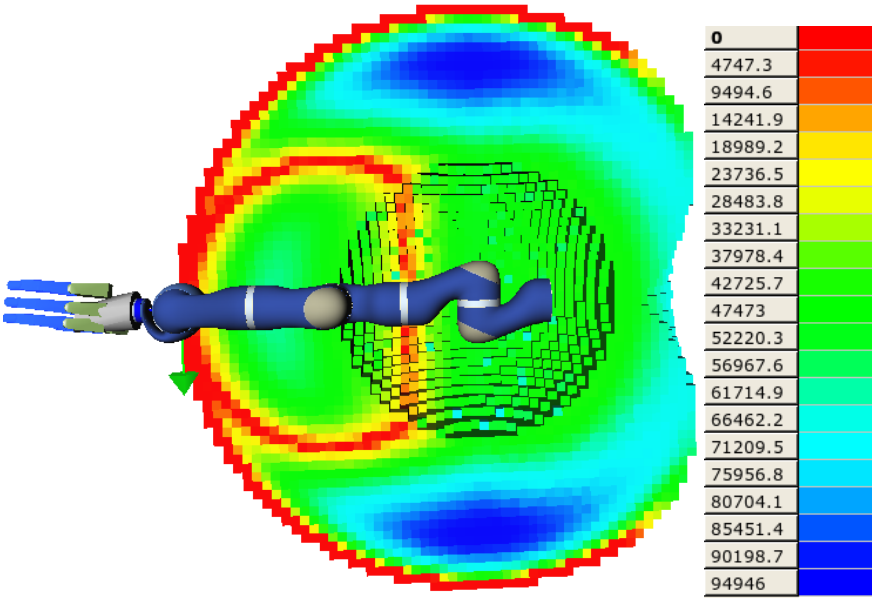


Fig. 3.1 Distribution of the minimum value of manipulability measure across the workspace of the DLR LWR arm. For computing the Jacobian and the singular values, the link lengths are in mm. The value of the manipulability measure is color-coded. Red voxels have a minimum manipulability measure near zero and therefore contain singular configurations.

criteria. However, through these additional criteria, the manipulability measure is even harder to interpret due to the additional unit dependency.

Due to the above mentioned problems this book refrains from using the manipulability ellipsoid or measures derived thereof to represent the robot arm capabilities. Instead, the next section concentrates on existing approaches to analyze the robot workspace in Cartesian space.

3.2 Workspace Analysis

Workspace analysis tries to represent the Cartesian workspace of a manipulator. The most basic kind of workspace analysis is the extraction of the inner and outer borders of the workspace [64] by iterating through the complete link ranges. This can be used to roughly compare robot arm kinematics. But also quantitative analysis of the reachable workspace has been reported. Gupta [36] sketches a global and a local evaluation method. It is reported for six DOF manipulators with a wrist where the last three axes intersect in one point, that the dexterous workspace decreases monotonically as the size of the hand increases. The hand size is taken to be the distance between the intersection of the wrist axis and the tool tip point. They also propose a local analysis at single points across the workspace in terms of approach angles

and approach lengths. The approach angle describes the orientations that are possible around the tool tip point, i.e. the possible angles of the tool axis. The approach length describes the linear axial motion of the tool away from the considered tool tip point. This analysis is only done for few points across the workspace and the results are stored. A similar analysis of whole regions of the workspace with respect to the directional reachability potentially results in a suitable representation of the reachable workspace.

Another approach to analyze the topology of the workspace is given by Lück et al. [69]. They analyze the topology of the self-motion manifolds of a redundant robot manipulator. Self-motion manifolds represent the null-space motion of the manipulator. They argue that singularity manifolds partition the configuration space as well as the workspace. Furthermore, they claim that a mapping can be found between these regions and can be expressed in a connectivity graph. For an 8 DOF spatial manipulator with link limits the connectivity graph is shown to be highly complex and is split in several parts. In another work, Lück et al. [68] propose to discretize non-uniformly the topology-defined partitions and use this discretization in path planning resulting in a bidirectional map between a discretized configuration space and workspace. However, this is possible for a 3 DOF manipulator, but it is infeasible for a manipulator with more DOF. The complexity of the connectivity graph for the 8 DOF manipulator shows this [69]. Even with this topology-based approach it is infeasible due to memory requirements and computation power.

A monte carlo approach to represent the reachable workspace was introduced by Guan et al. [35]. The workspace is enveloped by a cube and discretized. Via forward kinematics a randomly sampled configuration is mapped to the discretized workspace. In the visualization only those cubes are shown that are hit at least once. Klopčar et al. [52] compute a kinematic model for the human arm. They iterate through the value range of the arm links and map the position of the TCP to cubes of different granularity. To visualize the reachable workspace they only show those cubes containing at least one point. The visualized workspace is used to compare the reachable workspace between patients with different arm injuries. In both papers only the reachable workspace is represented. Orientation information is not represented.

To sum up, the information contained in most current representations of the workspace is not detailed enough to enable exploitation by high-level or low-level planners. No directional structure can be extracted from the representations. However, the approach by Gupta et al. [36] examined for specific points how they can be approached. If this approach is adapted and extended to cover the whole workspace it could lead to a suitable representation of the workspace.

3.3 Naive Sampling Based Approach

In this section a naive sampling based approach is examined which is first reported by Zacharias et al. [117]. The workspace of the robot arm is discretized as explained

in detail in Section 4.2.1. The configuration space is randomly sampled with a uniform distribution. The TCP frame is computed and mapped to a voxel of the Cartesian workspace. The number of randomly sampled configurations mapped to a voxel is used as a measure of reachability for the voxel that correlates with the voxel being easily reachable. Easily reachable means that the redundancy can be exploited and versatile manipulation is possible. A representation based on this measure appears to be an intuitive representation of the workspace. However, when a robot is in a singular configuration, large steps in the configuration space for the links causing the singularity result in small motions of the TCP in the Cartesian workspace. Therefore the TCP poses for configurations which are far apart in the configuration space are mapped to the same voxel in the Cartesian space. A large amount of sampled configurations accumulate in these voxels. The distribution of the number of randomly sampled configurations in the workspace is shown in Figure 3.2. In voxels that the robot arm can reach by singular configurations, samples accumulate represented by blue or green voxels. Therefore, the number of configurations mapped to a voxel cannot be used to find voxels that allow for versatile manipulation. Furthermore directional information is not represented.

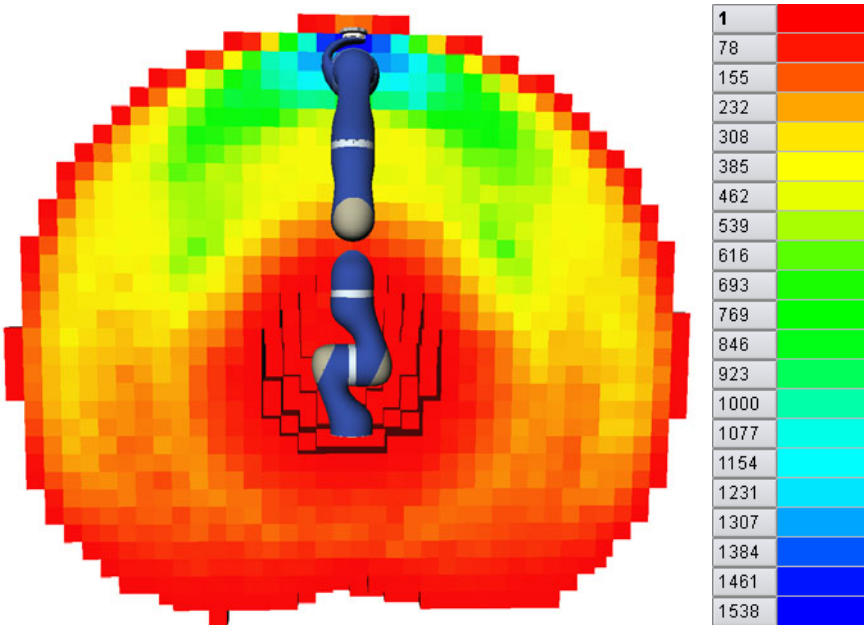


Fig. 3.2 Color encodes the number of randomly drawn configurations attributed to the discretized Cartesian voxels. In total $6 \cdot 10^6$ random configurations are drawn. The workspace is cut in half along the arm center axis. In voxels that the robot arm can reach with singular configurations, samples accumulate (blue and green voxels).

3.4 Model Requirements

As discussed in previous sections, existing criteria and methods are not suitable to represent the robot arm workspace. Most discussed criteria are not able to represent directional information. The manipulability ellipsoid is defined to evaluate the ability of the TCP to move in arbitrary directions. However, the axes of the manipulability ellipsoid mix rotational and translational components. Therefore they are difficult to interpret. In previously presented representations of the workspace, only the reachable workspace was modeled. It can be determined that regions are reachable but not from which directions. Therefore this section extracts a set of requirements that a representation of the robot arm workspace has to fulfill.

In humanoid robot tasks, objects have to be grasped and manipulated preferably using both arms. A high-level planner has to evaluate which arm can best grasp objects (Figure 3.3) and has to decide when to use which arm. Moreover, a humanoid robot with multi-fingered hands has to choose among an infinite number of alternative configurations that can be used to approach and grasp an object. Considering a mobile manipulator a further problem is the positioning of the mobile platform with respect to the operating area.

In general, a representation of manipulator capabilities is required that can be used to characterize which places are easily reached. If a specific direction is of interest, this direction is mapped to its discretized counterpart and its reachability is determined using the representation. Since different sensors have different accuracies and the positions of obstacles can change during a task, these issue should have no influence on the representation. They have to be addressed in algorithms that analyze the scene and the robot capabilities based on the representation. Furthermore,

Fig. 3.3 Justin has to decide which object to grasp with which arm and hand.

a visualization scheme is needed for the representation. It should enable to focus alternatively on regions of interest or on the whole workspace. The directional information represented for individual workspace regions might show similarities with other workspace regions. If areas with the same structural characteristics exist, a visualization of the representation should make this identifiable. The requirements are summarized in the following list.

- Direction and position information has to be represented.
- Data access has to be fast to enable use in online algorithms.
- The reachability of arbitrary poses has to be determinable.
- The memory requirement should be low to be able to keep the model in main memory.
- A 3D visualization scheme has to be provided.
- The visualization should enable inspection of the data at different levels of detail.
- The visualization should enable the focus on regions of interest.
- The visualization should enable to identify areas with the same structural characteristics.

3.5 Summary

The recent and existing performance criteria and workspace models can only be used to decide whether a position in the Cartesian workspace is reachable or whether a workspace region can be reached with a singular configuration. Directional information is not represented. Also the naive workspace representation examined in Section 3.3 proved to be unsuitable.

Therefore, requirements are extracted for a model of the workspace. The most important issue is the representation of directional information. For manipulation tasks, a region does not need to be reachable from all directions. However the model should show from which direction a region is reachable and how close a region is to belonging to the dexterous workspace, i.e. that it is reachable with arbitrary poses. In the next chapter, a new model is presented that fulfills the requirements defined in Section 3.4. The ideas of randomized sampling, workspace discretization, and the examination of a region from various directions are combined.

Chapter 4
Modeling the Robot Workspace

In general, every robot arm is designed differently, and therefore has different kinematic capabilities. These capabilities can result in directional structures specific to workspace regions. The robot's ability to manipulate objects depends on the relative position of the objects. Two-handed manipulation is limited to a region where the workspaces of both arms overlap. The best performance is achieved in an even smaller subspace. In the previous chapter, requirements were identified that a representation of the reachability throughout the workspace has to fulfill. The *reachability sphere map* is a representation that meets these requirements. The choice of this name becomes clear later. It was first introduced in [117].

As a first step, the construction of the reachability sphere map is described. A visualization scheme is introduced for the representation. It allows the detection of structure in the workspace and enables its visualization. In a second step, a compact abstraction is proposed for the data of the reachability sphere map. The approach is illustrated using a DLR light weight arm (LWR).

4.1 The Tool Frame (TCP)

The tool frame, also called TCP frame, is the reference frame for manipulation tasks where the robot uses the end effector to grasp or transport objects. The pose of the TCP is described by a homogeneous matrix F (Equation 4.1). It is also referred to as the TCP frame. In Figure 4.1 the TCP pose is indicated by the coordinate system at the end of the robot arm. The space containing all homogeneous matrices is called H in this book.

$$F : SE(3) \rightarrow H;\ (R,\mathbf{t}) \mapsto F(R,\mathbf{t}) := F = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{4.1}$$

The function $F(R,\mathbf{t})$ constructs the homogeneous matrix F from the rotation matrix $R \in SO(3)$ (Equation 4.2) and the position vector $\mathbf{t} \in \mathbb{R}^3$ (Equation 4.3). $SO(3)$ is the set of rotation matrices in $\mathbb{R}^3$ (see Section A.3 or Murray et al. [81]).

F. Zacharias: Knowledge Representations for Planning Manipulation Tasks, COSMOS 16, pp. 37–69.
springerlink.com

$$R = (\mathbf{x}\ \mathbf{y}\ \mathbf{z}) \tag{4.2}$$

$\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ are orthogonal basis vectors.

$$\mathbf{t} = (t_x, t_y, t_z) \tag{4.3}$$

A homogeneous matrix is often used to describe rigid motion or to define the pose of an object in the Cartesian space.

Let $K(\mathbf{q})$ describe the direct kinematics from Equation 2.1 which determines for a configuration $\mathbf{q} \in C$ the TCP pose T_{TCP}^{Base} in the robot base coordinate system.

$$K(\mathbf{q}) := T_{TCP}^{Base} \tag{4.4}$$

4.2 The Reachability Sphere Map

The reachable workspace W_R is the volume of space that the robot can reach in *at least one* orientation [19]. It is described in Equation 4.5 using set notation.

$$W_R := \left\{\mathbf{x} \in \mathbb{R}^3 \middle|\ \exists \mathbf{q} \in C \wedge R \in SO(3) : K(\mathbf{q}) = F(R,\mathbf{x})\right\} \subset \mathbb{R}^3 \tag{4.5}$$

However, in planning tasks a model of the reachable workspace is insufficient. Regions in the workspace have to be approached with specific orientations of the TCP. The dexterous workspace W_D describes the volume of space that the robot can reach in *all* orientations.

$$W_D := \left\{\mathbf{x} \in \mathbb{R}^3 \middle|\ \forall R \in SO(3) \exists \mathbf{q} \in C : K(\mathbf{q}) = F(R,\mathbf{x})\right\} \subset \mathbb{R}^3 \tag{4.6}$$

The dexterous workspace is a very small subset of the reachable workspace. For some robots it is empty. In real manipulation tasks, orientations are important but seldom all are needed. In this book, the **versatile workspace** W_V indicates with which orientations $R \in SO(3)$ a position $\mathbf{x} \in \mathbb{R}^3$ can be reached by the end effector and thus how far a position is from being a part of the dexterous workspace. This concept is formalized using $W(\mathbf{x})$, the set of orientations in which the TCP can reach $\mathbf{x}$.

$$W(\mathbf{x}) := \left\{R \in SO(3) | \mathbf{x} \in \mathbb{R}^3, \exists \mathbf{q} \in C : K(\mathbf{q}) = F(R,\mathbf{x})\right\} \tag{4.7}$$

The versatile workspace W_V is then defined as

$$\begin{aligned} W_V :&= \left\{(\mathbf{x},R) | \mathbf{x} \in \mathbb{R}^3, R \in W(\mathbf{x})\right\} \\ &= \left\{(\mathbf{x} \in \mathbb{R}^3, R \in SO(3)) | \exists \mathbf{q} \in C : K(\mathbf{q}) = F(R,\mathbf{x})\right\} \subset SE(3) \end{aligned} \tag{4.8}$$

The versatile workspace is a continuous space that can not be described analytically. The workspace representation introduced in this theses, hereafter called **reachability sphere map** represents the versatile workspace W_V by discretizing $\mathbb{R}^3$ and $SO(3)$. It is presented in the following sections.

4.2.1 Discretization of $\mathbb{R}^3$

The physical workspace of a robot arm can be encapsulated by an axis oriented bounding box with an edge length l_{ws} equal to two arm lengths. However, the reachable workspace of the arm is thereby overestimated. The axes of the bounding box are aligned with the robot arm base coordinate system situated at the position of the first link axis. The center of the bounding box is the position of the robot arm base (Figure 4.1). The bounding box is partitioned into an axis-aligned, regular grid of cubes. The homogeneous partitioning is called voxelization of $\mathbb{R}^3$. The discrete volume elements are named voxels. A so-called single-resolution voxel space is used, i.e. all voxels have the same edge length l_c, $l_c << l_{ws}$. In Figure 4.1, a coarse voxelization is shown for illustration.

Using this discretization, specific regions of the workspace can be analyzed in task planning processes. Each voxel stores information related to the covered volume. Each dimension of the Cartesian workspace is subdivided into n_c voxels.

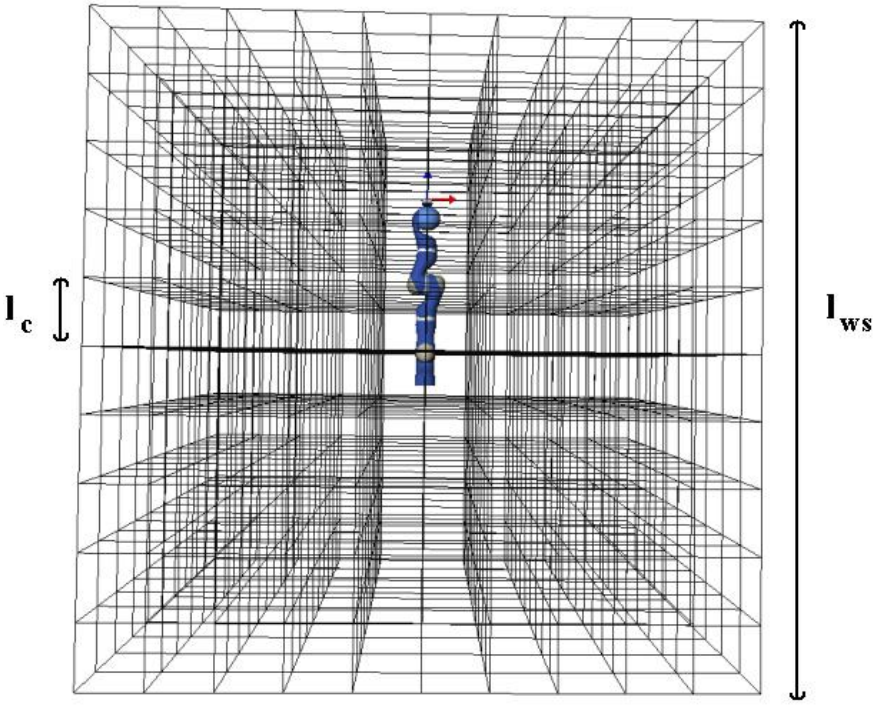


Fig. 4.1 A homogeneous partitioning of the robot workspace is performed. The maximum workspace of the DLR LWR is overestimated by the axis-aligned bounding box. For visualization, a coarse voxelization is chosen. The workspace is divided into voxels of 300 mm edge length.

$$n_c = \lceil l_{ws}/l_c \rceil \tag{4.9}$$

$\lceil x \rceil$ denotes the ceiling operator which always rounds to the smallest integer that is not less than x (Equation 4.10).

$$\lceil x \rceil = min\{n \in \mathbb{Z} | n \geq x\} \tag{4.10}$$

The whole voxel space V_{Robot} has n_c^3 voxels and is described by Equation 4.11.

$$V_{Robot} := \left\{ \mathbf{g} \in \mathbb{N}^3 | \ \|\mathbf{g}\|_\infty \leq n_c \right\} \subset \mathbb{N}^3 \tag{4.11}$$

Each voxel is mapped to a unique grid coordinate $\mathbf{g} \in \mathbb{N}^3$ that represents the center of the voxel. $\|.\|_\infty$ is the maximum norm of a vector. In the following, the term *voxel* and *grid coordinate* are used synonymously. In this book, the origin of V_{Robot}, i.e. $\mathbf{g} = (0,0,0)^T$ is the lower left corner of the bounding box. A grid coordinate $\mathbf{g}$ is element of $\mathbb{N}^3$. This is reflected in Equation 4.12 and Equation 4.13. The mapping of a Cartesian position $\mathbf{t} = (t_x, t_y, t_z) \in \mathbb{R}^3$ is given as

$$\begin{aligned} v : \mathbb{R}^3 \rightarrow \mathbb{N}^3; \mathbf{t} \mapsto \mathbf{g} := v(\mathbf{t}) &= \lceil \tfrac{1}{l_c} \cdot \mathbf{t} \rceil + (\tfrac{n_c}{2} - 1) \cdot \mathbf{1} \\ &= \begin{pmatrix} \lceil \frac{t_x}{l_c} \rceil \\ \lceil \frac{t_y}{l_c} \rceil \\ \lceil \frac{t_z}{l_c} \rceil \end{pmatrix} + \begin{pmatrix} \frac{n_c}{2} - 1 \\ \frac{n_c}{2} - 1 \\ \frac{n_c}{2} - 1 \end{pmatrix} \end{aligned} \tag{4.12}$$

If $\mathbf{t}$ is outside of the bounding cube with length l_{ws} encasing the robot workspace, it is not part of the robot workspace and thus by construction not in its representation. Correspondingly, the coordinate $\mathbf{t} \in \mathbb{R}^3$ of a voxel with grid coordinate $\mathbf{g}$ is $w(\mathbf{g})$.

$$w : \mathbb{N}^3 \rightarrow \mathbb{R}^3; \ \mathbf{g} \mapsto \mathbf{t} := w(\mathbf{g}) = (\mathbf{g} + (1 - \frac{n_c}{2}) \cdot \mathbf{1}) \cdot l_c - \frac{l_c}{2} \cdot \mathbf{1} \tag{4.13}$$

In this work, the voxels are stored in an one-dimensional, connected field called *linear voxel space*, which provides fast random access to each element [14]. The memory requirements are proportional to n_c^3. For fast access the whole representation is kept in main memory. Therefore, the size of the main memory restricts the minimum possible edge length l_c of the voxels. If arbitrarily small voxels are required, the representation of the voxel space through an octree allows to keep only the currently relevant parts of the workspace in main memory. However, for manipulation tasks as considered in this book, the linear voxel space implementation is sufficient.

4.2.2 Construction of the Model

In this section, the construction of the reachability sphere map based on the voxel space V_{Robot} is described. The pose of the TCP is described by a homogeneous matrix F (Equation 4.1). Using Equation 4.12, it is determined which voxel a TCP

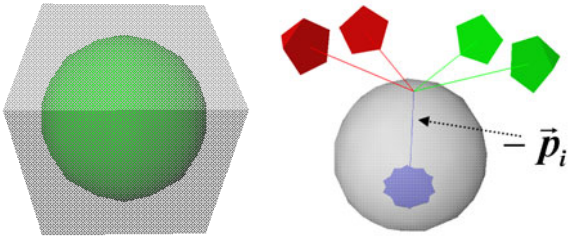


Fig. 4.2 Shows **(a)** a sphere inscribed into a voxel and **(b)** two exemplary frames F_{i,α_k} for a point on the sphere. The blue arrow corresponds to the z-axis, which is the same for both frames. The x-axis is shown in red and the y-axis is shown in green.

pose belongs to. The Cartesian position $\mathbf{t}$ is mapped to the reachability sphere map position $\mathbf{g}$, i.e. to the voxel located at this position.

The voxels serve as containers for the data represented in the reachability sphere map. The aspect that distinguishes the reachability sphere map from other representations of the workspace is the representation of position and orientation information. In this book, the voxels store for each member of a set of TCP poses whether it is reachable, i.e. an inverse kinematics solution exists. The set of TCP poses represents a discretization of $SO(3)$ and is generated as follows. Into each voxel, a sphere with a diameter equal to the edge length l_c of a voxel is inscribed (Figure 4.2(a)). The center of the sphere is identical with the center of the voxel. Using the spiral point algorithm proposed by Saff et al. [96], on a sphere a set P of n_p uniformly distributed points $\mathbf{p}_i$ is generated.

$$P := \left\{\mathbf{p}_0, .., \mathbf{p}_{n_p-1}\right\} \subset \mathbb{R}^3 \tag{4.14}$$

$\mathbf{p}_i$ denotes the position vector in a coordinate system placed at the center of the sphere. The set N_p denotes the set of point indices i.

$$N_p := \left\{0, .., n_p - 1\right\} \subset \mathbb{N} \tag{4.15}$$

For each point $\mathbf{p}_i$ with index i on the sphere, homogeneous coordinate frames are constructed. $-\mathbf{p}_i$ serves as the z-axis of the rotation matrix $R_{i,0}$ of the frame $F_{i,0}$. The x and y axes of $R_{i,0}$ are chosen to form a right hand coordinate system.

$$F_{i,0} := F(R_{i,0}, \mathbf{p}_i) = \begin{pmatrix} R_{i,0} & \mathbf{p}_i \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{4.16}$$

The resulting frame is then rotated by α_k around its z-axis according to a fixed stepsize Δ_o. Let N_o denote a set containing $m_o = \lfloor \frac{360^\circ}{\Delta_o} \rfloor$ orientation indices k.

$$N_o := \{0, .., m_o - 1\} \subset \mathbb{N} \tag{4.17}$$

$\lfloor x \rfloor$ denotes the floor operator which always rounds to the largest integer not greater than x (Equation 4.18).

$$\lfloor x \rfloor = max\{m \in \mathbb{Z} | m \leq x\} \tag{4.18}$$

Thus, the orientations α_k are defined as

$$\alpha_k = k \cdot \Delta_o; \;\; k \in N_o \tag{4.19}$$

and used to describe the frames $F_{i,\alpha_k} \in H$ as

$$F_{i,\alpha_k} = F_{i,0} \cdot F_z(\alpha_k) = F(Rot(i,k), \mathbf{p}_i) \tag{4.20}$$

$Rot(i,k)$ is the rotation matrix component of F_{i,α_k} and $F_z(\alpha_k)$ is a homogeneous matrix that performs a rotation by α_k about the z-axis.

$$F_z(\alpha_k) = \begin{pmatrix} R_z(\alpha_k) & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{4.21}$$

Thus each voxel contains $n_p \cdot m_o$ frames. These $n_p \cdot m_o$ frames F_{i,α_k} represent poses given in a coordinate system placed at the sphere center and are the same for all voxels. The resulting set of frames is referred to as O_S.

$$O_s := \left\{ F(Rot(i,k), \mathbf{p}_i) | \mathbf{p}_i \in P, i \in N_p, k \in N_o \right\} \subset H \tag{4.22}$$

In Figure 4.2(b) two exemplary frames are shown for a point $\mathbf{p}_i$ and orientation indices k and $k-1$. The x-axis (red) and the y-axis (green) are tangential to the sphere and the z-axis (blue) is pointing towards its center. The parameters (l_c, n_p, Δ_o) determine how well the model represents the versatile workspace. The effect of the parameter selection is examined in Section 4.3.

The inverse kinematics for the arm is used to determine for each voxel $\mathbf{g} \in V_{Robot}$ which frames F_{i,α_k} can be reached by the TCP of the arm. Let T^{Base}_{Sphere} be the transformation from the robot base frame to the coordinate system placed at the center of the voxel $\mathbf{g}$ which is also the center of the inscribed sphere. The TCP frame F^{Base}_{TCP} in the robot base coordinate system is computed given transformation T^{Base}_{Sphere} and the individual frames $F_{i,\alpha_k} \in O_s$.

$$T^{Base}_{Sphere}(\mathbf{g}) = F(I, w(\mathbf{g})) = \begin{pmatrix} I & w(\mathbf{g}) \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{4.23}$$

$$F^{Base}_{TCP} = T^{Base}_{Sphere}(\mathbf{g}) \cdot F_{i,\alpha_k} \tag{4.24}$$

I is a 3×3 identity matrix. Thus for a voxel $\mathbf{g}$ the set of frames

$$\left\{ T^{Base}_{Sphere}(\mathbf{g}) \cdot F_{i,\alpha_k} |\; i \in N_p,\; k \in N_o \right\} \tag{4.25}$$

is examined using the inverse kinematics. A frame F_{i,α_k} in voxel $\mathbf{g}$ is represented by a tuple $(\mathbf{g},i,k)$ in the reachability sphere map. Thus the reachability sphere map is described by the set M_S.

$$M_S = \left\{ (g_1,g_2,g_3,i,k) \in \mathbb{N}^5 |\ \mathbf{g} = (g_1,g_2,g_3)^T \in V_{Robot}, i \in N_p, k \in N_o \right\} \quad (4.26)$$

(g_1,g_2,g_3,i,k) and $(\mathbf{g},i,k)$ are used synonymously. If the inverse kinematics returns a valid solution for F_{TCP}^{Base}, the corresponding frame F_{i,α_k} is reachable. In the model this is represented by setting the respective entry $(\mathbf{g},i,k)$ in the data structure to true.

For most non-redundant robot arms an analytic inverse kinematics is available [19]. If the inverse kinematics problem is not solvable, the TCP pose is definitely not reachable. Using an analytical inverse kinematics the voxel space is traversed systematically and each sphere is examined only once.

However, the inverse kinematics of redundant arms cannot be solved analytically. Optimization procedures are applied in order to determine the value of the redundant degrees of freedom. Therefore, different initial configurations influence the success of the inverse kinematics algorithm. Its failure does not guarantee that the TCP pose is not reachable. In this book random configurations $\mathbf{q}_r \in C$ are drawn. Via the direct kinematics $K(\mathbf{q}_r) = F(R,\mathbf{x})$ the TCP pose is computed which determines the voxel $\mathbf{g} = v(\mathbf{x}) \in V_{Robot}$ to be examined next. Furthermore, $\mathbf{q}_r$ is used to start the inverse kinematics with different initial solutions and thus start the search for a solution in different parts of the C-Space. This effectively serves to explore the nullspace of the robot arm. Otherwise, the map represents only the strengths and weaknesses of the inverse kinematics algorithm in specific areas of the workspace. The configuration

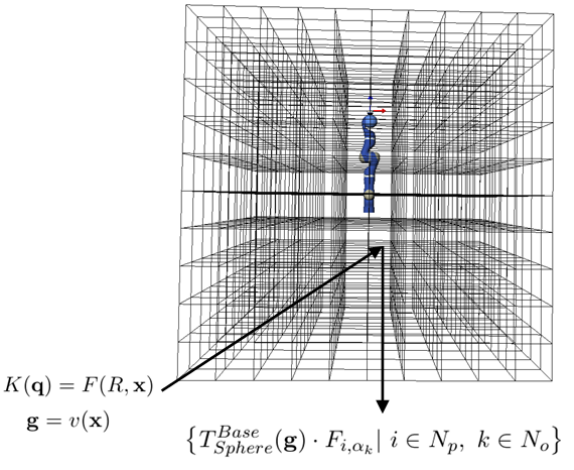


Fig. 4.3 For the random configuration q_r, the TCP Frame $F(R,\mathbf{x})$ is computed which is then mapped to its corresponding voxel $\mathbf{g}$. For the voxel $\mathbf{g}$ the frame set can be extracted which is then examined using the inverse kinematics.

Algorithm 1. BuildSphereMap(sphereMap,nrOfSamples)

```
/* n_p points on a sphere */
pointsPerSphere←sphereMap.GetPointsPerSphere()
/* m_o orientations about z */
zOrientations←sphereMap.GetZOrientationsPerPoint()
for j ∈ nrOfSamples do
   /* draw random configuration */
   q_r ← randomConfiguration()
   /* compute TCP pose */
   frame4TCP←directKinematics(q_r)
   /* see Equation 4.12 */
   sphereNr←sphereMap.map2Sphere(frame4TCP)
   for i ∈ pointsPerSphere do
      for k ∈ zOrientations do
         /* see Equation 4.20 */
         frame4IK←sphereMap.getTCPFrame(sphereNr, i, k)
         /* solve inverse kinematics */
         ikResult←solveIK(q_r, frame4IK, q_solution)
         /* if IK solution found */
         if ikResult then
            /* set entry in map to true */
            sphereMap.setPointAndOrientation(sphereNr, i, k)
         end if
      end for
   end for
end for
```

space is randomly sampled according to a uniform distribution by a pseudo random number generator. Here, it is important that the pseudo random number generator has a large period. Therefore, the Mersenne Twister random number generator [73] is used. The algorithm describing the map construction for a redundant robot is summarized in Algorithm 1.

The spheres represent the reachability of a voxel. Therefore, they are called **reachability spheres**. The reachability sphere map is the aggregation of all spheres. Two measures called the **reachability index** D (Equation 4.29) and the **reachability index with z-orientation** D_o (Equation 4.32) are assigned to each sphere characterizing the reachability of the region enclosed by the sphere $\mathbf{g}$. In Equation 4.27, n_p is the total number of points on a sphere. For each point $\mathbf{p}_i$, if *one* of the rotated frames F_{i,α_k} is reachable, $R(\mathbf{g})$ is increased by one.

$$R(\mathbf{g}) = \sum_{i=0}^{n_p-1} IsReachable(\mathbf{g}, \mathbf{p_i}) \tag{4.27}$$

$$IsReachable(\mathbf{g}, \mathbf{p_i}) = \begin{cases} 1 \text{ if } \exists k \in N_o : (\mathbf{g}, i, k) \in M_s \text{ is reachable} \\ \quad \text{in reachability sphere map} \\ 0 \text{ otherwise} \end{cases} \tag{4.28}$$

The reachability index $D(\mathbf{g})$ represents the percentage of *points* on the sphere $\mathbf{g}$ that have *at least one* inverse kinematics solution.

$$D(\mathbf{g}) = \frac{R(\mathbf{g})}{n_p} \cdot 100 \text{ with } R(\mathbf{g}) \leq n_p;\ \ D(\mathbf{g}) \in [0,100] \tag{4.29}$$

It is used when the rotation about the z-axis of the frame $F_{i,0}$ (compare Equation 4.20) is not important. The reachability index with z-orientation $D_o(\mathbf{g})$ measures how far the region is from being in the dexterous workspace. To compute $D_o(\mathbf{g})$, $R_o(\mathbf{g})$ is increased by one for *each* reachable frame F_{i,α_k}.

$$R_o(\mathbf{g}) = \sum_{i=0}^{n_p-1} \sum_{k=0}^{m_o-1} IsReachable(\mathbf{g}, \mathbf{p_i}, \alpha_k) \tag{4.30}$$

$$IsReachable(\mathbf{g}, \mathbf{p_i}, \alpha_k) = \begin{cases} 1 \text{ if } (\mathbf{g}, i, k) \in M_S \text{ is reachable} \\ \quad \text{in reachability sphere map} \\ 0 \text{ otherwise} \end{cases} \tag{4.31}$$

The reachability index with z-orientation $D_o(\mathbf{g})$ represents the percentage of *frames* on the sphere $\mathbf{g}$ that have an inverse kinematics solution.

$$D_o(\mathbf{g}) = \frac{R_o(\mathbf{g})}{n_p \cdot m_o} \cdot 100 \text{ with } R_o(\mathbf{g}) \leq n_p \cdot m_o;\ \ D_o(\mathbf{g}) \in [0,100] \tag{4.32}$$

A sphere where all frames are reachable receives $D(\mathbf{g}) = 100$ and $D_o(\mathbf{g}) = 100$. A region with a value $D_o(\mathbf{g}) = 100$ belongs to the dexterous workspace if the voxel size $l_c \to 0$.

4.2.3 Reachability Predictions

The reachability sphere map can directly be used to predict the reachability of a given TCP pose. The TCP pose is assumed to be presented as a homogeneous matrix. According to the reachability sphere map a frame F is reachable if the discretized correspondent in the map is reachable. Therefore frame F is mapped to its representation $(\mathbf{g}, i, k) \in M_S$ in the reachability sphere map.

First, the translation component $\mathbf{t}$ of the homogeneous matrix F is used to determine to which sphere with grid coordinate $\mathbf{g}$ (Equation 4.12) the frame belongs. Next, the point index i on the sphere is determined that best represents the frame. This is performed by searching for the point $\mathbf{p}_i$ on the sphere that has the smallest angle with the z-axis $\mathbf{z}$ of the frame F. In Equation 4.33, $-\mathbf{z}$ is used because the frames F_{i,α_k} are constructed with $-\mathbf{p}_i$ as their z-axis.

$$i = \text{argmin}_{j \in N_p}(\ \arccos(< \frac{\mathbf{p_j}}{\| \mathbf{p_j} \|}, \frac{-\mathbf{z}}{\| -\mathbf{z} \|} >)) \tag{4.33}$$

$<,>$ denotes the inner product between two vectors. In a last step, the best fitting discretized rotation about the z-axis is determined. Let P_{xy} be a matrix that projects onto the xy-plane.

$$P_{xy} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{4.34}$$

Let $R_{i,0}$ be the rotation matrix component of the homogeneous frame $F_{i,0}$ describing the frame for point p_i and rotation $\alpha_k = 0$ (Equation 4.16). The x-axis $\mathbf{x}$ of frame F is projected onto the xy-plane of the coordinate system given by $R_{i,0}$ using Equation 4.35.

$$\hat{\mathbf{x}} = R_{i,0} \cdot P_{xy} \cdot R_{i,0}^{-1} \cdot \mathbf{x} \tag{4.35}$$

Then the angle β between the projected x-axis $\hat{\mathbf{x}}$ and the x-axis $\mathbf{x}_{i,0}$ of $R_{i,0}$ is determined (Equation 4.36).

$$\beta = \arccos(< \frac{\hat{\mathbf{x}}}{\| \hat{\mathbf{x}} \|}, \frac{\mathbf{x}_{i,0}}{\| \mathbf{x}_{i,0} \|} >) \tag{4.36}$$

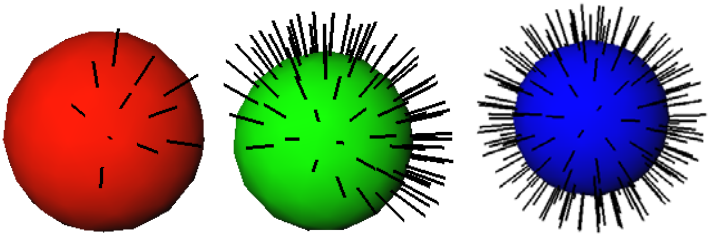
The index $k \in N_o$ of the discretized angle (Equation 4.37) represents the discretized rotation about the z-axis of frame $F_{i,0}$ that best matches frame F.

$$k = \lfloor \frac{\beta}{\Delta_o} + \frac{1}{2} \rfloor;\ k \in N_o \tag{4.37}$$

As a result, frame F has its correspondence in the sphere with coordinate $\mathbf{g} \in V_{Robot}$ in the sphere map, point $\mathbf{p}_i$ and rotation α_k in the map. If the tuple $(\mathbf{g}, i, k)^T \in M_s$ is marked reachable in the map, the examined frame F is reachable.

4.2.4 Visualization of the Model

The information stored in the reachability sphere map is 3D information, i.e. positions from $\mathbb{R}^3$ and rotations from $SO(3)$. Due to the amount of data represented by the reachability sphere map not every component of the 3D information can be visualized. For visualization of the reachability for each voxel, a point on the sphere is considered reachable if one of the corresponding TCP frames F_{i,α_k} is reachable. A sphere with diameter l_c is drawn in the voxel. For each reachable point on a sphere, a line segment originating in the sphere center is drawn (Figure 4.4). It perforates the sphere at the corresponding point. The reachability index can be used to visualize some structure inherent to the robot arm workspace. This is achieved by coloring the spheres with respect to their reachability index D or with respect to their reachability index with z-orientation D_o. In Figure 4.5, the change of the reachability index across the robot arm workspace is presented. Towards the interior of the workspace, the index increases and reaches its optimum in the blue region. Moving then towards the robot arm base, decreases the index. For better visibility, in some figures the full workspace is cut in half along the arm as shown in Figure 4.6.



(a) **(b)** **(c)**

Fig. 4.4 Spheres with different reachability indices are shown. The reachable points are shown as line segments. **(a)** The sphere has a reachability index $D = 5$. **(b)** The sphere has a reachability index $D = 45$. **(c)** The sphere has a reachability index $D = 98$.

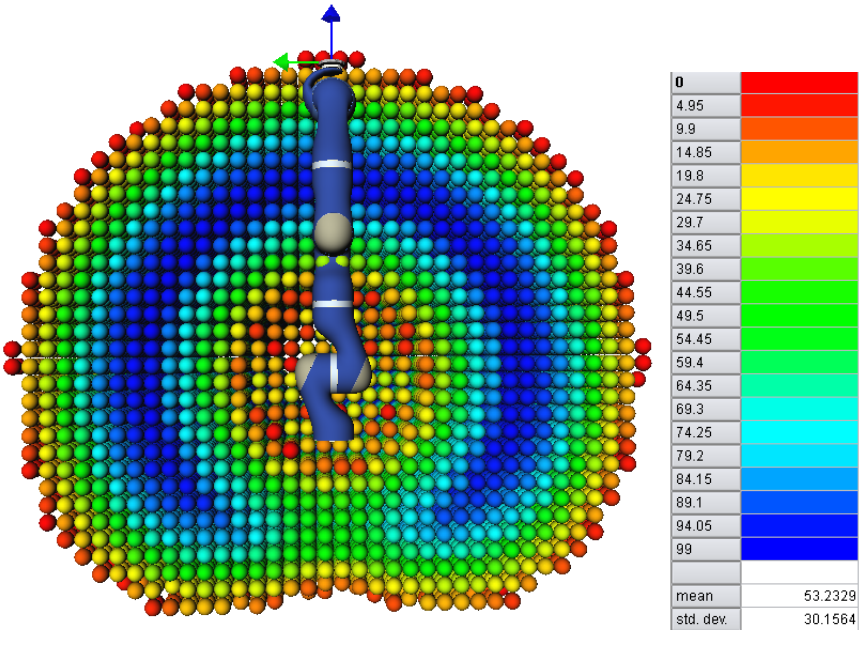


Fig. 4.5 The reachability spheres across the workspace colored with respect to the reachability index D. The workspace representation is cut as shown in Figure 4.6 for better visibility of the structure. The regions in the center of the workspace (blue regions) can be reached with the largest number of different poses.

In Figure 4.7, the spheres are colored with respect to D_o. Again, voxels in the middle of the workspace can be reached with the most orientations. Compared to Figure 4.5, it can be seen that voxels in the lower, middle part of the workspace can be reached with fewer poses than other voxels in middle of the workspace. The visualization using D_o therefore better reflects the effect of the link limits. The

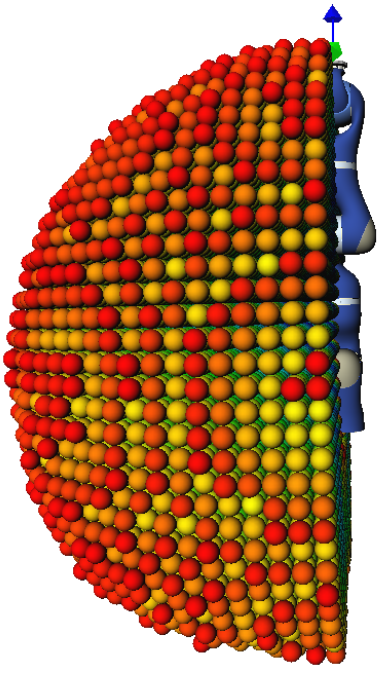


Fig. 4.6 The workspace of the robot arm is cut in half along the center axis of the arm. Only the left half-space is visualized.

reachability index D_o has a minimum value $D_o = 0$ and a maximum value $D_o = 74$ for the used robot arm. This shows that the dexterous workspace for the chosen TCP is empty. The pose of the TCP in Figure 4.7 is illustrated by the coordinate system attached to the flange of the robot. The z-axis is drawn as a blue arrow and the y-axis is drawn as a green arrow.

Figure 4.8 (top) shows all spheres with an index $D \in [0, 10]$ across the workspace. As expected spheres with a low value are found on the border of the workspace. Figure 4.8 (bottom) shows spheres with an reachability index $D \in [89, 99]$ across the workspace. It can be seen that spheres with a high value for index D lie on a sphere shell around the robot arm base (also compare with Figure 4.5) with a diameter of approximately half the robot arm length. Without link limits, a complete sphere shell would be obtained. The visualization validates the plausibility of the reachability sphere map. The kinematic capabilities of the robot arm at the border and in the center of the workspace appear as expected because the set of reachable poses in center regions is much larger than at the border of the workspace.

For Figure 4.5 to Figure 4.8, the map is constructed with 10^6 random samples. The spheres have a diameter of 50 mm and 200 points are distributed on a sphere. The stepsize Δ_o for turning the frame around its z-axis is 30° resulting in 12 frames per point on the sphere. In the next section, the effects of the parameters on validity of the reachability sphere map are examined.

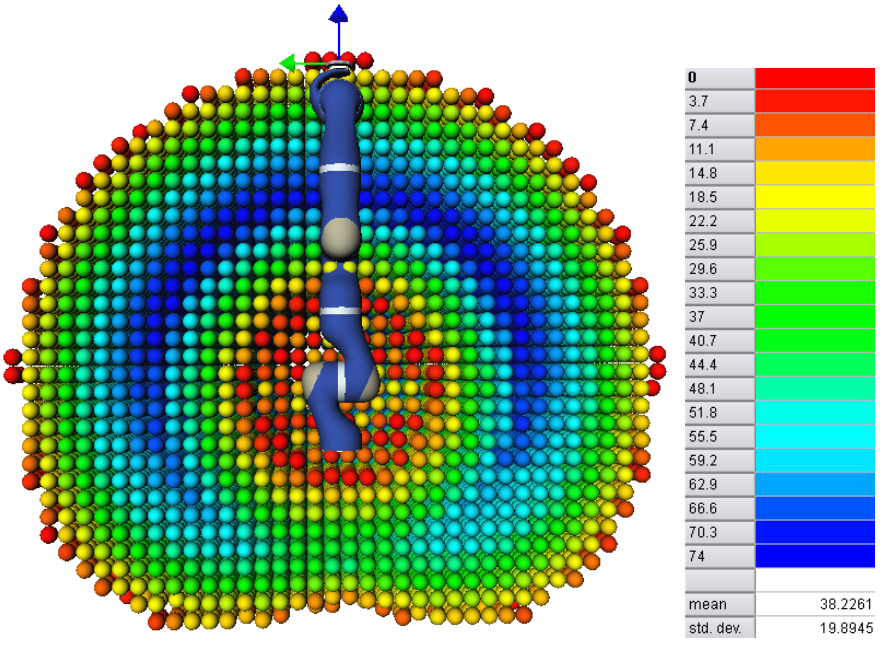


Fig. 4.7 The reachability spheres across the workspace colored according to the reachability index with z-orientations D_o. It can be seen that regions at the bottom of the workspace can be reached with fewer poses due to link limits.

4.3 Characteristics of the Representation

One of the fundamental assumptions of the developed model is that each reachability sphere describes the reachability characteristics for the corresponding voxel. For a given end-effector pose, the voxel and the representative frame on the sphere are determined using the method described in Section 4.2.3. If the representative $(\mathbf{g}, i, k) \in M_S$ in the map is reachable, the original pose is predicted to be reachable. The correctness of this prediction depends on the chosen sphere diameter (voxel size) and the number of points per sphere. Furthermore for redundant robots, it is influenced by the number of samples drawn during map building. In the following, the effect of the parameters on the quality of the reachability sphere map is examined. The number of samples, the sphere diameter l_c, the number of points n_p on the sphere and the z-orientation step size Δ_o are varied.

4.3.1 The Number of Samples

For a redundant robot arm, the number of samples affects how well the reachability sphere map represents the capabilities of the robot arm throughout the workspace. To examine the influence, maps are built for the DLR LWR with different numbers

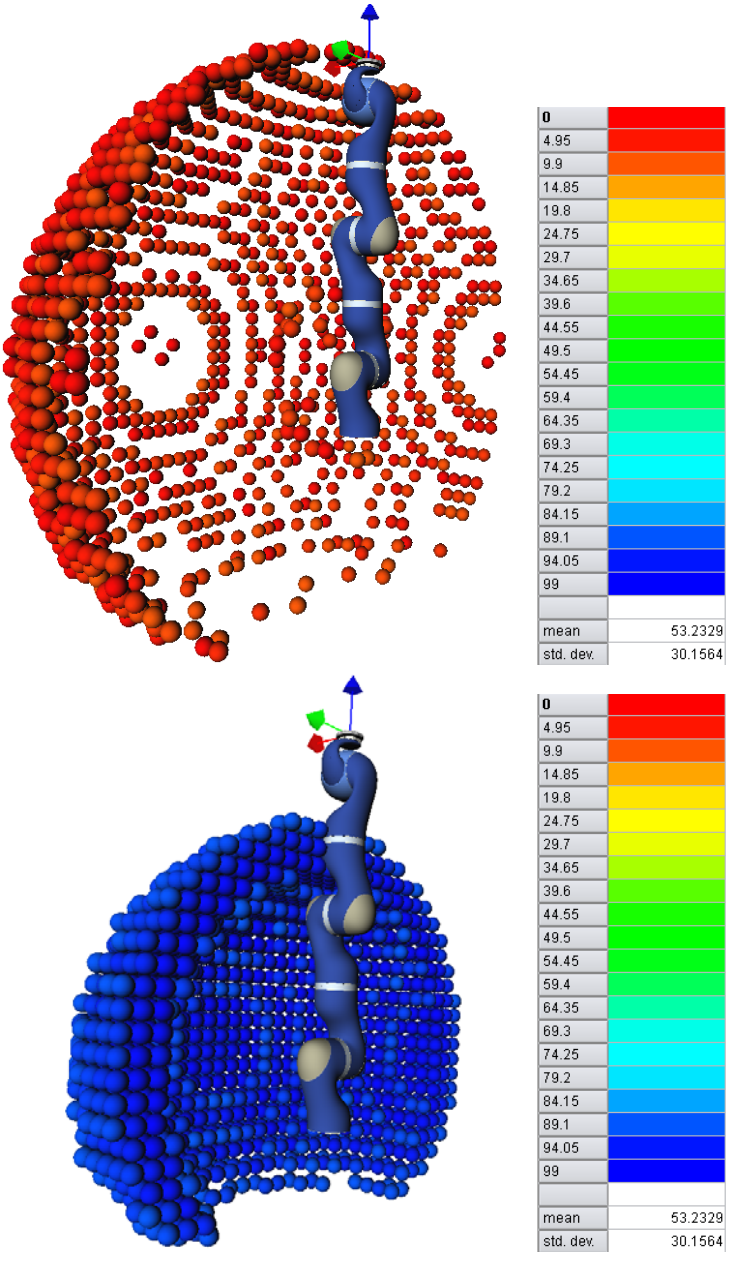


Fig. 4.8 The spheres with an index D in the lowest 10% of the reachability index **(top)** and in the upper 10% of the reachability index **(bottom)**. The spheres where the most frames are reachable lie in the center of the workspace. At the border of the workspace only few frames are reachable per sphere.

of samples. The kinematics description of this arm can be found in Section B.2 and the TCP frame is listed in Section B.7.1. In Figure 4.9 (a) it is shown how many formerly unknown spheres are examined and added to the map when the number of samples is increased from $2 \cdot 10^5$ to $2 \cdot 10^6$. The number of newly introduced spheres decreases strongly once 10^6 samples are drawn. Newly added spheres are mostly located at the outer and inner border of the workspace (Figure 4.10). As the number of samples approaches infinity, the number of newly added spheres approaches zero. This is shown in Figure 4.9 (a). An evaluation of the representation quality is provided in Section 4.3.5. Manipulation tasks take place in the center of the workspace where the possibilities to reach a region are manifold. The border regions can only be reached in few poses and are therefore not important for manipulation tasks. No manipulation will take place there. Nevertheless, the border of the workspace is well represented. It is concluded from this analysis that a sample size of 10^6 is sufficient to build a valid reachability sphere map for the 7 DOF LWR robot arm.

In Figure 4.11 color encodes how often spheres are visited during map building. The map is built with 10^6 samples. A large number of samples falls into regions that contain singular configurations of the robot arm. In singular configurations, large link movements result in small Cartesian displacements. Spheres that are visited less than 10 times are only found at the inner and outer border of the workspace.

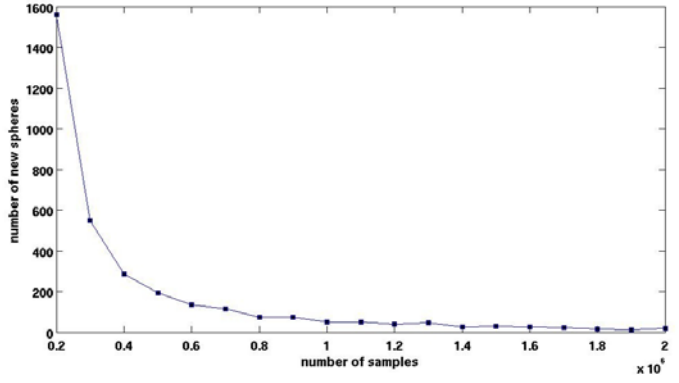
4.3.2 The Sphere Diameter l_c

The dependency of the prediction correctness on the sphere diameter i.e. voxel size is examined. The reachability sphere map for the DLR LWR is build as described using six different sphere diameters. All other parameters remain fixed. $n_p = 200$ points are distributed on the sphere and $\Delta_o = 30°$. Six versions of the capability map result.

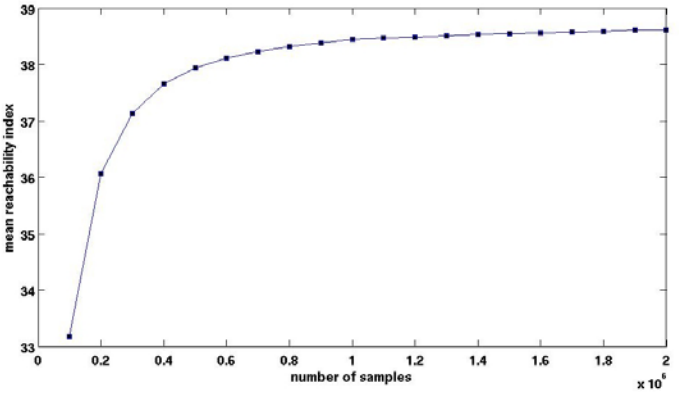
10^5 frames are randomly sampled. A frame is generated at a random position with a random orientation in the robot workspace in the following manner. A random position vector $\mathbf{t} = (x, y, z) \in \mathbb{R}^3$ is drawn from the volume of a sphere with radius $\frac{l_{ws}}{2}$ containing the workspace. The sphere's center equals the center of the voxelization, i.e. the robot arm base. The position is obtained via drawing random spherical coordinates. The sphere radius r is drawn from $[0, l_{ws}]$. The angle θ is drawn from $[0, 180°]$ and angle ϕ is drawn from $[0, 360°]$. Using these parameter a Cartesian position can be computed from the spherical coordinates as described in Equation 4.38.

$$\begin{aligned} x &= r \cdot \sin\theta \cdot \cos\phi \\ y &= r \cdot \sin\theta \cdot \sin\phi \\ z &= r \cdot \cos\theta \end{aligned} \tag{4.38}$$

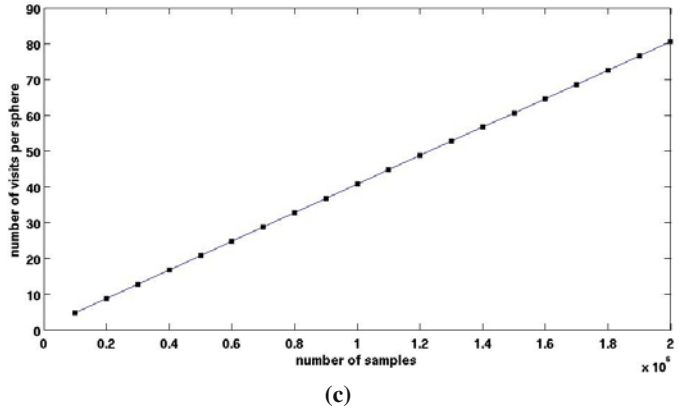
To generate an uniformly distributed random orientation, 1000 points are equally distributed on another sphere. A point index j is randomly drawn. A z-orientation γ_z is randomly drawn from $[0, 360°]$. These parameters are used to construct the rotation part of the frame F_{j,γ_z} as described in Section 4.2.2. Thus frames are obtained, that are uniformly distributed across the workspace.



(a)



(b)



(c)

Fig. 4.9 For the DLR light weight robot, the reachability sphere map is built with different numbers of samples. **(a)** The number of *new* spheres is plotted that are explored when the number of samples is increase from $2 \cdot 10^5$ to $2 \cdot 10^6$ in steps of $1 \cdot 10^5$. **(b)** The development of the mean reachability index D_o including z-orientations is shown as the number of samples is increased. **(c)** The mean number of visits per sphere is shown for increasing numbers of samples.

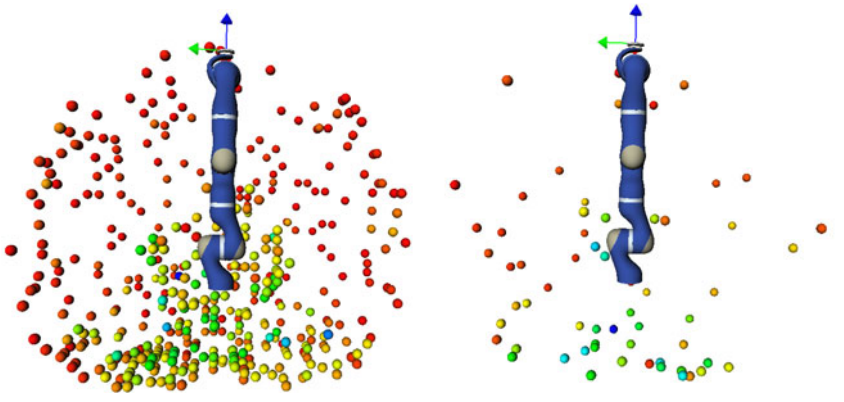


Fig. 4.10 The location of new spheres, that are added to the model, is shown. In both cases, spheres are only added at the inner and outer border of the workspace. **(left)** The spheres are shown that are added for a reachability map build with $4 \cdot 10^5$ samples compared to a map build with $2 \cdot 10^5$ samples. **(right)** The spheres are shown that are added for a reachability map build with $1 \cdot 10^6$ samples compared to a map build with $8 \cdot 10^5$. The number has decreased strongly.

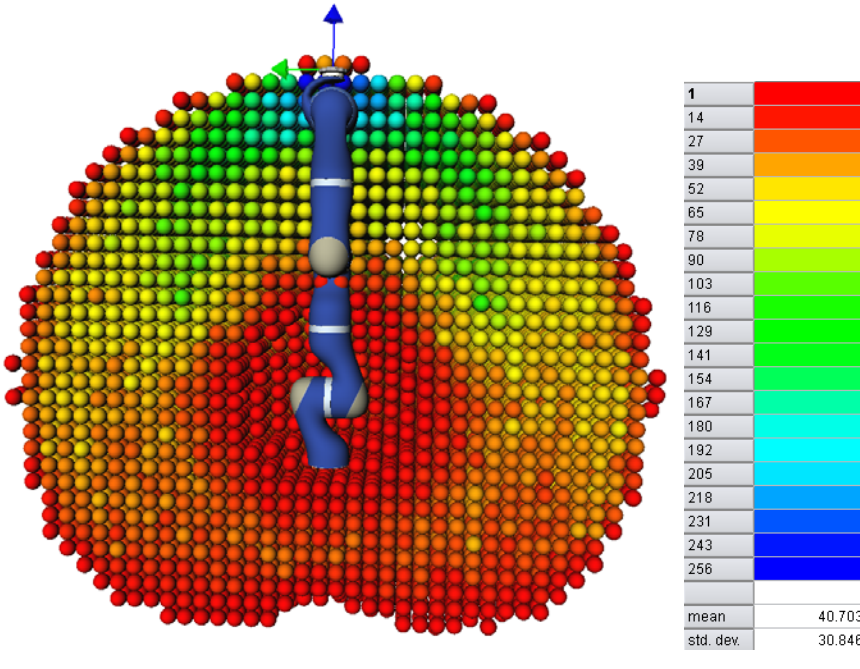


Fig. 4.11 The color encodes how often spheres are visited during building the reachability sphere map for the DLR LWR with 10^6 samples. Regions that contain singularities are often visited.

Table 4.1 The prediction accuracy is shown for different reachability sphere maps of the DLR LWR . The sphere diameter l_c is varied. All other parameters remain fixed. 200 points are distributed on each sphere and Δ_o is 30°. The prediction accuracy degrades with increasing sphere diameter.

sphere diameter l_c in mm	50	80	110	140	170	200
prediction accuracy in %	94.6	93.5	91.9	90.0	89.2	88.3
mean D_o	38.2	37.0	34.6	33.7	31.0	29.6

Using the different reachability sphere maps it is predicted whether the generated frame is reachable. The prediction is compared with the result of the inverse kinematics. Here, the inverse kinematics iterates the value in the initial configuration that corresponds to the link labeled *redundant*. This inverse kinematics is called iteratedIK. Thus, different parts of the C-space are searched for a solution. The prediction accuracy measures the correctness of predictions made by the reachability sphere map. The number of correct predictions is divided by the total number of predictions. To obtain a percentage, the result is multiplied by 100.

Table 4.1 shows the prediction accuracy for different sphere diameters. The accuracy degrades as the sphere diameter is increased. Also, the mean reachability index D_o is not representative anymore as the sphere diameter is increased. The highest prediction accuracy is achieved with a sphere diameter of 50 mm. With the current implementation, the diameter cannot be further decreased as the memory requirements for the reachability sphere map increase cubically with the decrease of the sphere diameter.

4.3.3 The Number of Points n_p per Sphere

While keeping all other parameters fixed, the number of points n_p per sphere is varied. The sphere diameter l_c with the best prediction accuracy as determined in the previous section is chosen. The sphere diameter is $l_c = 50mm$ and the orientation stepsize $\Delta_o = 30°$. The reachability sphere map for the DLR LWR is build with four different values for n_p. 10^5 frames are randomly sampled as described in the previous section. For each reachability sphere map the prediction accuracy is computed. In Table 4.2 the dependency of the prediction accuracy on the number of points on the sphere is shown. The prediction accuracy does not change strongly as the number of points is increased from 50 to 200. To achieve optimal representation, the reachability sphere map is built with 200 points per sphere for the examined robot arm.

Table 4.2 The prediction accuracy is shown for the reachability sphere map of the DLR LWR for different numbers of points n_p per sphere. The sphere diameter is $l_c = 50mm$.

points n_p per sphere	50	100	150	200
prediction accuracy in %	93.4	94.1	94.5	94.6

4.3.4 The z-Orientation Step Size Δ_o

The step size Δ_o for the rotation α_k around the z-axis (Equation 4.19) is varied. It determines the resolution of the angle α_k and thus the number of frames examined per point on the sphere. If $\Delta_o = 360°$ the z-orientation is neglected and only one frame F_{i,α_k} is examined per point index i. For very small Δ_o, the z-orientation is finely sampled and many frames F_{i,α_k} result.

All other parameters are fixed. The sphere diameter is $l_c = 50mm$ and $n_p = 200$ points are distributed on the sphere. The reachability sphere map for the DLR LWR is build with four different values for Δ_o. 10^5 frames are randomly sampled. For each reachability sphere map the prediction accuracy is computed. In Table 4.3 the dependency of the prediction accuracy on the z-orientation step size Δ_o is shown. The prediction accuracy does not change strongly as the number of points is increased from 15° to 60°. The step size $\Delta_o = 30°$ is determined to be most preferable. It provides a good compromise between the memory consumption of the representation and the prediction accuracy.

Table 4.3 For the reachability sphere map of the DLR LWR , the prediction accuracy is shown for different values of Δ_o (in degrees). The sphere diameter is 50 mm and 200 points are distributed on a sphere. The prediction accuracy degrades slightly as Δ_o is increased.

Δ_o	15	30	45	60
prediction accuracy in %	94.8	94.6	94.4	94.0

4.3.5 Reachability of TCP Poses

The reachability sphere map is developed to predict the reachability of a TCP pose and thereby facilitate scene reasoning. In this section it is evaluated in more detail how accurate predictions by the reachability sphere map are. The best parameters are extracted from the last sections, the sphere diameter $l_c = 50mm$, points per sphere $n_p = 200$ and $\Delta_o = 30°$. The maps of two redundant robot arms are examined, that of the Kuka LWR arm (Figure 4.12) and that of the DLR LWR arm (Figure 4.7). The kinematics description of the Kuka LWR is listed in Appendix B.3. The two robots use the same modular links but differ in their arrangement. Furthermore the last axis of the DLR LWR performs a rotation about the y-axis of the world whereas the Kuka LWR performs a rotation about the axis aligned with the center axis of the robot arm. The TCP frames with respect to the world coordinate system are chosen to be identical for both robot arms (Section B.7.1). Furthermore the reachability sphere maps for two non-redundant 6 DOF robot arms are examined, the Schunk PowerCube arm and the Kuka Kr16 industrial robot arm. The kinematics parameters for both are listed in Appendix B. A visualization of their reachability sphere maps can be found in Section 5.1. The Kuka Kr16 is twice as long as the other arm, therefore a sphere diameter of $l_c = 100mm$ is chosen.

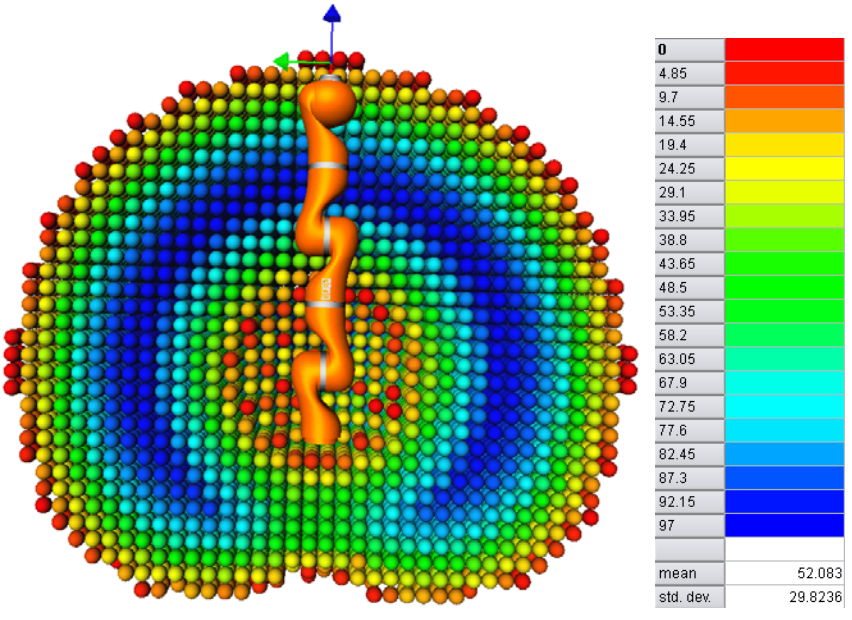


Fig. 4.12 The reachability sphere map for the Kuka LWR . The reachability spheres across the workspace colored with respect to the reachability index D_o. The regions in the center of the workspace (blue regions) can be reached with the largest number of different poses. The TCP frame is visualized as a coordinate system with the z-axis shown as a blue arrow.

10^5 frames are randomly sampled from the workspace. True positives are frames that are predicted to be reachable and can also be reached by the inverse kinematics. If a frame is predicted to be reachable but the inverse kinematics does not find a solution, this is a false positive. A true negative is a frame that is predicted to be unreachable and where the inverse kinematics solver fails to find a solution. A false negative is a frame predicted to be unreachable but where an inverse kinematics solution can be found. The sum of the true positives and the true negatives describes the percentage of correct predictions. The reachability sphere map has a prediction accuracy of 94.6% for the DLR LWR and 94.5% for the Kuka LWR (Table 4.4). Therefore the kinematic capabilities of both redundant robot arms are accurately represented by the reachability sphere map. The prediction accuracy for the maps of the PowerCube arm is 93.3% and for the Kuka Kr16 it is 94.5%. Thus the workspace of all examined robot arms is well represented.

4.4 Visualization of Workspace Structure

The reachability indices D and D_o are directionless measures. They help to recognize some basic structure, like which voxels are reachable in a versatile manner. The visualization scheme using spheres and lines introduced in Section 4.2.4 represents

Table 4.4 The prediction accuracy of the reachability sphere maps is evaluated for two 7 DOF robot arms and two 6 DOF robot arms using 100000 randomly sampled frames.

	DLR LWR	Kuka LWR	PowerCube arm	Kuka Kr16
true pos.	23.1	33.0	19.9	33.0
true neg.	71.5	61.5	73.3	61.5
accuracy	**94.6**	**94.5**	**93.3**	**94.5**
false pos.	2.5	2.5	3.0	2.5
false neg.	2.9	3.0	3.8	3.0

the simplest manner of visualizing the data. However, geometric structure present in the workspace cannot directly be recognized due to the huge amount of visualized data. In the following, it is examined how the geometric structure can be made visible.

4.4.1 Analyzing the Structures in the Workspace

To examine the structure typical for regions of the workspace, a line of spheres is extracted from the reachability sphere map. The line traverses the whole workspace. In Figure 4.13 (a) reachability spheres are shown in a line across the workspace marked by a box. For better recognizability, a zoomed view is shown in Figure 4.13 (b). When moving into the workspace (starting with the orange sphere) the number of points with valid inverse kinematics solutions increases. In detail, a cone like structure for the orange and light green spheres is observed. Moving further inward, these cones open up and the structure changes. A ring structure can be observed for the dark blue spheres. Double cone structures are also possible but not shown in Figure 4.13.

Capturing and approximating the structures using shape primitives such as cones and cylinders results in immense data reduction. Testing directions with these shape primitives is faster than testing single inverse kinematics solutions. To determine whether a TCP pose lies in a cone and thus is reachable, reduces to one computation of the angle between the shape's axis and the z-axis of the TCP frame. In Figure 4.14, the test of a TCP frame with a cone is shown.

4.4.2 Capturing the Structure to Construct a Map

In this section, it is described how and what shape primitives are fitted to the data. A measure is introduced for the relative error of an approximation. It is used to evaluate whether a shape primitive is a valid representation of the structure captured by the reachability sphere.

Resulting from the observations from the previous section, the shape primitives presented in Figure 4.15 are proposed. Cones can be used to approximate cone-like structures (Figure 4.15 (a)). The circular cone base area is used to represent the

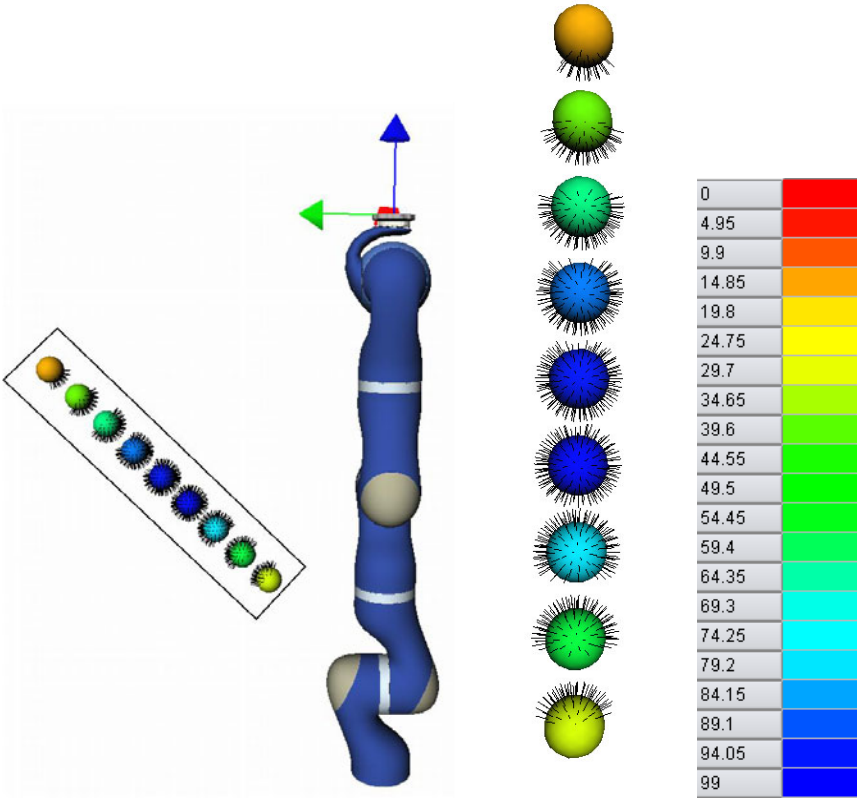


Fig. 4.13 **(a)** A line of sphere is extracted from the reachability sphere map of the robot arm. The line begins on the border of the workspace and moves towards the robot base. **(b)** A close up of the spheres with corresponding color table **(c)** for the reachability index D. The line covers continuous regions. The structures change gradually across the workspace.

data. A cylinder of type C1 (Figure 4.15 (b)) can capture double cone structures and a cylinder of type C2 (Figure 4.15 (c)) can capture ring structures. The cylinder of type C1 uses the two circular base areas to represent the data. The cylinder of type $C2$ uses the cylinder shell to represent the data. The process of fitting the shape primitives to the data involves optimizing the main axis of the shape primitive and its opening angle to best approximate the data. The axis is optimized using principal component analysis.

Principal component analysis (PCA) [44] is a powerful tool for analyzing and identifying patterns in data. Given a 3D dataset A, the eigenvectors of the dataset A are the principal axes of the data. The first principal axis describes the direction of the maximum data variation. The last principal axis is the direction of minimum data variation. Figure 4.16 shows the principal axes belonging to the two principal

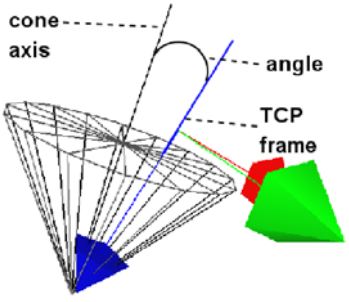


Fig. 4.14 For a TCP frame it is tested whether it is represented by the shape primitive.

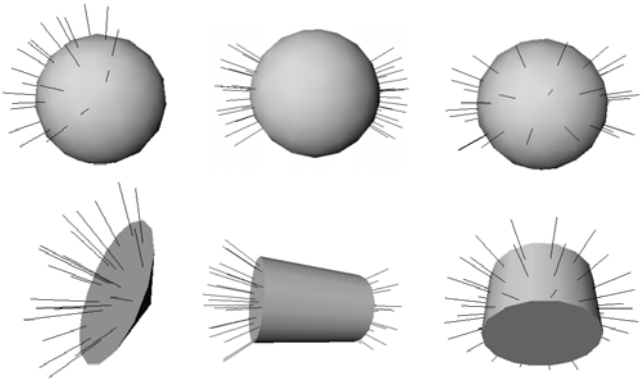


Fig. 4.15 Cones **(a)** and two cylinder types to capture structures. **(b)** Cylinder type C1, **(c)** Cylinder type C2

components for the given 2D data set. To determine which shape primitive best describes the present structure, the PCA is computed for the reachable points on the sphere. If they form a cone like structure, the principal axes for the first two components span the circular base of the cone. The last component collects the least variation. Its principal axis is taken to be the shape's axis. To fit a cylinder of type $C1$ to the data, the eigenvector for the largest eigenvalue is taken as the cylinders axis. To fit a cylinder of type $C2$ to the data, the eigenvalue for the smallest eigenvalue is assumed to be the cylinder's axis. For different opening angles, it is evaluated how well a shape primitive represents the data.

The representation quality of a shape primitive is determined by the number of valid inverse kinematics solutions that the shape captures (reachable points), and the number of points on the sphere that are covered, where no inverse kinematics solution can be computed (unreachable points). These observations are combined in the computation of the shape fit error (*SFE*). The SFE is computed for a shape primitive with axis **a** and opening angle γ (compare Figure 4.17). It is derived from the

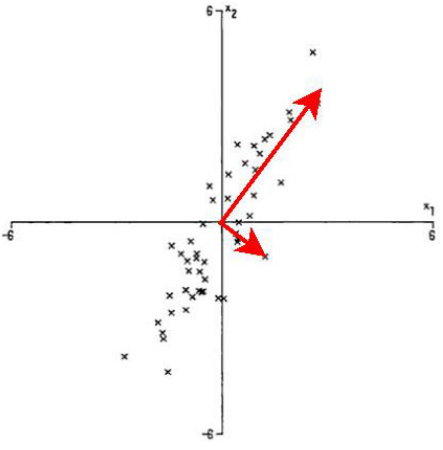


Fig. 4.16 A set of data points is distributed in the xy-plane. Principal Component Analysis is used to perform a dispersion analysis of the data. The resulting main axes of the dispersion are shown as red arrows.

relative error made by the shape fitting process to capture all inverse kinematics solutions available for a sphere. The ideal shape covers all $R(\mathbf{g})$ (Equation 4.27) inverse kinematics solutions available for a sphere $\mathbf{g} \in V_{Robot}$. Compared to an ideal approximation, a suboptimal shape fails to cover r reachable points and wrongly covers u unreachable points. The relative error of an approximation is defined by subtracting the ideal approximation value $R(\mathbf{g})$ from the value for a suboptimal shape and divide the result by the ideal value $R(\mathbf{g})$ (Equation 4.39). The scale factor 100 is introduced to obtain a percentage.

$$\left| \frac{(R(\mathbf{g}) - u - r) - R(\mathbf{g})}{R(\mathbf{g})} \right| = \frac{u+r}{R(\mathbf{g})} \; ; \; u, r, R(\mathbf{g}) \geq 0 \tag{4.39}$$

The shape fit error is then defined using this relative error.

$$SFE(\mathbf{g}, \mathbf{a}, \gamma) = \begin{cases} \frac{u+r}{R(\mathbf{g})} \cdot 100 & \text{if } u + r \leq R(\mathbf{g}) \\ 100 & \text{if } u + r > R(\mathbf{g}) \end{cases} \tag{4.40}$$

The SFE is limited to $[0, 100]$. If a shape has a relative error greater than 1, it is no better approximation than a shape with a relative error of 1. Both are unacceptable. Therefore both receive the maximum SFE of 100. The SFE allows to classify the data into the given structure categories.

Figure 4.17(b), (c) show a cylinder of type C1 with two different axes and opening angles fit to the same data set. While C1 collects all lines, it also covers many unreachable points distributed across the original sphere and receives the highest SFE of 100 as a result. The data is optimally represented by the second cylinder (Figure 4.17(c)) receiving the optimum SFE=0. Algorithm 2 details the process of

Algorithm 2. FitShape2SphereData(sphere)

```
/** sphere - contains reachablity information for frames */
/** extract the coordinates of all reachable points on the sphere */
currentData=ExtractReachablePoints(sphere)
/** for a number of iterations */
for i<maxIterations do
   /** compute the axis of the shape type to be fitted to the data */
   axis=ComputeShapeAxisUsingPCA(currentData)
   /** determine the opening angle of the shape
   for which the SFE is lowest*/
   angle,SFE=FindAngleWithBestSFE(currentData, axis)
   diffSFE=SFE-SFElastIteration
   /** if the SFE is improved, continue iterating, ε = 1e^-8 */
   if diffSFE<-ε then
      /** remember best values */
      bestAxis=axis
      bestAngle=angle
      SFElastIteration=SFE
   else
      break
   end if
   /** The best opening angle does not necessarily cover
   all data points. Using the reduced dataset a new shape
   axis is computed in the next iteration */
   currentData=GetFilteredDataset(currentData, axis, angle)
end for
```

fitting a shape to the data. First the shape axis is extracted using PCA. Then the opening angle is computed that results in the lowest SFE. Since the previously computed axis is not correct for the shape with the current opening angle, the process is repeated until a maximum number of iterations is exceeded or the SFE value for the shape does not improve anymore.

The shape fitting process can also be seen as a classification of the data. If the SFE for a shape is near zero, the data has a structure that is best represented by the respective shape class. The spheres are replaced by the best fitting cone or the best fitting cylinder of type C2. In Figure 4.18 and Figure 4.19, a cone shape map and cylinder shape map is shown. The color encodes the SFE. Shapes with a high SFE are red and shapes with a low SFE are blue. In the outer zones of the workspace cones are good approximations and have a low SFE shown by the blue coloring. In the inner workspace regions cones have high SFEs. Here, cylinders receive low SFEs (Figure 4.19). For the remaining workspace, the cylinder hypothesis is inadequate and the SFE is maximum. Hence, cone and cylinder representations complement each other. These results suggest that it is best to take the shapes with the lowest SFE from all three shape fitting processes. A mixed map optimally represents all structures found in the workspace for the DLR LWR.

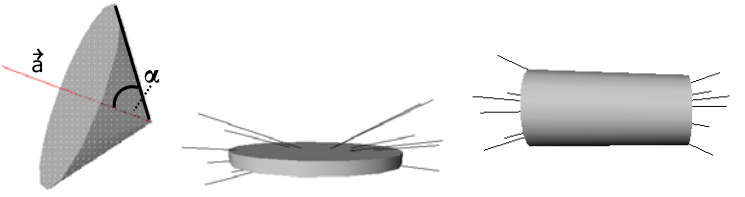


Fig. 4.17 **(a)** A cone with axis **a** and opening angle α. **(b)** The Cylinder with a SFE=100 is not fitted well to the data. **(c)** The cylinder is optimally fitted to the data used in (b) and has a SFE=0.

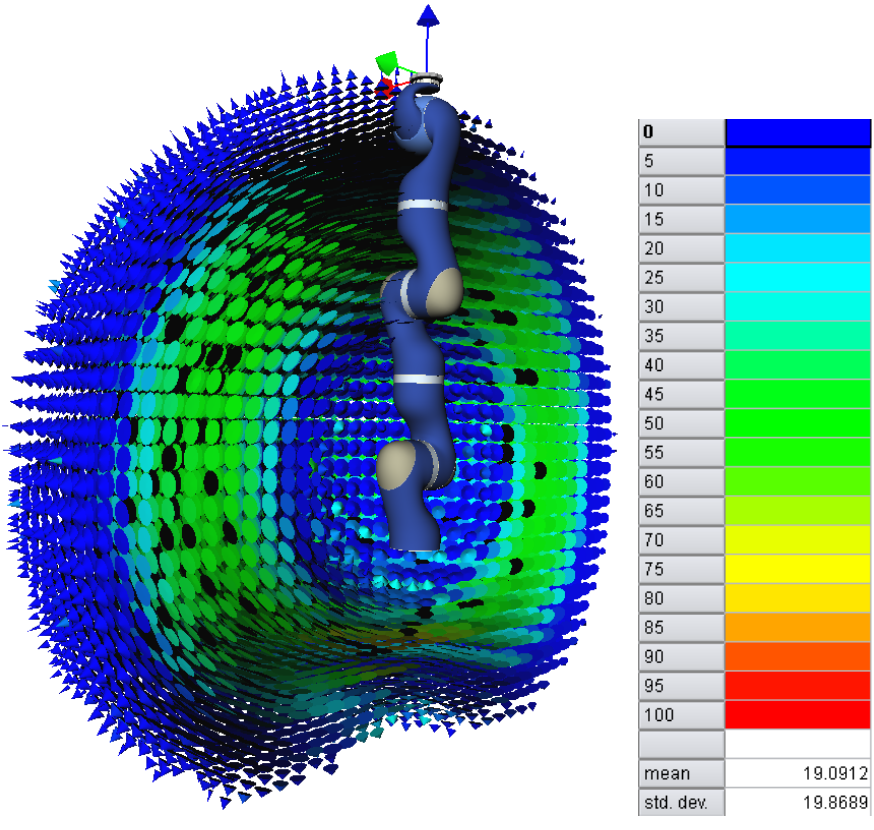


Fig. 4.18 In the shown map, the best fitting cone replaces the reachability sphere. The color encodes the SFE. Cones with a high SFE are red. Cones with a low SFE are blue.

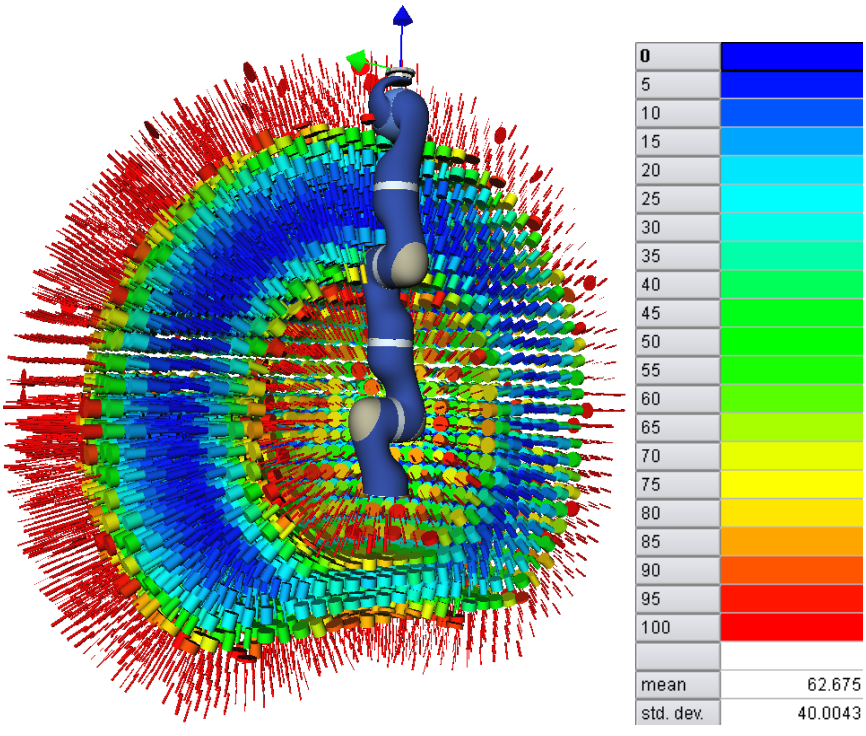


Fig. 4.19 In the map best fitting cylinder (type C2) replaces the reachability sphere. The color encodes the SFE. Cylinders with a high SFE are red. Cylinders with a low SFE are blue.

4.4.3 Evaluation of the Shape Maps

In this section the quality of the derived reachability shape maps is evaluated. The reachability sphere map for the DLR LWR computed in Section 4.2.2 is used. The cone map is a shape map where all reachability spheres are replaced by cone approximations. A cylinder map of type C1 or C2 is a shape map where all reachability spheres are replaced by cylinders of type C1 or C2. In the mixed map, the reachability spheres are replaced by the shape with the smallest SFE. The mean and the standard deviation of the SFE across the shape maps serve as performance measures. In Table 4.4.3, the mean SFE is shown for representing the workspace regions of the DLR LWR with spheres, cones, cylinders or a mix of shapes.

Using cones to approximate the structure in the workspace already provides a low mean SFE. Both cylinder maps have a mean SFE significantly higher than the cone map. Compared to the cones, only the inner part of the workspace can be represented well by cylinders, resulting in this high mean SFE. The mean SFE is lowest for the map that mixes cones and cylinders. The standard deviation has also decreased strongly. In Figure 4.20, the histogram of the absolute frequency of the SFE is shown. The occurrence of values of the SFE is counted based on all shape

Table 4.5 The SFE is evaluated for the shape maps for the DLR LWR . The reachability spheres are replaced by sphere, cones, cylinders C1, cylinders C2, or the best fitting shape. The mean and the standard deviation of the SFE rounded to one decimal is shown for the resulting shape map representations. The mean SFE for the mixed map is the lowest. The mixed map represents the data of the reachability map well. The reachability sphere map is built with sphere diameter l_c=50 mm, 10^6 samples, n_p=200 point per sphere and $\Delta_o = 30°$.

SFE	spheres	cones	cylinder C1	cylinder C2	mixed	optimal
mean	64.3	19.1	63.6	62.7	8.3	0
std. dev.	39.1	19.9	39.0	40.0	9.9	0

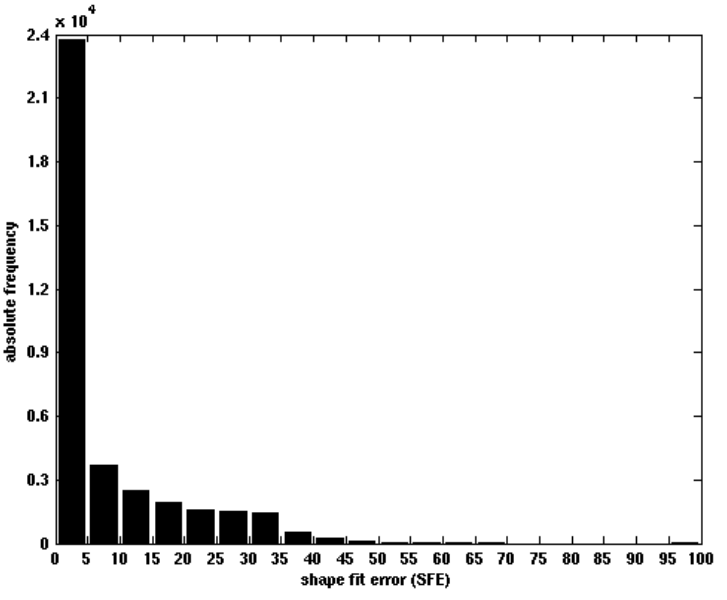


Fig. 4.20 Absolute frequency of the SFE for the mixed shape map of the DLR LWR is visualized as a histogram. The occurrence of values of the SFE is counted based on all shape primitives of the map. The majority of shapes have a low SFE. Therefore the mixed shape map represents the data well.

primitives of the map. The histogram confirms that the majority of the shapes fit the data well and have a low SFE. The mixed map is shown in Figure 4.21.

For the DLR LWR, it can be concluded that the mixed map is a suitable representation of the workspace structure and therefore the robot capabilities in its workspace. In Figure 4.22, a teabox is placed in different parts of the robot arm workspace. From the mixed shape map it is immediately evident from which

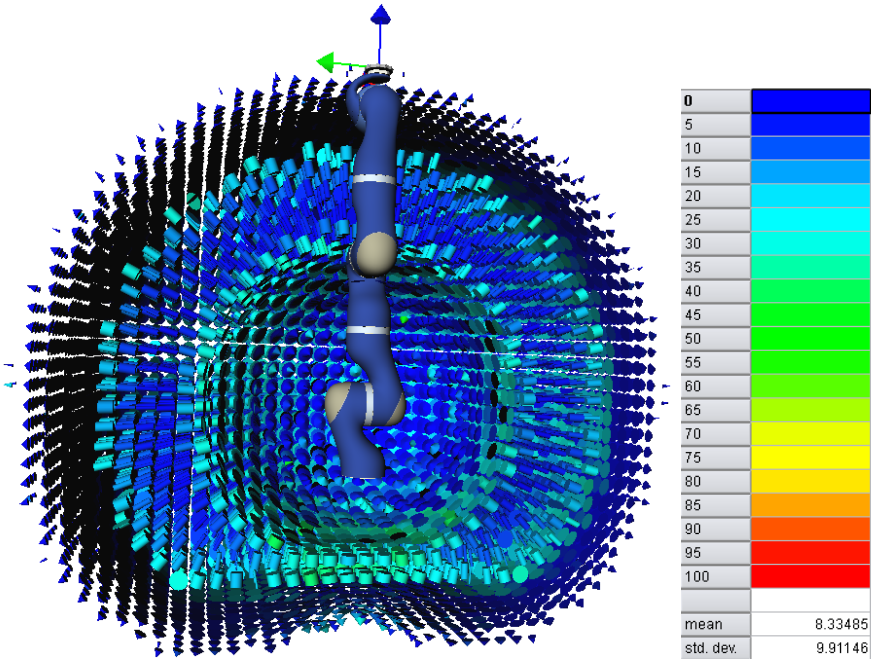
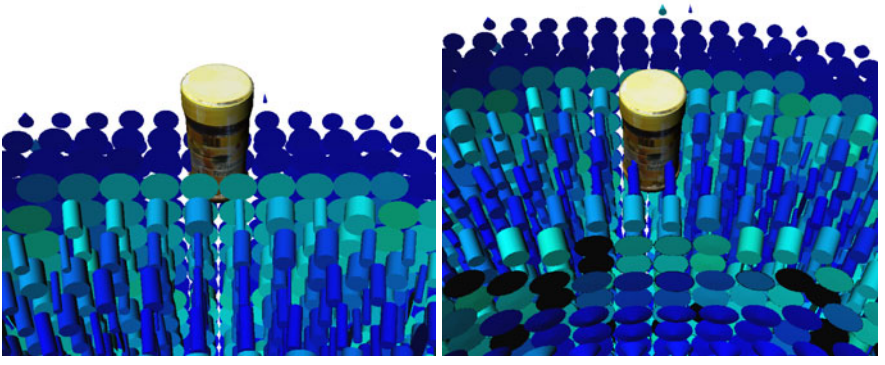


Fig. 4.21 The mixed shape map for the DLR LWR robot arm. The color encodes the SFE. Shapes with a high SFE are red. Shapes with a low SFE are blue. The mean SFE is low and the data is well represented. Shapes with SFE near 100 are only found at the outside and inside borders of the workspace and are very rare as Figurefig:histogram shows. They are not present in this figure.

direction the teabox is reachable. In Figure 4.22 (a), the teabox is placed near the border of the workspace and is only reachable from the front. In Figure 4.22 (b) it is placed in the center of the workspace and can be reached from the top, the sides and from below. Therefore the mixed shape map is an adequate tool for analysis and visualization involving the DLR LWR.

However, the workspace of other robot arms is not as well structured. In Table 4.4.3, the mean and the standard deviation of the SFE is shown for the shape maps fitted to the reachability sphere map of a 6 DOF Schunk PowerCube arm. The mean SFE for the mixed shape map is about double the value of the mixed map of the DLR LWR . Therefore, larger prediction error are expected than for the DLR LWR arm. The prediction accuracy is shown in Table 4.7. For the Schunk PowerCube arm, the reachability sphere map is the better representation of the data as Table 4.4 shows.



(a) **(b)**

Fig. 4.22 A yellow teabox is placed in different parts of the workspace. Using the mixed shape map it can immediately be seen from which directions it is reachable. **(a)** The teabox is placed at the border of the workspace. It can only be reached from the front. **(b)** The teabox is placed in the middle of the workspace. It can be reached from the right and left side and from above.

Table 4.6 The SFE is evaluated for the shape maps of the Schunk PowerCube arm. The reachability spheres are replaced by sphere, cones, cylinders C1, cylinders C2, or the best fitting shape. The mean and the standard deviation of the SFE rounded to one decimal is shown for the resulting shape map representations. The mean SFE for the mixed map is the lowest. However, it is nearly twice the value for the DLR LWR . Therefore, prediction errors can be expected. The reachability sphere map is built with sphere diameter l_c=50 mm, 10^6 samples, $n_p = 200$ point per sphere and $\Delta_o = 30°$.

SFE	spheres	cones	cylinder C1	cylinder C2	mixed	optimal
mean	99.9	16.1	95.1	97.6	14.0	0
std. dev.	0.1	22.9	13.2	8.4	17.5	0

4.4.4 Reachability Estimation Using the Mixed Shape Map

The mixed shape map also describes the capabilities of the robot arm. It can be used to predict the reachability of a given TCP pose and thereby facilitate scene reasoning. In this section, the suitability of the mixed shape map to predict the reachability of frames is examined. First, Equation 4.12 is used to determine the shape $\mathbf{g} \in V_{Robot}$, that the frame belongs to. Then it is determined whether the pose belongs to the shape. Using Equation 4.41 the angle ζ between the z-axis $-\mathbf{z}$ of frame F and the center axis $\mathbf{v_s}$ of the shape primitive is computed.

$$\zeta = \arccos(< \frac{\mathbf{v_s}}{\| \mathbf{v_s} \|}, \frac{-\mathbf{z}}{\| -\mathbf{z} \|} >) \tag{4.41}$$

If the shape is a cone or a cylinder of type $C1$, the frame F is represented by the shape if the angle ζ is smaller than the opening angle β of the cone or the cylinder. To test if a frame is represented by a cylinder of type $C2$, also ζ is computed. If ζ is smaller than the opening angle of the cylinder of type C2 this means that the tested vector $\mathbf{z}$ is element of the ground area of the cylinder and not part of the cylinder shell. Therefore it is not represented by the cylinder of type $C2$.

The prediction accuracy for the mixed shape map of the DLR LWR , Kuka LWR , the PowerCube arm and the Kuka Kr16 is examined. 10^5 frames are randomly sampled from the workspace. The prediction of the shape map is compared against the output of the inverse kinematics.

As shown in Table 4.7, the mixed shape map prediction is less accurate than the reachability sphere map prediction (Table 4.4) for all robots. The prediction accuracy of the map for the DLR LWR is reduced by 8%. The prediction accuracy of the Kuka LWR is reduced by 1.6%. The prediction accuracy of the map for the PowerCube arm is reduced by 4.2% and that of the map for the Kuka Kr16 is reduced by 5.2%.

During the shape fitting process the rotation about the z-axis is discarded. If the last link rotation is not identical with a rotation about the z-axis of the TCP frame, greater prediction errors are expected as with the use of the reachability sphere map. This is affirmed by the results in Table 4.7. The prediction accuracy for the mixed shape map of the DLR LWR has degraded the most. For the PowerCube arm and the Kuka Kr16 the predictions by the mixed shape maps are less accurate than those by the reachability sphere map. While the last link of both robots rotates the z-axis of the TCP, the mean SFE of the mixed shape maps shows that the reachability structure of the workspace is not accurately represented by the shapes.

Because of these results and because not every robot has a mixed shape map with a low SFE, the reachability sphere map is used to predict the reachability of frames.

4.4.5 Memory and Computation Time

The reachability sphere maps and reachability shape maps are computed offline once for each robot kinematics and a chosen TCP frame and saved on the hard drive. In Table 4.8, the computation time and the memory sizes for the maps saved in a human readable file are shown. The computation time of the reachability sphere map depends on the time needed to compute an inverse kinematics solution. The computation time for the reachability shape maps is the time to compute all shape map types based on a given reachability sphere map. For all four robots, the memory sizes for reachability shape maps are smaller than those for the reachability sphere maps by a factor between 20-30.

The sphere maps for the redundant robot arms, DLR LWR and Kuka LWR , are built with 10^6 samples, $l_c = 50\,mm$, $n_p = 200$ and $\Delta_o = 30°$. The reachability sphere map for the Schunk PowerCube arm is built with the same sphere map parameters (l_c, n_p, Δ_o). However, since the inverse kinematics has an analytical solution,

Table 4.7 For 100000 randomly sampled frames, the prediction of reachability by the mixed shape maps for the DLR LWR , the Kuka LWR , the 6 DOF PowerCube arm and the Kuka Kr16 is compared. The shape fitting process discards the reachability information concerning the rotation by α_k around the z-axis of the frame $F_{i,0}$. The predictions for the Kuka LWR , the PowerCube arm and the Kuka Kr16 are more accurate because its last axis rotates the z-axis of the TCP frame. The accuracy is highest for the Kuka LWR because the mean SFE of its map is the lowest.

	DLR LWR	Kuka LWR	PowerCube arm	Kuka Kr16
true pos.	23.3	32.2	17.7	21.6
true neg.	62.3	60.7	71.3	67.7
accuracy	**86.6**	**92.9**	**89.0**	**89.3**
false pos.	11.7	3.3	5.0	5.3
false neg.	2.8	3.8	6.0	5.4

Table 4.8 The computation time for computing reachability sphere maps and reachability shape maps are shown for different robot kinematics. Furthermore, the memory sizes for the maps saved in a human readable file are shown.

	DLR LWR	Kuka LWR	PowerCube arm	Kuka Kr16
sphereMap				
time	3.7 h	1.8 h	3.9 h	4 h
memory	62MB	64MB	40MB	125MB
shapeMap				
time	3 min	4 min	2 min	3 min
memory	2.3MB	2.3MB	1.6MB	3.6MB

the voxel space can be traversed iteratively. The same is true for the Kuka Kr16. However, for the Kuka Kr16 a sphere diameter $l_c = 100\ mm$ is used. The computation times were obtained on a computer with two Intel(R) Xeon(R) CPU W3520 2.67GHz processors and 6 GB main memory.

4.5 Summary

In this chapter, two representations of the versatile workspace were presented and analyzed, the reachability sphere map and the reachability shape map. Both representations are based on the voxelization of the workspace. To construct the reachability sphere map, voxels of the workspace are examined using inverse kinematics. For each voxel, the set of rotations $SO(3)$ in $\mathbb{R}^3$ is discretized and a set of frames is generated. The reachability of each frame is determined and noted in the representation. For the DLR LWR , a reachability sphere map build with 10^6 samples, a sphere diameter of $l_c = 50mm$, $n_p = 200$ points per sphere and $\Delta_o = 30^\circ$ is a valid representation of the versatile workspace. The visualization of the reachability sphere map allows the inspection of the workspace.

By capturing the data with shape primitives, it was shown that directional structure exists in the workspace of the DLR LWR robot arm. Continuous regions with similar structural properties can be recognized. However, representing the reachability sphere map by the mixed shape map is only possible when structure exists in the workspace, and when this structure can be represented by shape primitives with a low approximation error. This is not true for arbitrary robot arms.

The capability to predict the reachability of frames was analyzed for both representations. It was shown that the reachability sphere map is an accurate representation of the kinematic capabilities of a serial link robot arm. It provides more accurate predictions than the reachability shape map. The reachability sphere map can also be used for robot arms with parallel kinematics given a corresponding inverse kinematics.

In the following chapters, several applications of the reachability sphere map to different domains are presented. From now on, the reachability sphere map will be referred to as the **capability map** of the robot arm, since it describes a robot arm's capabilities in its workspace.

Chapter 5
Visualization and Setup Evaluation

One field of application for the capability map is the visualization and inspection of the robot arm workspace. In this chapter, the workspace is visualized for several robot arms and discussed with respect to potential tasks. Furthermore, the capability map is used to objectively evaluate the quality of a setup for human robot interaction.

5.1 Robot Arm Workspace Visualization

When no task specification is given, the capability map can be used to identify categories of tasks that the robot is suitable for. Given a task description, the suitability of the robot can be determined. The visualization of the capability map can be used for workspace and failure analysis. It can be determined which regions the robot can reach in a versatile manner.

5.1.1 Visualization for Industrial Robots

Kuka KR16

The Kuka KR16 [59] is an industrial robot. It is designed to have a big workspace and to be able to accomplish manufacturing tasks in its whole workspace. The KR16 is a general purpose robot and its capability map reflects this. In Figure 5.1, the capability map for the 6 DOF Kuka KR16 industrial robot is shown. The color encodes the reachability index D. The index D is high (blue region) throughout the whole workspace. The TCP frame is shown as a coordinate system at the flange of the robot. The last axis of the robot performs a rotation about the z-axis of this TCP. Because of the link limits of $\pm 350^\circ$ for this axis, the indices D and D_o are identical throughout the workspace. In Figure 5.1, it can be seen that the dexterous workspace volume for the chosen TCP is empty. The maximum length of the KR16 is 2.5 times that of the DLR LWR . Due to memory limitations, the sphere diameter for the capability map of this robot arm is $l_c = 0.1\ m$.

F. Zacharias: Knowledge Representations for Planning Manipulation Tasks, COSMOS 16, pp. 71–92.
springerlink.com

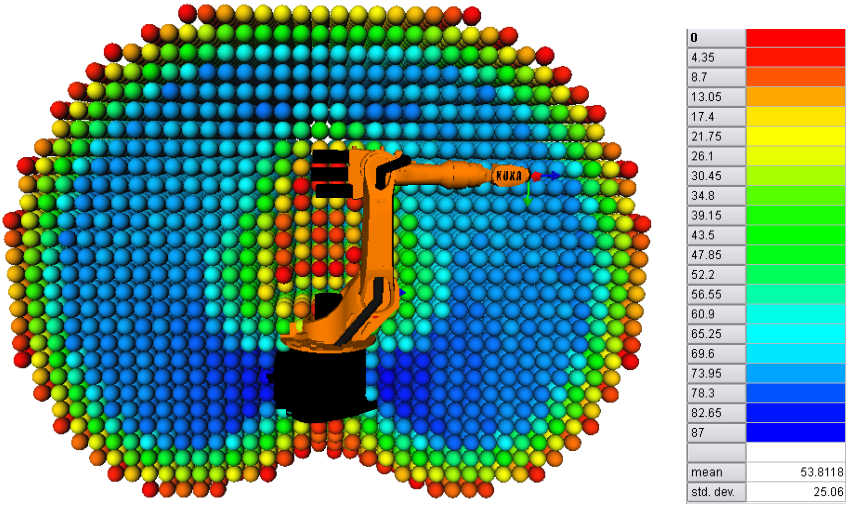


Fig. 5.1 Capability map is visualized for the Kuka KR16, a general purpose industrial robot. The color encodes the reachability index D. The reachability index is high (blue region) throughout the whole workspace. The coordinate system at the flange of the robot shows the pose of the TCP frame. The x-axis is shown by the red arrow, the y-axis by the green arrow and the z-axis by the blue arrow.

Schunk PowerCube Arm

The capability map for the Schunk 6 DOF PowerCube light weight robot arm [98] is significantly different. The workspace is smaller than that of the DLR LWR due to the shorter arm length (0.9 m) and the arm's capabilities in its workspace are much more restricted. In Figure 5.2, the capability map is shown. The color encodes the reachability index D_o. The arm can be attached to a mobile base as shown in Figure 5.3. To indicate the workspace to the front of the robot, the first rotation axis of the robot arm is shown by a white arrow. The map for the Kuka KR16 has a mean reachability index D_o of 52.8, while the map for the PowerCube robot arm has only $D_o = 19.5$. Also the maximum value of this index differs significantly. The KR16 robot has many regions (dark blue, $D_o = [78 - 87]$) that it can reach in a nearly dexterous manner, i.e. from all possible directions. In contrast the PowerCube arm, has a reachability index of 52 at maximum. As seen in Figure 5.2 the best reachable region for the Schunk arm is at shoulder height. If the arm should serve as the arm of a humanoid robot and accomplish human-like manipulation tasks this is clearly undesirable. The human often manipulates with the arm bend at 90°. Therefore, the best reachable region for a manipulator intended for human-like manipulation

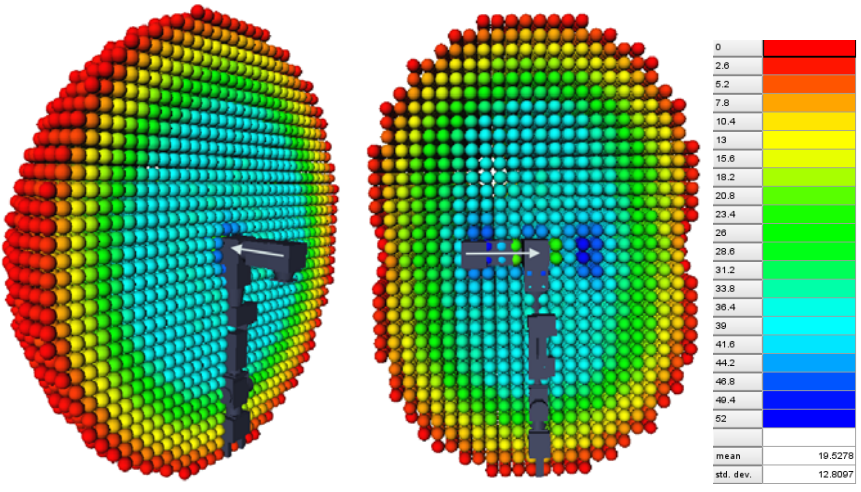


Fig. 5.2 Capability map for the Schunk 6 DOF light weight robot is shown. The first rotation axis of the robot arm is shown by a white arrow. The color encodes the reachability index with z-orientations D_o. The mean reachability index is low.

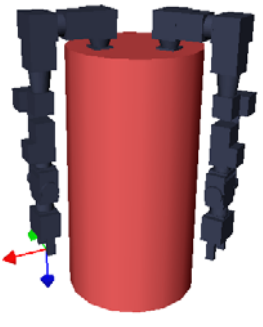


Fig. 5.3 Two Schunk 6 DOF light weight robot arms are attached to a mobile base. The TCP of the right arm is indicated by the coordinate system.

should be in the region in front of the body. The capability map is built with sphere diameter $l_c = 0.05$ m, $n_p = 200$ points per sphere, and $\Delta_o = 30^\circ$ z-rotation step size. The last axis of the robot performs a rotation about the z-axis of this TCP. Because of the discretization and the link limits of $\pm 170^\circ$ for this axis, the indices D and D_o are identical throughout the workspace.

5.1.2 Visualization for Specific Tasks

DEOS

The DEOS robot is attached to a carrier satellite. In orbital servicing scenarios, the carrier satellite approaches and captures target satellites either to deorbit them or to prolong their life-time. To accomplish this task, the robotic arm has to be attached to the carrier satellite so that the target satellite can be grasped. In Figure 5.4 the workspace of the DEOS robotic arm is shown for an exemplary attachment of the arm to the carrier satellite. It shows how close the satellite has to be to the carrier. In the blue region the most TCP poses are kinematically reachable and deviations or pose variations of the target satellite can be compensated best. Note that in space, the dynamics of both the robot and the satellites have to be considered because the TCP poses that allow for grasping of the satellite depend on the current robot configuration and the dynamics of the system. Due to the absence of gravity, each motion of the robot arm also causes its carrier satellite to move.

Justin

The humanoid robot *Rollin' Justin'* [29] consists of an humanoid upper body system mounted on a mobile base with variable footprint. The humanoid upper body is composed of a 3-DOF torso, two 7-DOF arms, two 4-finger hands and a 2-DOF head. The 7-DOF DLR light weight robot arm serves as the left and the right arm of

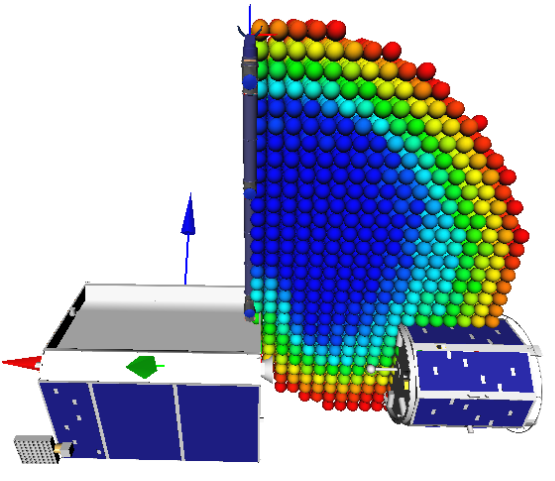


Fig. 5.4 For orbital servicing, the DEOS robot is attached to a carrier satellite. Its purpose is to capture other satellites. The capability map of the DEOS robot is visualized. It shows where with respect to the robot arm a satellite is kinematically reachable with the most poses and can be used for design analysis.

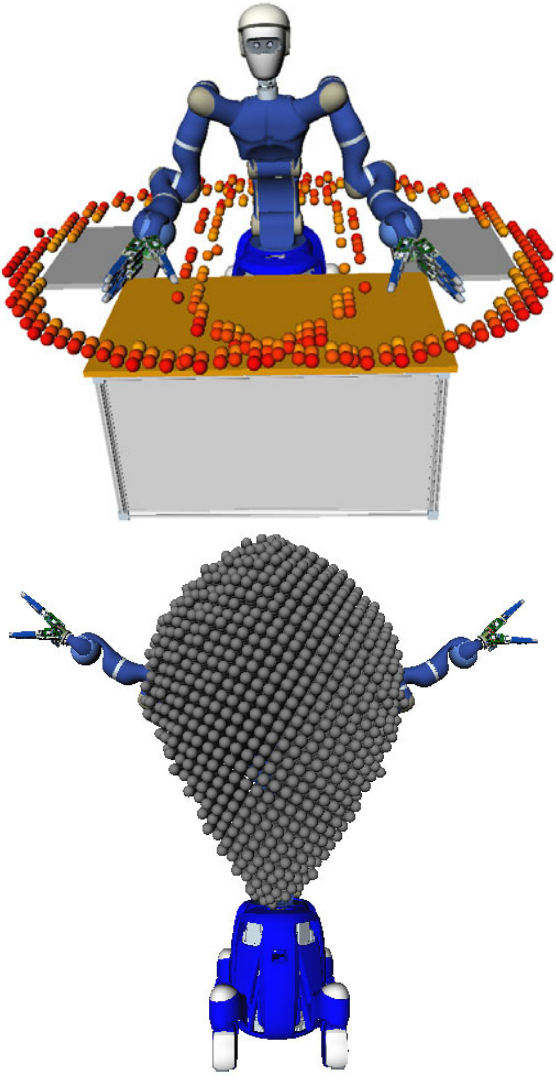


Fig. 5.5 The humanoid robot *Rollin' Justin* performs tasks at the table. **(top)** The borders of the workspaces of the left and the right arm are shown on the table surface using the capability map. **(bottom)** The combined workspace volume is shown that the left and the right arm can reach.

the robot *Rollin' Justin*. One task of the humanoid robot *Rollin' Justin* is to perform tasks at the table, e.g. to prepare coffee. Using both arms, various objects have to be grasped, moved, and used in individual subtasks. Hereby it is necessary to decide which arm to use to grasp an object. Furthermore, it is necessary to decide where

in the workspace to execute two-handed manipulation tasks. In Figure 5.5 (top) the border of the left and the right arm on the table surface is shown using the capability map. The region where two-handed manipulation is possible lies in the intersection of the two circles. The whole workspace volume that can be reached by both the left and the right arm is shown in Figure 5.5 (bottom). It can be seen that the attachment of the arms is not ideal for two-handed manipulation at table height. These kind of analyzes are valuable for the future design of humanoid robots.

When each arm of the humanoid robot works in a region that can only be reached by the respective arm, path planning for each arm can be done separately and in parallel. The arms do not interfere with each other. If both arms work in the workspace volume shown in Figure 5.5 (bottom) their motion has to be coordinated. A task planner can use this knowledge to structure the task and its execution. The path planners can then be parameterized accordingly as shown by Zacharias et al. [119].

5.2 Workspace Comparisons for Human-Robot Interaction

Systems designed to be operated by a human have to provide a human-machine interface for the user. Systems built for applications like virtual reality simulations or teleoperation require operators to interact intuitively with the system. Interfaces are needed that display the virtual environment as realistically as possible. For virtual assembly verification or remote maintenance tasks with haptic feedback, the quality of a haptic interface is crucial for successful task completion. The interface should allow flexible interaction with the operator. In the bi-manual human-robot interface (HRI) [42], [118] shown in Figure 5.6, the operator uses both hands to directly control the robotic arms by attached handles. The robots can be moved around and provide haptic feedback from a virtual environment to the user. Assuming a setup with two robotic arms, it has to be determined how the arms have to be attached to a base to allow for the best interaction with the operator. A simple approach is to evaluate various setups using a number of subjects who judge which system they are more comfortable with. Alternatively, the setups can be analyzed using functional criteria. In this section, a method is presented to objectively evaluate the quality of a setup. The workspaces of the robot and the operator's arm are represented using the capability map. Additional statistical data obtained with a portable tracking system allows to restrict the workspace comparison to the most significant regions.

5.2.1 Detailed Problem Analysis

Different attachments of the robot arms to a base are possible. Two fundamentally different setups can be distinguished. Either the human interacts head-on with the system (Figure 5.6 left) in *scenario 1*, or the human is enclosed by the system (Figure 5.6 right) in *scenario 2*. To evaluate the quality of one scenario with respect to the other, the most important question is how and what aspects can be compared. The first method of comparison that comes to mind is to determine which scenario is easier to use. Psychological questionnaires and experimental trials carried out by

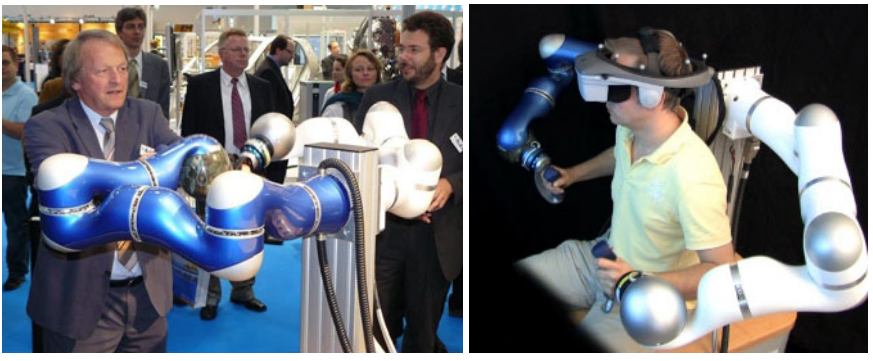


Fig. 5.6 A system with two robot arms attached to a common base provides bimanual haptic feedback. Two fundamentally different setups can be distinguished **(left)** the standard configuration, and **(right)** the ergonomic configuration of the bi-manual haptic interface.

human subjects are typically used to perform this kind of evaluation. However this method is time consuming, requires a significant number of subjects and provides no quantitative measures of performance.

An objective comparison should evaluate how well the bi-manual haptic interface is able to mimic typical bi-manual manipulation movements. These enable a broad range of applications for the haptic interface. A key point of the interaction is that the bi-manual interface does not hinder the operator's desired movements during task execution. When the operator wants to reach a position in a certain orientation, e.g. to install a car battery or turn a screw, the interface should accommodate that movement. This requires that the robotic interface covers the workspace of the human arm as well as possible. For the systems shown in Figure 5.6, a complete coverage is not possible due to collisions between the robot arms and the human. Therefore the workspace overlap has to be quantified. The simplest method of comparison is to compute the maximum volume that can be swept by the robotic arm or the operator's arm e.g. using methods such as presented in [1] or [95]. One could then determine how much of the volume for the operator's arm is covered by the robotic arm. However, a swept volume only specifies the positions that can be reached. No information regarding orientations is represented. This method is inadequate for cases where, for a set of positions, only certain orientations are of interest or where a certain region should be reachable in a versatile manner. Therefore, the workspace is represented using the capability map.

5.2.2 Kinematic Descriptions

In this section the kinematics for the robot arm and the human arm are specified. They are used for the computation of the capability map. Special emphasis is placed on the choice of the TCP frame i.e. the reference frame for the grasp.

5.2.2.1 Robot Arm Kinematics

In both setups, the bimanual haptic interface is composed of two DLR-Kuka light weight robot arms. Each arm has seven DOF, mimicking the human arm kinematics and is therefore well suited for use in human robot interaction. Both arms have an identical kinematics and only differ in their attachment to the base. The kinematics parameters and a visualization of the rotation axes can be found in the appendix in Table B.2 and in Figure B.2.

5.2.2.2 Human Arm Kinematics

This section details how the human arm kinematics is modeled. Two possibilities are discussed. The shoulder link is a spherical joint that is usually approximated by three serial axes. Tondu et al. [109] and Klopčar et al. [52] choose the first axis to point normal to the plane of the back of the human. This version is labeled *kinematics 1*. In contrast, Abdel-Malek [1] chose the first axis to point outwards to the side of the human shoulder. This version is labeled *kinematics 2*. Both versions are shown in Figure 5.7. The position and orientations of the link coordinate frames is shown. The coordinate axes are color-coded with the x-axis in red, the y-axis in green and the z-axis (=the rotation axis) in blue. One difference between these two kinematics is the location of their singularities. Singularities can be detected using

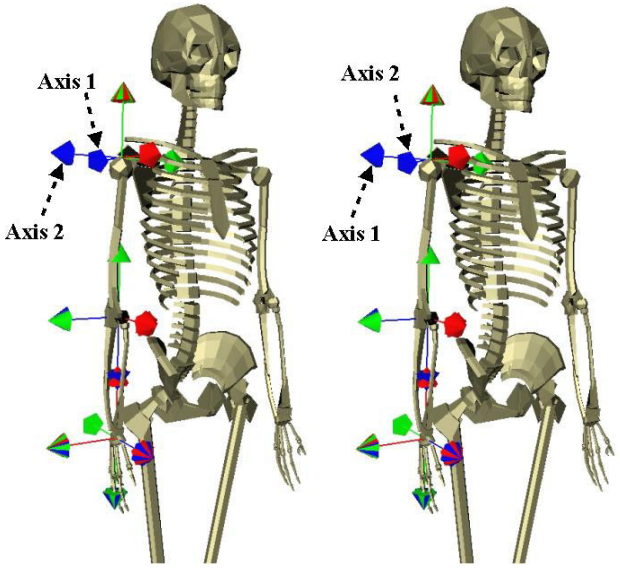


Fig. 5.7 Positions of the coordinate systems for the kinematics of the human right arm. **(left)** In *kinematics 1*, the first rotation axis points normal to the plane of human's back. **(right)** In *kinematics 2*, the first axis points to the side. (The graphics model of the virtual human provided by LAAS-CNRS was adapted for the individual kinematics.)

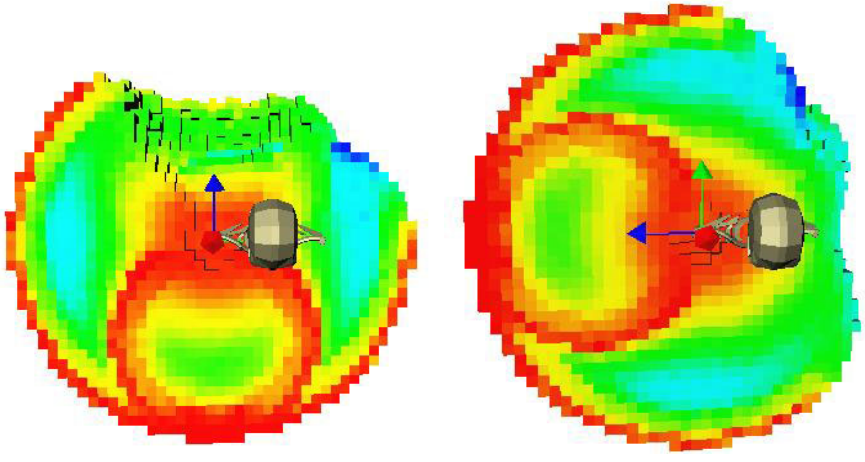


Fig. 5.8 For human arm kinematics, Cartesian regions that contain singularities are visualized using the minimum volume of the manipulability ellipsoid. A small value is shown in red, large values are represented by blue. If the volume is zero, the corresponding arm configuration is singular. The human is viewed from above. The workspace is cut in half at shoulder height. **(Left)** Singularities for human arm *kinematics 1* are shown. **(right)** Singularities for human arm *kinematics 2* are shown.

the volume of the manipulability ellipsoid as discussed in Section 3.1. In Figure 5.8 bright red regions signify Cartesian regions where corresponding configurations can be singular, i.e. the volume of the manipulability ellipsoid is zero. For *kinematics 1*, Cartesian regions with singular configurations lie to the front of the human body. For *kinematics 2*, they lie on the right side of the right shoulder. For both kinematics identical link limits are used. The link limits are set as proposed by Kapandji [47] and are listed in Table 5.1.

To determine which kinematics better represents the human arm kinematics, the capability map is computed for both versions. Figure 5.9 shows the regions that received a reachability index D in the top 20% of the value range (blue) of the reachability sphere map for *kinematics 1* and *2*. Figure 5.10 illustrate the arm's ability to reach regions at elbow level. In both Figures, it can be seen that whereas *kinematics 2* conveniently turns the singularities out of the area where manipulation most commonly occurs, it also rotates the best reachable region out of that area. *Kinematics 2* does not have a blue region at elbow level at all and thus could exploit fewer possibilities for positioning the hand when manipulating objects. This is clearly undesirable especially since Howard et al. [41] showed that the region the human hand moves in during every day tasks mostly lies to the front of the body. It is plausible to assume that the human arm's abilities are best in this region.

The wrist is also a spherical joint that has to be modeled by a sequential alignment of axes. The same analysis as for the shoulder can be performed. However due to the resolution of 0.05 m of the capability map and a comparably small distance from the wrist to the palm of the hand, different types of wrist kinematics do not introduce

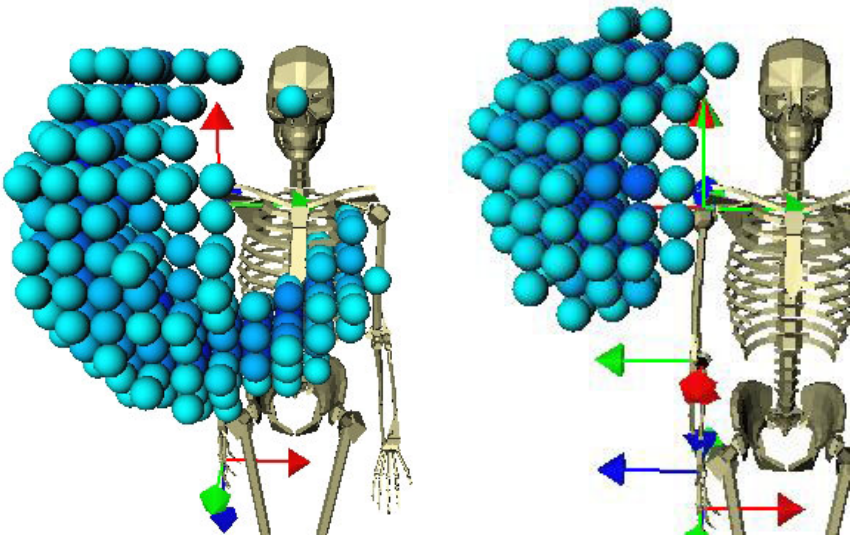


Fig. 5.9 The best reachable regions are shown for the human right arm modeled with *kinematics 1* or *kinematics 2*. Regions that received a reachability index D in the top 20% (blue) of the value range are visualized for **(left)** *kinematics 1* and **(right)** *kinematics 2*.

Table 5.1 DH parameters and link limits for the human arm kinematics 1.

i	α_{i-1}	a_{i-1}	d_i	θ_i	ll	ul
0	0	0	0	0	-30	180
1	0	90	0	-90	-50	180
2	0	90	0.27 m	0	-110	80
3	0	-90	0	0	0	145
4	0	90	0.22 m	0	-85	90
5	0	-90	0	90	-45	15
6	0	90	0	90	-85	85

significant changes with respect to the general location of the best reachable region. Therefore this analysis is not performed. For the workspace analysis, *kinematics 1* is chosen. It corresponds to the sequence and orientation of the link coordinate systems as detailed in Tondu et al. [109]. The kinematics (Table 5.1) are expressed using the Denavit-Hartenberg (DH) parameters as given by Craig [19]. The length of the limbs is provided by Ian Howard. It was computed from real data using the method proposed by Howard et al. [41].

The kind of analysis described in this section using the capability map to analyze the capabilities of specific kinematics was also used in the design of the DLR Hand Arm system (Figure 5.11) [34]. During design, the axes order and the link ranges were varied to obtain a kinematics where the region with the best reachability index

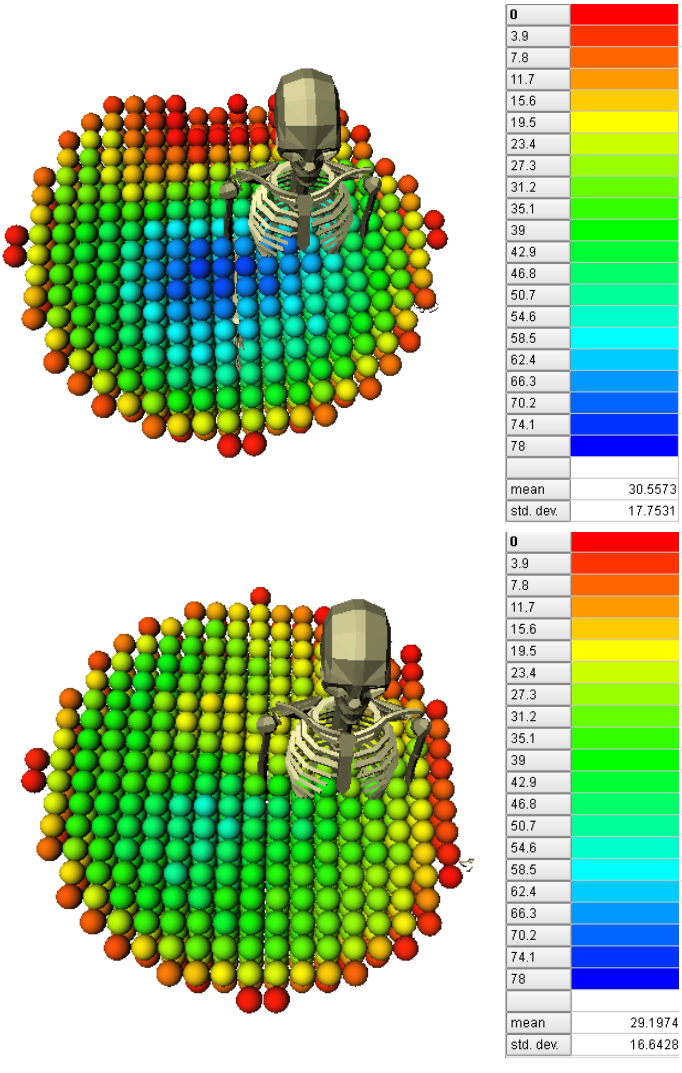


Fig. 5.10 The capability map is shown for **(top)** the human right arm modeled with *kinematics 1* and **(bottom)** the human right arm modeled with *kinematics 2*. The capability map is cut along a cutting plane at elbow level and only the element below the plane are shown from a birds eye perspective. The color scale is chosen to be the same for both images encoding the reachability index 0-78.

is at a location similar to the human. A further constraint was that the system must be realizable for the mechanical engineers. Therefore, the decision was made to implement *kinematics 2* because it allows for easier and more compact mechanical design. The human link ranges cannot be used if the goal is to have a kinematics that

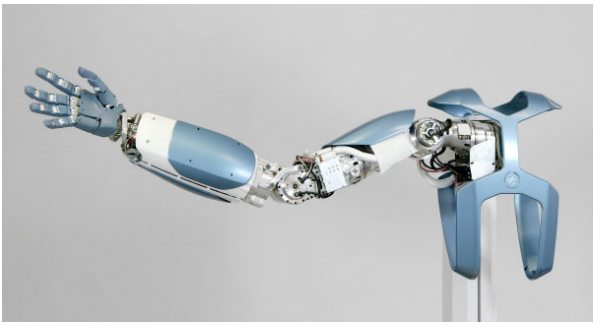


Fig. 5.11 The anthropomorphic hand arm system using variable stiffness actuation has been developed at DLR. It is aimed to reach its human archetype regarding size, weight and performance [34].

has the best region for manipulation in front of the body. Therefore the minimum and maximum link values were shifted appropriately and verified by computing the capability map for the kinematics.

5.2.2.3 The Selection of the Tool Center Point

The selection of the TCP is essential for the comparison of the workspaces. The importance of its selection can be easily recognized when considering the task of hitting a nail with a hammer. The structure and shape of the workspace of the arm with a tool attached is different from the workspace of the arm alone. The reason for this is the different location of the frame of manipulation, the TCP. The selection of the TCP is reflected in the structure of the capability map.

For both scenarios the TCP for the human operator is chosen so that it lies at the point in the palm where the handle attached to the robot arm (Figure 5.12 (a)) is grasped. The TCP of the robot arm differs between the two setups. The goal of the workspace comparison is to evaluate how well the robot arm is able to achieve the positions and orientations requested by the human operator. Therefore the pose of the robot TCP has to match the pose of the human arm TCP at the point of contact (Figure 5.12 a). The robot arm TCPs are shown as coordinate systems in Figure 5.12 (c) for *scenario 1* and in Figure 5.12 (d) for *scenario 2*. The human arm TCP is visualized in the human's palm.

5.2.3 Workspace Comparison

Using the TCPs defined in the last section, the capability maps are computed for the human arm and the robot arm. A visualization of a map for the robot arm is shown in Figure 5.13 and the map for the human arm is shown in Figure 5.14. The color encodes the reachability index D. The capability map for the right arm of the human matches the intuitive expectations. The space where the most frames are reachable, lies in front of the body and can be reached with the elbow link bent at 90 degrees.

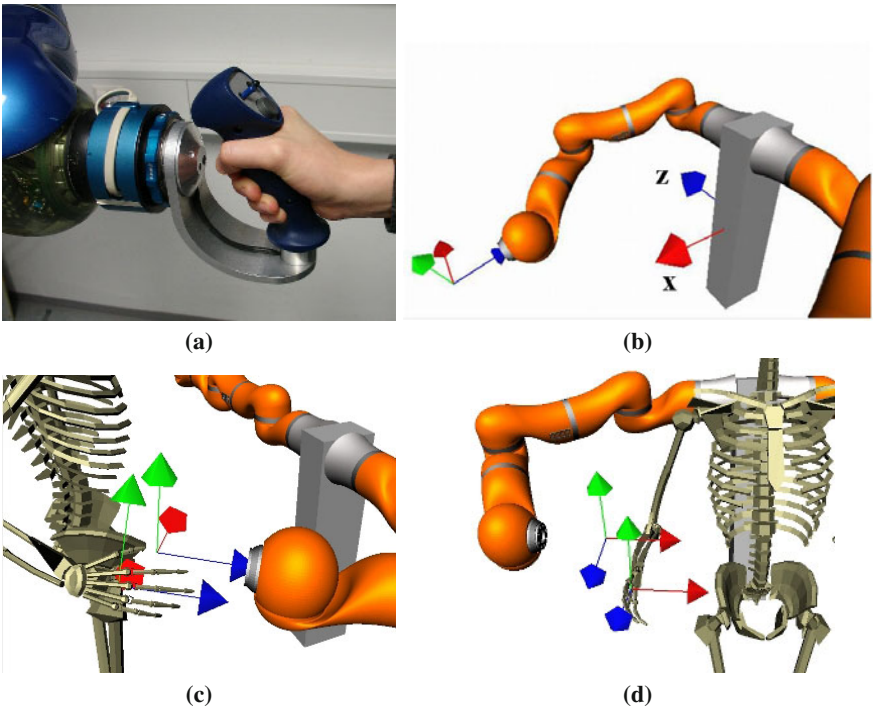


Fig. 5.12 To interact with the robot, the human hand grasps a handle attached to the robot. **(a)** A hand grasps the handle in *scenario 1*. **(b)** The base coordinates system is shown for the workspace analysis. The TCPs of the robot and the human arm are shown for **(c)** *scenario 1* and **(d)** *scenario 2*.

In the following analysis it is assumed that the pose of the human arm base T_{Human}^{world} and the pose of the robot arm base T_{Robot}^{world} in the world coordinate system are known. In order to compare the workspaces each TCP frame, i.e. its capability map representation $(\mathbf{g}, i, k) \in M_s$, that is labeled reachable in the human arm's capability map is transformed into the respective robot arm's coordinate system resulting in the TCP frame F_{TCP}^{Robot}.

$$F_{TCP}^{Robot} := F_{TCP}^{Robot}(\mathbf{g}, i, k) = (T_{Robot}^{world})^{-1} \cdot (T_{Human}^{world} \cdot T_{sphere}^{Human}(\mathbf{g}) \cdot F_{i,\alpha_k}) \tag{5.1}$$

$T_{sphere}^{Human}(\mathbf{g})$ is specified in Equation 4.23. F_{TCP}^{Robot} is then mapped to the best fitting frame in the robot arm's capability map. Let $f_{Robot}(F)$ perform the mapping of a frame $F \in H$ given in the robot base frame to its capability map representation $(\mathbf{g}, i, k) \in M_S$ as detailed in Section 4.2.3.

$$f_{Robot} : H \rightarrow M_S; F \mapsto f_{Robot}(F) := (\mathbf{g}, i, k) \tag{5.2}$$

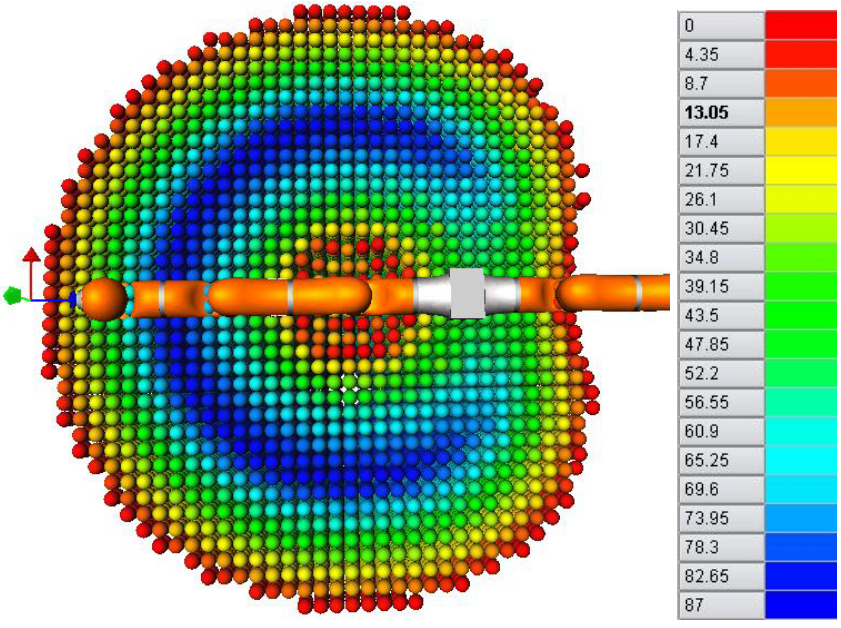


Fig. 5.13 The capability map of the Kuka LWR robot arm in *scenario 1* is shown. The spheres are colored according to their reachability index D.

The function $h(\mathbf{a})$

$$h : M_S \rightarrow \{0,1\}; \mathbf{a} \mapsto h(\mathbf{a}) := \begin{cases} 1 \text{ if } \mathbf{a} \text{ reachable according} \\ \quad \text{to capability map} \\ 0 \text{ if } \mathbf{a} \text{ not reachable} \end{cases} \tag{5.3}$$

is 1 if the tuple $\mathbf{a} \in M_S$ is reachable according to the capability map and 0 if it is not reachable. The percentage of reachable frames for the human that can also be reached by the robot quantifies the overlap between the workspaces of the human and the robot arm. Let V_{Human} be the voxelization of the capability map for the human arm, then the overlap OV is expressed in Equation 5.4.

$$OV = \frac{\sum_{\mathbf{g} \in V_{Human}} \sum_{i \in N_p} \sum_{k \in N_o} h_{Human}(\mathbf{g}, i, k) \cdot h_{Robot}(f_{Robot}(F_{TCP}^{Robot}(\mathbf{g}, i, k)))}{\sum_{\mathbf{g} \in V_{Human}} \sum_{i \in N_p} \sum_{k \in N_o} h_{Human}(\mathbf{g}, i, k)} \cdot 100 \tag{5.4}$$

The subscripts (Human od Robot) indicate on which capability maps the functions are based. The $(\mathbf{g}, i, k)$ is only drawn from the capability map of the human. The capability maps of the human and the robot have the same parameters (l_c, n_p, Δ_o). A method for computing the overlap is summarized in Algorithm 3. The robotic system itself is stationary. However the position of the human $\mathbf{t}_{Human} \in \mathbb{R}^3$ can vary with respect to the position of the robot system $\mathbf{t}_{Robot} \in \mathbb{R}^3$. The comparison is

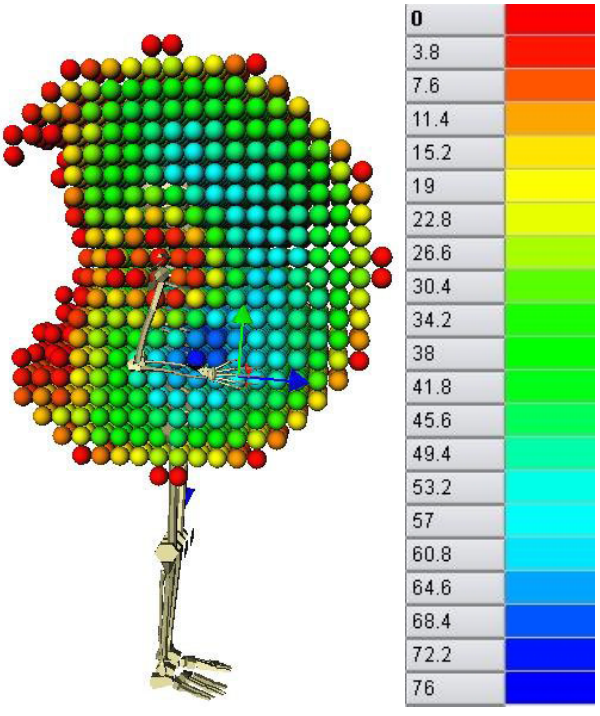


Fig. 5.14 The capability map of the human right arm kinematics 1 is shown. The map is cut in half for better visualization. The observer sees the human from the side and looks at the best workspace for the right arm. The spheres are colored according to their reachability index *D*. The human is best able to grasp objects with the elbow bent at 90°.

performed for a discrete set of positions of the human with respect to the robotic system. Both arms of the human should be able to interact with their corresponding robot arms equally well. Therefore the human shoulders are parallel to the robot system and the orientation of the human with respect to the robot is not changed. For the same reason the human is centered with respect to the base where the robots are attached.

5.2.4 Evaluation

In this section the method is evaluated. First the comparison is performed across the whole reachable workspace of the human arm. Recently, experiments as reported by Howard et al. [41] have shown that the locations of the wrists during daily tasks are concentrated in a subspace of the reachable workspace. The influence on setup comparisons when using this subspace is also demonstrated in this section. *Scenario 1* (Figure 5.15 left) and *scenario 2* (Figure 5.15 right) are evaluated using Algorithm 3. The main difference between the two scenarios is the pose of the human operator's

Algorithm 3. CompareWS(rMapHuman,rMapRobot,T_{Human}^{world},T_{Robot}^{world})

```
/** rMapHuman - reachability map for human arm
rMapRobot - reachability map for robot arm
T_Human^world, T_Robot^world - pose of human and robot in the world
F_TCP^Human, F_TCP^Robot - homogeneous matrices to represent poses of the human or robot TCP */

nrOfRobotReachableFrames←0
nrOfHumanReachableFrames←0
/** for all spheres of the capability map of the human */
for all spheres S_i of rMapHuman do
   /** for each frame on the sphere given in a coordinate system placed at human arm base
   */
   for F_TCP^Human ∈ S_i do
      /** is the frame reachable */
      reachable4Human←rMapHuman.IsReachable(F_TCP^Human)
      if reachable4Human then
         nrOfHumanReachableFrames++
         /** transform into robot coordinate system */
         F_TCP^Robot ← (T_Robot^world)^-1 · T_Human^world · F_TCP^Human
         /** map to its representation in the robot capability map */
         F̃_TCP^Robot ← rMapRobot.MapFrame(F_TCP^Robot)
         /** check reachability in robot capability map */
         reachable4Robot←rMapRobot.IsReachable(F̃_TCP^Robot)
         if reachable4Robot then
            nrOfRobotReachableFrames++
         end if
      end if
   end for
end for
/** compute workspace coverage */
coverageValue→ nrOfRobotReachableFrames / nrOfHumanReachableFrames · 100
return coverageValue
```

shoulder with respect to the robot arms and the attachment of the handle to the end of the robot arm. Between the two scenarios, the handle is rotated by 90 degrees.

5.2.4.1 Comparison without Workspace Restrictions

In *scenario 1* the right arm of the human interacts with the left arm of the two-arm robotic system. In *scenario 2* the right arm of the human interacts with the right arm of the robotic system. The base of the two-arm robotic system is positioned at the origin of the global or extrinsic reference coordinate system. The extrinsic coordinate system is displayed in Figure 5.12 (b). It is placed in the center of the base block below the robotic arms, i.e. 0.29 m below the robot arms' bases. The human's shoulder is repositioned in the *xy*-plane of the extrinsic coordinate system with respect to the robots. Movement in the *x* direction is equivalent with the human

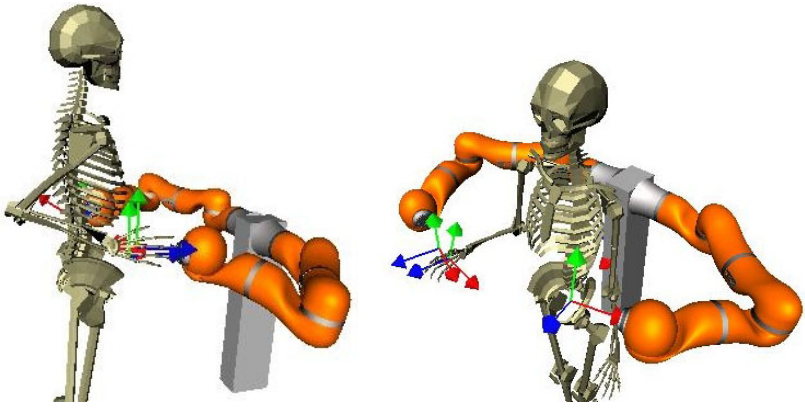


Fig. 5.15 The human is symbolized by the skeleton. The TCP frames of the robot arm and the human arm are shown as coordinate systems. For the robot arm, the TCP frame is placed where the handle (not shown) would be. The human arm TCP is in the human's palm. The human interacts with the two armed robot system in **(left)** *scenario 1* and in **(right)** *scenario 2*.

shoulder being displaced normal to the line connecting the two robot bases, i.e. moving away from the robot bases. Movement in the y direction is equivalent with varying the height of the human shoulder with respect to the robot bases. For each (x,y) position of the human's shoulder, it is determined how well the robot arm covers the human arm's workspace.

The results for *scenario 1* (Figure 5.16 (top)) are compared with the results for *scenario 2* (Figure 5.16 (bottom)). Although the maximum workspace coverage value does not differ significantly between the two scenarios, its peak location does. In *scenario 1* at maximum 59 % of the human workspace can be covered by the robot arm. In *scenario 2*, 63% coverage is obtained. So far, collisions between the human and the robot are not taken into account. However, such collisions affect the results of the analysis. In *scenario 1* the maximum coverage occurs when the human shoulder is placed at location $(x,y) = (0\ m, 0.29\ m)$ as shown in Figure 5.17. This is to be expected since the bases of the robot arm and the human arm coincide and the coverage is optimal. However at this location both arms are permanently in collision. Therefore the human operator cannot be positioned at the location of this maximum. The x-component of the human's shoulder position has to be greater than $x = 0.5\ m$ to avoid collisions with the robot and still be able to interact with the robot arms as shown in Figure 5.15. Thus, the maximum coverage decreases to $\approx 43\%$ at $(0\ m, 0.5\ m)$. In *scenario 2* the maximum coverage occurs for the human shoulder at location $(0.35\ m, 0.35\ m)$. Figure 5.15 (right) shows the human assuming this position. No collisions occur. In fact any position with an x-component $x \geq 0.2\ m$ is feasible and the human can operate the system as shown in Figure 5.15 (right). The z-component in extrinsic coordinates is fixed in these analyzes. It was chosen so that the human is centered between both robotic arms.

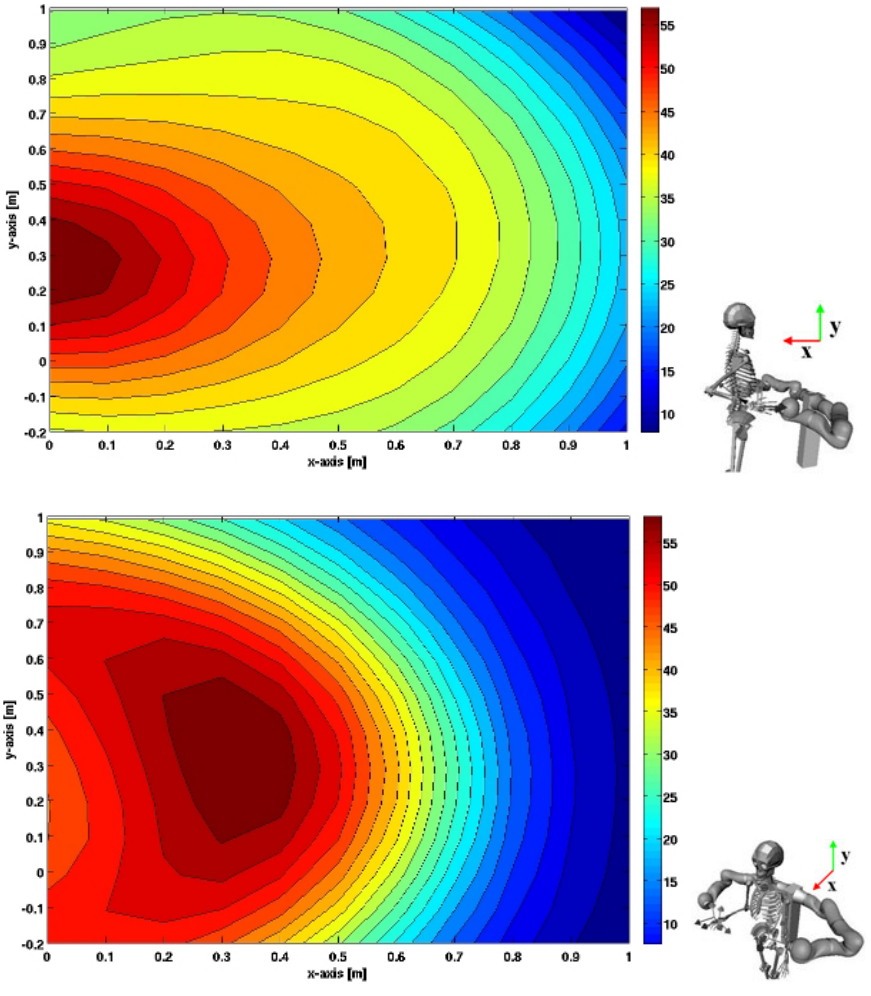


Fig. 5.16 Coverage of the workspace of the human right arm by the robot arm. The analysis is performed for different positions of the human in the *xy* plane of the global coordinate system. The right column shows how the human interacts with the system in the respective scenario and and shows the directions of the *x* and *y* axes. The coordinate system base lies as shown in Figure 5.12 (b). **(top)** The workspace coverage values are shown for *scenario 1*. **(bottom)** The workspace coverage values are shown for *scenario 2*. The maximum value of coverage (dark red) is nearly the same as shown by the color scale maximum. However, the (x,y) position of the maximum differs.

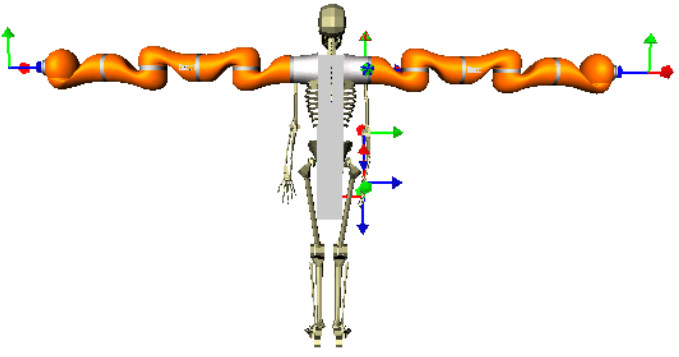


Fig. 5.17 The human is placed in the *scenario 1* for optimal workspace overlap. The bases of the robot arm and the human arm are at the same position. The robot and the human are in constant collision. Therefore this placement is impossible.

5.2.4.2 Comparison with a Restricted Human Workspace

The previous section evaluated the coverage of the entire reachable workspace of the human arm by the robotic arm. However, as recent research shows [41] only a limited portion of the human arm's workspace is used in every day life. When analyzing virtual manufacturing scenarios, additional ergonomics aspects can also be considered. Especially situations where a person manipulates above the head

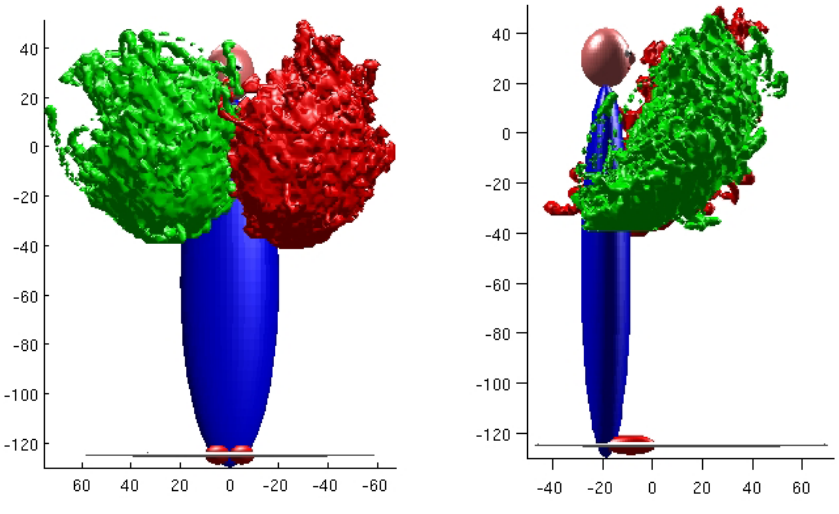


Fig. 5.18 The distribution of the wrist position was recorded with a portable tracking system as the subject went about his daily routines outside of a laboratory setting. (Source: Howard et al. [41])

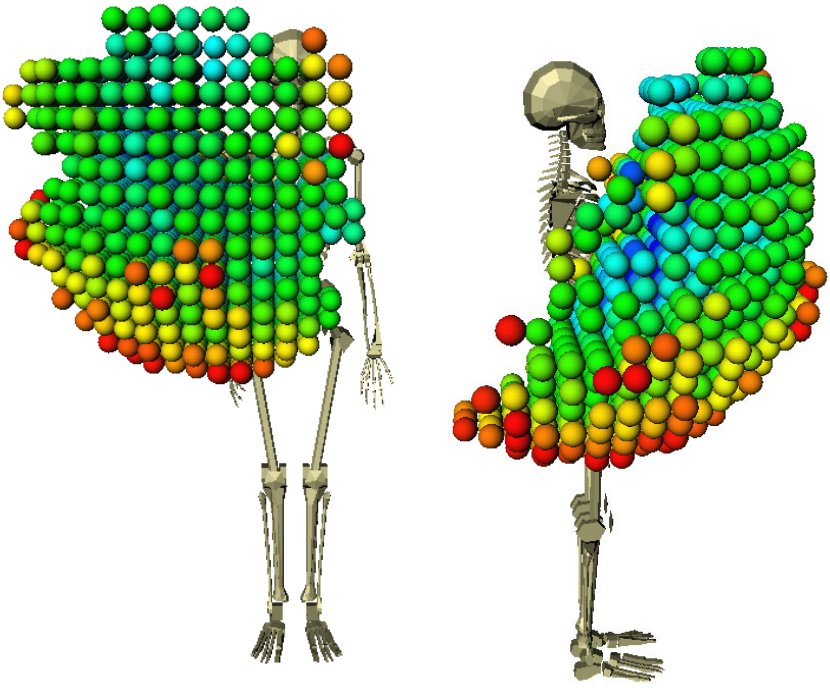


Fig. 5.19 The capability map for the human right arm is restricted to the volume that overlaps with the natural human movement data. The resulting filtered capability map is shown in two views.

should rarely occur in manufacturing tasks. The postures that the human would have to assume in these cases receive bad ergonomic scores [74]. If they are executed often, they result in damage to the musculoskeletal system of the person involved.

For these reasons the comparison of the workspaces is restricted to that region primarily used by the human. Howard et al. [41] collected data from subjects who wore a portable motion tracking system to record their arm movements as they went about their daily routines outside of a laboratory setting. No specific instructions were given and the system allowed the subject to engage spontaneously in normal everyday tasks. The results show that the majority of wrist positions fall within a region close to the body and also mostly to the front of the body. In Figure 5.18 two views of the dataset for one subject are shown.

For the following comparison, the dataset is used with the permission of Howard et al. [41]. The position of the palm is extrapolated from the wrist position. Therefore, the region shown in Figure 5.18 is slightly enlarged for the comparison. The comparison is restricted to that part of the human arm's workspace that overlaps with the natural movement data. Only the corresponding spheres from the human arm capability map are used. They are shown in Figure 5.19. Thus, the robot arm

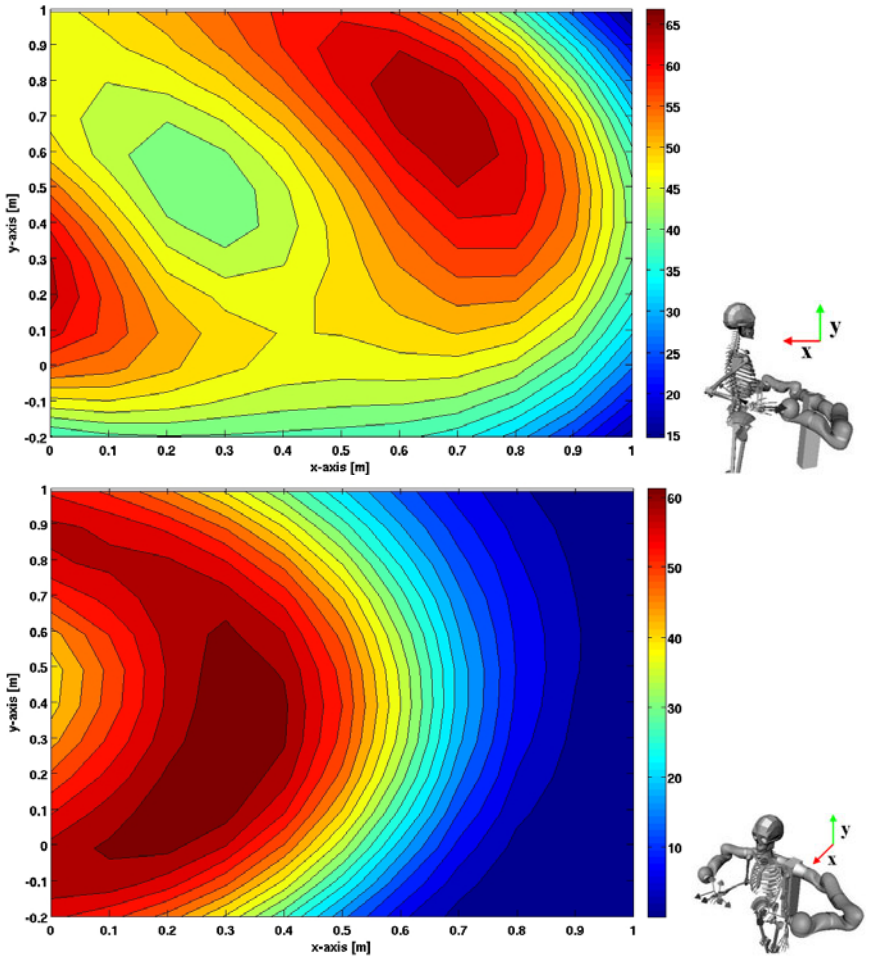


Fig. 5.20 It is shown how well the robot arm workspace covers the restricted workspace of the human right arm. The analysis is performed for different positions of the human in the *xy* plane of the global coordinate system. The right column shows how the human interacts with the system in the respective scenario and and shows the directions of the *x* and *y* axes. The coordinate system base lies as shown in Figure 5.12 (b). The workspace coverage values are shown for the human in **(top)** *scenario 1* and **(bottom)** *scenario 2*.

only has to cover this restricted workspace. For the restricted workspace, comparison results are shown in Figure 5.20 (top) for *scenario 1* and in Figure 5.20 (bottom) for *scenario 2*. It can be seen that again the height of the maximum (region in dark red) does not differ significantly. In *scenario 1* the maximum coverage is 66.8% and in *scenario 2* the maximum coverage is 64.4%. The Cartesian region where the human can be positioned to reach these maxima is collision free in both cases. *Scenario 2* is recommended, since the area of the region where the maximum coverage

occurs is larger. Furthermore its extension in the y-direction is larger. This implies that the shoulder height of the operator is allowed to vary over a greater range without the need to reposition the user. Thus different operators can more easily achieve a good performance with the system if they are located at a fixed position with respect to the robotic system. Additionally, in *scenario 2* the danger of the robot hindering the human by being in the way is reduced. In conclusion, the ergonomic *scenario 2* is more attractive. Especially since the location of the maximum coverage is approximately the same in the comparison across the whole workspace and in the comparison using the workspace filtered by the natural human movement data. Therefore, this system setup has greater potential for more general tasks.

5.3 Summary

This chapter demonstrated the application of the capability map for visualization of robot workspaces and setup evaluation. It was shown that the capability map supplements the design process for human-like or robot arm kinematics. The results concerning which serialized human arm kinematics better represents a human's capabilities can be used when building humanoid robots.

Furthermore, a method was presented to objectively evaluate robotic setups in which a human operator directly interacts with two robotic arms. The method was applied to two scenarios. The choice of data for the comparison was discussed and its effect on the evaluation was demonstrated. The placements of the human shoulder can be identified for which good performance is achieved. The proposed method can therefore be used as a quality criterion for optimizing human-robot interaction setups.

Chapter 6
Application in Planning

This chapter demonstrates the use of the capability map in planning tasks. Using two examples, it is demonstrated how the capability map can be used to restrict the search space. The algorithms thus address the gap between task planning and path planning that is indicated in Figure 6.1 by the red rectangle. In the first application covered in this chapter a robot is placed to perform a given trajectory. Its suitability for the task is evaluated. In the second application the capability map is used to

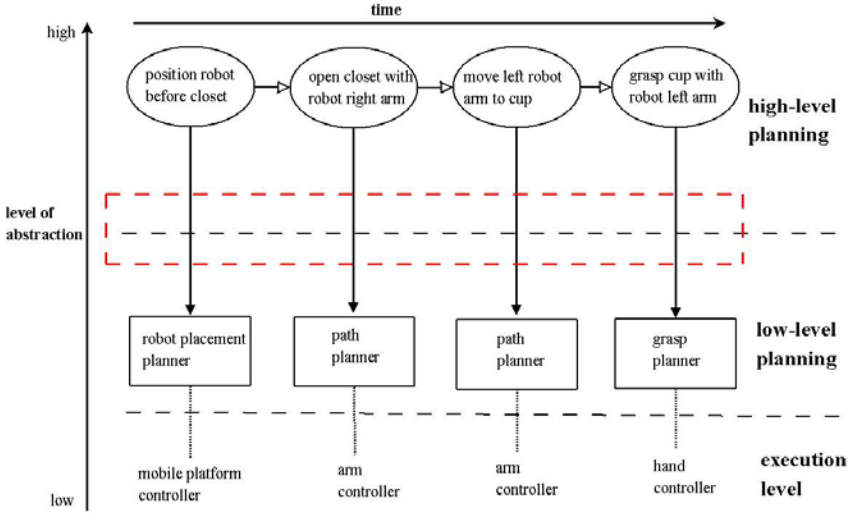


Fig. 6.1 Different levels of abstraction during the planning and execution of a service task are shown. The task of fetching a cup from a closet is used as an example. High-level planning is equivalent with task planning. Possibly parallel running actions are mapped to low-level planners. The planning results are then executed by divers robot controllers. The red rectangle indicates that the capability map is used to address the gap between task planning and path planning.

F. Zacharias: Knowledge Representations for Planning Manipulation Tasks, COSMOS 16, pp. 93–123.
springerlink.com

Fig. 6.2 The DLR mobile humanoid robot *Rollin' Justin*. Two DLR LWR arms are attached to an upper body. The omni-directional mobile base has a variable foot print.

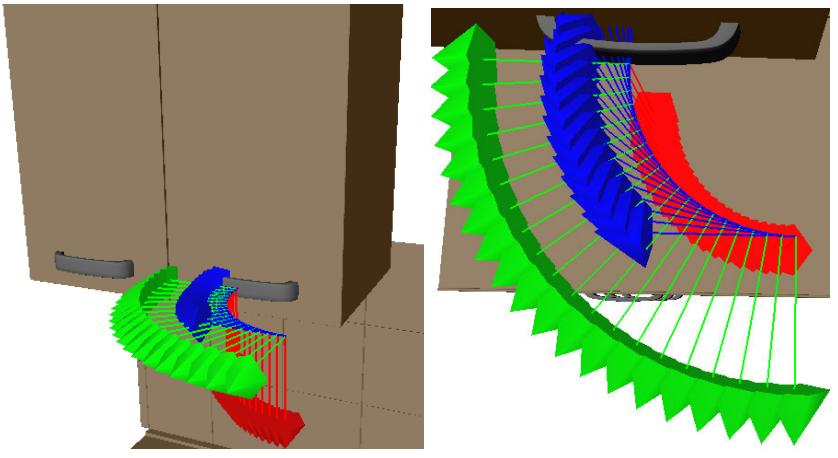
obtain good parameters for a path planner and bias the path planning process. The methods are evaluated using the mobile humanoid robot *Rollin' Justin* (Figure 6.2).

6.1 Placement for 3D Trajectories

Kitchen tasks require fetching and carrying things, as well as manipulating and interacting with the environment. To accomplish these tasks, the robot has to use knowledge about the specific environment and knowledge about its own capabilities. It has to know how to open a closet or how well it can grasp objects. The robot also has to decide when to use which part of its body. Not only the question when to use the upper body is important but also when to use the mobility of the base. To execute simple trajectories with the robot arm it is not always necessary to use the mobile base. On the contrary, if the mobile base is unnecessarily used, e.g. while opening a kitchen closet, additional forces acting on the arm have to be compensated. These forces are due to the fixed grasping of the door handle and the navigation errors of the mobile base.

6.1.1 Detailed Problem Analysis

In service robotic applications, unconstrained pick and place tasks are often encountered. In these tasks, the robot moves an object from a start to a goal position. To



(a) **(b)**

Fig. 6.3 When opening a closet, the hand that grasps the door handle is constrained to move on a specific trajectory. In the images, a coordinate system is fixed in a specific pose on the door handle. When the door is opened, it constantly changes its orientation while moving on a circle segment. The images show two views of this trajectory. **(a)** A trajectory for opening a closet is shown. **(b)** A zoomed view is shown.

perform these tasks, standard path planning algorithms can be queried for a suitable path. However, not only unconstrained, freely planable paths are needed. Interaction with the environment is subject to physical constraints. For opening a closet, the TCP of a robot arm is constrained to move on a circular path (Figure 6.3) [120]. If the TCP frame is attached to the handle of the closet, the orientation of its z-axis (blue arrow) constantly changes. The radius and orientation of this path depend on the design of the closet. The trajectory that the TCP has to follow is called a constrained trajectory. Due to the robot arm kinematics and link limits, executing constrained trajectories is not possible at arbitrary positions in a robot arm's workspace. Depending on a robot arm's capabilities or the arm's attachment to an upper body, some mobile manipulators may not be able to perform certain tasks at all. A method is needed to analyze the capabilities of a robot given an environment and typical tasks performed therein. Therefore, a method is needed that works for generic 3D trajectories $\in \mathbb{R}^3 \times SO(3)$ of the robot arm TCP and determines where these trajectories are situated in a robot arm's workspace. Given this information, a mobile manipulator can be placed easily. In this section, the capability map is used to determine for given 3D trajectories, where these trajectories are possible in a robot arm's workspace.

6.1.2 The Search Pattern

If a closet has to be opened, the end-effector grasps the handle and moves on an arc (Figure 6.3). The trajectory followed by the TCP is assumed to be given as a sequence of frames $F_l,\ l > 0$. A frame F_l is represented as a homogeneous matrix

$$F_l = \begin{pmatrix} R_l & \mathbf{t}_l \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{6.1}$$

with $R_l = (\mathbf{x}_l\ \mathbf{y}_l\ \mathbf{z}_l) \in SO(3)$ and $\mathbf{t}_l \in \mathbb{R}^3$ describing rigid body rotation and translation. The frames are defined with respect to a local reference system. The set of frames $\{F_l | l = 1..n\}$ is mapped to an equal number of discrete representations in the capability map. For simplicity it is assumed that the trajectory lies in the first octant of a 3D reference frame. Thus, the discretized representation p_l of a frame is element of $\mathbb{N}^5$. The set of frames is mapped to a search pattern $p = \{p_l | l = 1..n,\ p_l \in \mathbb{N}^5\}$ using $M(F_l)$,

$$\begin{aligned} &M(F_l) = p_l = (s_1, s_2, s_3, i, k)^T \in \mathbb{N}^5 \\ &\text{with } (s_1, s_2, s_3)^T \in \mathbb{N}^3,\ i \in N_p,\ k \in N_o \end{aligned} \tag{6.2}$$

which is an adaptation of the method specified in Section 4.3.5. The pattern is derived in the following manner.

For each position $\mathbf{t}_l$ of the Cartesian trajectory it is first determined which voxel and accordingly which sphere it maps to with respect to the reference point of the trajectory. Figure 6.4 (left) shows a trajectory superimposed on the workspace discretization of the reachability sphere map. The 2D projection is chosen for illustration. The filled spheres symbolize the spheres the trajectory is mapped to. Let f be the function that maps $\mathbf{t}_l$ to a sphere in the pattern given the discretization, i.e. the sphere diameter l_c of the underlying the reachability sphere map. Each sphere is

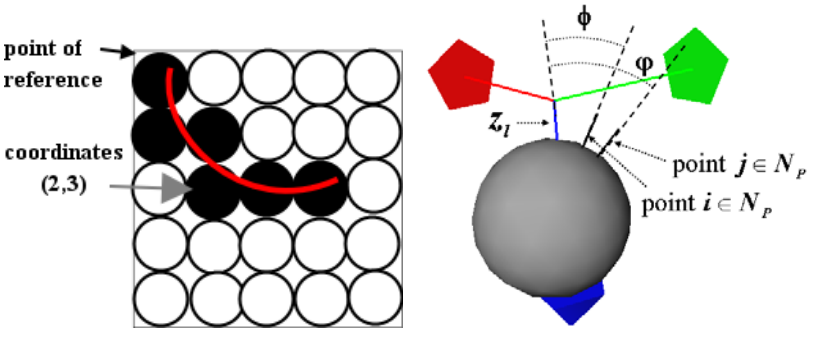


Fig. 6.4 Discretization of a trajectory. **(Left)** 2D view of mapping positions along the trajectory on the sphere map grid. **(Right)** Mapping of the frame orientation to a point on the sphere.

represented by an offset $(s_1, s_2, s_3)^T$ with respect to the reference point of the pattern as shown in Figure 6.4 (left) for a 2D case.

$$f(\mathbf{t}_l) : \mathbb{R}^3 \rightarrow \mathbb{N}^3 \; \mathbf{t}_l \mapsto \; f(\mathbf{t}_l) = (s_1, s_2, s_3)^T \tag{6.3}$$

In a second step, the z-axis $\mathbf{z}_l$ of frame F_l is mapped to the best fitting point with index i on the sphere.

$$i = \text{argmin}_{j \in N_p}(\arccos(< \mathbf{z}_j, -\mathbf{z}_l >)) \tag{6.4}$$

Let $F_{i,0}$ be the coordinate system attributed to the point with index i and nominal orientation $\alpha_0 = 0$ around $\mathbf{z}_{i,0}$. Note that $\mathbf{z}_{i,0} = \mathbf{z}_{i,m_o} = \mathbf{z}_i$ and $\| \mathbf{z}_i \| = \| \mathbf{z}_l \| = 1$. The mapping described by Equation 6.4 is illustrated in Figure 6.4 (right).

For the computation of the reachability sphere map the orientation around the z-axis of a frame (Figure 4.2(b)) is discretized into m_o steps. In the last part of the mapping process the orientation around the z-axis $\mathbf{z}_l$ of frame F_l is mapped to one of these m_o orientations. The x-axis $\mathbf{x}_l$ of frame F_l is projected onto the xy-plane of the coordinate system defined by the frame $F_{i,0}$.

$$\hat{\mathbf{x}}_l = R_{i,0} \cdot P_{xy} \cdot R_{i,0}^{-1} \cdot \mathbf{x}_l \tag{6.5}$$

Let P_{xy} be the projection matrix for projection onto the xy-plane. The angle β between the projected axis $\hat{\mathbf{x}}_l$ and the x-axis of frame $F_{i,0}$ is then computed.

$$\beta = \arccos\left(< \frac{\hat{\mathbf{x}}_l}{\| \hat{\mathbf{x}}_l \|}, \frac{\mathbf{x}_{i,0}}{\| \mathbf{x}_{i,0} \|} > \right) \tag{6.6}$$

The angle β is discretized using the discretization step width Δ_o of the orientation around the z-axis.

$$k = \lfloor \frac{\beta}{\Delta_o} + \frac{1}{2} \rfloor; \; k \in N_o \tag{6.7}$$

In Figure 6.5 an example trajectory is shown for opening a closet and the search pattern in the space of spheres. In Figure 6.5 (left) the large coordinate frames represent the original trajectory frames. The smaller frames represent the mapped frames in the sphere space. In Figure 6.5 (right), the red lines show to which lines on the sphere the frames are mapped.

6.1.3 The Search for the Trajectory in the Workspace

Cross-correlation is a standard technique in signal processing to determine the shift between two signals. The signals are specified over the same domain, e.g. $\mathbb{R} \rightarrow \mathbb{R}$ for audio signals over time or $\mathbb{R}^2 \rightarrow \mathbb{R}$ for gray images. The result of a correlation is a signal in the same domain, showing peaks at those locations where the two signals best match each other. This idea is used to find the search pattern p in the reachability sphere map M_S. The search is done by correlating the capability map with the given search pattern p. Figuratively speaking the pattern is moved across the

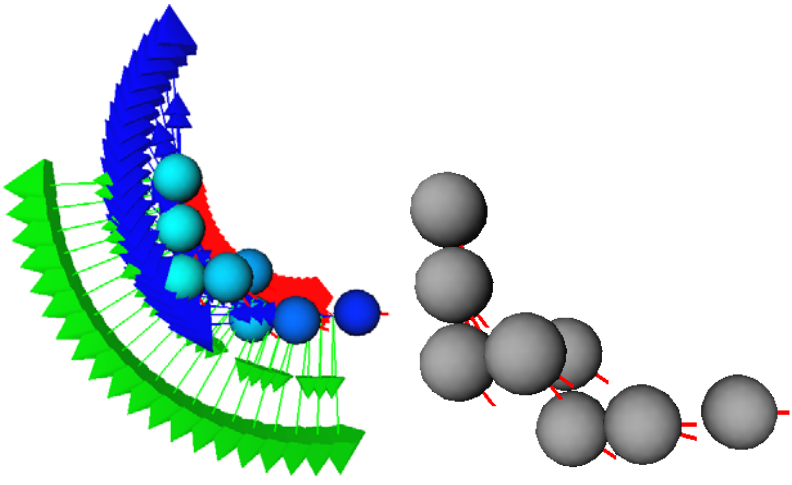


Fig. 6.5 The trajectory is represented as a sequence of frames which is mapped to its discrete representation in the sphere space. **(Left)** The large coordinate systems represent the original trajectory frames. The smaller frames represent the mapped frames in the sphere space. **(Right)** In a zoomed view, the red lines on the spheres show which points the frames are mapped to.

capability map and compared with the data present. M_S is the 3D data structure which represents the capability map (Equation 4.26). A discretized frame is represented by $\mathbf{a} \in M_S$. The function $h(\mathbf{a})$,

$$h : M_S \rightarrow \{0,1\}; \mathbf{a} \mapsto h(\mathbf{a}) := \begin{cases} 1 \text{ if } \mathbf{a} \text{ reachable according} \\ \quad \text{to capability map} \\ 0 \text{ if } \mathbf{a} \text{ not reachable} \end{cases} \tag{6.8}$$

is 1 if the tuple $\mathbf{a}$ is reachable according to the capability map and 0 if it is not reachable. (Equation 5.3 is repeated here for convenience.) The search pattern p is obtained as described in the previous section. Equation 6.9 defines the correlation between the capability map and the pattern if the pattern is added to voxel $\mathbf{g}$, i.e. the reference point of the pattern is placed there. Let $\mathbf{p}_{min} \in \mathbb{N}^3$ be the lower left corner and $\mathbf{p}_{max} \in \mathbb{N}^3$ be the upper right corner of the smallest axis-aligned bounding box enclosing the pattern p in the voxel space.

$$(M_S * p)(\mathbf{g}) = \sum_{l=0}^{|p|} h(f_s(g_1, g_2, g_3) + \mathbf{p}_l) \tag{6.9}$$

$$f_s : \mathbb{N}^3 \rightarrow \mathbb{N}^5; \mathbf{g} \mapsto f_s(\mathbf{g}) = (g_1, g_2, g_3, 0, 0)^T \in \mathbb{N}^5 \tag{6.10}$$

The function f_s translates a given sphere coordinate $\mathbf{g}$ into the starting point $\in M_s$ in the capability map at which to place the pattern. To get a description of how well

the pattern fits across the whole capability map, $g_1,\ g_2,\ g_3$ iterate over the whole capability map M_S minus the dimension of the bounding box for the 3D search pattern p. The pattern is moved across the whole capability map by adding the pattern element p_l to the current starting point (sphere) $\mathbf{g} \in V_{Robot}$ in the 3D sphere space and thus trying to fit the pattern there. The result $(M_S * p)$ is a 3D representation that describes how well the trajectory fits across the capability map. The places are searched in the robot arm workspace where the pattern fits completely and the correlation value equals the pattern length. In Figure 6.6 the correlation result is shown for opening a closet at a given height. Justin's torso is in its zero position where all active torso link values are zero. The trajectory is at about shoulder height (Figure 6.6 (top)). Since the trajectory is composed of 20 frames, the correlation result ranges from 0 to 20. To open a closet only correlation results are of interest that lie on the plane that also contains the closet handle. In Figure 6.6 (middle) and (bottom) the correlation results for this plane are interpolated and color coded. Note that the positive x-axis indicates the front of the robot. A value of 20 (dark red) means all frames of the trajectory are predicted to be reachable if the trajectory is started at the corresponding point in the robot arm workspace. It can be seen that the region in which the trajectory can be performed completely is composed of two parts, which lie to the front and to the back of the robot Justin.

Note that with this method, the pattern will not be found if it occurs in the image in a different orientation. The standard solution to this is to rotate the pattern using a fixed stepsize. Accordingly, the original Cartesian space trajectory can be rotated before being mapped to the pattern.

6.1.4 Computing the Robot Base Position

The capability map is computed with respect to the robot arm base. Thus if the base is moved, the map moves accordingly. Once the position of the trajectory is computed in the workspace with respect to the robot arm base, the position of the mobile manipulator with respect to the world can be determined easily.

The z-axis of the world system is assumed to point upwards. The transformation from the old to the new robot base position involves only a rotation R around the z-axis of the world system and a translation $\mathbf{t}$ in the xy plane. Let $\mathbf{t} = (x, y, z)^T$ be a translation and R be a orientation of the search pattern with respect to the map space for which the correlation of Equation 6.9 reaches the maximum. Equation 6.11 gives the target position of the arm base in a reference frame *world*, e.g. the world base. The placement of the mobile base follows directly.

$$T_{arm}^{world} = T_{object}^{world} \cdot (T_{object}^{arm})^{-1} = T_{object}^{world} \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}^{-1} \tag{6.11}$$

T_B^A are homogeneous matrices that describe the pose of frame B in coordinates of reference frame A.

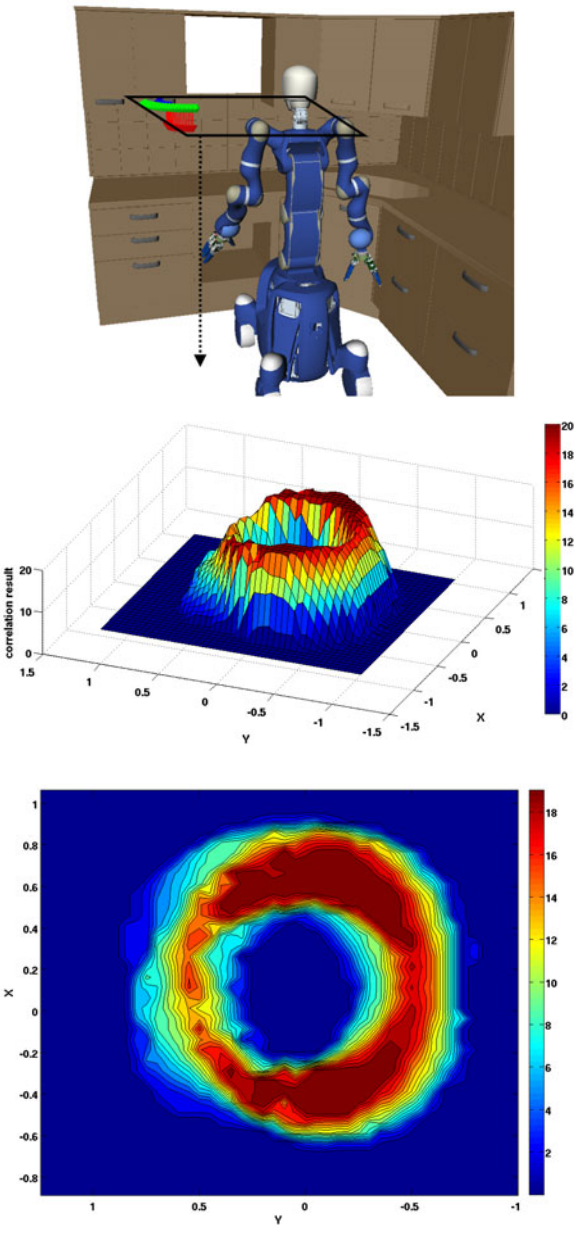


Fig. 6.6 The humanoid robot has to open a kitchen closet. **(top)** The trajectory is drawn beginning at the closet handle. Only correlation results are of interest that lie on the plane that also contains the closet handle. The plane is indicated by the black rectangle. The correlation results for this plane are interpolated and color-coded. **(middle)** A relief view of the correlation result is shown. **(bottom)** A contour view of correlation result is shown.

6.1.5 Computational Complexity

The computational costs of performing the trajectory search in position space are compared with the costs of performing the search in the frequency domain. For cross-correlation in image processing two images are first transformed into frequency domain using the Fourier transformation. In frequency domain, the spectra are multiplied component-wise and the result of the multiplication is transformed back using the inverse Fourier transformation. The capability map and the search pattern are the correspondences of the images. It is known that the fast discrete Fourier transform (FFT) using Cooley-Tuckey's radix-2 algorithm has a time complexity of $O(N\log(N))$, where N is a power of factor 2. Let $N_{\mathrm{D}}^3 = n_c^3$ denote the volume of the discretized robot workspace. The complexity is highest if the search pattern has the dimension of the workspace, i.e. $N_{PA} = N_D$. According to the number of multiplications for a single FFT [17], the total cost for two Fourier transformations and for the multiplication in the frequency domain involved are

$$\mathrm{Cost}_{\mathrm{freq}} = |O|N_{\mathrm{D}}^3 \log_2(N_{\mathrm{D}}^3) + |O|N_{\mathrm{D}}^3 \tag{6.12}$$

where $|O| = n_p \cdot m_o$ with n_p points on the sphere and m_o the number of discretized orientations around $\mathbf{z}$. Note that the cost of the Fourier transformation of the reachability map is neglected because it can be computed once and used for different planning tasks. In the case of the discretization of Justin's arm workspace with side length $N_{\mathrm{D}} = 40$ spheres and $|O| = 200 \cdot 12$ orientations ($n_p = 200,\ m_o = 12$), a total number of $\mathrm{Cost}_{\mathrm{freq}} = 2.6 \cdot 10^9$ multiplications is needed.

In contrast, the number of multiplications for general cross-correlation in the space domain is $|O| \cdot N_{\mathrm{D}}^3 \cdot N_{\mathrm{PA}}^3 = 1.5 \cdot 10^{11}$, whereas a side length $N_{\mathrm{PA}} = 10$ spheres of the pattern is assumed. Contrary to the general case, significant optimizations can be applied here because only a few entries in the trajectory volume are non-zero. Let $|p|$ denote the length of the discretized trajectory, i.e. the number of pattern elements then the multiplications amount to

$$\mathrm{Cost}_{\mathrm{space}} = (N_{\mathrm{D}} - N_{\mathrm{PA}})^3 \cdot |p| \,. \tag{6.13}$$

Only a sub volume of the robots capability map $(N_{\mathrm{D}} - N_{\mathrm{PA}})$ is considered, because the trajectory is requested to be completely within the workspace of the robot. Whenever $|p| = N_{\mathrm{PA}}$, the costs reach their maximum at $N_{\mathrm{PA}} = N_{\mathrm{D}}/4$. In the case of Justin's arm, this corresponds to $N_{PA} = 10$ and only $\mathrm{Cost}_{\mathrm{space}} = 2.7 \cdot 10^5$ multiplications[1], which are four orders of magnitude fewer then assessed for the correlation with the Fourier transformations. Therefore, the correlation is done in the space domain.

[1] Since both volumes are binary, the multiplication can be replaced with a simple binary AND operation.

6.1.6 Discretization Issues

To unambiguously represent the task-specific 3D trajectory template, the trajectory has to be sampled according to the Nyquist-Shannon sampling theorem [99]. If this theorem is violated and too few frames characterize the trajectory, the trajectory cannot be correctly reproduced and the pattern does not correctly represent the trajectory. In this case the trajectory is aliased with a less frequent one. In Figure 6.7 (left) a 2D trajectory is represented by too few samples. When reproducing a continuous trajectory from the sample points, the gray line is obtained instead of the original trajectory. Using only the three samples indicated by squares, the two trajectories cannot be distinguished. From this follows that the search results for the sampled trajectory are not valid for the original trajectory and prediction errors occur. This has to be considered when obtaining the trajectory search template.

For trajectories with low amplitude i.e. $amplitude < sphere\ radius$ (Figure 6.7 right) it is assumed that the interpolation assumption holds which underlies the capability map. This states that at each point of a voxel the same orientations are reachable as on the sphere located at its center. Thus, the pattern can be generated as described in Section 6.1.2 provided that the trajectory is correctly sampled.

6.1.7 Brute-Force Search vs. Model-Based Search

Without a capability map, a brute force search can be used to determine whether a given trajectory can be performed by the robot. The start point for the trajectory is randomly sampled using a uniform distribution. However a sampled start point only contributes to the computed success rate if the position of the first and last frame of the trajectory lies within the hull of the reachable workspace of the robot arm. Each trajectory is checked for reachability using inverse kinematics. A relative success measure is computed by determining how many trajectories are predicted by the capability map to be reachable and how many are reachable as determined by the inverse kinematics. In Table 6.1 the likelihood is reported to find the arc for opening a closet at the height shown in Figure 6.6 (top) for the torso in its zero position. Results are reported for the brute force search and the capability map based

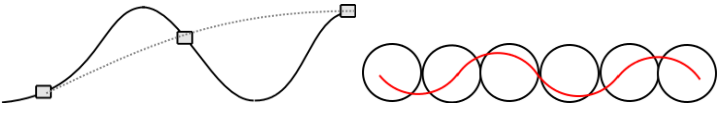


Fig. 6.7 The trajectory has to be correctly discretized. Otherwise, it is indistinguishable from one of lower frequency. This effect is called aliasing. From this follows that the correlation results of the trajectory search are not valid for the original trajectory. **(left)** The aliasing effect is shown for a 2D trajectory where the sampling violates the Nyquist Shannon sampling theorem. The squares represent sampling points. When reproducing the trajectory the gray line is obtained instead of the original trajectory. **(right)** A trajectory is shown that has an amplitude that is smaller than the sphere diameter.

Table 6.1 Results of the brute force search are compared with the capability map-based approach. The terms *samples* and *predicted* are synonymous with potential candidates. The brute force search wastes time in regions where the trajectory does not exist. The ratio of evaluated to valid trajectories is much better for the search based on the capability map.

brute force search			capability map search		
samples	valid	%	predicted	valid	%
$1 \cdot 10^6$	39807	3.98	119	96	80.7

search for the trajectory in one orientation. Both approaches are evaluated on the same computer. The results for the brute force approach show that a lot of effort is wasted on regions where the trajectory does not exist. While the presented efficient approach needs 1.6 s to find and verify the 96 trajectories, the brute-force approach finds the same number of valid trajectories in 20 s. Considering that the trajectories may still be inconsistent or colliding, the advantages of the model-based approach are emphasized. Note that the prediction accuracy for finding whole trajectories is not as high as for isolated frames (Table 4.4) because errors accumulate.

6.1.8 Validation of Solutions

The correlation process itself is very efficient. It needs 25 ms on a computer with Intel Pentium D 3 GHZ processor and 2 GB main memory to find all occurrences of an arc trajectory with 20 frames in one orientation within the capability map. Only the correlation results are considered that lie in an area of the workspace that is of interest for the given task, i.e. at the height of the closet handle. However, the solutions have to be validated because of the following reasons:

6.1.8.1 Generalization Assumption

The correlation process provides a number of starting points (the center of a voxel) for trajectories. The trajectory started at these positions needs to be checked for actual reachability since the capability map assumes that entries in the map generalize over the complete voxel. It is assumed that at each point of a voxel the same orientations are reachable as on the sphere located at its center, i.e. that the reachability structure does not change for small increments in space. However at the inner and outer border of the workspace or at the border of structurally different regions, this assumption may be violated. As Section 4.3 shows the prediction accuracy is 94.6% for single TCP frames. For sequences of frames, the prediction accuracy may be lower.

6.1.8.2 Collision-Free and Consistent Trajectories

Using the capability map, the reachability of the trajectory is independently evaluated for the individual frames. It is not guaranteed that the robot is able to follow

a smooth trajectory between these frames, e.g., in the vicinity of singularities reconfigurations of the robot arm will occur. Therefore each trajectory is checked for consistency by setting a threshold on the allowed link-wise change between two configurations for two adjacent path steps. Currently, the threshold is empirically chosen and is set to 23° (0.4 rad). However if an execution time is given for the trajectory, the threshold should be dependent on the maximum link velocities and the time. Additionally, the trajectory is checked for collisions of the robot with the environment and for collisions of the robot arm with the head or the upper body.

The complete algorithm for searching 3D trajectories $\in \mathbb{R}^3 \times SO(3)$ in the capability map is summarized by Figure 6.8.

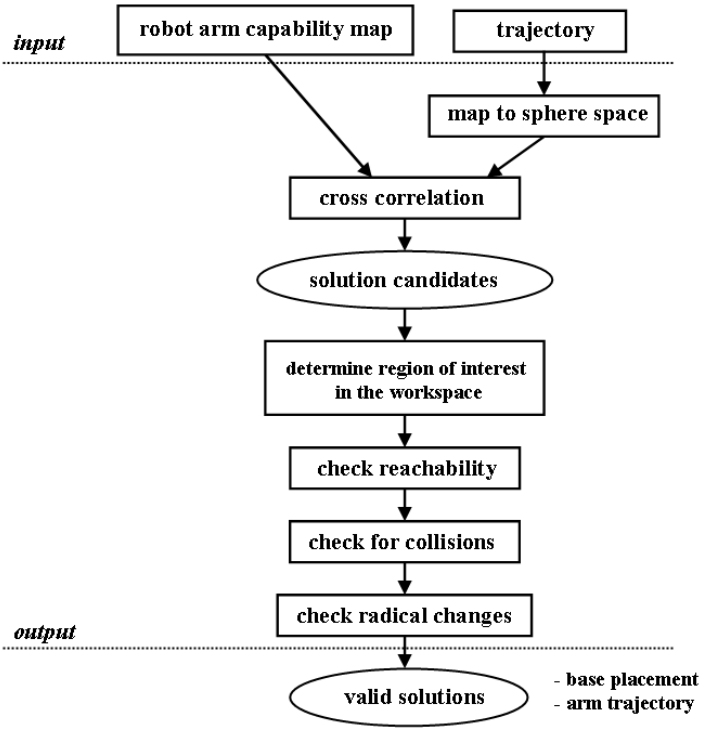


Fig. 6.8 The algorithm for the search of trajectories in the capability map is shown at a glance. Inputs and outputs are listed.

6.1.9 Robots Working in the Kitchen

In a kitchen, a robot should above all be able to open closets, to extract dishes, or fill the dishwasher. Placements of the mobile manipulator (Figure 6.2) are computed using the presented method and validated for opening the door of a closet (Figure 6.3). The trajectory search is only performed for the original orientation of the trajectory.

Table 6.2 The arc trajectory is searched for the torso in configuration C1 and C2. The same closet is opened. For C1 about half the number of solutions are predicted as for configuration C2. Only a third of the number of valid solutions found for C2 is valid for C1. The column *reconf.* states how many solutions involve reconfigurations of the arm. The column *colliding* reports the number of solutions that are in collision with the environment.

	nr. of solutions				
torso conf.	predicted	real	reconf.	colliding	valid
C1	119	96	70	82	7
C2	204	195	97	157	27

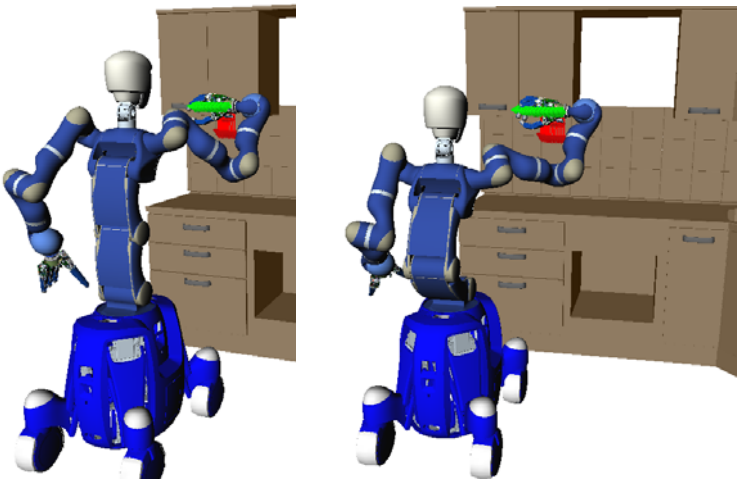
In Table 6.2 results are reported for two different configurations $C1$ and $C2$ of the movable upper-body of the robot *Rollin' Justin*. The results show significant differences for the two torso configurations. For configuration $C2$ more reachable solutions are found. Therefore the presented approach is able to determine a beneficial torso configuration for a given task. In Figure 6.9 a robot placement for each torso configuration is shown. After the validation step, a task planner can choose from a set of valid solutions. For both configurations, the number of collisions is striking. Since the workspace of the arm is nearly symmetric solutions are found in front of and behind the torso of Justin (compare Figure 6.6). In the latter case Justin has to be placed inside the kitchen closets to execute the trajectories. Furthermore, often collisions of the arm and the head occur. The results for the check for consistency and freedom of collision are a proof of concept. The inverse kinematics currently computes independent solutions for the individual frames and does not exploit the null space of the 7-DOF robot arm to avoid collisions. For trajectories that are currently invalid, it is expected that there exist alternative arm configurations that lead to consistent and collision-free solutions. The time consumption is measured for each part of the algorithm for the torso in configuration C1. The computation times are summarized in Table 6.3. The test is performed on a computer with Intel Pentium D 3 GHZ processor and 2 GB main memory.

Table 6.3 Computation times for the steps of the algorithm to search for 3D trajectories $\in \mathbb{R}^3 \times SO(3)$. The test is performed on a computer with Intel Pentium D 3 GHZ processor and 2 GB memory.

correlation	reachability	reconfiguration	collision	total
25 ms	1564 ms	0.113 ms	31 ms	1620 ms

6.1.10 Following the Motion of a Tumbling Satellite

In the future, robot arms could be attached to carrier satellites to perform on orbit servicing missions (Figure 6.10) in space for defect satellites or satellites that have run out of fuel. At the DLR, the ESS demonstrator [60] simulates the capture of a tumbling target satellite using a robot mounted on a carrier satelite. However, the Kuka industrial robot used in the ESS demonstrator cannot be mounted on a real



(a) (b)

Fig. 6.9 Rollin' Justin is placed to open a closet. Solutions are shown for two configurations of the torso. **(a)** The torso is in configuration C1. **(b)** The torso is in configuration C2.

Fig. 6.10 The ESS demonstrator simulates the approach and docking of a robot arm to a tumbling satellite. A Kuka industrial robot approaches a satellite and captures it using the capture tool developed at the DLR.

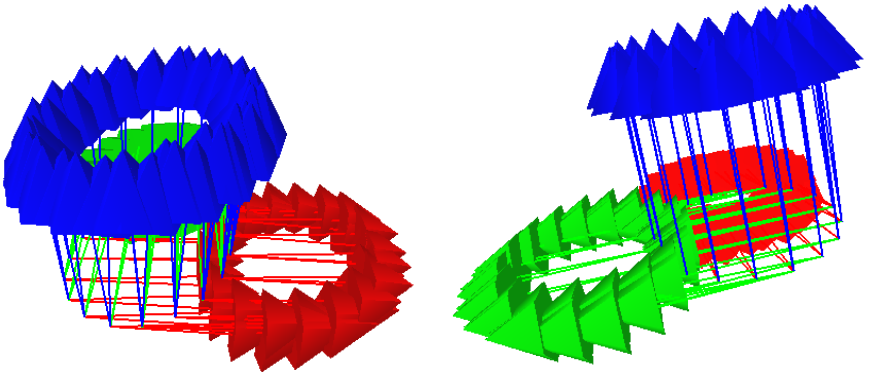


Fig. 6.11 To capture a defect satellite, a robot arm attached to a carrier satellite has to be able to follow the tumble motion of the defect satellite. A coordinate system is attached to the center of gravity of the tumbling satellite. The trajectory of the center of gravity is shown in two orientations with respect to the same reference coordinate system.

satellite because it is too heavy. A Kuka LWR weights only 14 kg. Therefore, it presents an alternative which could be mounted on a satellite.

In the following the capability of the Kuka LWR to follow the motion of a tumbling satellite is examined. The presented method for 3D trajectory search is used to determine whether the Kuka LWR arm can follow the motion of a tumbling satellite. A coordinate system attached to the center of gravity of the satellite is assumed to tumble on a circular path as depicted in Figure 6.11. The trajectory of the satellite can appear in different orientations with respect to the robot. Therefore, 50 representative orientations are generated. For this purpose 50 points are distributed uniformly on a sphere using the algorithm by Saff et al. [96]. The points are used for the generation of frames $F_i,\ (i = 1..50)$ that describe the orientation i of the trajectory in the 3D Cartesian space. The frames are generated as already detailed in Section 4.2.2. The resulting frames are shown in Figure 6.12. Each frame F_i is used to reorient the trajectory $T = \{T_j | j = 1..n\}$ before the search.

$$\forall\, T_j :\ T_{i,j} = F_i \cdot T_j \tag{6.14}$$

The trajectory is searched in each of the orientations. In Figure 6.13 the number of reachable trajectories is shown that are found for each orientation i of the trajectory with respect to the robot arm base. It can be seen that for orientations 10 to 13 of the trajectory with respect to the robot arm only a quarter of the solutions are found compared to orientation 16 to 19. In Figure 6.14 the number of consistent trajectories are shown, i.e. trajectories without reconfigurations. From the results, it is evident that the task can be performed with the given robot. The orientation of the tumbling satellite with respect to the robot has significant influence on the number of solutions. Possibilities extracted from Figure 6.14 are to position the robot arm so that the trajectory appears in orientation 16 to 19 or 41 to 44 in the robot arm's

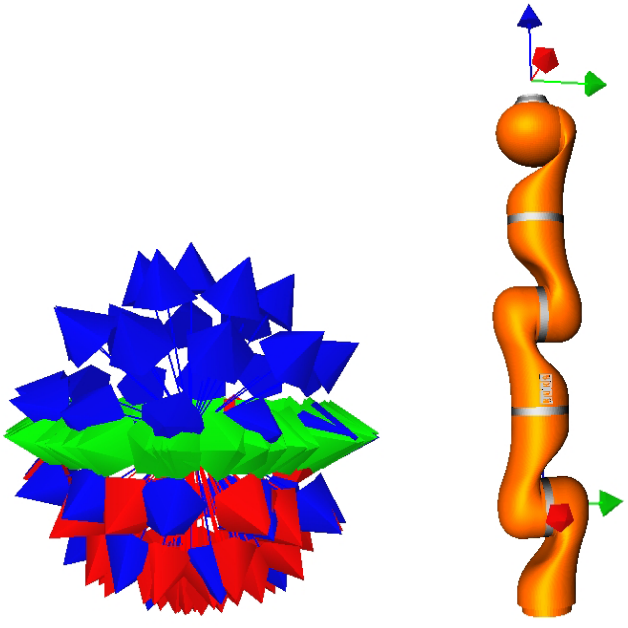


Fig. 6.12 The trajectory is searched in 50 different orientations. **(left)** The new base frames for the trajectory are visualized as coordinate systems. **(right)** The TCP frame of the robot is shown attached to the robot arm. It is used to build the capability map. The base frame of the robot is located in the first link. It is also shown by a coordinate system.

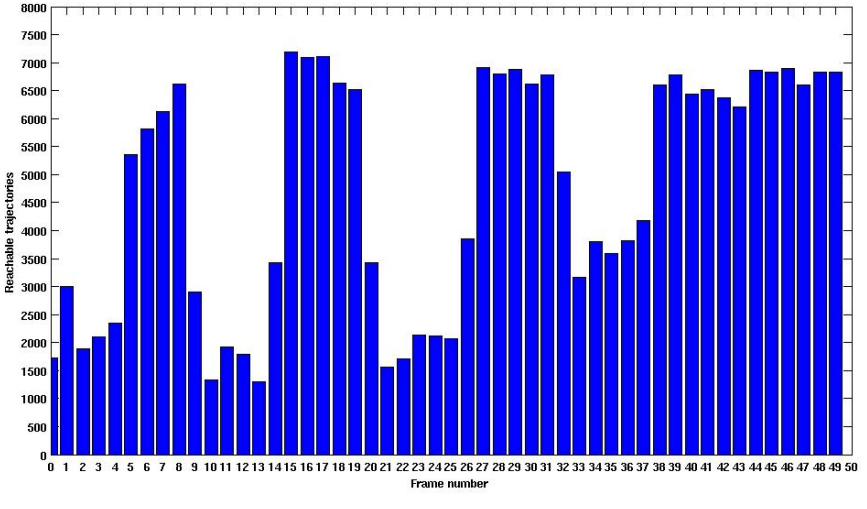


Fig. 6.13 For each orientation i (x axis) of the trajectory the number of voxels of the capability map is counted (y axis) where the trajectory can be executed.

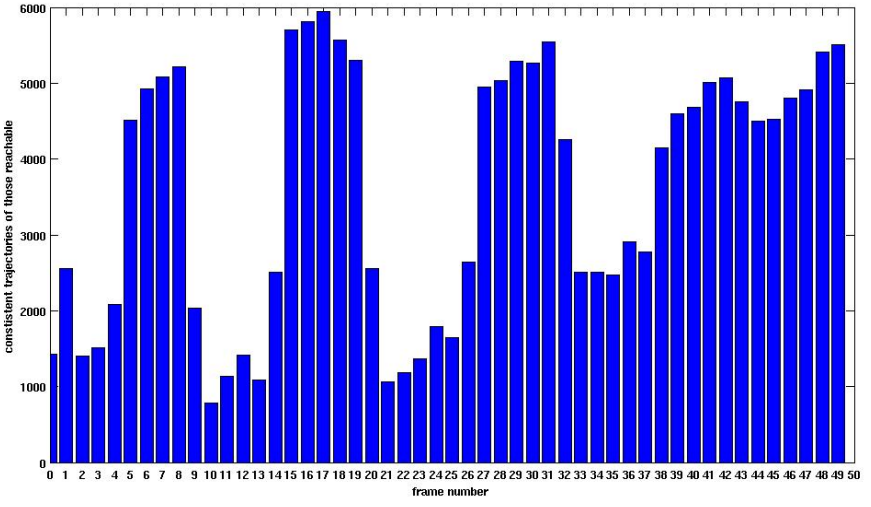


Fig. 6.14 For each orientation *i* (x axis) of the trajectory, the number of consistent solutions is counted (y-axis). Large differences in the number of solutions exist. Therefore the orientation of the robot with respect to the trajectory is very important to successfully capture the satellite.

workspace. It should be noted that also the dynamics of the robot and the satellite have to be considered. In space, the movement of the robot causes the carrier satellite to move. Therefore, not all consistent trajectories will fulfill the dynamics constraints. However, the results can directly be used to construct a new demonstrator for the laboratory that includes the Kuka LWR.

6.2 Human-Like Path Planning

People have strong expectations of how other persons act and move in a given situation. This may be a source of confusion when a human interacts with humanoid robots, and the robot does not move in a predictable or expected manner. If a humanoid robot like Rollin' Justin (Figure 6.15) uses awkward arm configurations (Figure 6.16), or trajectories to perform tasks, the human may feel uncertain and insecure.

The principles of path planning, especially probabilistic path planning have been extensively examined in the last decade. However, the paths that the planner computes are unpredictable and may seem awkward. While the problem of getting to an object is solved, the solution may not be comprehensible for the human observer or coworker. This section shows that the choice of an arm's target configuration strongly effects planning time and how human-like a planned path appears. Human-like goal configurations are found using a criterion from ergonomics research, the rapid upper limb assessment (RULA) [74]. By restricting the planning process to the regions of the robot arm workspace where human-like manipulation is possible, planning times and motion quality are enhanced. The knowledge of which pose of

Fig. 6.15 DLR humanoid robot Justin looks quite human-like and friendly when the arms are in a configuration that appears comfortable for a human. Justin can move in a very human-like manner, but it can also perform movements that look very awkward for a human observer.

Fig. 6.16 Justin places cubes on the table using both arms. While the left arm is able to put the cube on the table, it is in an awkward configuration. The elbow is bent inwards. This arm configuration is impossible for a human.

the TCP can be reached in a natural manner is encapsulated in a restricted capability map for the robot arm. This map is also used to determine which grasps from a given grasp set are usable for human-like manipulation. The proposed concepts are evaluated on the humanoid robot *Justin*.

6.2.1 The RULA Evaluation Criterion

Rapid Upper Limb Assessment (RULA) is a survey method for the fast and easy evaluation of ergonomic conditions at manual workplaces [74]. RULA introduces a

scoring system which investigators employ for a selection of possibly critical work postures in order to assess the resulting stresses and strains on the human musculoskeletal system. Here, RULA is used to identify human-like arm postures for humanoid robots under the assumption that natural and convenient human arm postures try to avoid stresses and strains and thus feature good RULA scores. The Institute of Man-Machine Interaction (MMI) at RWTH University Aachen implemented RULA for the Virtual Human in order to evaluate ergonomic conditions at manual workplaces. The assessment is carried out for a single work posture, yielding a score $S_{rula} \in \{\mathbb{N}|1 \dots 7\}$, where a lower score stands for an ergonomically better posture. For the assessment, the positions of the upper arm, lower arm and the wrist position are evaluated in terms of rough angles between the arm segments and anatomical planes of the body. The angles yield intermediate scores for each of the positions, based on which the summarized RULA score S_{rula} is extracted from tables. The anthropomorphic multi-agent approach [28] provides the basis to calculate RULA scores for humanoid kinematics. The Virtual Human and other humanoid kinematics are modeled as kinematic trees. An anthropomorphic multi-agent system model of Justin was implemented for ergonomic evaluation (see Figure 6.17) by MMI. This system is used to evaluate robot arm configurations of Justin using RULA .

For the adaption of RULA to anthropomorphic robot arm kinematics, arm configurations are penalized which operate beyond the scope of human joints. If the joint values grow beyond human-like limits, the *strict function* P_{strict} instantly yields a penalty of 7. The penalties are combined with regular RULA scores based on the maximum function:

$$S_{rula}^{strict} = max\{S_{rula}, P_{strict}\} \tag{6.15}$$

The joint limit penalties are an artificial extension to the original assessment, and are not necessary for the evaluation of human workers at manual workplaces. In Figure 6.17-Figure 6.19, the upper arm, lower arm and wrist scores are shown as well as the final RULA scores S_{rula} (first green value), S_{rula}^{strict} (second green value) for some exemplary arm postures. Only the RULA score for the arm is computed and the configuration of the upper body does not influence the RULA score. An arm configuration with the elbow bent at 90° and the upper arm and wrist as shown in Figure 6.17 receives the best RULA value 1. In Figure 6.18 the robot grasps something from above. The RULA score 5 indicates an awkward arm configuration. In Figure 6.19 the robot arm is in a configuration that is impossible for a human arm. Therefore, the RULA score with strict penalty of 7 is assigned.

To verify the adaptation of the RULA scores for Justin, the mean RULA value is computed for the voxels of the workspace discretization introduced in Section 4.2.1. During the computation of the capability map at maximum 100 configurations are saved per voxel and evaluated using the adapted RULA implementation by MMI. For each configuration, the corresponding TCP pose is mapped to a voxel of the workspace discretization. With the computed RULA score, a new mean is computed for the voxel. As expected the regions with the best mean RULA values lie around the arm configuration where the elbow is bend at 90°. The mean RULA value across the robot arm's workspace is shown in Figure 6.20. From studying the

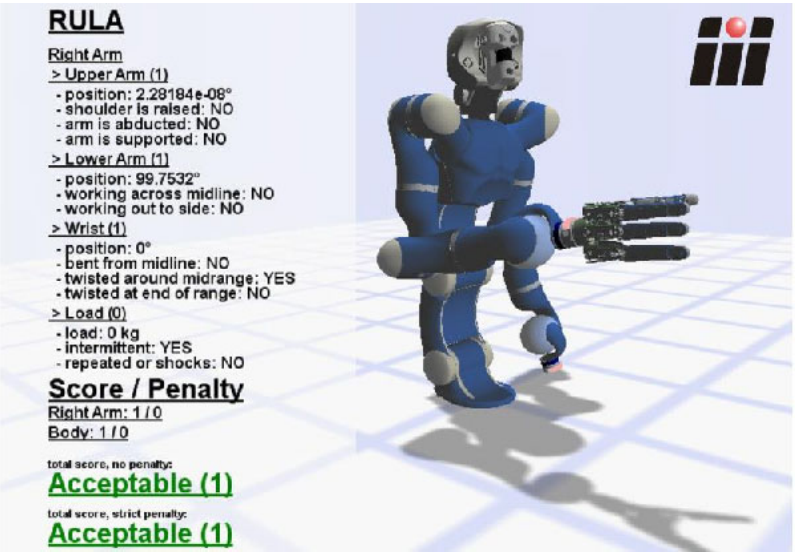


Fig. 6.17 MMI's anthropomorphic multi-agent model of Justin in an 'Acceptable' posture. An arm configuration with the elbow bent at 90° receives the best RULA value 1. The scores of the upper arm, the lower arm and the wrist are combined to obtain the final RULA value for the right arm of Justin.

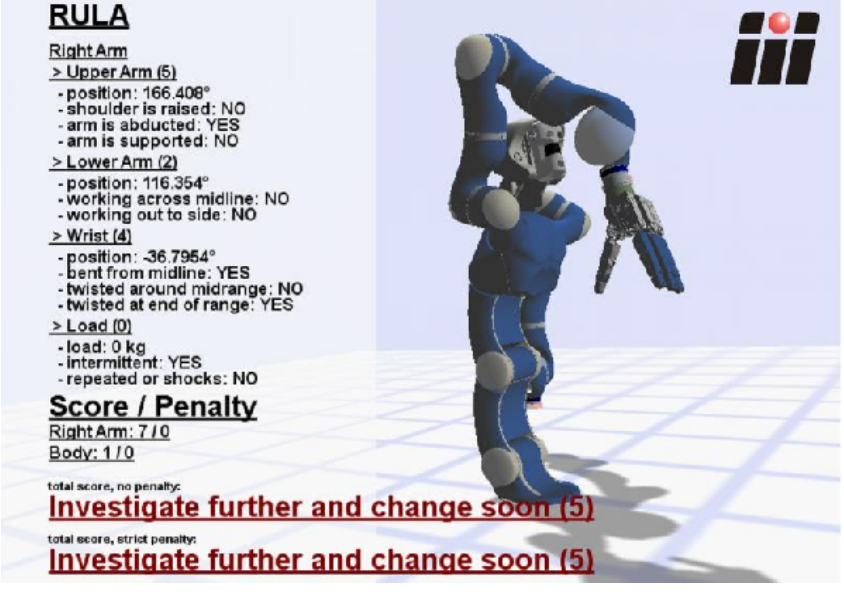


Fig. 6.18 MMI's anthropomorphic multi-agent model of Justin in an ergonomically strenuous posture. The RULA score 5 indicates an awkward arm configuration.

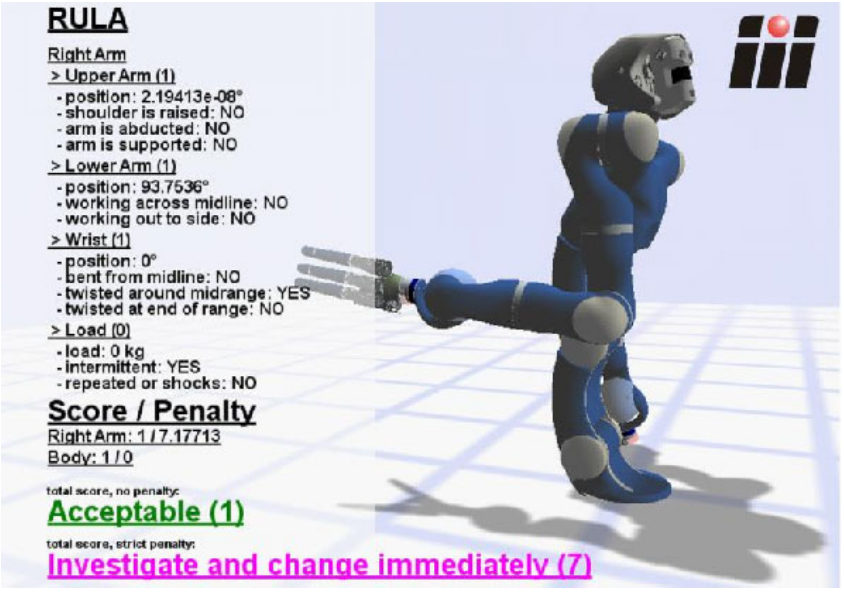


Fig. 6.19 MMI's anthropomorphic multi-agent model of Justin in a posture beyond the scope of human joints. Therefore, the RULA score with strict penalty of 7 is assigned. The normal RULA scoring system is not able to deal with this configuration because the human arm is not able to assume this configuration.

RULA scoring system, the mean RULA value 3 is expected for natural looking arm configurations.

6.2.2 Combining RULA and Capability Maps

Combining RULA and capability maps is straightforward. In the map building process, the reachability of a set of TCP frames is determined using an inverse kinematics solver. If the inverse kinematics is successful and returns a robot configuration as solution, its RULA score is computed. If the score is above a given threshold the configuration is ignored. After the map is built, the TCP frame is labeled unreachable if no configuration was found with a RULA score below the threshold. The resulting map is called RULA restricted capability map (*rulaCapMap*).

Figure 6.21 (right column) provides two views of the rulaCapMap. The map is build by restricting the data to frames reachable with configurations that have RULA values between 1 and 4. In Figure 6.21 the original capability map (left) is compared with the rulaCapMap (right). The manipulation capabilities predicted with the rulaCapMap are less versatile. In Figure 6.21 (c) and (d) the best reachable regions in the capability map and the rulaCapMap are compared. In the first case, the region is symmetric about the robot arm since the robot arm has no muscles or tendons inducing discomfort or strain. In the second case, the robot bends the elbow at 90° to

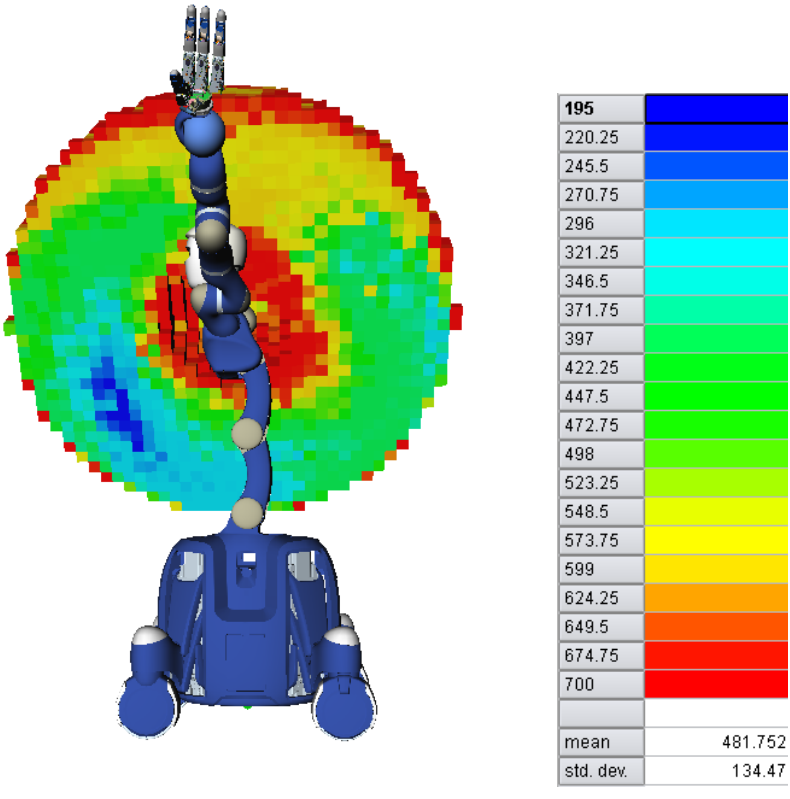


Fig. 6.20 For arm configurations with a TCP pose distributed across the whole workspace the RULA score is computed. The TCP pose is mapped to a voxel of the workspace discretization. Using the RULA score a new mean is computed for the voxel. The mean RULA value is shown for voxels across the robot arm's workspace. The color encodes the mean RULA value multiplied by 100.

reach the best region. This reflects the intuition with respect to the preferred region a human manipulates in. The region is a subset of the preferred region a human uses during every day tasks [41].

6.2.3 RULA and the Inverse Kinematics

The inverse kinematics solver [55] for the 7 DOF arm of Justin starts its search and optimization from a given initial configuration. Providing the inverse kinematics with a configuration with a good RULA value should result in its solution receiving similar RULA values. During the creation of the rulaCapMap a database of configurations with good RULA values is created. This database uses the same voxelization of the workspace as the capability map. For each of these voxels at maximum 10 configurations with RULA values in a given value range $[min, max]$ are stored.

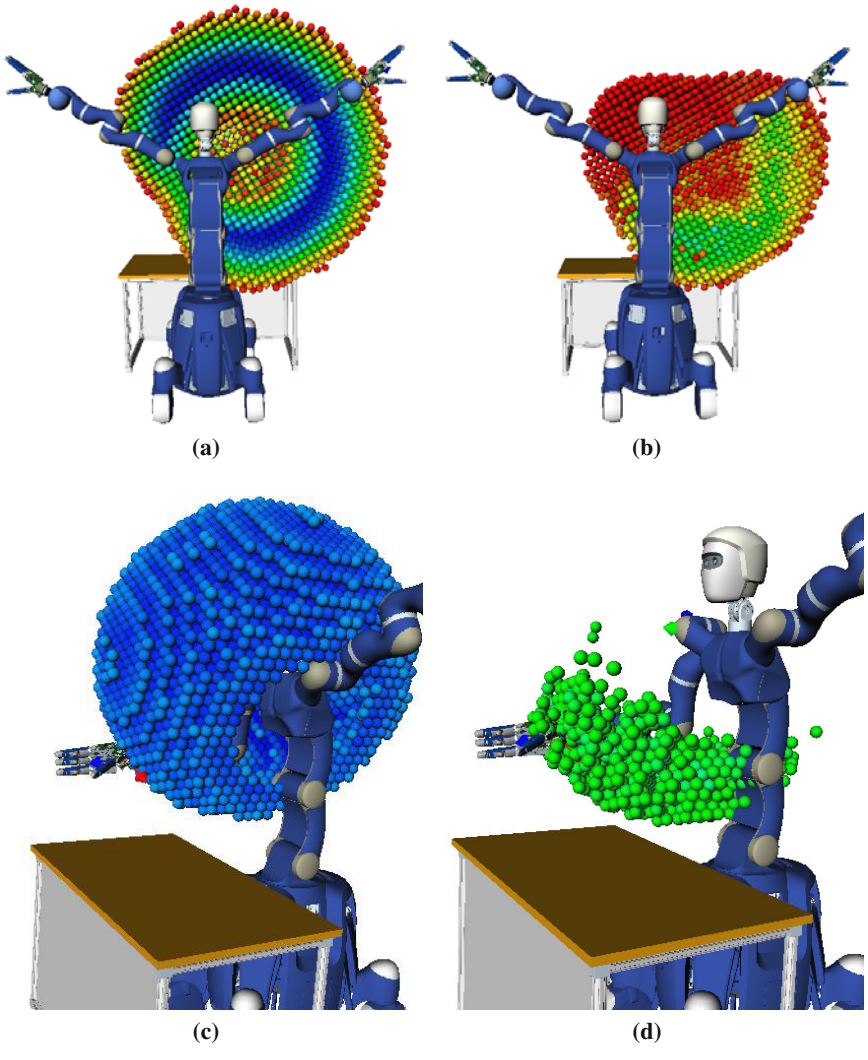


Fig. 6.21 For the robot's right arm, the original capability map **(left column)** is compared with the rulaCapMap **(right column)**. To build the rulaCapMap, configurations that do not have a RULA score between 1 and 4 are ignored. This restriction has the effect that the number of reachable voxels and their reachability index is reduced. (c) The best reachable region in the capability map is symmetric about the arm's longitudinal axes. (d) For the rulaCapMap the best reachable region lies in front of the body. It can be reached with the elbow bent at 90°.

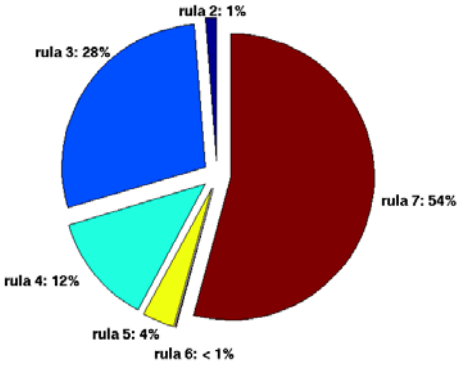
Given the pose of the TCP, the rulaCapMap is used to determine if it can be reached with a configuration with a RULA value $\in [min, max]$. If that is the case, a configuration is retrieved from the respective voxel of the database and provided as initial configuration to the inverse kinematics. The TCP corresponding to this initial configuration is close to the desired solution, therefore a solution is likely also natural looking. This approach is called *RulaIK* and is summarized in Algorithm 4.

Algorithm 4. RulaIK(T, sol, rMap, rConfDB)

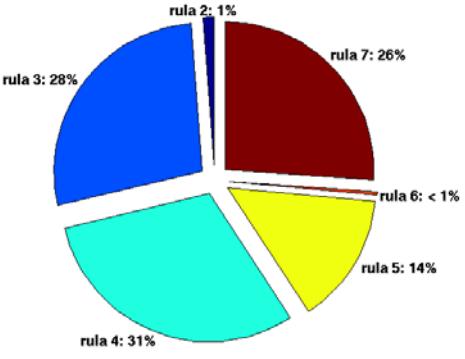
```
/** T - Tcp Pose
sol - ik solution
rMap - RULA restricted capability map
rConfDB - configurations with RULA score ∈ [min,max] */
/** if pose is reachable */
if rMap.IsReachable(T) then
   /** get voxel coordinates */
   index←rConfDB.GetVoxelIndex(T)
   /** get nr of configurations saved in voxel */
   nrOfConfs←rConfDB.GetNrOfConfs4Voxel(index)
   /** for a random configuration in voxel
   (each configuration only used once) */
   for nr ∈ range(nrOfConfs) do
      /** get configuration */
      initConf ← rConfDB.GetConf4Voxel(index, nr)
      /** solve IK with configuration as initial solution*/
      solFound ← ikSolver.Solve(initConf, T, sol)
      /** if solution found, report success */
      if solFound then
         return true
      end if
   end for
end if
return false
```

The RulaIK is compared with the same inverse kinematics algorithm provided with arbitrary initial configurations, called the *iteratedIK*. The link of the robot arm declared as redundant is randomly sampled. The resulting configuration is provided as initial solution to the inverse kinematics. Since the inverse kinematics involves local optimization and search, no solution may be found if the initial configuration is too far from a solution. Therefore, the process is repeated n times. In Algorithm 5 the *iteratedIK* is defined.

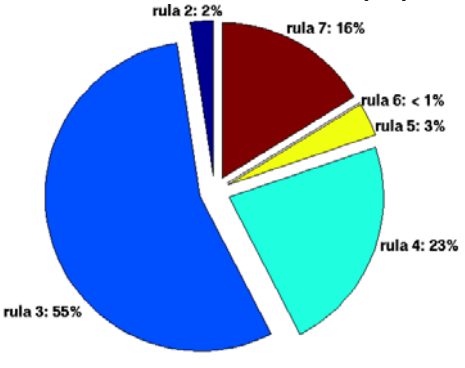
To compare the two approaches, 10^4 frames are randomly sampled and the rulaIK and the iteratedIK are used to compute inverse kinematics solutions. For the corresponding robot arm configurations, the RULA value is computed. For the whole set of configurations the occurance of each RULA value is determined. In Figure 6.22 it is shown that inverse kinematics solutions obtained with Algorithm 4 have significantly improved RULA values. In Figure 6.22 (a) the mean RULA values are



(a) iteratedIK



(b) rulaIK with RULA values $\in [1,4]$



(c) rulaIK with RULA values $\in [1,3]$

Fig. 6.22 The RULA scores for configurations computed with the iteratedIK and the rulaIK are compared. For each inverse kinematics the percentage of configurations with a RULA score i ($i = 1..7$) is shown. **(a)** The solutions of the iteratedIK are evaluated. About 59% have RULA scores ≥ 5. **(b)** The solutions of the rulaIK are evaluated that uses a rulaCapMap and configuration database with RULA values $\in [1,4]$. About 40% of the solutions have RULA scores ≥ 5. **(c)** The solutions of the rulaIK are evaluated that uses a rulaCapMap and configuration database with RULA values $\in [1,3]$. Only 20% have RULA scores ≥ 5.

Algorithm 5. IteratedIK(T, sol, nrOfRedundantLink)

```
/** T - Tcp Pose
sol - IK solution
nrOfRedundantLink - index of the redundant link */
/* vector of length robot DOF */
initSol←defaultInitSol
for i=1 to nrIterations do
   /** sample redundant DOF */
   initSol[nrOfRedundantLink]←getRandomLinkValue()
   /** solve IK with initial solution*/
   solFound ← ikSolver.Solve(T, initConf, sol)
   /** if solution found, report success */
   if solFound then
      return true
   end if
end for
return false
```

reported for the iterated inverse kinematics where RULA values are not considered. In Figure 6.22 (b) the used database contains only configurations with RULA values $\in [1,4]$ and in Figure 6.22 (c) it contains only configurations with RULA values $\in [1,3]$. As can be seen, not for all TCP poses the good RULA value for the configuration found during the map building process can be reproduced. Out of the configuration set saved for the respective voxel not necessarily the configuration is used that leads to the best possible RULA evaluation. A solution for this can be to compute a configuration with the inverse kinematics and then evaluate it using RULA . Several initial solutions for the inverse kinematics can be tried till a certain quality threshold is met. Furthermore, the RULA score can be included as an optimization criterion in the inverse kinematics algorithm. That however is a matter for future work. Here, it is shown that the presented concept already leads to substantial improvements.

6.2.4 RULA Sampling for Path Planners

Not only the start and goal configuration of a task should be human-like. Also the path should resemble a path a human would take. Normally, a RRT path planner uses a sampling routine that uniformly samples robot arm configurations from the configuration space. Without smoothing, the paths often are awkward and take roundabout routes. Even with smoothing and due to the fact that the whole range of valid link configurations is sampled, the found paths are often not human-like. Therefore, the sampling routine is adapted to use the database of configurations with RULA values in a given value range $[min,\ max]$. Instead of directly drawing a configuration from configuration space, a voxel of the database is drawn randomly. From the set of associated configurations, one is drawn randomly and represents the return value of the sampling routine. This sampling method is referred to as RULA sampling. It

represents a non-uniform sampling strategy in configuration space. The advantages and disadvantages of this method are examined in the next section.

6.2.5 *Evaluation*

The task of the robot is to place cubes in a predefined pattern on the table. Equally sized cubes are chosen to concentrate on the concepts to be explored instead of on grasping and object recognition issues. For each cube the position to which it is moved is predetermined by the given sequence of subtasks. A subtask where the robot arm moves to the next cube is called transit task. A subtask where the robot arm transports a cube to its destination is called transfer task [3]. A set of 40 precision grasps is generated for a cube placed on a table surface using the method described by Borst et al. [16]. This method computes grasps by only considering the isolated hand. In Figure 6.23, two grasps out of this set are shown. The reachability of the respective grasps is checked when the specific task is known, i.e. from where to move a cube to which location.

The rulaCapMap and the configuration database are used to enhance path planning speed and facilitate the choice of goal configurations for a robotic arm. In all experiments, the mobile humanoid robot Rollin' Justin is used. For path planning, the BiRRT planner [57] implementation in OpenRAVE [22], an open source planning architecture, is used. The RULA values are assessed with the help of MMI's anthropomorphic multi-agent model of Justin. For the planning experiments the rulaCapMap based on configurations with RULA values $\in [1,4]$ is used. While the map restricted to RULA values $\in [1,3]$ leads to more natural arm configurations, it is also very restrictive, i.e. fewer poses are considered reachable. To be able to use this map, a task planner is needed that determines the subtask sequence e.g which cube is put where or whether to reposition the upper body. For the experiments reported here, it is assumed that a sequence of subtasks is given and the torso configuration is fixed.

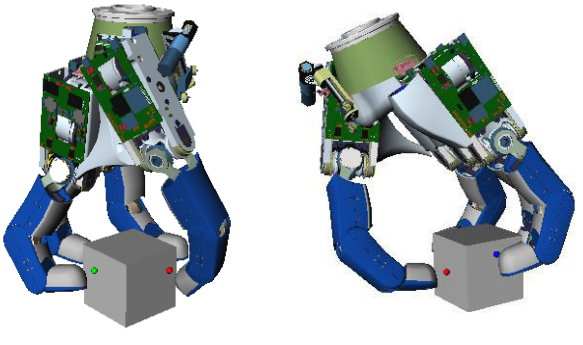


Fig. 6.23 An object can be grasped from different directions and with a variety of finger configurations. Two precision grasps are shown for the DLR 4-fingered hand and a cube. The colored spheres represent the point contacts of the fingers.

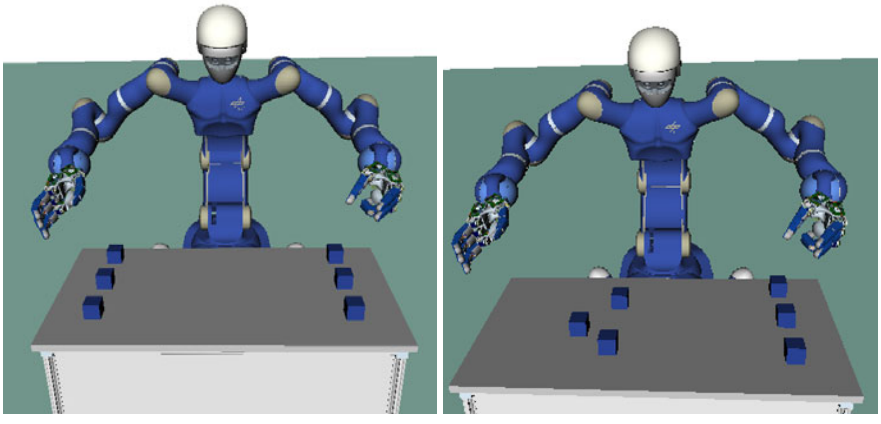


Fig. 6.24 Justin has to transport the cubes from their initial to their goal positions. **(left)** The initial state of the task is shown. **(right)** The right arm has transported the cubes to their goal positions.

A rough outline of the arrangement of planning steps is shown in Algorithm 6. In Figure 6.24 the initial and the final state of the task are shown. Only the right arm moves in this task. The initial position of the cubes and their target positions are assumed to be given. For each cube the reachable grasps are determined. Here, the rulaIK filters out grasps that are estimated to be unreachable with a human-like arm configuration. The grasp has to be reachable at the initial cube position and at its target position as no regrasp operation is allowed. However, the path planning task to reach a grasping position can fail. Therefore, potentially all grasps from the list of reachable grasps are tried to complete the task, i.e. move to, grasp and transport all cubes. The choice of the grasp can also be integrated in the path planner [23], however the focus in this section is on a proof of concept and not on obtaining the fastest system possible. Moreover, the aim is to let a task planner decide which grasp to try first. If only few grasps are available because of collisions this can be an indicator for a difficult path planning problem. A task planner can then decide to remove a disruptive obstacle to gain more freedom of action.

The results shown in Figure 6.25 and Figure 6.26 are averaged over 50 runs of the individual transit and transfer subtasks. The task is solved with four different setups of the planner. In the first setup the task is solved with a planner using the *iteratedIK and the standard sampling* (dark blue bars). Standard sampling here means uniform random sampling. The inverse kinematics is used to determine start and goal configurations to reach a selected grasp at the initial and the final position for a cube. In the second setup, the IKFast method [21] from OpenRAVE is used as inverse kinematics and standard sampling is used (cyan bars). To use IKFast one link of the arm is declared to be redundant. For the remaining 6 DOF, IKFast solves the inverse kinematics analytically. If necessary, samples for the redundant link are drawn from configuration space to find a solution and the inverse kinematics algorithm is

Algorithm 6. PlanTask(cubes2Move, goal4Cubes, graspSet)

```
/** cubes2Move - current positions of cubes
    goal4Cubes - goal positions for cubes
    graspSet - a set of precision grasps for a cube */
/** for each cube */
for cube ∈ cubes2Move do
   /** determine the reachable grasps */
   graspList← getReachableGrasps(graspSet,cube)
   /** get the number of reachable grasps*/
   nrOfGrasps← graspList.size()
   /** for each reachable grasp */
   for grasp ∈ graspList do
      /** plan path to grasp cube */
      solTransit=planTransitPath(grasp, cube)
      if not solTransit then
         /** if no path found, discard grasp */
         continue
      end if
      /** perform grasp */
      applyGrasp(grasp,cube)
      /** plan transport to goal position */
      solTransfer← planTransferPath(cube, goal4Cube)
      if solTransfer then
         /** if successful proceed to next cube */
         break
      end if
      if not solTransfer and isLastGrasp(grasp) then
         return false
      end if
   end for
   /** release cube */
   releaseObject(cube)
end for
return true
```

restarted. However, the current robot arm configuration is tried as initial solution first. If this is already a human-like configuration the IKFast solution is probably also human-like. Therefore, this method is conceptually in between the iteratedIK and the rulaIK. In the third setup, the *rulaIK and standard sampling* is used (yellow bars). In the fourth setup, the *rulaIK and RULA sampling* is used (red bars). In Figure 6.25, it can be seen that in all cases the proposed rulaIK with RULA sampling outperforms the iteratedIK with standard sampling. Only for cube4 transfer and cube5 transfer do *IKFast with standard sampling* and *rulaIK with RULA sampling* perform comparable. For all other cases *rulaIK with RULA sampling* is faster and provides better mean RULA values for the trajectories. Computation time here denotes the sum of path planning time and the path optimization time. Both are shown in the graph. The computation times were obtained on a computer with two Intel(R)

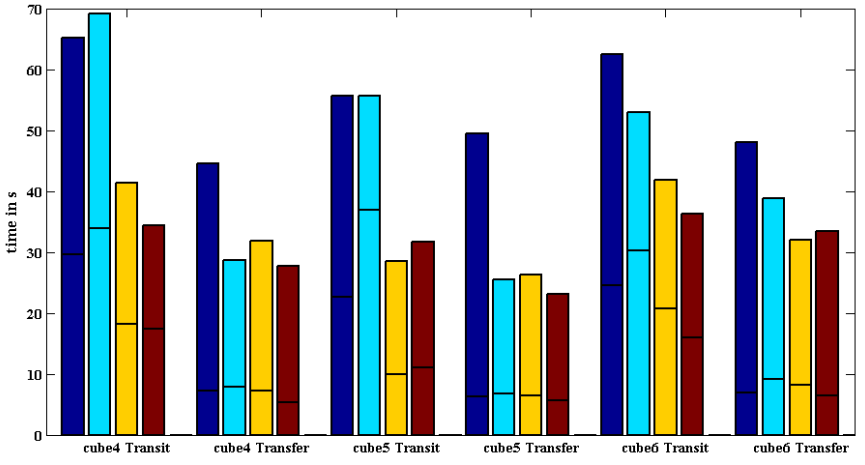


Fig. 6.25 The task is solved with four different setups of the planner. For each setup and each subtask the computation times are shown. The computation time is split in planning and optimization time. The colors symbolize the individual setups. (Blue) *Standard sampling and iteratedIK* are used. (Cyan) *Standard sampling and the IKFast* are used. (Yellow) *Standard sampling and the rulaIK* are used. (Red) *RULA sampling and the rulaIK* are used. In all cases the rulaIK with RULA sampling outperforms the iteratedIK with standard sampling.

Xeon(R) CPU W3520 2.67GHz processors and 6 GB main memory. In Figure 6.26 the mean RULA values are averaged over the length of the transfer and transit trajectories. Each configuration of the trajectory and its RULA value contribute to the mean RULA value of the trajectory. It can be seen that the best trajectory has a mean RULA value of 3. The plausibility of this result is verified by Figure 6.20. Here, it is shown that the mean RULA value is always above 2. The region with a mean RULA value up to 3 is quite small. Therefore at best a mean RULA value of 3 or 4 can be expected.

The trajectories computed with the method *rulaIK with RULA sampling* have the best mean RULA values. The results for the setup *rulaIK and standard sampling* (yellow bars) show that RULA sampling definitely needs to complement the rulaIK. Both, the planning and the smoothing step of the path planning algorithm are sped up. Results for the combination iteratedIK and RULA sampling are not shown here. If the inverse kinematics chooses an unnatural goal configuration, it can happen that no path is found because of the workspace restrictions introduced through the RULA sampling.

In general, guiding the planning with knowledge representations results in an improvement of the quality of manipulation motions and a speed up of the planning process. This effect is also observed by Hauser et al. [38] for planning the walking motions for a humanoid robot on uneven terrain.

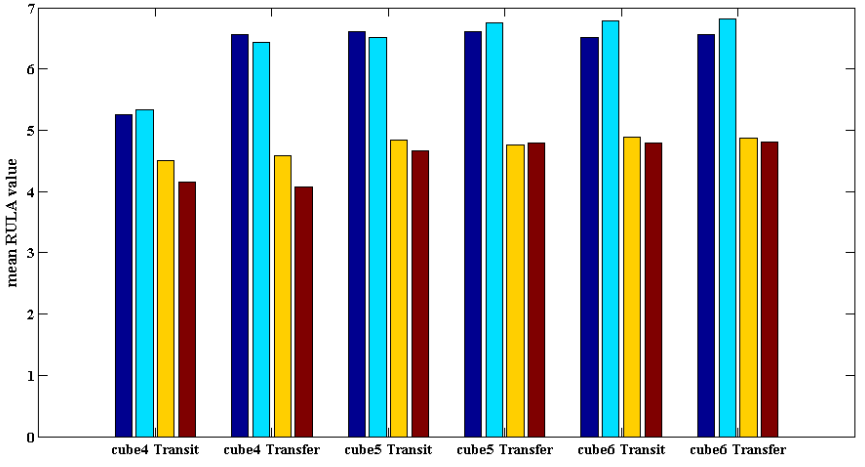


Fig. 6.26 The mean RULA value is averaged over the length of the trajectories and over all planning runs of a subtask. The colors symbolize the individual setups. (Blue) *Standard sampling and iteratedIK* are used. (Cyan) *Standard sampling and the IKFast* are used. (Yellow) *Standard sampling and the rulaIK* are used. (Red) *RULA sampling and the rulaIK* are used.

6.3 Summary

This chapter demonstrated the use of the capability map in planning applications. First, an algorithm was presented that used the capability map to position a mobile manipulator to execute 3D trajectories. Trajectories are localized in the capability map using correlation and then validated. Once a trajectory is found, the corresponding mobile manipulator position is computed. To illustrate the method the humanoid robot *Rollin' Justin* was positioned to open a kitchen closet. In general, the trajectory search method can be used to evaluate how well a robot is suited for specific environments or tasks. The determined number of solutions for the trajectory can be assumed to correlate with the ability of the robot to cope with disturbances e.g. objects left behind by a human, or a human standing in the way.

The method is especially suited to decide whether or not a task can be performed without using the mobile base. This information can be used by a task planner to decide which planner or execution component to trigger.

In a path planning task, the RULA criterion from ergonomic research was used to determine the naturalness of robot arm configurations. More natural start and goal configurations and the RULA sampling routine were provided to a state of the art RRT path planner. The path planner was then able to plan more human-like robot arm motion. The computation times and the mean RULA value along the trajectory were significantly improved. By restricting the planner to human-like start and goal configurations the humanoid robot's movement capabilities are restricted. It may be that no human-like start or goal configuration is found due to obstacles. In these cases, a task planner is needed to determine which obstacles to displace.

Chapter 7
Conclusion and Outlook

This chapter summarizes the achievements presented in this book, provides some concluding remarks as well as an outlook on potential future applications and future research directions.

7.1 Conclusion

In this book, the **versatile workspace** of a robot arm describes with which orientations of the end effector a position can be reached. A general representation of the versatile workspace, the **capability map**, was introduced. The capability map discretizes the workspace into voxels and determines for each voxel how it can be reached and how close it is to the dexterous workspace. For each voxel, a set of representative TCP frames is generated. The capability map represents the reachability of each of these TCP frames. Therefore, the capability map represents position and orientation information in contrast to current state of the art workspace representations. The capability map is attached to the base frame of a robot arm. Thus if this base frame moves because e.g. the robot torso is moved, the map is moved along.

An intuitive visualization scheme reveals the capabilities of the robot arm in its workspace and thus allows the graphical inspection of the capability map. The robot-specific map has to be built only once and allows fast random access. Through parameter analysis a recommended set of parameter values for the capability map construction was extracted. Using four different robot arms, it was verified that the workspace of arbitrary arm kinematics can be modeled and the reachability of TCP frames can be predicted accurately.

The capability map is specifically designed to support planning and scene analysis processes as shown by examples from several distinct application domains. The capability map was used to visualize and analyze robot workspaces. In this context, it was used to evaluate which serialized human arm kinematics better represents a human arm's capabilities. The results of this analysis can supplement the design process for humanoid robot design.

In setup evaluation, the capability map was used to evaluate an interface for human-robot interaction. The presented bi-manual human-robot interface allows the

F. Zacharias: Knowledge Representations for Planning Manipulation Tasks, COSMOS 16, pp. 125–127.
springerlink.com

operator to use both hands to directly control two robotic arms by attached handles. An algorithm was presented that uses the capability map to objectively evaluate different attachments of the robotic arms to a base and determine the quality of the interaction.

In planning tasks the capability map was used to parameterize path planners and select grasps. During a manipulation task for a humanoid robot, the capability map was used to bias the path planner and obtain more human-like start and goal configurations for the robot arm. The capability map focused the search and enabled the path planner to plan more human-like motion while simultaneously reducing the planning time.

In task reasoning the capability map was used to determine how well a robot is suited for a task. In an example a humanoid robot had to perform a task involving 3D trajectories. Regions that enable the execution of the task were extracted from the capability map and the suitability of the robot was inferred. This information can also be used by a task planner for the selection of the appropriate planner type. The proposed method can also be used during the robot design phase to determine how to attach the robot arms to an upper body. Or it can be employed when setting up demonstration scenarios that involve a robot that has to be able to perform a specific class of trajectories.

In summary, the capability map is a representation that can be machine processed and used in automatic planning. Its visualization is intuitively comprehensible for the human. The capability map and derived algorithms are a valuable source of information for task planners, path planners or robot base placement planners. The selection of good parameters for low-level planners is facilitated and good parameterizations for planners can be determined.

7.2 Future Work

This book introduced a novel representation of a robot arm's workspace. Its benefits were exemplarily shown in several applications. The capability map was used for scene analysis and as an information source for various planners. However, many other fields of application can be considered. Three are outlined in the following paragraphs.

Capability Maps with Arbitrarily Small Discretization. For the current implementation of the capability map, the minimum possible voxel edge length depends on the total length of the robot arm and the available main memory. The linear voxel space implementation of the capability map does not allow arbitrarily fine-grained workspace discretizations that can still be be kept in main memory. For minimally invasive surgery analysis involving entry point constraints, arbitrarily fine-grained capability maps or capability maps with different granularity levels could be desirable. Hence an implementation of the capability map could be beneficial that allows this and can be partially kept in main memory for fast access. For instance, hierarchical spatial data structures like Octrees are an alternative to the linear voxel space. They allow the representation of different granularity levels and keep only part of

the tree in main memory. The whole tree is located on the hard drive and the respective parts are loaded on demand.

Estimation of Task Difficulty. A task planner divides a task in elementary subtasks. For each of these elementary tasks it has to be determined whether they are solvable. The capability map could be used to estimate the difficulty of a planning problem or of a scene in general. The task planner could then use this information to estimate the probability of the success of a task. For instance, a task to approach and grasp a glass could have a low probability of success because of many obstacles standing close. The task planner could then trigger a rearrangement planner that displaces some obstacles and thus make the associated path planning problem easier to solve. This approach would avoid time consuming trail and error processes.

Compensation of Uncertainty. The trajectory search method provides a continuous region of solutions based on the capability map. To compensate uncertainty it is recommended to choose a solution in the center of the region of solutions. This will enable the compensation of variations. A robot with a mobile base that has as certain positioning error would still be able to open a closet door. However, in the future uncertainties could be explicitly modeled given the capability map and the task description. It could be estimated what type and amount of uncertainty occurs and what can be counterbalanced. This would enable a task planner to choose a manipulation strategy that is able to cope with these issues. If large uncertainties are present, the task execution could be observed closely and replanning could be triggered if necessary.

Graspability of Objects. To be able to grasp an object, a robot has to know where the reachable grasps are. The capability map data structure can be used to discretize the volume around an object. The resulting graspability map [94] represents for a particular object the positions and orientations that a given mechanical hand can adopt to achieve a force closure precision grasp. It is assumed that the hand is not attached to a robot arm. Thus the graspability map represents the capabilities of the isolated hand. Therefore the graspability maps can be used for comparing the grasp capabilities of different mechanical hands with respect to some benchmark objects.

The graspability maps also have potential applications in online grasp and manipulation planning. The capability map of the robot arm could be intersected with the graspability map of an object placed at a specific position in the workspace. The resulting representation yields the reachable grasp frames for the given object. This information can be used to plan manipulation tasks or to perform scene reasoning.

Appendix A
Abbreviations and Glossary

Abbreviations are defined and the mathematical symbols and notations used in this book are specified. Furthermore, the random number generator used in this book is referenced.

A.1 Abbreviations

arccos	arccosine
BiRRT	Bidirectional rapidly growing random tree
C-space	Configuration space
DH	Denavit-Hartenberg
DLR	German aerospace center
DOF	Degree of freedom
FFT	Fast fourier transformation
IK	Inverse kinematics
HRI	Human-Robot interface
LWR	Light weight robot
MMI	Institute of Man-Machine interaction
PCA	Principal component analysis
PRM	Probabilistic road map
RRT	Rapidly growing random tree
rulaCapMap	Rula-restricted capability map
RULA	Rapid upper limb assessment
SFE	Shape fit error
TCP	Tool center point
OV	workspace overlap

A.2 Mathematical Symbols

C	configuration space
$K(\mathbf{q})$	direct kinematics
H	set of all homogeneous matrices
W_R	reachable workspace
W_D	dexterous workspace
W_V	versatile workspace
$F(R,\mathbf{x})$	function that maps to a homogeneous matrix
V_{Robot}	voxel space for the robot arm
V_{Human}	voxel space for the human arm
P	set of points on the sphere
N_p	set of point indices for the points on the sphere
N_o	set of orientation indices
O_S	set of all homogeneous frames distributed on a sphere
M_S	capability map

A.3 Mathematical Notations

a	scalar value
$\mathbf{a}$	vector
$\mathbf{a}^T$	vector transposed
A	matrix
A^T	matrix transposed
$<\mathbf{a},\mathbf{b}>$	inner product
$SO(3)$	group of rotation matrices $\in \mathbb{R}^3$
	$SO(3) := \left\{R \in \mathbb{R}^{3\times 3} \mid RR^T = I, \det R = +1\right\}$
$SE(3)$	$\mathbb{R}^3 \times SO(3)$
T_B^A	reference frame B given in coordinates of reference frame A
$\lceil a \rceil$	ceiling function
$\lfloor a \rfloor$	floor function

A.4 Random Sampling

In this book, the drawing of random samples is often used. Unless otherwise stated, the samples are always drawn from an uniform distribution. The Mersenne Twister random number generator [73] is used to generate uniformly distributed random numbers.

Appendix B
Kinematics Descriptions

This section describes homogeneous matrices and gives the specifications of arm kinematics used in this book in Denavit-Hartenberg (DH) parameters as proposed by Craig [19].

B.1 Homogeneous Matrices

Once a base coordinate system A is chosen, any point in the 3D Cartesian space can be denoted by a $3x1$ vector.

$$\mathbf{t}_A = (t_x, t_y, t_z)^T \tag{B.1}$$

The orientation of a body in Cartesian space is described by a $3x3$ rotation matrix $R \in SO(3)$. One way to describe the body-attached coordinate system B, is to write the unit vectors of its principal axes in terms of the coordinate system A. Let $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, be the unit vectors that give the principal directions of coordinate system B in terms of coordinate system A. If they are inserted as the columns of a $3x3$ matrix, a rotation matrix results.

$$R = (\mathbf{x}\ \mathbf{y}\ \mathbf{z}) \tag{B.2}$$

The information needed to completely specify the pose of a manipulator hand is a position and an orientation. The point on the manipulator hand whose position is described is chosen as the origin of the body-attached frame. A frame is a set of four vectors defining the position and orientation of an attached body in space. A homogeneous matrix is a $4x4$ matrix

$$T = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{B.3}$$

used to describe a frame. A frame is a coordinate system where in addition to the orientation, a position vector is given which locates its origin relative to some other embedding frame [19]. Using frames the rigid body motion of objects in Cartesian space can be described.

B.2 DLR LWR Kinematics

In Table B.1 the kinematics is described using DH-parameters. The lower limits are shown in column *ll* and the upper limits in column *ul*. In Figure B.1 the individual positions and directions of the rotation axes are shown. The rotation axis of a link is always the z-axis shown as a blue arrow.

Table B.1 DH parameters for the DLR LWR robot arm. Link lengths are in *m*. Angles are given in degrees. The robot base frame is identical with the frame of the first link.

i	α_{i-1}	a_{i-1}	d_i	θ_i	ll	ul
0	0	0	0	0	-170	170
1	0	90	0	0	-120	120
2	0	-90	0.4	-90	-170	170
3	0	90	0	0	-120	120
4	0	-90	0.39	180	-170	170
5	0	90	0	90	-45	80
6	0	-90	0	90	-45	135

B.3 Kuka LWR Kinematics

In Table B.2 the kinematics is described using DH-parameters. The lower limits are shown in column *ll* and the upper limits in column *ul*. In Figure B.2 the individual positions and directions of the rotation axes are shown. The rotation axis of a link is always the z-axis shown as a blue arrow.

Table B.2 DH parameters for the Kuka LWR robot arm. Link lengths are in *m*. Angles are given in degrees.

i	α_{i-1}	a_{i-1}	d_i	θ_i	ll	ul
0	0	0	0	0	-170	170
1	0	90	0	0	-120	120
2	0	-90	0.4	0	-170	170
3	0	-90	0	0	-120	120
4	0	90	0.39	0	-170	170
5	0	90	0	0	-130	130
6	0	-90	0	0	-170	170

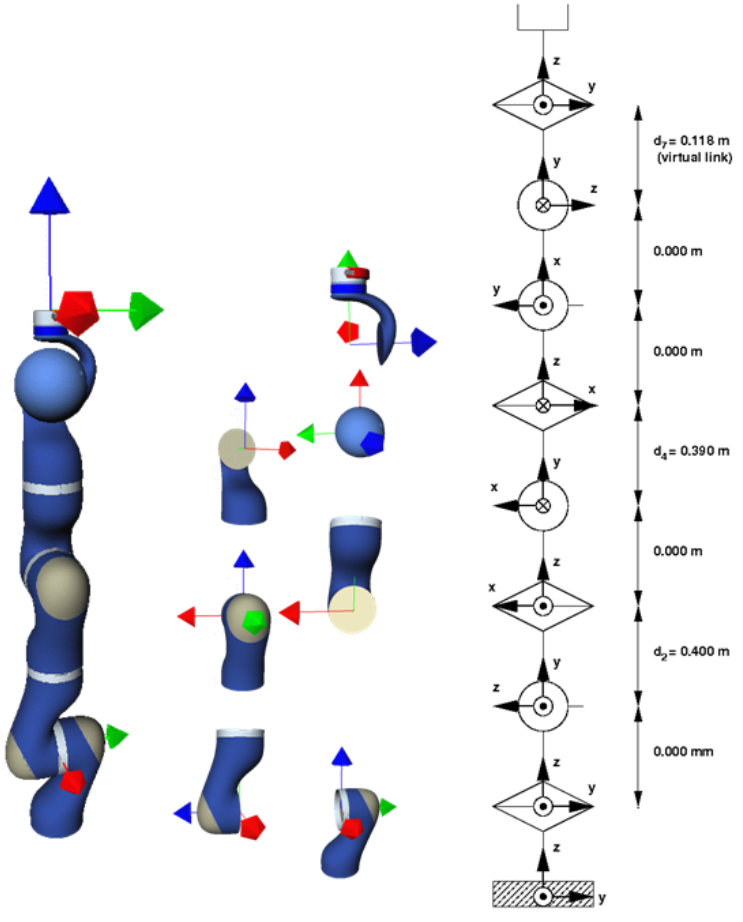


Fig. B.1 Visualization of the kinematics of the DLR LWR robot arm that serves as the right arm for the humanoid robot Justin. The positions and directions of the link frames are shown. Each link rotates about its z-axis (blue arrow). A circle with a dot signifies that the respective axis is normal to the picture plane and is pointing at the observer. A circle with a cross signifies that the respective axis is normal to the picture plane and is pointing away from observer.

B.4 Schunk PowerCube Arm Kinematics

In Table B.3 the DH parameter and link limits are listed for the 6 DOF Schunk PowerCube arm. The kinematics is e.g. used in Section 4.4.3. The TCP used for the generation of the reachability sphere map is listed in the last row.

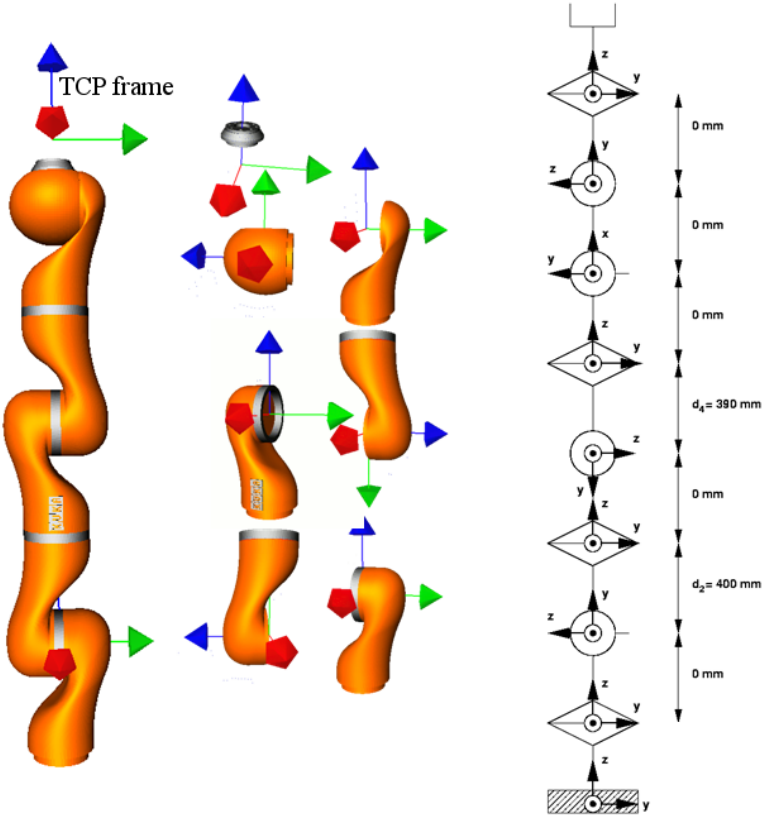


Fig. B.2 Visualization of the kinematics of the Kuka LWR robot arm. The positions and directions of the link frames are shown. Each link rotates about its z-axis (blue arrow).

Table B.3 DH parameters and link limits for the 6 DOF PowerCube arm. The DH parameters for the transformation to the robot arm TCP are shown in the last row of the table. Link lengths are in *m*. Angles are given in degrees.

i	a_{i-1}	α_{i-1}	d	θ_i	ll	ul
1	0	0	0	0	-270	270
2	0	-90	0.26	-90	-270	270
3	0	-90	0	180	-135	135
4	0	-90	0.31	180	-270	270
5	0	-90	0	180	-120	120
6	0	-90	0	180	-270	270
TCP	0	0	0.265	0		

B.5 Kuka Kr16 Kinematics

In Table B.4, the DH-parameters are given for the industrial robot arm Kuka Kr16. It is significantly larger than the other robot arms used in this book. Fully extended it has a length of about 2 *m*.

Table B.4 DH parameters and link limits for the 6 DOF Kuka Kr16 industrial robot arm. The DH parameters for the transformation from the last link to the robot arm TCP are shown in the last row of the table. Link lengths are in *m*. Angles are given in degrees.

i	a_{i-1}	α_{i-1}	d	θ_i	ll	ul
1	0	180	-0.675	0	-185	185
2	0.26	90	0	0	-155	35
3	0.68	0	0	-90	-130	154
4	-0.035	90	-0.67	0	-350	350
5	0	-90	0	0	-130	130
6	0	-90	0.158	90	-350	350
TCP	0	0	0	0		

B.6 Human Kinematics

In Table B.5 the DH parameters are listed for the *kinematics 2* that is examined in Section 5.2.2.2 as a candidate for the kinematics of the human right arm.

Table B.5 DH parameters and link limits for the human arm kinematics 2. Link lengths are in *m*. Angles are given in degrees.

i	α_{i-1}	a_{i-1}	d_i	θ_i	ll	ul
0	0	0	90	0	-50	180
1	0	-90	0	-90	-30	180
2	0	90	0.27	-90	-110	80
3	0	-90	0	0	0	145
4	0	90	0.22	0	-85	90
5	0	-90	0	90	-45	15
6	0	90	0	90	-85	85

B.7 TCP Frames

B.7.1 TCP Frames Used in Section 4.3

The TCP used in Section 4.3 for the DLR LWR arm which serves as the right arm for the robot Justin is given in DH-parameters in Equation B.4. The TCP frame is placed at the end of the robot arm.

$$(\alpha_{i-1}, a_{i-1}, d_i, \theta_i) = (0, -90, 0.118\,m, -180) \tag{B.4}$$

The TCP used in Section 4.3 for the Kuka LWR arm is given in DH-parameters in Equation B.5.

$$(\alpha_{i-1}, a_{i-1}, d_i, \theta_i) = (0, 0, 0.118\,m, 0) \tag{B.5}$$

The TCPs have the same pose with respect to the robot arm base to have arms with comparable length and comparable kinematics. Only if the TCPs are chosen like this can the capability maps directly be compared because in this case the robots try to reach the same poses. Otherwise, kinematics can be compared only with respect to given tasks.

B.7.2 TCP Frames Used in Section 5.2.2.3

The TCP for the human arm kinematics used in Section 5.2.2.3 is given in DH-parameters in Equation B.6.

$$(\alpha_{i-1}, a_{i-1}, d_i, \theta_i) = (0, -90, 0.065\,m, 180) \tag{B.6}$$

The TCP for the right robot arm in *scenario 1* is given in Equation B.7. For the right robot arm in *scenario 2* the TCP frame is given in Equation B.8.

$$(\alpha_{i-1}, a_{i-1}, d_i, \theta_i) = (0, 180, -0.23\,m, 0) \tag{B.7}$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0.23\,m \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{B.8}$$

References

1. Abdel-Malek, K., Yang, J., Brand, R., Tanbour, E.: Towards understanding the workspace of human limbs. Journal of Ergonomics 47, 1386–1405 (2004)
2. Abdel-Malek, K., Yu, W.: Placement of robot manipulators to maximize dexterity. J. Robotics and Automation 19(1), 6–15 (2004)
3. Alami, R., Laumond, J.P., Simèon, T.: Two manipulation planning algorithms. Algorithmic Foundations of Robotics, 109–125 (1995)
4. Amato, N., Bayazit, O., Dale, L., Jones, C., Vallejo, D.: Obprm: An obstacle based prm for 3d workspaces. In: Proc. of Workshop on the Algorithmic Foundations of Robotics, WAFR (1998)
5. Asfour, T., Regenstein, K., Azad, P., Schröder, J., Bierbaum, A., Vahrenkamp, N., Dillmann, R.: Armar-iii: An integrated humanoid plattfrom for sensory-motor control. In: Proc. IEEE Intl. Conf. on Humanoid Robots (2006)
6. Baginski, B.: Motion Planning for Manipulators with Many Degrees of Freedom - The BB-Method. PhD thesis, Technische Universität Muenchen (August 1998)
7. Beetz, M., Bandouch, J., Kirsch, A., Maldonado, A., Müller, A., Rusu, R.B.: The assistive kitchen - a demonstration scenario for cognitive technical systems. In: Proc. COE Workshop on Human Adaptive Mechatronics, HAM (2007)
8. Beetz, M., Stulp, F., Esden-Tempski, P., Fedrizzi, A., Klank, U., Kresse, I., Maldonado, A., Ruiz-Ugalde, F.: Generality and legibility in mobile manipulation. Autonomous Robots Journal (Special Issue on Mobile Manipulation) (2010)
9. Berenson, D., Kuffner, J., Choset, H.: An optimization approach to planning for mobile manipulation. In: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 1187–1192 (2008)
10. Berenson, D., Srinivasa, S., Ferguson, D., Kuffner, J.: Manipulation planning on constraint manifolds. In: IEEE Intl. Conf. on Robotics and Automation, ICRA (2009)
11. Berenson, D., Srinivasa, S., Ferguson, D., Romea, A.C., Kuffner, J.: Manipulation planning with workspace goal regions. In: IEEE Intl. Conf. on Robotics and Automation, ICRA (2009)
12. Bertoli, P., Botea, A., Fratini, S.: Report of the board of judges. In: Third Intl. Competition on Knowledge Engineering for Planning and Scheduling (2009)
13. Blum, A., Furst, M.: Fast planning through planning graph analysis. Artificial Intelligence 90, 281–300 (1997)

14. Bodenmüller, T.: Streaming Surface Reconstruction from Real Time 3D Measurements. PhD thesis, Lehrstuhl für Reakzeit-Computersysteme, Technische Universität München (2009), http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20091019-795498-1-5
15. Borst, C., Fischer, M., Hirzinger, G.: Grasping the Dice by Dicing the Grasp. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 3692–3697 (2003)
16. Borst, C., Fischer, M., Hirzinger, G.: Efficient and precise grasp planning for real world objects. Springer Tracts in Advanced Robotics 18, 91–111 (2005)
17. Bronstein, I.N., Semendjajew, K.A., Musiol, G., Mühlig, H.: Taschenbuch der Mathematik. In: Harri Deutsch, 6th edn. (2005)
18. Calino, S., Guenter, F., Billard, A.: On learning, representing and generalizing a task in a humanoid robot. Transactions on Systems, Man, and Cybernetics 37(2), 286–298 (2007)
19. Craig, J.: Introduction to Robotics: Mechanics and Control. Addison-Wesley (1989)
20. Cutkosky, M.R.: On grasp choice, grasp models, and the design of hands for manufacturing tasks. IEEE Trans. Robotics and Automation 5, 269–279 (1989)
21. Diankov, R.: Automated Construction of Robotics Manipulation Programs. PhD thesis, Robotics Institute. Carnegie Mellon University (2010)
22. Diankov, R., Kuffner, J.: OpenRAVE: A Planning Architecture for Autonomous Robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute (July 2008)
23. Diankov, R., Ratliff, N., Ferguson, D., Srinivasa, S., Kuffner, J.: Bispace planning: Concurrent multi-space exploration. In: Proc. Intl. Conf. on Robotics: Science and Systems (2008)
24. Dornhege, C., Eyerich, P., Keller, T., Trüg, S., Brenner, M., Nebel, B.: Semantic attachments for domain-independent planning systems. In: Intl. Conf. on Automated Planning and Scheduling, ICAPS (2009)
25. Dornhege, C., Gissler, M., Teschner, M., Nebel, B.: Integrating symbolic and geometric planning for mobile manipulation. In: Intl. Symposium on Robotic Research, ISRR (2009)
26. Drumwright, E., Jenkins, O.C., Matarić, M.: Exemplar-based primitives for humanoid movement classification and control. In: IEEE Intl. Conf. on Robotics and Automation, ICRA (2004)
27. Fikes, R.E., Nilsson, N.: Strips: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2, 189–208 (1971)
28. Freund, E., Rossmann, J., Schlette, C.: Controlling anthropomorphic kinematics as multi-agent systems. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems, IROS (2003)
29. Fuchs, M., Borst, C., Giordano, P.R., Baumann, A., Krämer, E., Langwald, J., Gruber, R., Seitz, N., Plank, G., Kunze, K., Burger, R., Schmidt, F., Wimboeck, T., Hirzinger, G.: Rollin′ Justin - Design Considerations and Realization of a Mobile Platform for a Humanoid Upper Body. In: Proc. IEEE Intl. Conf. on Robotics and Automation, ICRA (2009)
30. Ghallab, M., Nau, D., Traverso, P.: Automated Planning - theory and practice. Morgan Kaufmann (2004)
31. Gienger, M., Toussaint, M., Jetchev, N., Bendig, A., Goerick, C.: Optimization of fluent approach and grasp motions. In: Proc. IEEE Intl. Conf. on Humanoid Robots, pp. 111–117 (2008)

32. Google 3D Warehouse. 3D model of a kitchen (December 2010), http://sketchup.google.com/3dwarehouse/details?mid=25e8368f5b81a1312ed4225c1c283c73&prevstart=0
33. Gravot, F., Cambon, S., Alami, R.: Asymov: A planner that deals with intricate symbolic and geometric problems. In: Proc. Intl. Symposium on Robotics Research, Springer Tracts in Advanced Robotics, pp. 100–110 (2003)
34. Grebenstein, M., Albu-Schäffer, A., Bahls, T., Chalon, M., Eiberger, O., Friedl, W., Gruber, R., Haddadin, S., Hagn, U., Haslinger, R., Höppner, H., Jörg, S., Nickl, M., Nothhelfer, A., Petit, F., Reill, J., Seitz, N., Wimböck, T., Wolf, S., Wüsthoff, T., Hirzinger, G.: The DLR Hand Arm System. In: Proc. IEEE Intl. Conf. on Robotics and Automation, ICRA (2011)
35. Guan, Y., Yokoi, K.: Reachable space generation of a humanoid robot using the monte carlo method. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 1984–1989 (2006)
36. Gupta, K.C.: On the nature of robot workspace. J. Robotics Research 5(2), 112–121 (1986)
37. Harada, K., Kaneko, K., Kanehiro, F.: Fast grasp planning for hand/arm systems based on convex model. In: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 1162–1168 (2008)
38. Hauser, K., Bretl, T., Latombe, J.-C.: Using motion primitives in probabilistic sample-based planning for humanoid robots. In: Workshop on the Algorithmic Foundations of Robotics, WAFR (2006)
39. Hauser, K., Latombe, J.-C.: Integrating task and prm motion planning: Dealing with many infeasible motion planning queries. In: ICAPS Workshop on Bridging the Gap Between Task and Motion Planning (2009)
40. Hoffmann, J.: FF: The fast-forward planning system. AI Magazin 22 (2001)
41. Howard, I., Ingram, J., Körding, K., Wolpert, D.: Statistics of natural movements are reflected in motor errors. J. Neurophysiol. 102(3), 1902–1910 (2009)
42. Hulin, T., Sagardia, M., Artigas, J., Schaetzle, S., Kremer, P., Preusche, C.: Human-scale bimanual haptic interface. In: Proc. 5th Intl. Conf. on Enactive Interfaces (2008)
43. Jain, D., Moesenlechner, L., Beetz, M.: Equipping robot control programs with first-order probabilistic reasoning capabilities. In: Proc. IEEE Intl. Conf. on Robotics and Automation, ICRA (2009)
44. Jolliffe, I.T.: Principal Component Analysis. Springer, Heidelberg (2002)
45. Kaelbling, L.P., Lozano-Perez, T.: Hierarchical planning in the now. In: IEEE Intl. Conf. on Robotics and Automation, Workshop on Mobile Manipulation (2010)
46. Kaneko, K., Kanehiro, F., Kajita, S., Hirukawa, H., Kawasaki, T., Hirata, M., Akachi, K., Isozumi, T.: Humanoid robot hrp-2. In: Proc. IEEE Intl. Conf. on Robotics and Automation, ICRA (2004)
47. Kapandji, I.: The Physiology of the Joints - Upper Limb, Churchill Livingstone, vol. 1 (1982)
48. Kärcher: Rc 3000 saugbot (December 2010), http://saugrobot.de/kaercher-rc3000.php
49. Kavraki, L., Svestka, P., Latombe, J.-C., Overmars, M.: Probabilistic roadmaps for path planning in high dimensional configuration spaces. IEEE Trans. on Robotics and Automation 12, 566–580 (1996)
50. Kawato, M.: Internal models for motor control and trajectory planning. Curr. Opin. Neurobiol. 9(6), 718–727 (1999)
51. Klein, C.A., Blaho, B.E.: Dexterity measures for the design and control of kinematically redundant manipulators. J. Robotics Research 6(2), 72–83 (1987)

52. Klopčar, N., Lenarčič, J.: Kinematic model for determination of human arm reachable workspace. Meccanica 40, 203–219 (2005)
53. Koga, Y., Kondo, K., Kuffner, J., Latombe, J.: Planning motions with intentions. In: Proc. of SIGGRAPH, pp. 395–408 (1994)
54. Konietschke, R.: Planning of Workplaces with Multiple Kinematically Redundant Robots. PhD thesis, Technische Universität München, Munich, Germany (2007), http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20070618-622197-1-5
55. Konietschke, R., Hirzinger, G.: Inverse kinematics with closed form solutions for highly redundant robotic systems. In: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 2017–2022 (2009)
56. Konietschke, R., Ortmaier, T., Weiß, H., Engelke, R., Hirzinger, G.: Optimal Design of a Medical Robot for Minimally Invasive Surgery. In: 2. Jahrestagung der Deutschen Gesellschaft für Computer- und Roboterassistierte Chirurgie (CURAC), Nürnberg, Germany, pp. 4–7 (November 2003)
57. Kuffner, J., LaValle, S.: RRT-connect: An efficient approach to single-query path planning. In: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 995–1001 (2000)
58. Kuka. youbot (December 2010), http://www.kuka-youbot.com
59. Kuka robotics. Homepage (2010), http://www.kuka-robotics.com
60. Landzettel, K., Brunner, B., Lampariello, R., Preusche, C., Reintsema, D., Hirzinger, G.: System prerequisites and operational modes for on-orbit-servicing. In: Int. Symposium on Space Technology and Science, ISTS (2004)
61. Latombe, J.-C.: Robot Motion Planning. Kluwer Academic Publishers (1991)
62. LaValle, S.M.: Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University (October 1998)
63. Leidner, D.: Planung und Ausführung von alltäglichen zweihändigen Manipulationsaufgaben. Master's thesis (2010)
64. Lenarčič, J., Stanič, U., Oblak, P.: Some kinematic considerations for the design of robot manipulators. Robotics and Computer-Integrated Manufacturing 5(2/3), 235–241 (1989)
65. Lifschitz, V.: On the semantics of strips. In: Workshop on Reasoning about Actions and Plans (1986)
66. Lopez-Damian, E., Sidobre, D., Alami, R.: A grasp planner based on inertial properties. In: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 754–759 (2005)
67. Lozano-Pérez, T., Jones, J.L., Mazer, E., O'Donnell, P.A.: Task-level planning of pick-and-place robot motions. IEEE Magazine Computer 22(3), 21–29 (1989)
68. Lück, C.L.: Robot cartography: A topology-driven discretization for redundant manipulators. In: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 1446–1451 (1996)
69. Lück, C.L., Lee, S.: Redundant manipulator self-motion topology under joint limits with an 8 dof case study. In: Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), Yokohama, Japan (1996)
70. Ma, O., Angeles, J.: Optimum architecture design of platform manipulators. In: Proc. Intl. Conf. on Advanced Robotics (ICAR), pp. 1130–1135 (1991)
71. Mason, M.T.: Mechanics of Robotic Manipulation. MIT Press (2001)
72. Matarić, M.: Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics. In: Imitation in Animals and Artifacts, pp. 391–422. MIT Press (2002)

73. Matsumoto, M., Nishimura, T.: Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation 8(1), 3–30 (1998)
74. McAtamney, L., Corlett, E.: RULA: a survey method for the investigation of work-related upper limb disorders. Applied Ergonomics 24, 91–99 (1993)
75. McCarthy, J., Hayes, P.J.: Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligence 4, 463–502 (1969)
76. McDermott, D.: The Formal Semantics of Processes in PDDL. In: Proc. ICAPS Workshop on PDDL (2003)
77. Miller, A.T., Allen, P.K.: Graspit!: A versatile simulator for robotic grasping. IEEE Robotics and Automation Magazine 11(4), 110–122 (2004)
78. Miller, A.T., Knoop, S., Christensen, H.I., Allen, P.K.: Automatic grasp planning using shape primitives. In: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 1824–1829 (2003)
79. Mishra, B.: Grasp metrics: Optimality and complexity. Technical Report 1995-680, Robotics and Manufacturing Laboratory, Courant Institute. New York University (1995)
80. Müller, A.: Transformational Plans for Autonomous Household Robots Using Libraries of Robust and Flexible Plans. PhD thesis, IAS Group, Technische Universität München (2008)
81. Murray, R.M., Li, Z., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press (1994)
82. Neo, E., Sakaguchi, T., Yokoi, K., Kawai, Y., Maruyama, K.: A behavior level operation system for humanoid robots. In: Proc. IEEE-RAS Intl. Conf. on Humanoid Robots, pp. 327–332 (2006)
83. Nof, S.Y. (ed.): Handbook of Industrial Robotics. John Wiley & Sons (1999)
84. Okada, K., Haneda, A., Nakai, H., Inaba, M., Inoue, H.: Environment manipulation planer for humanoid robots using task graph that generates action sequence. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 1174–1179 (2004)
85. Oriolo, G., Mongillo, C.: Motion planning for mobile manipulators along given end-effector paths. In: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), Barcelona, Spain, pp. 2154–2160 (April 2005)
86. Park, F., Brockett, R.: Kinematic dexterity of robotic mechanisms. J. Robotics Research 13(1), 1–15 (1994)
87. Park, J.-Y., Chang, P.-H., Yang, J.-Y.: Task-oriented design of robot kinematics using the grid method. Advanced Robotics 17(9), 879–907 (2003)
88. Pettré, J., Laumond, J.-P., Siméon, T.: A 2-stage locomotion planner for digital actors. In: Proc. Eurographics Symposium on Computer Animation, SCA 2003 (2003)
89. Pin, F.G., Culioli, J.-C.: Optimal positioning of combined mobile platform-manipulator systems for material handling tasks. J. Intelligent and Robotic Systems 6(2-3), 165–182 (1992)
90. Press, W.H., Teukolsky, S., Vetterling, W.T., Flannery, B.P.: Numerical recipes. Cambridge University Press (2007)
91. Reif, J.: Complexity of the mover's problem and generalizations. In: Proc. 20th Symp. Foundations of Computer Science (1979)
92. Reif, J.: Complexity of the Generalized Mover's Problem. In: Planning, Geometry and Complexity of Robot Motion, pp. 267–281. Ablex Pub. (1987)
93. Reiser, U., Connette, C., Fischer, J., Kubacki, J., Bubeck, A., Weisshardt, F., Jacobs, T., Parlitz, C., Hägele, M., Verl, A.: Care-O-bot 3 - Creating a product vision for service robot applications by integrating design and technology. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems, IROS (2009)

94. Roa, M.A., Hertkorn, K., Zacharias, F., Borst, C., Hirzinger, G.: Graspability map: A tool for evaluating grasp capabilities. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems, IROS (2011)
95. Rodriquez, I., Peinado, M., Boulic, R., Meziat, D.: Bringing the human arm reachable space to a virtual environment for its analysis. In: Proc. Intl. Conf. on Multimedia and Expo, pp. 229–232 (2003)
96. Saff, E., Kuijlaars, A.: Distributing many points on the sphere. Mathematical Intelligencer 19(1), 5–11 (1997)
97. Schaal, S., Peters, J., Nakanishi, J., Ijspeert, A.: Learning movement primitives. In: Intl. Symposium on Robotics Ressearch, ISRR (2004)
98. Schunk. Manipulators (December 2010), http://www.schunk-modular-robotics.com
99. Shannon, C.E.: Communication in the presence of noise. Proc. Institute of Radio Engineers 37(1), 10–21 (1949)
100. Siciliano, B., Khatib, O. (eds.): Springer Handbook of Robotics. Springer, Heidelberg (2008)
101. Stilman, M.: Task constrained motion planning in robot joint space. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 3074–3081 (2007)
102. Stilman, M., Schamburek, J.-U., Kuffner, J., Asfour, T.: Manipulation planning among movable obstacles. In: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 3327–3331 (2007)
103. Stocco, L., Salcudean, S.E., Sassani, F.: Fast constrained global minimax optimization of robot parameters. Robotica 16(6), 595–605 (1998)
104. Stulp, F., Federizzi, A., Zacharias, F., Tenorth, M., Bandouch, J., Beetz, M.: Combining analysis, imitation, and experience-based learning to acquire a concept of reachability in robot mobile manipulation. In: Proc. IEEE Intl. Conf. on Humanoid Robots, Humanoids (2009)
105. Stulp, F., Fedrizzi, A., Beetz, M.: Action-related place-based mobile manipulation. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems, IROS (2009)
106. Stulp, F., Oztop, E., Pastor, P., Beetz, M., Schaal, S.: Compact models of motor primitive variations for predictable reaching and obstacle avoidance. In: 9th IEEE-RAS International Conference on Humanoid Robots (2009)
107. Sturges, R.: A quantification of machine dexterity applied to an assembly task. J. Robotics Research 9(3), 49–62 (1990)
108. Terasaki, H., Hasegawa, T., Takahachi, H., Arakawa, T.: Automatic grasping and regrasping by space characterization for pick-and-place operations. In: Workshop on Intelligence for Mechanical Systems at IEEE Intl. Conf. Intelligent Robots and Systems, IROS (1991)
109. Tondu, B., Ippolito, S., Guiochet, J.: A seven-degrees-of-freedom robot-arm driven by pneumatic artificial muscles for humanoid robots. J. Robotics Research 24(4), 257–274 (2005)
110. Watanabe, T., Yoshikawa, T.: Optimization of Grasping by Using a Reqired External Force Set. In: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), Taipei, Taiwan, pp. 1127–1132 (September 2003)
111. Woesch, T., Neubauer, W.: Grasp and place tasks for domestic robot assistants. In: Intl. Workshop on Advances in Service Robots, Stuttgart, Germany (May 2004)
112. Yang, Y., Brock, O.: Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation. In: Proc. Robotics: Science and Systems, RSS (2006)
113. Yao, Z., Gupta, K.: Path planning with general end-effector constraints: using task space to guide configuration space search. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 1875–1880 (2005)

114. Yoshida, E., Kanoun, O., Esteves, C., Laumond, J.-P.: Task-driven support polygon reshaping for humanoids. In: Proc. IEEE Intl. Conf. on Humanoid Robots, pp. 208–213 (2006)
115. Yoshikawa, T.: Foundations of Robotics: Analysis and Control. MIT Press, Cambridge (1990)
116. Zacharias, F., Borst, C., Hirzinger, G.: Bridging the gap between task planning and path planning. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 4490–4495 (2006)
117. Zacharias, F., Borst, C., Hirzinger, G.: Capturing robot workspace structure: Representing robot capabilities. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems, IROS (2007)
118. Zacharias, F., Howard, I., Hulin, T., Hirzinger, G.: Workspace comparison of setup configurations for human-robot interactions. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems, IROS (2010)
119. Zacharias, F., Schlette, C., Schmidt, F., Borst, C., Rossmann, J., Hirzinger, G.: Making planned paths look more human-like in humanoid robot manipulation planning. In: IEEE Intl. Conf. on on Robotics and Automation, ICRA (2011)
120. Zacharias, F., Sepp, W., Borst, C., Hirzinger, G.: Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories. In: Proc. IEEE Intl. Conf. on Humanoid Robots (2009)
121. Ziparo, V.A., Iocchi, L., Leonetti, M., Nardi, D.: A probabilistic action duration model for plan selection and monitoring. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems, IROS (2010)
122. Zöllner, R., Asfour, T., Dillmann, R.: Programming by demonstration: Dual-arm manipulation tasks for humanoid robots. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems, IROS (2004)

Cognitive Systems Monographs

Edited by R. Dillmann, Y. Nakamura, S. Schaal and D. Vernon

Vol. 1: Arena, P.; Patanè, L. (Eds.)
Spatial Temporal Patterns for
Action-Oriented Perception
in Roving Robots
425 p. 2009 [978-3-540-88463-7]

Vol. 2: Ivancevic, T.T.; Jovanovic, B.;
Djukic, S.; Djukic, M.; Markovic, S.
Complex Sports Biodynamics
326 p. 2009 [978-3-540-89970-9]

Vol. 3: Magnani, L.
Abductive Cognition
534 p. 2009 [978-3-642-03630-9]

Vol. 4: Azad, P.
Visual Perception for Manipulation
and Imitation in Humanoid Robots
270 p. 2009 [978-3-642-04228-7]

Vol. 5: de Aguiar, E.
Animation and Performance Capture
Using Digitized Models
168 p. 2010 [978-3-642-10315-5]

Vol. 6: Ritter, H.; Sagerer, G.;
Dillmann, R.; Buss, M.:
Human Centered Robot Systems
216 p. 2009 [978-3-642-10402-2]

Vol. 7: Levi, P.; Kernbach, S. (Eds.):
Symbiotic Multi-Robot Organisms
467 p. 2010 [978-3-642-11691-9]

Vol. 8: Christensen, H.I.;
Kruijff, G.-J.M.; Wyatt, J.L. (Eds.):
Cognitive Systems
491 p. 2010 [978-3-642-11693-3]

Vol. 9: Hamann, H.:
Space-Time Continuous Models of Swarm
Robotic Systems
147 p. 2010 [978-3-642-13376-3]

Vol. 10: Allerkamp, D.:
Tactile Perception of Textiles in a
Virtual-Reality System
120 p. 2010 [978-3-642-13973-4]

Vol. 11: Vernon, D.; von Hofsten, C.; Fadiga, L.:
A Roadmap for Cognitive Development
in Humanoid Robots
227 p. 2010 [978-3-642-16903-8]

Vol. 12: Ivancevic, T.T.; Jovanovic, B.;
Jovanovic, S.; Djukic, M.; Djukic, N.;
Lukman, A.:
Paradigm Shift for Future Tennis
375 p. 2011 [978-3-642-17094-2]

Vol. 13: Bardone, E.:
Seeking Chances
168 p. 2011 [978-3-642-19632-4]

Vol. 14: Jakimovski, B.:
Biologically Inspired Approaches for
Locomotion, Anomaly Detection and
Reconfiguration for Walking Robots
201 p. 2011 [978-3-642-22504-8]

Vol. 15: Der, R.; Martius, G.:
The Playful Machine
336 p. 2012 [978-3-642-20252-0]

Vol. 16: Zacharias, F.:
Knowledge Representations for Planning
Manipulation Tasks
143 p. 2012 [978-3-642-25181-8]