TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Echtzeitsysteme und Robotik

Analysis of Serial Chain Manipulator Structures with Respect to Efficient Actuation in Context of Abstractly Represented Tasks

Susanne M. E. Petsch

Vollständiger Abdruck der von der Fakultät der Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Daniel Cremers
Prüfer der Dissertation: 1. Univ.-Prof. Dr. Darius Burschka
2. Ao. Univ.-Prof. Dr. Markus Vincze, Technische Universität Wien/ Österreich

Die Dissertation wurde am 24. Oktober 2013 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 13. Mai 2014 angenommen.

Abstract

Robots get deployed in more and more application areas. The range of possible tasks is large, reaching from kitchen scenarios to medical applications. However, the suitability of a robot for a task has been hardly analyzed in general. If, e.g., several robots are available, which one can perform a desired task best? What happens if one or several of the robot's joints fail? Can the robot compensate lost capabilities due to joint failure? Several criteria can be considered to answer these questions. The criteria include, e.g., perceptual capabilities, mobility or manipulation capabilities. This thesis focuses on the analysis of manipulator *structures*. Such an analysis provides essential information about the robot's manipulation capabilities. An inter- and intra-robot analysis is provided. Possible joint failures and the potential of path optimization are considered.

The presented analysis of manipulator structures focuses on efficient actuation. Slow and smooth joint motions are desirable as well as an equal distribution of the workload on all joints. The context of the analysis is formed by abstractly represented tasks. Hence, the robot's capabilities are analyzed for a general task description. Therefore, an abstract task representation needs to be provided as well. Such an abstract representation allows to choose the final path for a task execution in a less constrained manner. The robot is not limited to a pre-defined path. The resultant freedom can be utilized to optimize paths with respect to efficient control. This, in turn, enables an analysis of the potential of path optimization in the case of joint failures. The representation of the tasks is object-centric, since the objects are in the focus of the task. The analysis itself has a robot-centric perspective, since the capabilities of the robot are analyzed. In order to show and discuss approaches based on different perspectives in general, a survey of research in robotic manipulation with respect to bio-mimicry, bio-inspiration and technical approaches is provided.

Experiments are performed on the overall analysis and its underlying single components. The results show the task-specific suitability of the analyzed robots. The capabilities of the robots are depicted with respect to the desired tasks. The important joints of the robots are identified. Moreover, the potential of path optimization is presented.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Dr.-Ing. Burschka who gave me the opportunity to do my Ph.D. in his group. I want to thank him for his guidance, support and advice. The instructive scientific discussions that I had with him provided useful ideas and impulses to my work.

My work was funded by the GRASP project and the German Aerospace Center (DLR). They made the development of this thesis possible. Many thanks go to everybody who enabled this funding.

I want to thank everyone, who helped me with words and deeds during the conceptualization, development and completion of my thesis.

I am especially grateful to my colleagues and “cooks” in Hochbrück, namely, Chavo, Elmar, Juan-Carlos and Olli (sorted alphabetically), who let me profit from their experiences. I want to thank them for the constructive scientific discussions and great lunch times with them. Many thanks go to all the nice and open-minded people who I met at conferences the last years. The conversations and discussions with them were adjuvant and encouraging. I wish to thank them for the great times.

My special thanks goes to my father for his extensive personal support that helped me especially in tough times of my Ph.D. pursuit.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robots which are supposed to support a human worker need sophisticated manipulation capabilities and abilities. This requires not only appropriate physical capabilities from the hardware, but also advanced knowledge about the manipulation. The requirements on the robot can, therefore, be very demanding. The question arises, whether a certain robot is able to perform a desired task which could, e.g., require specific manipulation capabilities. If several robots are available, it might be interesting to know which robot is the most appropriate one for the task. Such an analysis of the *task-specific suitability* of robots has hardly been presented yet.
Several criteria can be relevant for the choice of a robot. These criteria include, e.g., cognition, physical capabilities, special requirements set by the task, interaction and psychological aspects. The robot might need to be able to observe and to interact with the environment. This includes, e.g., the detection and the manipulation of objects. Moreover, tasks can require specific properties from the system. For example, hygienic aspects play an important role in the field of medical robotics. Hence, it has to be possible to disinfect the involved hardware properly. If humans are involved in the task, a safe interaction has to be enabled between the human and the robot. Psychological aspects in the interaction can be important to allow a comfortable interaction for the human. This, in turn, should be considered, since it can be a future criteria to deploy a certain robotic system.

1.1 Motivation

This thesis focuses on the *analysis* of *serial chain manipulator structures*. Such a serial chain manipulator can be a part of a larger robotic system (e.g., a humanoid robot, a manipulator attached to a mobile base). The serial chain manipulator is of special interest, since it plays a very important role in the performance of manipulations. The required *manipulation capabilities* to perform such a task are analyzed here. They are the basis to allow manipulations at all. We do neither aim to analyze single arbitrary points in the robot's workspace nor do we analyze the entire workspace of the robot. We focus on the areas relevant for the desired

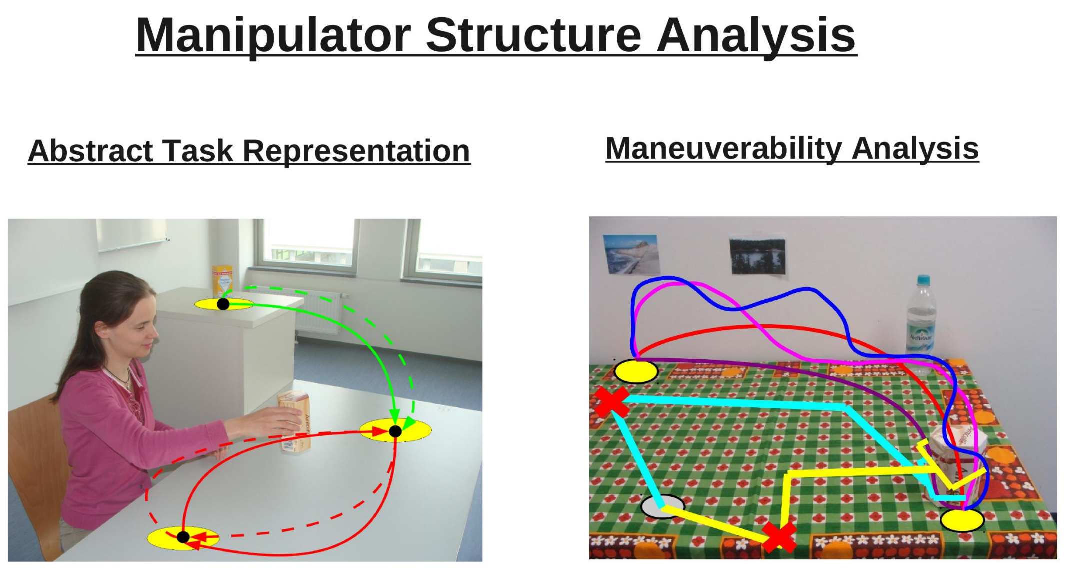


Figure 1.1: Maneuverability analysis for abstractly represented tasks. The Figure illustrates both main parts of the maneuverability analysis for abstractly represented tasks: The abstract task representation and the maneuverability analysis. *Left:* The system builds up an abstract representation of a task demonstrated by the human [2]. Rather than storing the trajectories of the observed demonstrations, the system extracts the characteristic properties of the manipulation. *Right:* Two exemplary manipulators (yellow and cyan manipulators on the gray base) perform a task. Their performance can be affected through joint failures and the choice of the path. The red crosses on the manipulators symbolize joint failures. The curves between the yellow start, resp., end position illustrate different paths for a manipulation. The desired maneuverability analysis has to be applicable to any arbitrary serial chain manipulator. It has to consider possible joint failures and the potential of path optimization.

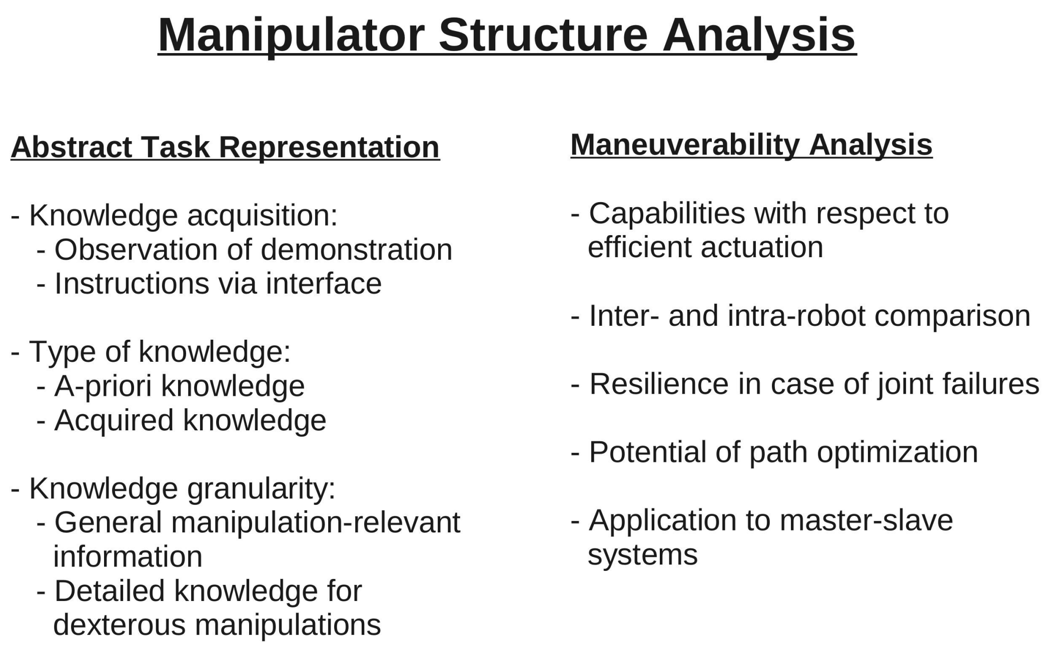
task. Just these areas are analyzed with respect to the required capabilities. The aim is not only an analysis whether a desired area is reachable for the robot. This is just a necessary pre-condition for a further analysis. The capabilities of the robot at these areas are of interest here. A further important aspect is the resilience in the case of joint failures. If one or several of the joints fail, the remaining capabilities still need to be analyzable. Of course, the desired task plays also an important role, since we are interested in a *task-specific* analysis. Hence, an appropriate representation of tasks is required as well.
Fig. 1.1 illustrates both main parts of this thesis: The *task representation* and the *maneuverability analysis*. An overview of the relevant aspects of both main parts is given in Fig. 1.2. The required components are shown in Fig. 1.3.

The required task knowledge has to cover a wide range of possible tasks, reaching from transportation tasks to very dexterous manipulations. The system should, e.g., know how an object can be picked up and transported. If a specific manipulation has to be performed in a task, much more detailed knowledge about the manipulation is additionally necessary. A daily-life task like knot-tying seems to be easy and, hence, not extraordinary to many humans. In fact, it is a complex task, which requires detailed information how the threads have to be handled. Moreover, many tasks cannot only be performed on one fixed path. Variations of the path can also be used to perform a desired task. This can already be observed during humans' demonstrations. A human hardly uses the same path for his/her actions, if it is not really necessary to keep a certain path. For example, if a human carries a cup from the kitchen into the office several times, the path will, most likely, change every time. A human is probably not even able to use exactly the same path a second time. On the one hand, this means, that the robot can use a set of paths to perform a task. On the other hand, the robot needs to deal with a variety of possible actions during human observation. It has to store the information in such an *abstract* manner, that the characteristic properties of the actions reflect the actual idea of the manipulation. A simple record of observed trajectories is neither sufficient nor appropriate for this purpose.

The storage of knowledge at an abstract level allows to describe the actual idea of the manipulation. The system needs, then, to be able to make use of the entire scope of this knowledge representation. As described, a set of possible paths fulfilling the properties of the desired manipulation is available to the robot. Obstacle avoidance can be considered as well as path optimization. In the context of manipulation analysis, the potential of path optimization is of special interest. Path optimization can be utilized to compensate joint failures. Hence, an appropriate method is required. Furthermore, the abstract task representation enables the adaption of the path to the current situation in the environment, since the path is not fixed to certain x,y,z-coordinates in space.

The analysis itself focuses on the capabilities of the manipulators with respect to *efficient actuation*. Smooth and slow motions are, hence, desirable for every joint of the manipulator. Moreover, the workload should be distributed equally among the joints. A side effect is the reduced stain on the hardware. In the context of path optimization, we need, consequently, an optimization with respect to efficient actuation.

It is important to point out, that the analysis has to be independent of the manipulator's structure. It should be applicable to any robot without further modifications. Hence, the representation of the knowledge needs to be entirely independent of the robot. It should be possible to use the representation on any robot. A transfer of the knowledge to another robot is easily possible, then. The analysis can, hence, be performed on different robots which have the same knowledge about the task. The focus of the knowledge is on the objects which are manipulated. Therefore, an object-centric representation is developed. In the case of the path optimization, we need a robot-centric perspective. The path has, of course, to be optimized with respect to the robot to achieve efficient actuation. However, the optimization should be applicable to any manipulator without demanding modifications of the basic optimization.
The just described requirements enable an application on any manipulator, independently of its number of Degrees-of-Freedom (DoF). Hence, an analysis of a manipulator with one or multiple



Manipulator Structure Analysis

Abstract Task Representation

- Knowledge acquisition:
  - Observation of demonstration
  - Instructions via interface
- Type of knowledge:
  - A-priori knowledge
  - Acquired knowledge
- Knowledge granularity:
  - General manipulation-relevant information
  - Detailed knowledge for dexterous manipulations

Maneuverability Analysis

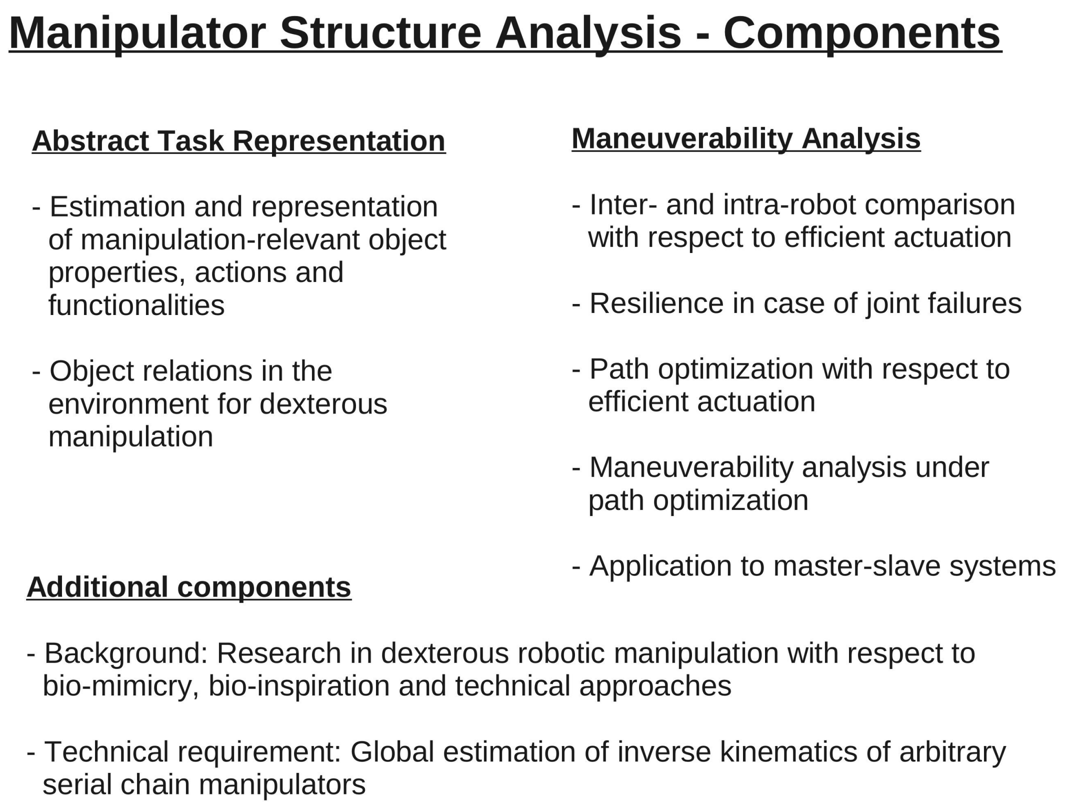
- Capabilities with respect to efficient actuation
- Inter- and intra-robot comparison
- Resilience in case of joint failures
- Potential of path optimization
- Application to master-slave systems

Figure 1.2: Maneuverability analysis for abstractly represented tasks - Overview. The Figure gives an overview of the relevant aspects in both main parts illustrated in Fig. 1.1. In the context of the abstract task representation, we discuss the knowledge acquisition, the type of knowledge and the granularity of the knowledge. As motivated in this Section, the Maneuverability Analysis is built in a manner, such that the robot's capabilities are analyzed with respect to efficient actuation. It is an inter- and intra-robot comparison, which considers possible joint failures. The potential of path optimization is analyzed, moreover. A further application area (master-slave systems) is indicated.

broken joint(s) is possible as well. This allows an inter- and intra-robot comparison of capabilities. The proposed analysis in combination with the provided abstract task representation cannot only be used to determine, whether a robot is able to reach a certain *point*. It is also possible to analyze whether and at which effort a desired *task* can be performed. Furthermore, the consequences of possible joint failures can be examined to answer, e.g., the following questions: Is the robot still able to perform the required tasks? Which capabilities of the robot get limited and which ones are lost?

We focus on the analysis of the manipulator *structure*. The *motion* capabilities of the structure are analyzed. Consequently, we do not consider the mass, mass-distribution or other physical properties of the robot, since such topics are out of the scope of this work.

As described, we analyze serial chain manipulators, since they are usually used for manipulation tasks. The tasks are performed non-remotely by the robot itself. One could also consider a master-slave system. There, the slave-system is operated remotely from a master-system, which is, in turn, controlled by a human user. Therefore, the application of a human-centric



Manipulator Structure Analysis - Components

Abstract Task Representation

- Estimation and representation of manipulation-relevant object properties, actions and functionalities

- Object relations in the environment for dexterous manipulation

Maneuverability Analysis

- Inter- and intra-robot comparison with respect to efficient actuation

- Resilience in case of joint failures

- Path optimization with respect to efficient actuation

- Maneuverability analysis under path optimization

- Application to master-slave systems

Additional components

- Background: Research in dexterous robotic manipulation with respect to bio-mimicry, bio-inspiration and technical approaches

- Technical requirement: Global estimation of inverse kinematics of arbitrary serial chain manipulators

Figure 1.3: Maneuverability analysis for abstractly represented tasks - Components. The Figure shows the components for both main parts and the additionally required components. The “estimation and representation of manipulation-relevant object properties, actions and functionalities” includes a-priori knowledge as well as knowledge extracted from human observation. It contains general manipulation-relevant information. The “object relations in the environment for dexterous manipulations” allow to represent detailed knowledge for dexterous manipulations. It can be acquired through human observation or through instructions from an interface (e.g., graphical). The maneuverability analysis is built in such a manner, that an “inter- and intra-robot comparison with respect to efficient actuation” is possible. The “resilience in case of joint failures” is considered in the analysis as well. The “path optimization with respect to efficient actuation” is developed separately before it is applied to the maneuverability analysis. A further application area are master-slave systems, which require a change of perspective from the object-/ robot-centric one to a human-centered one. Moreover, such different perspectives are discussed in the context of robotic manipulation. Additionally, a exhaustive global estimation of inverse kinematics of arbitrary serial chain manipulators has to be developed to enable an appropriate maneuverability analysis.

perspective is expedient in this area, as we will explain in detail later on. Experiments on the analysis of master-slave systems can indicate a further application area of the proposed methods. We provide basic experiments on the analysis of such *remote-controlled, non-autonomous* master-slave systems.

The relevant aspects of both main parts of this thesis (task representation and maneuverability analysis) are shown in Fig. 1.2. The required components are shown in Fig. 1.3. Besides the already described components, it depicts additionally required components.
The first one is an overview of the current state of the art in robotic grasping and dexterous manipulation. The approaches in this field can be categorized with respect to bio-mimetic, bio-inspired and technical approaches. Of course, there are arguments for the one or the other perspective. The overview forms an appropriate base for the discussion of the advances of the different perspectives.
The second additional component depicted in Fig. 1.3 is a technical requirement: An appropriate estimation of the inverse kinematics of arbitrary serial chain manipulators. We motivated already, that the analysis has to be independent of the manipulator's structure. Hence, an estimation of the inverse kinematics has to be provided, which can be applied to any manipulator independently of its structure.

1.2 Related Work

In this Section, the related work is presented. It is subdivided into the associated fields of inverse kinematics of arbitrary serial chain manipulators, representations of manipulation-relevant knowledge and dexterous manipulations, path optimization, manipulator structure analyses and analyses of master-slave systems.

1.2.1 Estimation of Inverse Kinematics of Arbitrary Serial Chain Manipulators and Human-Like Robotic Hands

In order to perform a structure analysis on any arbitrary robot, we need an appropriate estimation of inverse kinematics. In contrast to existing work, we aim to estimate the inverse kinematics of an *arbitrary* serial chain manipulator (position and orientation). The solution has to be general, so that it can be applied to any serial chain manipulator even under the presence of redundancy. Moreover, the estimation has to be able to work without any pre-knowledge about (possibly) advantageous of known (start-)configurations. We also provide an estimation for a human-like robotic hand to complete the experiments: The object needs to be grasped (at pre-defined points), before it can be manipulated (see also Chapter 2).

The field of inverse kinematics has been studied for a long time. Many applications and problems depend on inverse kinematics, e.g., [8]. A general introduction to the problem of inverse kinematics can be found in [9]. Solutions for special manipulators have already been presented there as, for example, Pieper's solution [10]. Moreover, solutions for special redundant manipulators have been introduced as, for example, in [11]. Others made use of the null-space, which accompanies a redundant manipulator, e.g., [12]. Combinations of different methods have been developed (analytic, numerical methods; including optimization), e.g., in [13] or [14]. The Jacobian (including its inverse, resp., pseudo inverse) was also often used as, e.g., in [15]. In general, optimization methods have already been applied on the problem of inverse kinematics (e.g., [16]). The determination of the inverse kinematics of parallel chain manipulators has been presented as well (e.g., [17]). Inverse kinematics and path planning were integrated in [18] to move a manipulator arm from an initial configuration.

The concept of the virtual finger was originally used in the context of mapping a human grasp to a grasp for a robotic hand [19]. We adapt the original idea to our approach for the estimation of inverse kinematics. We just use a thumb and one virtual finger. In our case, the virtual finger is a combination of the real fingers without the thumb.

As already motivated, we aim to estimate the inverse kinematics of an arbitrary serial chain manipulator and an arbitrary human-like hand (resp., position and orientation). The solution has to be general, so that it can be applied to any serial chain manipulator, resp., robotic hand even under the presence of redundancy. At first, we want to estimate whether a solution exists. If one or more solutions exit, we want to determine them. Otherwise, we are interested in a solution close to the goal. We do not assume any pre-knowledge about (possibly) advantageous of known (start-)configurations. Nevertheless, local minima may not cause any problem. In contrast to local approaches (e.g., null-space), it has to be possible to find several optima, even if they are significantly separated. Although local optimizer can possibly find a solution of inverse kinematics, we need to apply a global optimization method to achieve the previously described aims. Additionally, the computation time should stay within a reasonable boundary.

It is important to distinguish our approach from other fields like path planning. We do not aim to, e.g., find a path between two points or analyze the entire workspace of the robot. We are interested in the *discrete* analysis of inverse kinematics for one or more *given* points.

1.2.2 Estimation and Representation of Manipulation - Relevant Object Properties and Actions from Human Observation

As we motivated at the beginning, we are interested in an abstract representation of manipulation-relevant object properties and actions. Rather than storing observed x,y,z-trajectories, we want to extract the characteristic properties of an observed manipulation. We use an object-centric perspective. It allows not only a knowledge transfer to any robot, but, also the reusage of knowledge in similar situations (see also Section 3.1).

In general, the estimation and representation of knowledge is related to three fields of research: Artificial Intelligence in the context of knowledge representation, Computer Vision for the observation of humans, and Robotics, in which the knowledge is applied.

In the field of Artificial Intelligence, different representations of knowledge exist. In [20], the environment of the robot was categorized according to the properties of task environments. The categorization can be applied as follows.

The environment is partially observable to the robot, since not all relevant aspects of manipulation properties can be observed all the time, e.g., because of occlusions or noise. The environment appears stochastic to the robot, since the environment is partially observable and the model of the world is non-complete. The robot can detect known and, therefore, deterministic observations. Moreover, the robot has to be able to handle unknown (stochastic) observations. In a real world problem, the robot should, in fact, not just handle such observations. It should make use of it to improve its model of the environment.

Furthermore, the robot's environment is dynamic, since, e.g., a human can change it. The changes in the environment happen continuously, leading to a continuous-state and continuous-time problem. Due to the dynamic environment, a static model is not enough to deal with such changes. Therefore, the permanently non-complete model has to be extended, using the learning capabilities of the system.

A lot of actions in the environment depend on other actions. Hence, the robot acts in a environment of sequential events. The system can make use of this property, e.g., during the observation of procedures of human actions. The dependencies of manipulation sequences on each other are not examined in the representation.

The multi-agent environment of the robot is used for the acquisition of new knowledge through

the observation of an other agent.
To sum up, the robot has to handle new information and unknown circumstances, (contingency problem, [20]). The aim is to provide a representation, which allows (1) to detect new information efficiently and (2) to represent the knowledge in manner, such that the detected new information can be utilized.
The knowledge, which the robot attains, can be seen as constraints, which the robot has to consider for its own manipulation afterward.

In the field of workflow acquisition and analysis, Hidden Markov Models (HMMs) [21] have been applied often. Padoy *et al.* [22] proposed Workflow-HMMs for workflow monitoring based on 3D motion features. The analysis of a workflow sequence, which is split into segments, is also of interest here, but the focus is on the analysis of the manipulation properties of the object itself and its functionality in the environment. Reiley and Hager [23] used HMMs for surgical skill evaluation of robotic minimally invasive surgery. Discrete HMMs were built at task level and at the level of surgical gestures. The analysis of the different skill levels was based on the check of the efficiency of the procedure planning. We also analyze gestemes, but in the context of motion characteristics in object manipulation. Webel *et al.* [24] used HMMs for the workflow acquisition of assembly skills, but they classified properties related to the human hand and not the object itself. Another observer-centric approach was presented in [25], in which HMMs were used for action representation and recognition of action primitives.

Object detection as well as object tracking are important elements for the acquisition of knowledge. They are related to the field of Computer Vision. Even though, they are not the focus of here, they have to meet certain criteria. The object detection and recognition has to be done in a manner, such that the object is classified according to the manipulation-relevant properties of the object. In contrast, a pattern or color based recognition does not consider these properties. Hence, the object recognition approach in [26] is considered here. The tracking algorithm has to be able to observe the object during a human demonstration. Therefore, it has to be real-time capable. We use the algorithm presented in [27].

Extensive work exists in the field of imitation learning. In [28], HMMs were used for imitation learning of arm movements in manipulation tasks for humanoid robots. There, the aim was a human-like reproduction of the motions. Moreover, further non-object-centric approaches have been presented. They include, e.g., the imitation/ learning of motor skills or the imitation of movements with Dynamic Movement Primitives (DMP), which encoded directly the trajectories themselves [29], [30]. Approaches related to Reinforcement Learning [31] were also applied in imitation learning. They utilized the observations of humans as reward [32], [33]. Calinon *et. al* used imitation learning, in order to learn control strategies [34], [35]. HMMs and Gaussian Mixture Regression (GMR) were deployed for the imitation of human motions [35]. In [34], they considered the object's relative position, but they were reproducing a certain task using a dimensionality reduction method (Principal Component Analysis) and mixture models for the generalization of variations in the given task. An object-centric model was presented

in [36]. It integrated variations along the object's trajectory (by using the corresponding mean and variance of multiple demonstrations).
In contrast to the described work in the field of imitation learning, we do neither aim to imitate the humans demonstrations nor do we aim to encode the trajectories as, e.g., DMP or the model in [36]. We want to provide a more general representation of object properties and their functionality in the environment, in order to get a further understanding of the knowledge about different tasks and environments. The objects themselves are in the focus. Hence, we provide an object-centric approach. This aim goes beyond imitation learning. We want to generalize the observation to cope with variations in repetitive human actions.

The intention in imitation tasks was addressed by Jansen and Belpaeme [37]. They trained their agent in a grid with blocks in a computer simulation. In contrast, our work deals with more complex, real-world environments. Furthermore, our system needs much less training instances than the one presented in [37]. A real-world example of capturing the user's intention about sequential task constraints was presented in [38], [39]. Their system reasoned about the existence of sequential dependencies of operations. Moreover, the system was refined incrementally in [39]. Here, we want to attain a further understanding of the object's functionality itself. This offers possible operations to the robot later on. The knowledge can be refined incrementally.

In order to achieve a further understanding of the object's functionality, the object's motion has to be analyzed. Basic motion properties of an object were analyzed for a child's play in [40]. The proposed system goes further, since the motion properties related to the possible states of the object and its function in an environment are of interest. The work in [41] is closer to the proposed object-centric approach, even though, it was in the context of imitation learning. It took into account the effect on the object (position, orientation). The object properties and its state are also of interest here. However, the acquired knowledge is a significant difference between the approach in [41] and ours. We are interested in the manipulation properties directly related to the object and, furthermore, the objects functionality in the environment. Function from motion was analyzed in [42] for "primitive motions", which were translations or rotations relative to the main axes of primitive objects. Our approach goes further to more and more general manipulation-relevant object properties. Additionally, we focus on functionality in context of the environment. In [43], functional roles of objects like "pour out" have been introduced explicitly. These roles do not refer to the object's properties, which are directly observable during manipulation. Moreover, it is important to distinguish our analysis of the object's functionality observed from human demonstrations from reasoning about shape descriptions for object functions [44].

It should be to pointed out, that the reconstruction or the analysis of the environment, like [45], is not of interest. We focus on the objects and their functionalities in the environment. In [45], the relative/ absolute position of objects to each other has been used for the consideration of the environment in manipulation properties. A perceptual space (for the color and shape object properties) and a situation space (for the displacement of the objects in the

scene) were introduced in [46]. In contrast, the object properties in our representation are beyond the pure visual appearance of the object, since we are interested in the manipulation-relevant object properties. Furthermore, the Functionality Map does not deal with the object's relative position to each other. It aims to understand the objects and their functionalities in the environment.
Moreover, an analysis or a semantic labeling of the entire scene is not our aim. In [47], the semantic labeling of spatial entities was done for the entire scene. We reduce the analysis to the object of interest. This is also a difference to [48], which dealt with spatial and visual appearance based on navigation and mapping.

Other related areas are planning and navigation. For example, [49] provided a representation for constraint manipulation planning of programming by demonstrations of certain tasks. In contrast, the representation in this Section is more general. It does not aim to acquire or to represent knowledge for a specific task or to plan a certain task.

In the context of robot navigation, surprise-based learning has been applied [50], [51]. There, basic visual features (color matching) were used for a prediction model based on rules (logic) in a static environment. Our approach works on more general object properties, which are relevant for manipulation.

1.2.3 Representation of Object Relations in the Environment for Dexterous Manipulations

We also develop a representation for dexterous manipulations. Such manipulations require a much more detailed knowledge in comparison to the already described representation of manipulation-relevant object knowledge (related work in Section 1.2.2). Usually, a manipulation aims to achieve a desired goal state in the environment. We use a contact-state perspective to represent such states (see also Section 3.2).

The changing contact state (sometimes also contact "mode") between an object and a robotic hand has been analyzed multiple times. For example, in [52], [53] [54] and [55], a randomized manipulation planner was presented for a multi-fingered hand and switching contact modes. The contact state between a robot and an object (passive, hybrid and active closure) was discussed in [56]. A contact space dynamic model and active joint space model were developed in [57]. A manipulation of an object using the entire body was discussed in [58]. Whole body contact manipulation methods were presented in [59], [60]. For example, the method in [60] allowed, e.g., the physically demanding task of transferring a patient from a bed to a wheelchair.
Moreover, approaches regarding a contact between an object and its environment have been proposed. For example, the kinematics analysis of such a manipulation was presented in [61]. A contact state transition graph for graspless manipulation was proposed in [62]. Srinivasa *et al.* [63] considered the contact between the robot and the object as well as the contact between the object and the environment in control synthesis for dynamic contact manipulation.

In contrast to existing work, we do not focus on the contact between a robot and its environment, respectively, objects. Our perspective has the aim to describe the environment by the contacts within the environment. In contrast to assembling tasks, we present a general representation for dexterous manipulation in the environment. Just the current and the desired contact state descriptions are known here. This distinguishes our work also from [64]. There, sequential constraints of tasks were learned through a Programming by Demonstration approach. The process of the manipulation itself was in the foreground there. Our starting point is the aim state of the manipulation. Moreover, we focus on a representation for very dexterous manipulations.

Possible applications of our approach include dexterous manipulations in medical applications. A good overview of surgical and interventional robotics can be found in [65], [66], [67], [68] and [69]. The broad range of the area becomes, e.g., clear in special issues like in [70]. An abstract representation of surgical tasks was, e.g., suggested in [71] based on [2] (for [2], see also Section 1.2.2). It is important to distinguish our work from the aim of surgical skill teaching in general. Reiley and Hager [23] used HMMs for surgical skill evaluation of robotic minimally invasive surgery. Discrete HMMs were built at task level and at the level of surgical gestures. The analysis of different skill levels was based on an efficiency check of procedure planning. Padoy *et al.* [22] proposed Workflow-HMMs for monitoring the workflow based on 3D motion features. They used a hierarchical HMM with phase-probability variables, in order to model the dependencies between distinct phases at the top level and to model the dependencies within individual phases. A statistical modeling and recognition method for surgical workflow was presented in [72]. In [73], a Human Machine Collaborative (HMC) system was proposed to perform portions of surgical tasks autonomously and other parts manually. HMMs were used to learn subtasks, which were performed autonomously later on. Konietschke *et al.* [74] presented a multi-modal training platform for minimally invasive robotic surgery. The presented training included skills on workspace constraints of the console as well as on workspace constraints of the executing manipulator.

Our representation is independent from the deployed robotic system. The authors in [75] validated robotic surgery training assessment across different training platforms. We do not focus on a *certain execution* of a task with a *certain robot*, but on the abstract description of a dexterous manipulation task itself. Our aim is an abstract representation of *dexterous* manipulations in *general.* Medical procedures are one area of possible applications. It is important to point out, that this description is not considered to be used directly for automated surgical operations. It is an abstract representation of required capabilities for a dexterous manipulation.

1.2.4 Path Optimization for Abstractly Represented Tasks with Respect to Efficient Actuation

As motivated in Section 1.1, a path optimization with respect to efficient actuation is required in the context of abstractly represented tasks, since we want to analyze the potential of path optimization in the case of joint failures. The optimization has to be able to utilize the freedom in path planning which accompanies with the abstract task representation. It is important to point out, that we do not aim to search a path (e.g., [76], [77]). We want to optimize a path of an *abstractly represented task* with respect to *efficient* actuation.

Path optimization has been done with respect to different criteria. For example, points on the path were optimized to increase the distance to obstacles in [78]. Kinematics singularities were considered in [79]. Smith *et al.* proposed an optimal path planning for surveillance with temporal-logic constraints [80]. Other authors included dynamics in the optimization process. In [81], the authors dealt with the complicated dynamics of large space manipulators. Rieswijk *et al.* incorporated actuator and jerk constraints in [82].

If we want to optimize the path with respect to efficient actuation, a solution with the minimal distance between the start and end points in 3D space could be an intuitive solution. As we will show in our experiments [3] (see Section 7.4.1), the shortest path is not necessarily the most efficient one with respect to efficient actuation. Hence, we propose an exhaustive path optimization with the Elastic Power Path concept.

The name of our concept "Elastic Power Path" might cause an association with the "Elastic Strip Framework" [83], especially in the context of path planning. Both concepts have the elasticity of the path in common. However, the aims are different. We want to use the elasticity of the path to improve the efficiency in actuation. In [84], [85], a minimization of energy (the instantaneous kinetic energy of the robot) was processed among others. A special Jacobian was used there. In contrast, our optimization is independent of any Jacobian J. Moreover, δx and $\delta\theta$ are unknown (e.g., in $\delta x = J(\theta)\delta\theta$). Furthermore, we use a global optimization method to overcome local minima. The aim of our path optimization is efficient actuation under the consideration of the abstract characteristic properties of a desired manipulation.

1.2.5 Structure Analysis of Manipulators

This thesis aims to provide a general, task-specific analysis of manipulator structures with respect to efficient actuation. Such an analysis has not been presented yet. In the field of manipulator structure analysis, the Jacobian Matrix is often incorporated. For example, the manipulability measure in [86] was based on the determinant of the Jacobian Matrices. Kim and Khosla [87] investigated further in the Jacobian as dexterity measure of a manipulator. They introduced a measure of isotropy which can be used to develop a manipulator close to an isotropic configuration for a given position in 3D space. Reachability and joint limits were considered for the kinematic design of serial link manipulators in [88].

Asada [89] proposed a dynamic analysis of a manipulator's workspace based on a graphical representation of inertia ellipsoids. Lee [90] compared manipulability ellipsoids and manipulability polytopes. The Dynamic Capability Equations in [91] described the acceleration and force capabilities of a robot at a particular configuration under the consideration of the limitations of the manipulator's motor torques. The translational and rotational quantities were mapped into representations of actuator torques with same units.

Dimensionality reduction techniques were presented for different robotic applications as, e.g., in grasping. In [92], a dimensionality reduction was applied on hand-independent dexterous robotics grasping. The spatial and temporal context of human grasping actions was studied in [93].

In contrast to the described approaches, we aim to analyze any arbitrary manipulator with respect to the change of the efficiency of its actuation under possible joint failures in the context of an abstractly represented task. We are interested in (1) the inter- and intra-robot analysis of the efficiency of the robot's motion, (2) the variety of directions in which the end-effector can be moved efficiently and (3) the capability to change the orientation of the end-effector without a change in position. The manipulator can have an arbitrary number of joints. Therefore, it could be highly redundant, too. It is important to notice, that we do not aspire to model torque bounds or something similar. The analysis of parallel manipulators (e.g., [94]) is not our aim, we focus on serial link manipulators.

1.2.6 Analysis of Master-Slave Systems

As we mentioned already at the beginning, we want to apply the proposed analysis of manipulator structures in a further application area: Master-slave systems. A general overview of teleoperation systems can be found in [95]. Yokokohji and Yoshikawa [96] discussed the analysis and design of master-slave teleoperation systems with respect to stability and transparency in general. They focused on an "ideal response" in the context of large time delay in teleoperation (e.g., [97]). Stability and transparency were desirable there, even under time delay. Transparency is the correspondence between the positions of the master and the slave system [96], [98]. Yokokohji and Yoshikawa [99] analyzed also the maneuverability and stability of micro-teleoperation systems. Lawrence [100] dealt with stability and transparency in bilateral teleoperation under significant communication delays. A local force feedback to the human user has, e.g., been applied in [101].

The authors in [102] made use of a redundant slave manipulator to guide the user away from singularities in a teleoperated application. The performance of a redundant slave robot in teleoperation was analyzed in [103]. An overview of the state of the art in telemanipulation for remote minimally invasive surgery as well as requirements for an ideal telemanipulator were given in [104]. A detailed background of robot-assisted minimally invasive surgery systems can be found in [105]. The human, resp., the convenient usage of the system for the human user is the most important aspect in our analysis. Hannaford [106] pointed already out the importance of the consideration of the human operator in the analysis of teleoperation systems. His work focused on stability and performance tradeoffs in the context of time delay.

The related work in the area of surgical and interventional robotics including surgical skill teaching has already been presented in Section 1.2.3. The following paragraph is just shown for completeness within this Section. A good overview of surgical and interventional robotics can be found in [65], [66], [67], [68] and [69]. The broad range of the area becomes, e.g., clear in special issues like in [70]. An abstract representation of surgical tasks was, e.g., suggested in [71] based on [2] (for [2], see also Section 1.2.2). It is important to distinguish our work from the aim of surgical skill teaching in general. A review of methods for the objective evaluation of surgical skills can be found in [107]. Reiley and Hager [23] used HMMs for surgical skill evaluation of robotic minimally invasive surgery. Discrete HMMs were built at task level and at the level of surgical gestures. The analysis of different skill levels was based on an efficiency check of procedure planning. Padoy *et al.* [22] proposed Workflow-HMMs for monitoring the workflow based on 3D motion features. They used a hierarchical HMM with phase-probability variables, in order to model the dependencies between distinct phases an the top level and to model the dependencies within individual phases. A statistical modeling and recognition method for surgical workflow was presented in [72]. In [73], a Human Machine Collaborative system was proposed to perform portions of surgical tasks autonomously and other parts manually. HMMs were used to learn subtasks, which were performed autonomously later on. Konietschke *et al.* [74] presented a multi-modal training platform for minimally invasive robotic surgery. The presented training included skills on workspace constraints of the console as well as on workspace constraints of the executing manipulator. The authors in [75] validated robotic surgery training assessment across different training platforms.

In contrast to existing work, we do not aim to optimize the performance of the human for one existing system. We provide a framework for the analysis of the human's performance on arbitrary systems with respect to convenient usage. Hence, we do not analyze the trajectory performed by a human, but the human's motions with respect to the work the human has to perform. Medical procedures are one area of possible applications. Moreover, we do not work in the area of autonomous surgery as, e.g., [108]. The authors in [108] discussed the formal verification of properties of autonomous surgery devices as well as the formal verification of a simple puncturing action plan.

Nisky *et al.* [109] presented a framework for the analysis of surgeon arm posture variability. Their work has a similar direction as ours. However, they analyzed surgeon arm posture variability in relation to the performance with respect to stabilization and variability in hand trajectory change as well as the end-point error. Our work focuses on the overall work which is performed by the human.

1.3 Research in Dexterous Robotic Manipulation with Respect to Bio-mimicry, Bio-inspiration and Technical Approaches

The aim of this thesis is the analysis of serial chain manipulator structures with respect to efficient actuation in the context of abstractly represented tasks. The underlying perspective is mainly object-centric, resp., robot-centric, depending on the respective component of the analysis. This is a significant difference to anthropocentric approaches. Many approaches in robotic grasping and dexterous manipulation are inspired by the human's comprehensive ability to grasp and manipulate objects with his/ her hand. Of course, there are arguments for the one or the other perspective. Hence, we want to discuss the advances of bio-mimetic, bio-inspired and technical approaches. First, we provide an overview of the current state of the art in robotic grasping and dexterous manipulation with respect to the different perspectives. It forms an appropriate base for the discussion. In order to keep the overview at a reasonable size, we put our focus mainly on the most recent developments in the last years.
We choose the topic "robotic grasping and dexterous manipulations" for the discussion, since this thesis is associated with this field. Moreover, grasping and manipulation capabilities are an important and demanding field in robotics. They enable an interaction with the environment as well as cooperation with humans. Such interactions require the consideration of many different aspects, e.g., regarding physical capabilities and psychological influence. The information about the environment is also important in the context of interactions. Many questions arise: Which (possibly unknown) objects are in the scene and how are they related? How to grasp and handle the object? How does the human act? Is the environment changing?
The available information about the environment can be incomplete, ambiguous and complex. However, humans have sophisticated grasping and dexterous abilities. The acquisition of similar abilities is an important step to deploy robots in daily-life of humans.

The just mentioned human's comprehensive ability to grasp and manipulate objects with his/ her hands inspired many approaches in robotic grasping and dexterous manipulation. Such inspirations affect not only the hand's physical structure, but also its capabilities. At the same time, there are approaches and physical structures which are purely or partly based on technical requirements. These approaches have a different perspective than bio-mimetic and bio-inspired approaches. The technical approaches use the desired final functionality as starting point to develop possible solutions.
The chosen perspective of the approach can also depend on the question, whether really a grasping, resp., manipulation task is present. If a movement of the hand is desired for another purpose like the performance of a gesture in human-robot communication, one could argue, that a human-inspired type of approach could lead to a significant advantage. Our focus is here on grasping and manipulation themselves. This can, of course, include an implicit type of communication through a cooperation between the robot and a human. Hence, we give a brief

overview of psychological aspects in human-robot interaction and include it in our discussion. If one considers haptic interfaces, which are used to control a robot remotely (e.g., [110]), the requirements differ from the ones on robots for non-remote grasping and manipulation actions. We do not cover haptic interfaces here, we focus on non-remote grasping and dexterous manipulation tasks in our overview and discussion.

We distinguish two different types of approaches. The first type consists of bio-mimicry and bio-inspired approaches, which directly copy ideas or structures from nature or which are inspired by nature. Approaches which are build on technical requirements form the second type. We give an overview of the current state of the art in robotic grasping and dexterous manipulation with respect to both described types.

Moreover, we want to compare research in dexterous robotic manipulation with another technical field which faced a similar situation regarding the types of approaches. We are interested in the developments and the experiences in the other field. At least some research results of the other technical field have to be in successful daily use today. The approaches can, then, be assigned to one or more types. We choose the design of airplanes for comparison. The design of airplanes fulfills the above requirements, since there were different approaches in the history of aviation, which can be assigned to one (or more) of the described types (further details in Section 1.3.5).

Overviews and surveys about robotic manipulation have already been presented. For example, Okamura *et al.* [111] gave an overview of dexterous manipulation. They described already, that two different types of approaches can be observed: While some researchers focused on an anthropomorphic approach, others pointed out a fundamental difference between human and robotic hands. Similarly, Bicchi summarized the state of the art in the field of robot hands and depicted a basic distinction between human-like hands and “minimalistic hands”, e.g., in terms of the least number of actuators [112].
In contrast to existing work, we give a current overview of grasping and dexterous manipulation. Moreover, we want to compare the development in the area of grasping and dexterous manipulation with the development in another technical field. In [113], the state of robotics today was generally compared with the state of computers in the 1970s. In contrast to computers, grasping and dexterous manipulation require the consideration of advanced physical “interactions” with the environment. Hence, we want to compare grasping and dexterous manipulation with a technical field, which also needs to deal with such an interaction and which is meanwhile part of our daily-life: Flying. We focus on a comparison with respect to bio-mimicry, bio-inspiration and purely technical approaches.

First, we give an overview of the current state of the art in robotic grasping and dexterous manipulation with respect to bio-mimicry/ bio-inspiration and technical approaches. In order to keep the overview at a reasonable size, we put our focus mainly on the most recent developments in the last years. The overview includes the following subfields: physical structures of robotic hands, grasping and manipulation approaches, usage of sensors and analyses of psychological aspects in human-robot interaction. Afterward, we introduce another field of engineering to illustrate why bio-mimicry and bio-inspiration are not necessarily the best way to design technical systems: The design of airplanes. Hence, the history of aviation is briefly described. The current state of the art in robotic grasping and manipulation is discussed thereafter.

1.3.1 Physical structure of robotic hands

The physical design of robotic hands can already be assorted in human-like hands and technical hands. In some cases, both types of approaches, bio-inspiration and technical requirements, are included in the construction. We, then, assort the hands according to its main appearance and refer to the bio-inspired part, respectively, technical part.

First, we give an overview of human-like hands, before the technical hands are described. The hands of the widely known ASIMO [114] have respectively 2 DoF [115] to grasp objects. Schunk developed in cooperation with DLR and HIT the SAH anthropomorphic hand [113]. The hand has four fingers with sensors to determine force and position values. The Michelangelo hand of Otto Bock [116] aimed to recreate as many hand functions as possible in a prosthesis. The DLR hand arm system [117] was designed to reach its human archetype regarding robustness, dynamic performance and dexterity. However, they did not simply copy the human hand, but they based the design on abstract, fundamental functionalities of the human hand. A special focus on the functional analysis of the thumb was presented in [118]. In contrast to a pure mechanical copy of the thumb, guidelines for a design regarding the functionality of the thumb were presented. The authors in [119] built an anthropomimetic musculoskeletal upper torso, which mimics the mechanical structures of the human body. It includes a hand with one artificial muscle to close and open the hand.

The physical structure of other robotic hands has been based on technical requirements. Dollar and Howe [120] presented a simple and robust grasping system. It consists of a four-fingered hand. Respectively two fingers were arranged parallel to each other. The resulting two pairs of fingers were opposed to each other. The entire hand was only driven by a single actuator. Nevertheless, the system was able to deal with uncertainty which accompanies unstructured environments. It made use of compliance to achieve adaptability and robustness in robotic power grasping. Such a design makes precision grasping and in-hand manipulation more difficult. An in-hand manipulation with an underactuated elastic hand has been demonstrated in [121]. Underactuated fingers were used in [122] to grasp very thin objects from flat surfaces by flipping the objects. The corresponding single steps were similar to a human's action. A robotic hand which differs significantly from others robotic hands was built at TU Berlin [123], [124], [125]: The hand was mainly made of rubber and it was actuated by a pneumatic system. The authors made use of the advantage of compliance to grasp, e.g., bottles or glasses, while still using simple control.

1.3.2 Grasping and manipulation

Similarly to the physical design of robotic hands, grasping and manipulation approaches can be categorized in human-like ones and technical ones.

We start with the anthropocentric approaches. A lot of work exists in the field of imitation and learning from human observation. Calinon *et al.* used imitation learning, in order to learn control strategies [35]. Approaches related to Reinforcement Learning [31] were also applied in the context of imitation learning. They utilized the observations of humans as reward [32], [33]. The mapping of human grasps to manipulator grasps is, e.g., addressed by Kang and Ikeuchi [126]. They made use of the concept of the virtual finger [127]. A metric for the evaluation of human-to-robot mapping of grasps was based on this concept [128]. Ciocarlie *et al.* [92] utilized results in neuroscience research for hand-independent dexterous robotic grasping. They reduced the high number of DoF to a lower dimensionality by introducing eigengrasps. Eigengrasps are a number of basis vectors, which determine a low dimensional subspace of the DoF space. Their approach was bio-inspired, but the application was not restricted to human-like robotic hands. Romero *et al.* [93] studied how different grasps are performed. They distinguished between the actual grasping phase and the approach phase of the grasp. The authors in [129] used a fully data-driven approach to generalize human grasping for a multi-fingered robotic hand. After learning efficient grasp representations from human demonstrations, the contact points were wrapped onto new objects. Synchronized reach-and-grasp movements were optimized before execution.

The just described approaches have an anthropocentric perspective. Object-centric and robot-centric approaches have been presented as well. For example, the effects on a manipulated object (position, orientation) were taken into account in [41]. The object properties and its state were also analyzed there. In [2] (see Section 3.1.2), a clear object-centric perspective was used. Not only information about the manipulation properties of the object was obtained, but also the object's functionality in the environment. A robot-centric perspective has been applied in [3], [5] (see Chapter 4) for the analysis of the efficiency of the generated motions (see Fig. 1.4). Ma and Dollar [130] focused on more generalized tasks and object-centric definitions of dexterity. They discussed arm vs. hand dexterity and provided a classification of in-hand manipulation without a constraint of anthropomorphism. For a two-armed manipulation task, the task structure was exploited in [131]. These bimanual operations are an additional challenge due to the paragraphization and relation of both involved arms.

1.3.3 Usage of sensors in grasping and manipulation

Perception plays an important role in robotic grasping and dexterous manipulation. It enables the necessary interaction with objects and the environment, since information about the environment is provided. If the perception is incomplete, the system needs either pre-knowledge or an approach which allows to deal with the incomplete information.

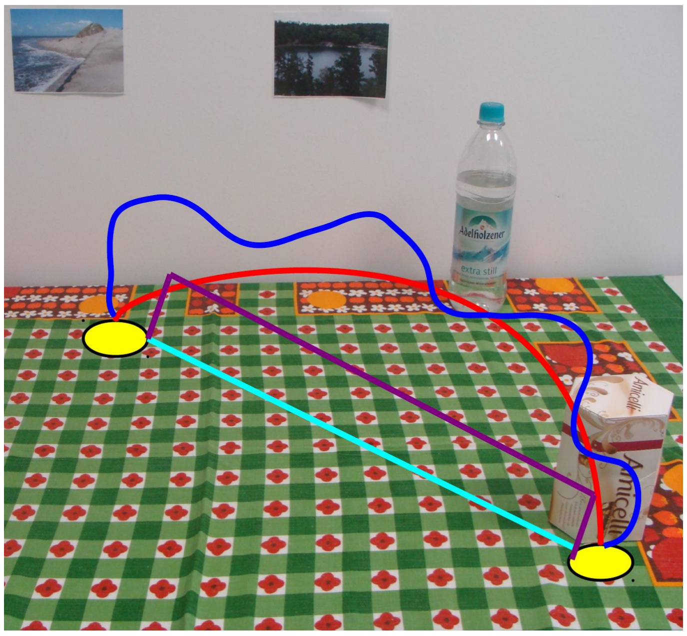


Figure 1.4: Paths with different properties for a manipulation between two areas. A robot-centric perspective has been applied in [3] (see Section 4.1): Paths with different properties (e.g., motion shapes) are shown for a manipulation between two areas (yellow). Which path should be chosen to achieve efficient actuation during the manipulation?

In general, many different types of sensors exist as, e.g., visual, acoustic or pressure sensors. In the context of robotic grasping and manipulation, not all possibly available sensors are usually used.

Similarly to human's eyes, visual sensors can observe a scene during the execution of the robot's actions or during demonstrations by a human user. It is important to notice, that the observation of a human's action does not necessarily mean, that the action is directly imitated. Instead, the knowledge can be extracted at a much more abstract level as in [1], [2] (see Section 3.1.1). The less pre-defined the environment is, the more required the information from sensors and the more necessary more autonomous systems get. For example, piles of unknown objects were grasped in the context of an empty the basket-scenario in [132]. There, the shape based learning approach needed to deal with incomplete point cloud data. An interaction with a cluttered environment was presented in [133], where unstructured groups of objects were singulated through interaction.
The choice of the placement of the visual sensor can be inspired by the place of the human's eyes or by the requirements of the desired task. A sensor placed at the upper part of the robot [134], [135], [136] is clearly inspired by the human's eyes' position. This enables an overview of a larger area. In contrast, a camera can be used to observe a very small area. For example, the camera on the forearm of the robot Baxter [136] enables a close view of objects. Obviously,

the forearm position of camera was based on technical requirements and not inspired by the human's eye's position.

Haptic feedback is another important way to perceive the environment. This enables interaction with objects and humans. For example, the DLR-Hand II [137] has a six-dimensional finger tip force torque sensor on each finger (strain gauge sensor). Similarly to a human's skin, artificial skin was, is and will be developed as, e.g., in [138], [139].

A robotic hand can also contain sensors *in* the hand, and not only sensors *on*, respectively, *outside* the hand. For example, sensors in the SAH anthropomorphic hand [113] enable the measurement of forces and torques. Another example is a force-torque sensor, which has been placed in the wrist of the robot ARMAR-III [140].

Sensors are usually deployed to get more information about the environment and possible uncertainties. A more implicit way of dealing with uncertainty in unstructured environments is the usage of compliance as Dollar and Howe [120] showed.

1.3.4 Psychological aspects in human-robot interaction

Of course, psychological aspects can play an important role in human-robot interaction. One could argue, that a human-like behavior is more predictable and, hence, more acceptable for a human user. The opposite argument is anxiety which can arise due to a really human-like robot.

In this paragraph, we do not limit ourselves to robotic hands. General human-robot interactions provide us a much larger range of studies, which is necessary here. The analyzed psychological aspects vary between the studies. For example, anxiety toward robots was measured, e.g., with respect to human-like robots [141]. Depending on the study, the term "human-like" is related to different properties of the robot, e.g. its behavior or its appearance. Many studies in this area suffer from a small number of participants, the lack of several real robots or non-significant differences in the preference of human-like and non-human-like robot properties.

A (non-representative) group of people was asked to grade properties of (imagined) robots on a "positive-negative" scale in [142]. A machine-like robot appearance seemed to generate more positive feelings than a human-like or an animal-like appearance. The robot was seen as a useful tool, which should, e.g., be easy to use, be save and behave correctly. A robot which acted too personally seemed too be less liked in public [143]. In [144], expectations of younger and older adults were examined. The participants had to imagine a domestic robot and to fill out a questionnaire. The analysis of the questionnaires, which were sent back, showed, that the imagined robots were seen as helpful, purposeful devices and less as socially-intelligent devices. Other aspects like the cultural background or human attributes as, e.g., sex and age of the user, were considered in other studies, as for example in [145], [146]. Breazeal [147] pointed out the importance to consider the perspective of the human *and* the robot. She argued, that benefits from endowing a robot with social skills and capabilities are beyond the interface value for an interacting person. Kanda *et al.* [148] showed, that the difference in appearance

of two divers humanoids did not affect the verbal behaviors, but the non-verbal ones such as distances. In a further, large study, Kamide *et al.* [149] identified nine factors for the evaluation of general impressions of humans regarding humanoids. The factors were extracted based on eleven existing humanoids.
In the case of gestures in human-robot communication, an imitation of the human gestures could be useful to enable an intuitive communication for the human with the robot. This nearby idea does not necessarily seem to be true as [150] showed.

In general, psychological aspects in human-robot interaction are complex, intricate and depend on many factors. Hence, an overall answer regarding a preferable robot's appearance and behavior in general (including different cultures, applications, etc.) does not seem to be possible, at least not at the moment. Moreover, an ethnical aspect has to be discussed in this context (see, e.g., [151]).

1.3.5 Relation of Bio-Mimicry and -Inspiration to Technical Approaches Using the Example of the Design of Airplanes

To better illustrate why mimicry is not necessarily the ideal way for technical systems, let us compare it to another field of engineering: The design of airplanes. In order to provide a feasible comparison, we need an overview of the history of aviation with respect to bio-mimicry, bio-inspiration and technical approaches.

A lot of literature can be found about the history of aviation. We build up this overview (including all information, conclusions, etc., in this Section 1.3.5) based on [152], [153], [154], [155], [156], [157], [158] and [159]. Our aim is a general overview of the history of aviation. Hence, we do not discuss inconsistent information in the references (e.g., dates). We choose the information, which is given in most of the references.

1.3.5.1 Early attempts

People have been dreaming of flying for a very long time. Today, it is part of the daily-life of many people. Hence, it seems, that the dream of flying became true. However, if one considers early ideas of flying, e.g., Icarus in Greek mythology, it becomes clear, that people did not imagine something like today's airplanes back then. They thought about a flying mechanism like a bird with feathers and beating wings.

In contrast to these bio-inspired ideas, approaches based on technical designs had been suggested. Such designs were, e.g., machines, which should fly, since they can get lighter than air: Balloons. The disadvantage of these types of aircraft is obvious: The more weight should be carried, the larger the balloon has to be.

Sir George Cayley observed accurately the flight of birds. Moreover, he performed systematic experiments as well as mathematical computations. The substantial progress in his work were wings, which produced an ascending force and not a propelling force any more. To conclude, his work was based on bio-inspiration and technical requirements. Based on Cayley's publications, Samuel Henson worked on airplanes which were actuated by steam-engines. Even though, he found temporary funding for his company at the beginning, he did not find further funding. The limitation of actuation to steam-engines was a general problem at this time.

Clément Ader was the first one who claimed, that he has solved the problem of motor-aviation. His steam-powered Éole made a 50 m-long hopper. Even though, it was a beginning, it was not a long-term, steered flight.

Most of the developers in motor-aviation did hardly think about the way how the machine would fly after the take-off. In contrast, representatives of the gliding flight approaches aimed to achieve further progress through flying experiences. Otto Lilienthal was the most prominent "birdman". He built different types of wings, which he wanted to use to fly like a bird. Lilienthal's work was much more scientific and useful than it seemed at first. He concluded from the precise observation of birds, that a curved wing is advantageous for ascending forces. His practical work showed him, that fixed wings are much more useful than beating wings. His machines were able to fly. In general, Lilienthal performed a lot of flights. The machines did not have any steering system, hence, he had to throw his body into different directions to keep the balance in the airflow. Lilienthal died during one of experiments. Nevertheless, others were inspired by his work. Moreover, he had shown the importance of steering systems. Lilienthal's work was based on a combination of bio-inspiration and practical experiments.

1.3.5.2 The first flights

The work of the Wright brothers was very systematic from the beginning. They studied existing work of Cayley, Lilienthaland others. They focused on steering, which appeared to be a hardly noticed area. In contrast to previous work, they assumed, that flying an airplane is closer to riding a bicycle than to driving a car. One had to keep the balance during flying, similarly to cycling. Perhaps, it was helpful for them, that they were running a bicycle shop. Another important finding was achieved through their observation of floating buzzards. The movements of the wing tips is essential for the animals to keep the balance. The invention of the sloped wings began.

At the beginning, the Wright brothers started many testings with a gliding biplane, which was quite similar to a machine used for successful motor-driven flights later on. The experiments showed the brothers not only the way towards the improvement of their own work, but also errors and imprecision in the previous work of others. Moreover, they were confronted with the problem of curve flying in their experiments. They introduced movable vertical tail fins to be able to control the machine in curves.

After their success with gliding biplanes, they started to work on motor-driven airplanes. The motor was built by one of their assistants (Charlie Taylor), since the available motors did not fulfill their requirements. The result was an outstanding motor at this time. After solving problems regarding the propeller shape and some occurring failures, they succeed in the first steered and controlled flight with an airplane heavier than air on December, 17th 1903.

To conclude, the contributions of the brothers Wright were based on technical approaches (e.g., comparison to bicycles) as well as bio-inspiration (e.g., floating buzzards). Practical experiments were a further important milestone towards their success.

1.3.5.3 Technology Acceptance

The acceptance of new technologies is an important aspect. If the technology is not accepted or just accepted at a low level, its deployment can be limited significantly. Hence, it could be of benefit to know, what might cause a possible fear regarding a technology. Therefore, we are interested in the question, where the fear of flying might come from. This fear is still existing, even though, flying is a part of the daily-life of many people.

If just statistics are taken into account, there does not seem to be a reason to fear flying. Significantly more people die in car accidents than in regular flights every year. Big accidents are seldom, but, nevertheless, they seem to have a very important influence on surprisingly many people, to whom accidents are strictly inseparable from flying. Others argue, that other reasons can cause such a fear. A fear of heights, a fear of being locked-in and a fear of being caught without control of the own situation are prominent ones. There are different possibilities to teach people to handle their fear of flying. For example, appropriate seminars seem to be a successful method.

1.3.6 Discussion of the Current State of the Art

After the overview of the current state of the art in robotic grasping and dexterous manipulation, the discussion follows in this Section. Moreover, we compare the current state to the other field of engineering: aviation. Of course, such a comparison can hardly be done 1-by-1. An abstract comparison is much more feasible.

The development of airplanes, which were really able to fly, was influenced by several factors. The observations of animals gave important insights into biological systems and their performance. The realization of practical experiments lead to important practical and theoretical insights. The development of the theoretical background was a further essential step towards the development of airplanes. To conclude, a mixture of bio-inspired and technical approaches based on theoretical analyses and practical experiences have lead to the long-term success in aviation today.

A similar development is observable in robotic grasping and dexterous manipulation, as Section 1.3.1-1.3.4 shows. The concepts are based on bio-mimicry, bio-inspiration and technical requirements. Bio-mimicry is, e.g., applied in the construction of prosthesis. The inspiration from biological systems is used to build up advanced technical systems. The analysis of desired grasping and manipulation capabilities and abilities results in the requirements of technical approaches. Approaches which combine both types can be found as well. For example, the

design of the DLR hand arm system [117] was based on fundamental functionalities of the human hand.
A significant inspiration from the human's archetype is noticeable in sensing methods in the context of grasping and manipulation. A large amount of work on vision-based sensors is human-inspired. The idea of an artificial skin is, obviously, inspired similarly. Of course, one could argue, that a robot's working environment is usually human-made. A human-made environment could provide advantages to human-mimicry approaches. However, this does not necessarily mean, that a pure human-mimetic approach will lead to the final solution. One could also argue, that today's airplanes are not a pure copy of birds, as we discussed already. Even though, a bird's environment has not been developed for the bird, the birds are specialists in flying *in* their environment. Hence, copying a bird might sound like a promising direction towards "flying humans". However, we have seen in the history of aviation, a combination of both types of approaches has lead to the long-term success in aviation.

Possible similarities between humans and humanoid robots are often analyzed in psychological studies as shown in Section 1.3.4. At the same time, results of other studies indicate, that the robot is perceived as a useful tool. However, psychological studies with robots lack from a couple of problems. Further studies analyzing, e.g., human-like and machine-like designed robots with respect to psychological aspects would be necessary. The importance of technology acceptance can also be seen in aviation. Despite today's success in aviation, the fear of flying is still a contemporary issue. This fear seems to be due to many reasons (see Section 1.3.5.3).

Another interesting issue is the question, when a problem is solved. This depends on the definition of the problem itself and the definition of the state, when the defined problem is solved. In the history of aviation, Clément Ader was the first one who claimed, that he has solved the problem of motor-aviation. His steam-powered Éole was able to manage a 50 m-long hopper. This is, of course, not a long-term, steered flight as we would define a flight today. Nevertheless, his steam-powered machine lifted off from the ground. Hence, we can see, that the definitions of problems and corresponding descriptions, when the problems are solved, are important. Only such definitions allow to make a decision, whether a problem is solved, appropriately. The question, how "grasping" is defined and when it is solved, was also discussed in [160], [161]. As the overview in robotic grasping and dexterous manipulation in Section 1.3.1-1.3.4 shows, there are problems which are "solved" in that sense, that successful demonstrations have been presented. For example, the DLR hand arm system [117] was designed to reach its human archetype regarding robustness, dynamic performance and dexterity. Robotic power grasping in unstructured environments was demonstrated in [121]. However, if we define the entire grasping and manipulation capabilities and abilities (including independence of humans and knowledge about, e.g., physical and handling properties of unknown objects) in any environment (including the capability to interact with any human) as benchmark, a successful demonstration is still missing.

This leads automatically to the question of missing capabilities and abilities. For example, different mechanical designs of robotic hands exist, but sensing methods as well as advanced sensor data processing methods are limited. Such a feedback from the environment can be useful in many situations, like in human-robot interactions or during a sensitive interaction with unknown objects. Currently, a lot of work exists in the field of vision-based perception. Advanced haptic sensing capabilities of the hand and sophisticated sensor information usage (e.g., artificial sensitive skin on an arm in [138]) could be an example for enhanced sensing methods. How up-to-date the topic is, can be seen in currently ongoing and future work. For example, a workshop "Research frontiers in electronic skin technology: Multi-functional bendable and stretchable electronic skin for robots and beyond" [162] was just recently organized at ICRA 2013. In future work at DLR [139], this missing "feel of touch" was mentioned. The limitations in sensing methods could be compared to the limited number of motor types in aviation at Cayley's time. The only available motors were steam-engine motors.
A further important, currently very limited component in robotic grasping and manipulation is a general independence of human teachers. If cumbersome and time-consuming demonstration processes are necessary for a daily use in the future, users will be annoyed and the deployment of robots could be limited significantly. In order to achieve a further independence in robotic grasping and manipulation applications, progress is required with respect to several issues. For example, knowledge about physical and handling properties of unknown objects is an important issue. Another example are the often limited or structured environments in the applications. Applications working in any environment are desirable.

To sum up, the overview shows many successful approaches for a lot of aspects of the overall problem. However, there are still missing or incomplete components like limited sensing capabilities or a further independence of human teachers in unstructured environments. The entire grasping and manipulation capabilities and abilities (including independence of humans and knowledge about, e.g., physical and handling properties of unknown objects) in any environment (including the capability to interact with any human) have not been achieved yet by a robotic system.

The discussion the entire Section show, that bio-mimicry, bio-inspiration and technical approaches made and make important contributions. This might sound nearby. However, the comparison to another field of engineering shows, that there were similar experiences. Moreover, both types do not necessarily exclude each other as, e.g., the current development in robotic grasping and dexterous manipulation [163] shows. To conclude, both, research focusing on a technical perspective and research on anthropomorphic approaches, should be considered in general.

The components of this thesis contribute to the approaches as follows.
The proposed knowledge extraction and representation is *object*-centric. Hence, requirements set by the handled objects are in the focus. Similarly, the representation of dexterous manipulations provides object-centric knowledge in a the contact-state perspective . Both representations

contribute towards autonomous systems which acquire knowledge efficiently and which allow a reusage of the knowledge in similar situations (Chapter 3).
The path optimization approach and the manipulator structure analysis (Chapters 4 and 5) focus on the *robot* itself. They are built up in a manner, such that the robot's capabilities are optimized, resp., analyzed with respect to its own control rather than copying the movements of a human demonstrator. Such a check of the task-specific suitability of a robotic system has not been presented yet.
A *human*-centric approach is presented in the additional application of the analysis on master-slave systems (Chapter 6). There, the convenient system usage for the human is in the focus.

1.4 Overview of the Contributions

This section sums up the contributions of the thesis. We focus on the contributions in the approach. Of course, the related work and the experiments are presented for all areas. The context of our work are grasping and manipulations. Hence, the current state of the art in grasping and dexterous manipulation with respect to bio-mimicry, bio-inspiration and technical approaches is additionally summarized and discussed. It shows not only the current state of the art and open problems, but it illustrates also, that bio-mimicry and bio-inspiration are not necessarily the best and only way to develop technical systems.

Estimation of Inverse Kinematics of Arbitrary Serial Chain Manipulators and Human-Like Robotic Hands We provide very general tools for the global estimation of inverse kinematics

1. at single points for arbitrary serial chain manipulators,
2. along entire paths with consecutive configurations for arbitrary serial chain manipulators, and
3. of a complex human-like robotic hand.

Manipulation-Relevant Knowledge Representation The abstract task representation is presented. It allows the desired analysis in the context of abstractly represented tasks. At first, we provide a representation for manipulation-relevant object properties and actions. The framework allows

1. to estimate manipulation-relevant properties,
2. to distinguish between background and more important foreground information in the scene,
3. to represent the characteristics of the manipulation in a separated long-term and short-term memory,
4. to split the current knowledge into object-related properties and object functionalities in the environment, and
5. to detect new information efficiently at an abstract level.

A representation of the environment for dexterous manipulations follows. The representation is chosen in a manner, such that

1. it extends the representation for manipulation-relevant object properties and actions with respect to a detailed information representation for dexterous manipulations,

2. it is clearly structured to enable an easy usage,
3. it is reusable even in changing environments, and
4. path alternatives can be build up if feasible.

Path Optimization for Abstractly Represented Tasks with Respect to Efficient Actuation The concept of the Elastic Power Path is explained in Chapter 4. It allows to optimize paths

1. with respect to efficient actuation
2. in task-specific contexts
3. independently of the manipulator's structure.

Analysis of Manipulator Structures with Respect to Efficient Actuation in Task-Specific Contexts The analysis of manipulator structures with respect to efficient actuation in task-specific contexts is presented, afterward. We provide an inter- and intra-robot comparison

1. applicable to manipulators of any structure,
2. with respect to efficient actuation,
3. in task-specific contexts,
4. under the consideration of joint failures
5. including the potential of path optimization.

Analysis of Master-Slave Systems with Respect to Efficient Actuation in Task-Specific Contexts At the end, we provide a basic simulation in the context of the analysis of arbitrary telemanipulation systems. The previously developed methods are adapted appropriately to telemanipulation systems.

1.5 Outline

The thesis is structured as follows:

The required estimation of inverse kinematics of arbitrary serial chain manipulators is presented in Chapter 2. The estimation is a technical requirement to allow the structure analysis. Afterward, the representation of manipulation-relevant knowledge is introduced in Chapter 3. The estimation and representation of manipulation-relevant object properties and actions are described in Section 3.1. The representation of dexterous manipulations follows in Section 3.2. It is based on a contact state description of object relations in the environment. The path optimization method is proposed in Chapter 4. The manipulator structure analysis is presented in Chapter 5. It includes an analysis under path optimization. The analysis of master-slave systems is described in Chapter 6.

The experiments are presented in Chapter 7. The validation scenarios, the implementation and the results of all experiments are shown there.

The thesis ends with conclusions and future work. In the attachment, further information about the used vision data, parameter descriptions, an explanation of the mathematical notation as well as a list of abbreviations are available.

Chapter 2

Estimation of Inverse Kinematics of Arbitrary Serial Chain Manipulators and Human-Like Robotic Hands

The analysis of manipulator structures requires an advanced estimation technique of inverse kinematics. It has to be applicable to any arbitrary serial chain manipulator without circuitous modifications. Moreover, a *global* estimation is required to ensure, that the best solutions can be found. Hence, we provide a general estimation of the inverse kinematics of serial chain manipulators.

Usually, a robot is supposed to reach one (or several) certain position(s) with its end-effector. Then, the robot needs to be moved to an appropriate configuration to place the end-effector in the desired position. In many situations, it is advantageous to have a system which takes the desired end-effector position as input and gives one, several or all possible goal configurations as output. The necessary mapping from the robot's workspace to its joint space is called inverse kinematics (see exemplary illustration in Fig. 2.1). The general solution of inverse kinematics is a known problem for a long time. Sometimes, the inverse kinematics can be computed explicitly for manipulators with certain structures. A general estimation framework for the inverse kinematics of an arbitrary serial robot is still missing. It can already be enough to know, whether a solution exits. In other cases, the knowledge about one, several or all solutions is desirable.

We focus on the *discrete* analysis of the configuration-space. Hence, we provide a punctual analysis. Such an analysis is, e.g., necessary for the evaluation of an arbitrary manipulator structure. We are interested in the inverse kinematics of an *arbitrary serial chain manipulator*

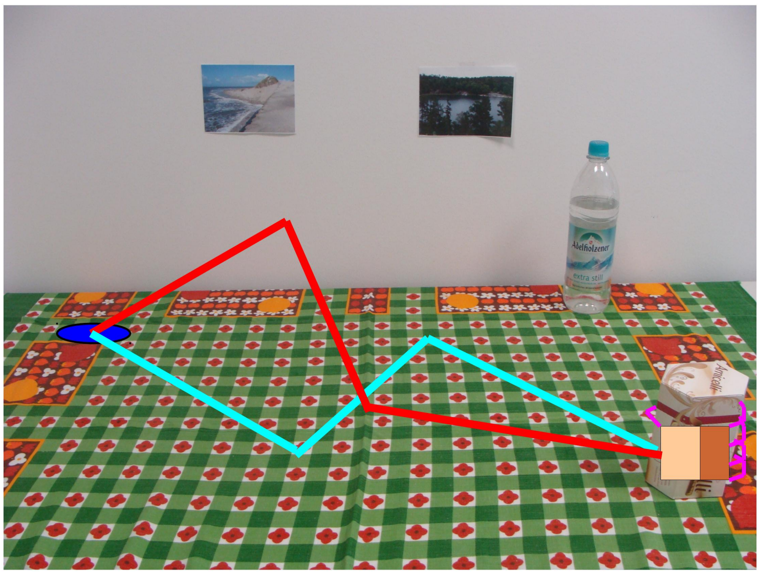


Figure 2.1: Inverse kinematics in a manipulation scenario [6]. The figure illustrates the problem of inverse kinematics for a manipulation scenario: The box should be transported. This means, first, that the object has to be grasped by the robotic hand (brown palm, magenta fingers). A serial chain manipulator has to put the hand into an appropriate position, such that a successful grasp is possible. Several configurations of the manipulator could be possible to reach a point (e.g., blue and red configurations of the robot). Then, the manipulator has to reach several desired positions along a trajectory for the transportation (not depicted).

at a given point without any incremental methods based on intermediate positions or (partial) derivatives (e.g., through the application of Jacobians). We do not assume any advantageous start configurations or something similar. Consequently, we formulate the problem of inverse kinematics as optimization problem. We look for configurations of the robot which minimize the distance between the robot's end-effector and its desired position (3D: position; 6D: pose/ position with orientation). As for other optimization problems, local minima can occur. Of course, we need to be able to overcome these, since we want to know, whether it is possible to reach the desired positions. Furthermore, we want to be able to find several different solutions without getting stuck in local minima. The solutions could be separated through one or several large maxima. This means, that we need to formulate our problem as a *global* optimization problem. Therefore, the entire configuration space is considered for the optimization. Such a global approach does not exist yet for an arbitrary robot. For example, the widely known Jacobian method depends on the start configuration of the robot. Hence, it can also suffer from local minima. The desired global optimizer has to run within a short computation time. We apply the stochastic approach for global minimization presented in [26], since it meets the above requirements. The optimization in [26] was developed in another context. Therefore, we need to develop a cost function in order to (1) check if a solution exists and to (2) find several solutions if there are more present. Hence, we cannot only minimize the distance between the desired and the real position of the end-effector in our cost function. If we have found one solution, we need to be able to find further solutions. Moreover, if no solutions exists, it is desirable to estimate at least a configuration which moves the end-effector as close as possible to the desired position.

Furthermore, we want to determine a convenient solution of the inverse kinematics along an entire path, e.g., for the manipulation of an object. Of course, it is desirable to achieve consecutive configurations which are close to each other. Therefore, we introduce the concept of *adaptive tunneling*. Up to now, we formulated our optimization problem in a manner, such that we find one or more solutions for a single point. We are looking for these solutions in the entire search space, e.g., from 0 to 360 degree for each joint. Now, each configuration should be close to its ancestor. Hence, we limit the search space to a region around the configuration of the ancestor. A small search space is desirable to achieve consecutive configurations which are really close to each other. At the same time, a small space reduces the chance to find a good solution to attain the goal. Therefore, the size of the search space has to be adapted appropriately to ensure consecutive configurations close to each other, while the desired points are reached. Our concept of adaptive tunneling provides a framework to achieve this goal.

Of course, it is desirable to estimate also the inverse kinematics of more complex systems, while keeping the run time very small. However, the more DoF have to be solved by the optimizer, the longer it will take. An example is a human-like hand with 20 DoF, which should grasp an object. If the desired positions of the finger tips and the thumb's tip are known, the position and the orientation of the hand as well as its 20 DoF have to be determined in such a manner, that the five tips can reach the goal positions. In order to deal with this complex problem, we build on the concept of the "virtual finger" [19], which is a combination of real fingers. An object can be grasped through a shut of the object between a thumb and a virtual finger. We call such a grasp a *virtual shut grasp*. The application of the virtual shut grasp reduces the number of DoF significantly from all DoF of the hand (e.g. 20) to a small number of DoF in form of a serial chain (7 DoF in our example, which is shown later).

To sum up, we provide very general tools for the global estimation of inverse kinematics (1) at single points for arbitrary serial chain manipulators, (2) along entire paths with consecutive configurations for arbitrary serial chain manipulators, and (3) of a complex human-like robotic hand.

2.1 Description of the used variables and parameters

In general, appropriate joint angles θ_j have to be determined for all joints j in the joint space in such a manner, that the robot's end-effector is able to reach a desired position $\boldsymbol{p}_d$ in the workspace. This is called inverse kinematics. The mapping from the robot's joint space into the workspace can be described by a function F:

$$\boldsymbol{p}_d = \mathrm{F}(\boldsymbol{\Theta}) \tag{2.1}$$

with $\boldsymbol{\Theta} = (\theta_1, ..., \theta_D)$.

We describe a robotic system in the DH-convention suggested by Denavit and Hartenberg [164] in the form shown in [9]. The orientation $\boldsymbol{o}$ of the robot's end-effector is described as Z-Y-X Euler angles [9]. More detailed information is provided in Appendix B.

The robot is supposed to reach a desired point $\boldsymbol{p}_d$. The currently reached, real point is labeled with $\boldsymbol{p}_r$. The euclidean distance e describes the remaining distance between $\boldsymbol{p}_d$ and $\boldsymbol{p}_r$. A desired position is reached, if e is smaller than a certain tolerance t. A trajectory consists of N desired points $\boldsymbol{p}_i$ including the start point and end point. The configuration $\boldsymbol{c}_i = (\theta_1, ...\theta_D)$ for all joints j of the robot corresponds to point $\boldsymbol{p}_i$ in the trajectory. The euclidean distance between two configurations is labeled with $e_{\boldsymbol{c}}$. The total number of joints is D.

2.2 Inverse kinematics for a single point

In order to estimate the inverse kinematics for a single point, we want to minimize the distance between the real and the desired position. Therefore, we search for the minimum in the following objective function O_k:

$$\mathrm{O}_k = \|\boldsymbol{p}_r - \boldsymbol{p}_d\| + r \tag{2.2}$$

with residual r. The residual r is zero for the search of the first solution of O_k. If more solutions should be found for the same point $\boldsymbol{p}_d$, we need to search for further, different minima of O_k. Configurations close to already determined configurations should be avoided. Therefore, r is increased, when a new configuration is close to a previous, already known one. The minimal distance, that is necessary to declare two configurations as *different*, is labeled with u. If the euclidean distance $e_{\boldsymbol{c}}$ of the joint configurations is smaller than the distance u, the residual r is increased. r is, then, increased for each joint j in the configuration, if its distance $e_{\boldsymbol{c},j}$ is smaller than u/D:

$$r := r + \gamma \cdot |e_{\boldsymbol{c},j} - u/D| \tag{2.3}$$

with γ as a scaling factor. The higher the scaling factor, the higher the penalization of known configurations.

If we want to determine the 3D position of the end-effector as well as its orientation, we extend the previous objective function O_k to the following objective function O_o:

$$\mathrm{O}_o = \|\boldsymbol{p}_r - \boldsymbol{p}_d\| + \alpha \cdot \|\boldsymbol{o}_r - \boldsymbol{o}_d\| + r \tag{2.4}$$

with $\boldsymbol{o}_d$ as desired orientation of the end-effector and $\boldsymbol{o}_r$ as the current orientation of the end-effector. The factor α is used as scaling, since the terms have different ranges. The settings of the parameters described in this Chapter are discussed in detail in the experiments (Section 7.3).

If we want to transport an object, e.g., a cup filled with coffee, we are interested in keeping the object upright. However, the orientation around the vertical axis of the object is allowed to change. Then, the objective function O_o needs just to be adapted slightly. Instead of including the orientations around all axes, we just use the orientations around the horizontal axes of the object in O_o. All described axes in this paragraph are, of course, in an object-centric point of view.

It should be pointed out, that the formulation of the objective functions is independent of the number of DoF of the manipulator. For example, the inverse kinematics of a redundant manipulator could be estimated as well.

If the result of the first estimation of the objective function is higher than allowed, no solution is found. In this case, we get automatically the robot configuration closest to the goal, since we search for the global minimum of the objective function.

2.3 Adaptive tunneling

As already motivated, we want to estimate the inverse kinematics of an arbitrary serial chain manipulator along a trajectory. Of course, it is desirable to get robot configurations which are close to each other along the trajectory. This allows to achieve smooth motions. Hence, we do not need to evaluate the entire search space. We just change the search space $\mathcal{S}$ of the optimizer. At the beginning, we use the entire search space ranging from 0 to 360 degree for each joint to determine the robot's configuration at the start point of the trajectory. For each following point $\boldsymbol{p}_i$ with $i > 0$, the search space is limited to a smaller search space $\mathcal{S}'_j$:

$$\mathcal{S}'_j = [\theta_{j,i-1} - l_j ; \theta_{j,i-1} + l_j] \tag{2.5}$$

for each joint j. $\theta_{j,i-1}$ is the configuration of joint j at the previous point $\boldsymbol{p}_{i-1}$. The variable l_j limits the search space around the previous configuration for joint j. At first, l_j is set to a small constant l_c. If the manipulator is able to reach the desired position, we are done. Otherwise, the limit l_j is adapted:

$$l_j := l_j \pm \delta \tag{2.6}$$

with

$$\delta = m \cdot |(e - t)/D| \text{ and } |e| > t \tag{2.7}$$

The sign in Eq. 2.6 is positive for the upper bound and negative for the lower bound. The adaption factor δ (Eq. 2.7) depends on the distance e between the desired and the real position. Hence, if the goal position is far away from the currently reachable positions, the search space is extended by a large step to overcome the gap to the goal. In the case of an adaption, $|e| > t$ must hold. Each l_j is, then, increased by the (proportionate, scaled) difference between the current distance e and the tolerance t. The distance $e-t$ is divided by D, since l_j limits the joint space for one joint j. The scaling m determines the step size of the adaption. The smaller m,

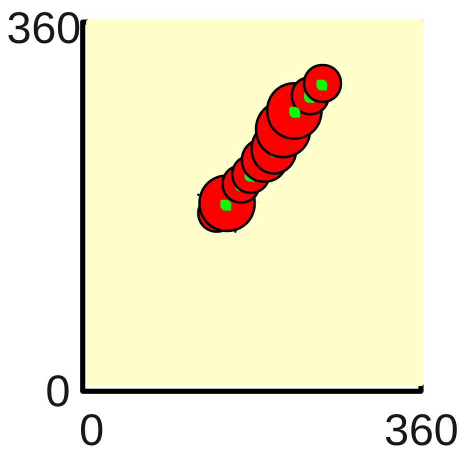


Figure 2.2: Adaptive tunneling [6]. The figure shows a tunnel in a 2D search space for a 2 DoF manipulator (overall range: $0°$-$360°$ for each joint). The green dots are consecutive configurations. The red circles show the (possibly adapted) search space around the each configuration. Each consecutive configuration is searched within the red circle of its ancestor.

the closer the consecutive configurations can be. However, the run time can increase at the same time, since more iterations can be necessary to reach the desired points. The process of adaption can be repeated until either a solution is found or if $\mathcal{S} = \mathcal{S}'$ without any solution. The objective functions O_k and O_o themselves (see Section 2.2) are not changed. We just modify the search space.

This concept is independent of (partial) derivatives. Moreover, it is able to find solutions, even if the step size between consecutive points is large.

If we concatenate the search spaces along the trajectory, we can illustrate them as a tunnel. The diameter of the tunnel depends on the size of the search space. The concept of adaptive tunneling is illustrated in Fig. 2.2. It supports consecutive manipulator configurations close to each other, while avoiding extensive computation times.

2.4 Virtual shut grasp

A virtual shut grasp is built up as follows. We construct a “virtual finger” [19]. A virtual finger is a combination of real fingers. In our concept, it is the mean of all real fingers without the thumb (mean of, resp., base, link length, orientation). We concatenate the real thumb and the virtual finger to a serial chain (from the tip of the thumb over the hand carpus to the tip of the virtual finger). An object can be grasped through a shut of the object between a thumb and a virtual finger. We call such a grasp a *virtual shut grasp*. Now, we attach the tip of the thumb to its 3D goal position. Then, we just need to determine the thumb’s orientation and

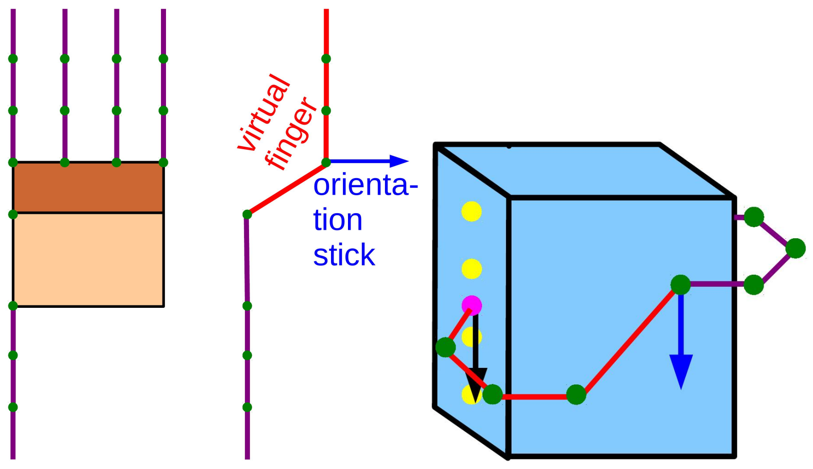


Figure 2.3: Human-like robotic hand and virtual shut grasp [6]. *Left:* Illustration of the human-like robotic hand with the brown palm of the hand, the dark magenta fingers and the green joints. *Middle:* Model of the human-like robotic hand with the red virtual finger and the blue orientation stick. *Right:* Illustration of a grasp with the virtual finger and the orientation stick: The yellow circles are the desired positions of the real fingers. The virtual finger should reach the light magenta mean of these positions. The blue orientation stick should be kept in the right orientation: It should be parallel to the black aim stick and point into the same direction.

the small number of DoF in the serial chain, such that the tip of the virtual finger reaches its desired position (the mean of the goal positions of all fingers).

Additionally, we need to make sure, that the serial chain has the right orientation. It has to be oriented in a such manner, that the real fingers are able to reach their goal positions afterward. Hence, we attach an "orientation stick" to the root $\boldsymbol{p}_{v,root}$ of the virtual finger. The stick is aligned with the joint axis of the root of the virtual finger and points towards the root $\boldsymbol{p}_{l,root}$ of the finger, which is the last one or defined as the last one. We use the little finger as the last finger. The orientation stick can be formulated as the vector $\overrightarrow{\boldsymbol{p}_{v,root}\boldsymbol{p}_{l,root}}$. This stick has to be parallel to an "aim stick" and it has to point in the same direction. The desired aim stick is the vector from the goal position of the virtual finger to a point $\boldsymbol{p}_{g,goal}$ ($\overrightarrow{\boldsymbol{p}_{v,goal}\boldsymbol{p}_{g,goal}}$). To construct the point $\boldsymbol{p}_{g,goal}$, we imagine a plane $\boldsymbol{p}_{vFP}$ within which the tip of the virtual finger can move. If a goal position $\boldsymbol{p}_{f,goal}$ of a real finger is on the same side of the plane $\boldsymbol{p}_{vFP}$ as the goal position of the last finger $\boldsymbol{p}_{l,goal}$, we introduce a new point $\boldsymbol{p}'_{f,goal} = \boldsymbol{p}_{f,goal}$. Otherwise, we mirror $\boldsymbol{p}_{f,goal}$ on $\boldsymbol{p}_{vFP}$ to create $\boldsymbol{p}'_{f,goal}$. The mean of all $\boldsymbol{p}'_{f,goal}$ and $\boldsymbol{p}_{l,goal}$ is the desired point $\boldsymbol{p}_{g,goal}$, which is used for the aim stick. Fig. 2.3 illustrates the principle of the virtual shut grasp.

The determination of the virtual shut grasp gives us the orientation of the thumb as well as the position and orientation of the hand carpus. Afterward, just the joints of the real fingers need to be estimated (independently of each other). It is important to point out, that our approach is independent of the number of fingers and the form of the hand.

A robotic hand can be attached to the end-effector of a robotic serial chain manipulator. If we want to use a hand and a manipulator for an action, e.g., object transportation, we can go on as follows. First, we determine the joint parameters as well as the orientation and position of the hand to grasp the object. Afterward, we just need to deal with the serial chain manipulator. We treat the hand carpus as the manipulator's end-effector, which has, first, to position the hand for the grasp and, second, to keep the object upright during the transportation. Its inverse kinematics can be estimated as described in Section 2.2 and 2.3.

Chapter 3

Manipulation-Relevant Knowledge Representation

The desired analysis is processed in the context of abstractly represented tasks. Hence, an appropriate representation is required. We provide such a representation for manipulation-relevant object properties and actions as well as a representation for dexterous manipulations. The first representation consists of a-priori knowledge about objects *(Atlas)* and current knowledge about the scene *(Working Memory)*. There, we distinguish between properties, which are directly related to the object (*Object Container*), and typical action areas affiliated with the environment (*Functionality Map*). The work is published in [1], [2]. The second representation uses a contact state based perspective to represent dexterous manipulations. It is based on *object relations* in the environment. The approach and the corresponding experiments are published in [7].

3.1 Estimation and Representation of Manipulation - Relevant Object Properties, Actions and Functionalities

The first representation deals with manipulation-relevant properties, actions and functionalities. The proposed framework allows a sensor-based estimation of the desired information. It is extracted from observations of human actions in natural scenes. An important aspect is the abstraction of observations. The resulting characteristic properties of the observed objects are stored in a model which allows a reusage or transfer of the knowledge. The representation of the properties and actions is split into an a-priori knowledge *(Atlas)* and current knowledge about the scene *(Working Memory)*. The Atlas contains already inferred properties of the objects, as well as a-priori knowledge about the object classes and the corresponding handling properties. The a-priori knowledge contains information, which cannot be observed by a vision-based system or by the analysis of an object's trajectory. The handling properties can differ for the same object depending on the context. These alternatives are stored in the Atlas. The general knowledge of the Atlas can be mapped into the current context. The mapped information is represented in the Working Memory, in order to relate it to the current scene. The scene itself is classified into relevant objects (*foreground*) and *background* geometry which is only important for collision avoidance. The human operator identifies the relevant objects through a direct interaction with them (manipulation). Hence, the system does not need to identify and to learn about all objects in the scene but only about the objects that are used by the human (Fig. 3.1).

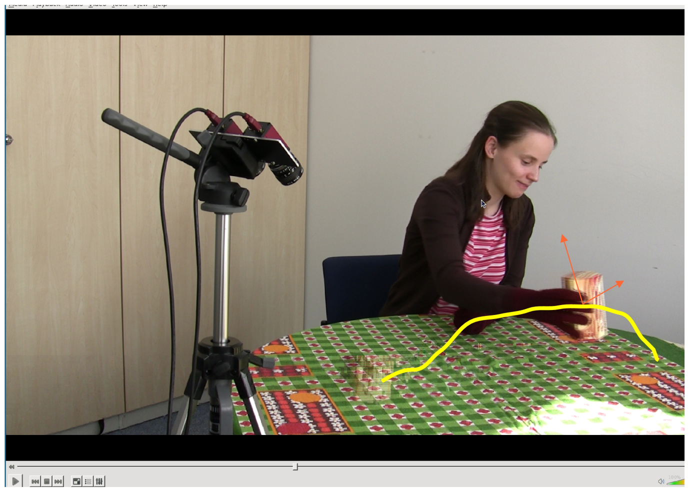


Figure 3.1: Knowledge extraction from human observation [1]. The system observes human actions and completes the internal knowledge representation for the observed objects.

A very important aspect is, that not only the object itself (e.g., its properties or physical states) is defining the way, how it is manipulated, but also the location at which the manipulation is performed. Certain actions takes usually place at specific locations, which have certain properties. For example, washing the dishes is usually done in the sink and not on a flat table without any water source around. The conclusion is, that we need not only a collection of object properties, but also a map, which links environment locations to the specific way how objects are handled at that locations (see Fig. 3.2). It is important to notice, that we are not interested in the exact registration of the actions to the environment in the sense of navigation, but in an abstract representation of the functionalities in the environment. We are not using any semantic information about the environment. Furthermore, the system does not rely on any linguistic information.

The representation of the knowledge is split into an object-centric representation, reflecting the physical properties of an object (*Object Container*), and a *Functionality Map* representing possible actions related to the environment. While the Object Container is linked only to the

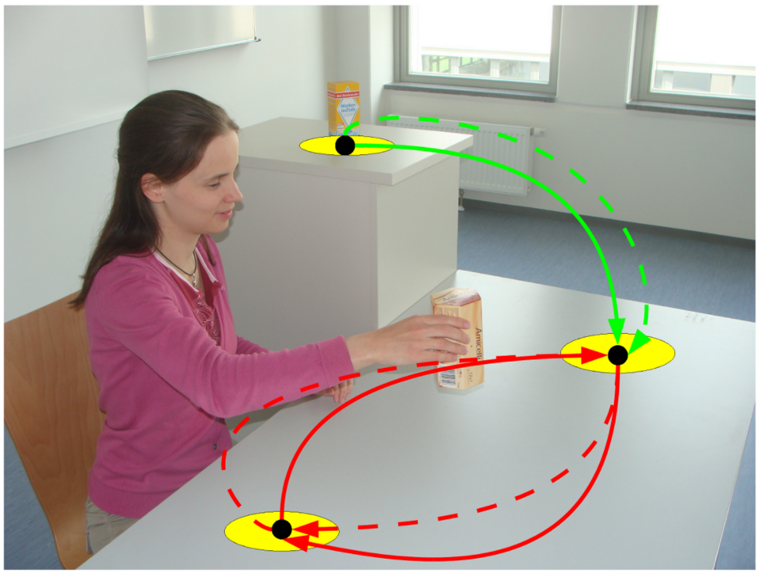


Figure 3.2: Creation of an abstract map of possible manipulation actions and goals [2]. The system creates an abstract map of possible manipulation actions (symbolized by dashed/ continuous arrows in different colors) and characteristic areas (symbolized in yellow) in the environment.

object, the Functionality Map is anchored to the geometric model of the environment. This framework allows us to detect changes in the physical state or the function of an object. Hence, the system is insensitive to variations in the execution of the same actions.

Of course, it is desirable, that the robot is able to learn unsupervised through the observation of human actions. Unfortunately, humans do not follow exact trajectories, while performing repetitive manipulation tasks. The system needs to focus on characteristic information, which is necessary to accomplish a manipulation or to cooperate with a human in a given environment. Here, we can make use of the abstract representation of manipulation actions. At this abstract level, the system can compare its expectation as an observer system with a current human action. A mismatch occurs only in situations, in which the change appears to be a result of a change at an abstract level. We will call such a mismatch a *surprise* event in the following. A surprise event requires the modification of the stored information. Just in the case of a surprise, really new information is detected. For example, motion constraints can change (e.g., a cup carried always upright is now tilted arbitrarily). Another example is an object which is suddenly placed on an unexpected place (e.g., a cup on the floor). These observations are usually an indication, that the physical properties of the object (e.g., the level of the liquid in the object) or their function (not a drinking cup, but a dirty dish) changed. This refers to a detection of really new information which requires an update of the knowledge.

Predictions about the current state or function of an object are based on the information stored in the Object Container or in the Functionality Map. Therefore, mismatches between these predictions and observations (surprise events) occur just at an abstract level. They signal the *right moment* to update the stored information.

It is important to consider, that the robot might face different types of input like, e.g., vision data or data recorded with an external tracking system. Here, trajectories of the human motions are neither used for workflow applications nor for planning of actions. We focus on the object-centric constraints and the object's function in the environment. Our system does not rely on a trajectory in a certain representation, like x,y,z-coordinates, or on a certain colored pattern of an object. The properties acquired in our system have to be sufficient to represent a manipulation task appropriately. At the same time, they have to be generic and extractable from different sources.

To sum up, we provide a framework which allows (1) to estimate manipulation-relevant properties, (2) to distinguish between background and more important foreground information in the scene, (3) to represent the characteristics of the manipulation in a separated long-term and short-term memory, (4) to split the current knowledge into object-related properties and object functionalities in the environment, and (5) to detect new information efficiently at an abstract level.

3.1.1 Estimation and Representation of Manipulation - Relevant Object Properties and Actions from Human Observation

The estimation and representation of manipulation-relevant object properties and actions from human observations is presented in this section.

A robot should be able to acquire manipulation-relevant knowledge as independent as possible. It needs to be able to detect and to extract relevant information. Moreover, it has to process the information in a such manner, that it can be reused. An important ability is the distinction of *relevant* and *new* knowledge from varying, but similar demonstrations. The robot needs to deduce the characteristics of the observation, instead of storing the exact x,y,z-trajectory. The exact trajectory could be used for a pure repetition of the observation. In contrast, the system should determine the relevant knowledge from varying human demonstrations.

The underlying model has to be general. Hence, we introduce the *Atlas* and the *Working Memory* to store the knowledge. The Atlas contains already inferred properties of the objects, as well as a-priori knowledge about the object classes and the corresponding handling properties. The a-priori knowledge contains information, which cannot be observed by a vision-based system or extracted from an object's trajectory. The information about the object in the current scene is stored in the Working Memory, which is the Short-term memory.

It is important to consider, that not only the object itself (e.g., its physical state) is defining the properties of the manipulation, but also the environment. Different locations in the environment are used for different actions, which have certain properties. For example, washing the dishes is normally done in the sink and not on the flat table without any water source in the neighborhood. The conclusion is, that not only a collection of object properties (*Object Container*) is needed, but also a map of the environment providing the functionalities in this environment (*Functionality Map*, see, e.g., Fig. 3.2). It is important to notice, that the exact reconstruction of the environment in the sense of navigation is not of interest here, but an abstract representation of the functionalities in the environment.

An important property of the proposed system is the general and object-centric representation. The combination of the two involved representations, the Object Container and the Functionality Map, achieves the following advantages.
The acquired knowledge is not only reusable in the situation in which the observation took place. A knowledge transfer to similar situations is possible. The similarity consists of two parts: First, the observed objects have to be similar to the known objects in the Object Container. It is important to consider, that the system does not rely on, e.g., a certain colored pattern of an object. It works on manipulation-relevant properties of the objects. The second part of the similarity are the characteristic locations, where the manipulation can start or end. These areas have to be similar to known locations in the Functionality Map. If the current situation is similar to the known one, the conversant properties can be applied on manipulations.
A further advantage of the proposed system is the efficient detection of new knowledge. New knowledge is achieved through the observation of unknown actions or unknown objects in the current environment (*surprise detection*). In order to detect such actions or objects, the expected situation and the current situation are compared. A mismatch refers to an unknown event, which provides new information. Such a surprise detection occurs just for a change of properties at an abstract level, since the system consists of general and object-centric knowledge. The number of false positive alerts, caused by varying user demonstrations, can be reduced to allow an efficient surprise detection.
Moreover, the robot might face different sources of information. One source is, of course, the observation of humans with, e.g., a vision system, what can be useful in a household. However, there are also other sources of knowledge, like a data base, which provides knowledge in another form (e.g., the exact trajectories of actions in a chemical laboratory). The proposed system does not rely on a trajectory in a certain representation like x,y,z-coordinates. It is built on properties of the object and its functionalities. The underlying properties have to be powerful enough to provide information for a manipulation. At the same time, they have to be general and extractable from different sources.
Additionally, the knowledge extraction has to be possible through the observation of a small number of user demonstrations. A large amount of demonstrations has several disadvantages, like time-consumption and annoyed users.
To sum up, the described requirements lead to a further understanding of the actions in the environment in contrast to a pure repetition of actions.

The aim of this part of the work is the representation of manipulation-relevant knowledge in a manner, such that the described requirements can be met.

3.1.1.1 Focus of Attention

A robot's environment can be very complex. It can, e.g., contain a lot of objects in rooms with different furniture. Moreover, the robot has to deal with demonstrations of humans in these complex scenes. Such a complex scene requires to focus the attention on the relevant information in the scene. Here, manipulated objects are relevant to the robot, since it could observe new information. Hence, the robot needs to detect and observe the object manipulated by the human. This mission-relevant object is the foreground. In contrast, the remaining geometric structure is the background used for obstacle avoidance. The selection of the foreground object and its monitoring is triggered by the human's interaction with it. This concept has an analogy to the Vision Interaction Cues (VICs) approach [165]: For each object, its actions and its monitoring space around itself are defined by the object. The processing of the human actions can be speed up this way.

Fig. 3.3 illustrates in the entire concept. It shows the Atlas as well as the Working Memory with the fore- and background.

3.1.1.2 Determination of the Object Candidates

The mentioned mission-relevant object is placed somewhere in the scene. It has to be determined. A simple idea to detect the mission-relevant object is the observation of the entire scene, until a change occurs (e.g., movements of an object or a human). Such an approach is not efficient. Changes could be detected, which are not of interest (false positives; e.g., changed lighting conditions). Therefore, we determine the objects, which can be manipulated by a human (object candidates) in the current scene, first. Then, just these candidates have to be observed.
We apply a plane-subtraction as described in [166], [167]. It makes use of the homography between the (u,v,D) coordinates of the disparity image ([u,v] are the image coordinates and D denotes the disparity at [u,v]) and the corresponding Cartesian coordinates from the 3D scene. In [166], the plane P_r is represented as

$$\mathrm{P}_r : a_r x + b_r y + c_r z = d_r. \tag{3.1}$$

According to [167], the equivalent disparity plane with the normal vector $\boldsymbol{n}_r^*$ is given by

$$\mathrm{D}(u,v) = \begin{pmatrix}\rho_1\\ \rho_2\\ \rho_3\end{pmatrix} \cdot \begin{pmatrix}u\\ v\\ 1\end{pmatrix} = \boldsymbol{n}_r^* \cdot \begin{pmatrix}u\\ v\\ 1\end{pmatrix} \tag{3.2}$$

with the disparity D(u,v) at image coordinates (u,v), $\rho_1 = \frac{a_r}{k}$, $\rho_2 = \frac{b_r}{k}$, $\rho_3 = \frac{c_r}{k}$, $k = \frac{d_r}{B}$ and the baseline B.

In order to estimate the normal vector $\boldsymbol{n}_r^*$ of the plane, an area of low gradients of the same direction (planar candidate) has to be found. The biggest planar candidate is assumed to be a

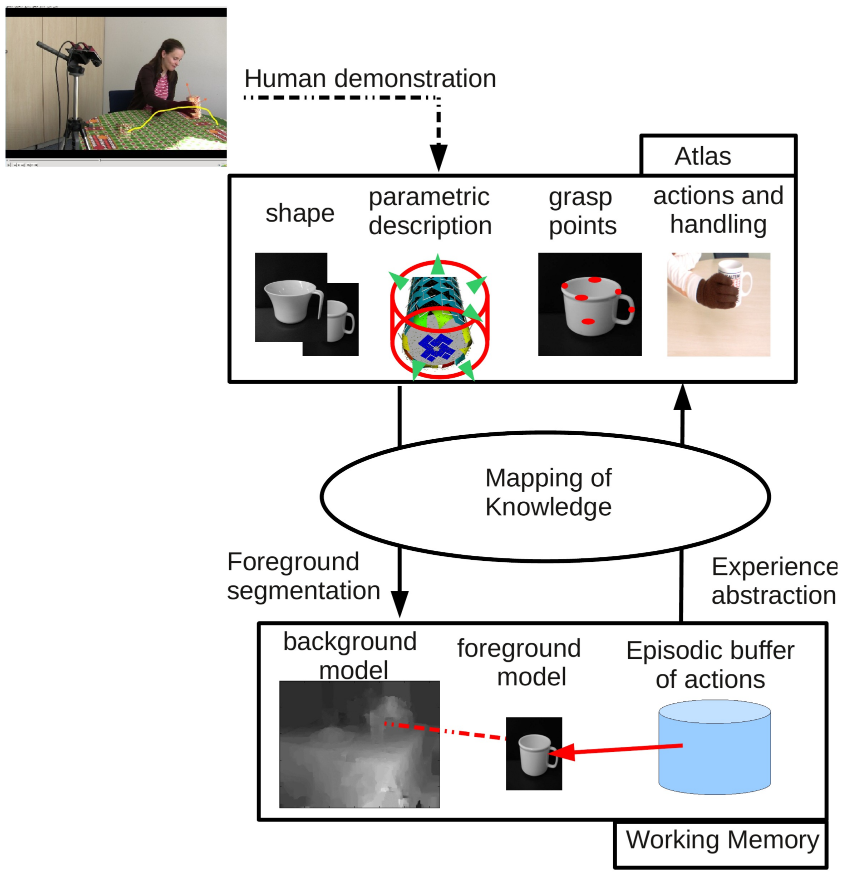


Figure 3.3: Atlas and Working Memory [1]. The figure illustrates the concept of the Atlas and the Working Memory. The general knowledge of the Atlas is mapped into the current scene and stored in the Working Memory.

part of the plane. It is used for the estimation of the normal vector $\boldsymbol{n}_r^*$ of the plane according to (6) in [167]:

$$\begin{pmatrix} \sum u_i \cdot D_i \\ \sum v_i \cdot D_i \\ \sum D_i \end{pmatrix} = \begin{pmatrix} \sum u_i^2 & \sum u_i v_i & \sum u_i \\ \sum u_i v_i & \sum v_i^2 & \sum v_i \\ \sum u_i & \sum v_i & \sum 1 \end{pmatrix} \cdot \boldsymbol{n}_r^* \tag{3.3}$$

All pixel, which belong to the plane, are deleted in the disparity-map. Consequently, the objects placed on the plane (e.g., a table) remain in the disparity-map as object candidates. An example of the plane subtraction is shown in Fig. 3.4.

3.1.1.3 Selection of an Object as Region of Interest

One of the observed object candidates is the mission-relevant object. It is selected by the human, who grasps it. The grasped object forms, now, the Region of Interest (ROI).

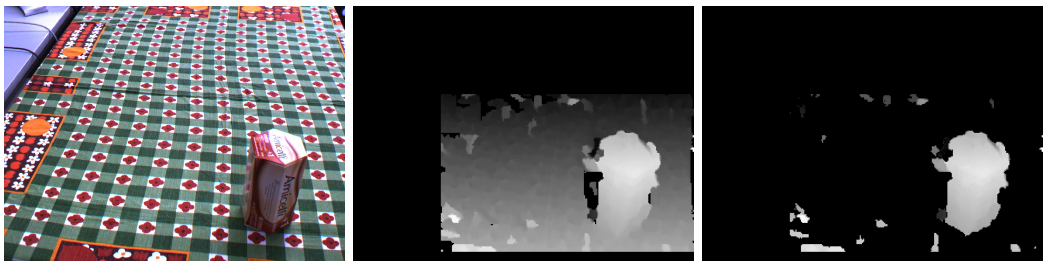


Figure 3.4: Plane subtraction [1]. *Left*: Original color image. *Middle*: Disparity image of the color image on its left. *Right*: Remaining object in the disparity image after the plane subtraction.

In order to perceive the human grasp, the contact between the hand and the object has to be determined. The object candidates have already been found, as described before. The detection of the hand can be done with different methods. A blob-detector is used here. Hence, the hand is determined through the detection of a connected region, which has the characteristic color of the hand. Of course, this method faces the problem, that the color of the hand is not unique. First, the color of human's hands can differ significantly. Therefore, a hand and a possible contact cannot be detected using the color of the hand of *one* human in general. Moreover, the object and/ or the environment can contain blobs of the same color and size. This can lead to false-positive alerts for the contact detection. Similarly, the size of the hand can change in a 2D color image depending on the orientation of the hand and its distance to the camera. An advanced hand detection would, hence, require the inclusion of a further characterizing property of the hand: Its shape. However, such an advanced hand detection is a complex topic on its own (see, e.g., in [168]). The blob-detector can be implemented easily and it allows a simple detection of the hand. Hence, the blob-detector can fulfill its purpose. We use a glove in the manipulations to simplify the detection. The blob-detector is, then, based on significant colors. If several blobs of sufficient size are detected, the largest one is chosen in our case.

In order to determine a contact between the hand and the object, we need to check for a sufficient overlap of both. In principle, a pixel-wise comparison could be performed, but, this method can require extensive computation times. Hence, we introduce an outer bounding box in form of a rectangle R_o around the object. The outer bounding box is slightly reduced, since we want to avoid false positive alerts for a contact caused by, e.g., accidental contacts at the borders of the bounding box. Moreover, the borders of the original bounding box might even be out of the object. Therefore, the original outer bounding box R_o is reduced by b pixel on each side. The resulting rectangle R_b is defined through its four corners r_c with pixel-coordinates u_{r_c}, v_{r_c} (top left: r_1; top right: r_2; bottom left: r_3; bottom right: r_4). Now, we check for all pixel i of the hand blob, whether they are within the four corners r_c. Hence, a contact candidate ($contact_c$) is found, if the following equation holds:

$$contact_c : \exists (u_i, v_i) \in hand\_blob : u_{r_1} <= u_i \ \wedge \ u_i <= u_{r_2} \ \wedge \ v_{r_1} <= v_i \ \wedge \ v_i <= v_{r_3} \qquad (3.4)$$

If a pixel is within the rectangle R_b, we have found a first contact candidate. The object is selected as ROI, as soon as a sufficient number (n_c) of contacts candidates has been seen in a row. This check for a contact candidate is much faster, than a direct pixel-wise comparison, since it just checks the borders of the rectangle R_b. If the blob of the hand consists of H pixel, the proposed method requires $O(n_c \cdot 4 \cdot H)$ checks to find out, whether the hand blob is within the rectangle R_b. Now, we label the number of pixel of the object with H_O. For the pixel-wise comparison, each pixel of the hand needs to be compared with each pixel of the object, resulting in $O(n_c \cdot H_O \cdot H)$ comparisons. Normally, we can assume, that H_O is significantly larger than four. Hence, we would need significantly more checks for the pixel-based comparison.
We further discuss the settings of the parameters described in this Chapter in the experiments (Section 7.1).

3.1.1.4 Object Tracking and Determination of the Object's Type

After the selection of an object as ROI, the object needs to be observed. First, we focus on the actions on the object. We are interested in the changes of the object's position and orientation. Such changes can provide information about characteristic properties of the observed object. Consequently, we need to extract the object's 6D-trajectory (position: $\boldsymbol{p}_o$; orientation: $\boldsymbol{o}_o$).

In principle, any tracking system can be used as long as it provides the 6D-trajectory. We want to focus on two types in this thesis: An external tracking system and a vision system based on features. The first one requires external markers fixed to the object. These markers are tracked, then. However, the external markers plus a setup observing them needs to be integrated in the original scene.
In contrast, the vision system can directly work on features, which are already available in the scene. Of course, appropriate features need to be found, first, and, afterward, tracked in the natural scene. Moreover, it is important to consider, that we are interested in the object's 6D-trajectory. Hence, 3D information is necessary. A 3D tracking can, however, require a long computation time. A tracking in 2D can speed up the processing. Therefore, we use the following procedure: At the time of contact, we store one depth image as well as the corresponding color image. In the color image, we extract 2D-features on the object. For each feature, we know the underlying 3D information, since the depth image is given. The object's 2D-features are tracked in 2D during the manipulation. It is possible to compute the 6D-trajectory of the object based on the recorded trace of the 2D-features and the depth image at the time of contact. V-GPS is applied for this purpose [169].

As the name V-GPS [169] already indicates, the concept has similarities to the GPS system. It uses natural landmarks in the environment for localization. GPS uses satellites for the same purpose. Originally, V-GPS was applied for the estimation of a camera pose based on natural landmarks. The camera was moved, there. In our context, the camera is fixed, while the observed object is moved. Therefore, we use V-GPS in the opposite direction: We estimate the pose of the moved object from a fixed camera system. Visual features on the object are used

for the pose estimation similarly to the natural landmarks in the original concept.
Eq. 1 in [169] shows the projection of a point ${}^{c}\boldsymbol{p}$ (given in the camera frame) into the image plane with coordinates $(u, v)^T$ (assuming a perspective projection [170]):

$$\pi({}^{c}\boldsymbol{p}) = \begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{z} \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.5}$$

Originally, an internal model of the environment was represented by 3D coordinates ${}^{m}\boldsymbol{p}_i$ in [169]. In our case, the object is represented by these points.
The goal of the original concept was the estimation of a transformation matrix ${}^{c}\boldsymbol{X}_m$ in a manner, such that each image point $(u_i, v_i)^T$ corresponds to a model point ${}^{m}\boldsymbol{p}_i$ (Eq. 2 in [169]):

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \pi({}^{c}\boldsymbol{X}_m({}^{m}\boldsymbol{p}_i)) \tag{3.6}$$

The transformation matrix ${}^{c}\boldsymbol{X}_m$ describes the transformation between a known point ${}^{m}\boldsymbol{p}_i$ and its corresponding observed point $\boldsymbol{p}_i^*$. It is defined as

$${}^{c}\boldsymbol{X}_m = \begin{bmatrix} \widetilde{\boldsymbol{R}} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \tag{3.7}$$

with rotation matrix $\widetilde{\boldsymbol{R}}$ and translation vector $\boldsymbol{t}$ (Eq. 12 in [169]). The direction towards the observed point $\boldsymbol{p}_i^*$ is known given the image point $(u_i, v_i)^T$. It can be described by a unit vector $\boldsymbol{n}_i^*$. However, the distance λ_i to $\boldsymbol{p}_i^*$ is unknown. It can be estimated through the initial distance D, resp., the distance in the previous frame (within a sequence of images). To sum up, the observed point $\boldsymbol{p}_i^*$ can be described as follows (Eq. 14 in [169]):

$$\boldsymbol{p}_i^* = \lambda_i \cdot \boldsymbol{n}_i^* = \widetilde{\boldsymbol{R}} \cdot {}^{m}\boldsymbol{p}_i + \boldsymbol{t} \tag{3.8}$$

If at least three pairs of non-collinear points ${}^{m}\boldsymbol{p}_i$ and $\boldsymbol{p}_i^*$ are known, the rotation matrix $\widetilde{\boldsymbol{R}}$ and the translation vector $\boldsymbol{t}$ can be estimated using the following least-squares problem (Eq. 16 in [169]):

$$\min_{\widetilde{R}, \boldsymbol{t}} \sum_{i=1}^{n_O} \|\widetilde{\boldsymbol{R}} \cdot {}^{m}\boldsymbol{p}_i + \boldsymbol{t} - \boldsymbol{p}_i^*\|^2, \text{ subject to } \boldsymbol{R}^T\boldsymbol{R} = I \tag{3.9}$$

The total number of point pairs is n_O. The rotation matrix $\widetilde{\boldsymbol{R}}$ and the translation vector $\boldsymbol{t}$ are estimated online. This allows to use, resp., the distance in the previous frame as an initial estimation of the distance λ_i
The labeling of the variables of the equations above (Eq. 3.5-3.9) differ slightly from their original presentation in [169]. They are adapted to the mathematical notation in the thesis.

In order to be able to apply the above method, the current image-coordinates of the initial features need to be tracked. The online application of V-GPS provides an advantage in this context: It allows the application of V-GPS for a re-initialization of lost features [169]. The rotation matrix $\widetilde{\boldsymbol{R}}$ and the translation vector $\boldsymbol{t}$ are estimated for each frame online. They can be utilized to project a lost feature on its assumed position. The tracking has, then, a much

better chance to find the appropriate correspondence again. However, the tracking (and the subsequent V-GPS application with the re-initialization of lost features) needs to be real-time-capable for this purpose.
Of course, it important to find the corresponding features in a subsequent image. Such correct correspondences are essential, since wrong ones (e.g., similar looking features) will disturb the correct computation of the 6D-trajectory. In such a case, the least-squares problem (Eq. 3.9) is solved based on wrong assumptions (Eq. 3.8 does not hold). Therefore, it is of great interest to get correct correspondences. If a large number of features is available, it could be of benefit to loose features which cannot be assigned properly in the subsequent image. Hence, we want to use a tracking system which searches for subsequent features in a close area around the previous position of the feature. A search in such an area restricts the possibility to find false correspondences in the form of similar, but other features. Additionally, the processing is faster, since it is not necessary to search for the features in the entire image. At the same time, the features are just allowed to move within the restricted regions around the previous position of the feature. Hence, the frame rate has to be high enough to ensure, that the subsequent features are within this region. Otherwise, the region needs to be enlarged. However, such an increased region reduces, of course, the just described desirable effects.
An exemplary object selection and an exemplary tracking of the features are shown in Fig. 3.5.

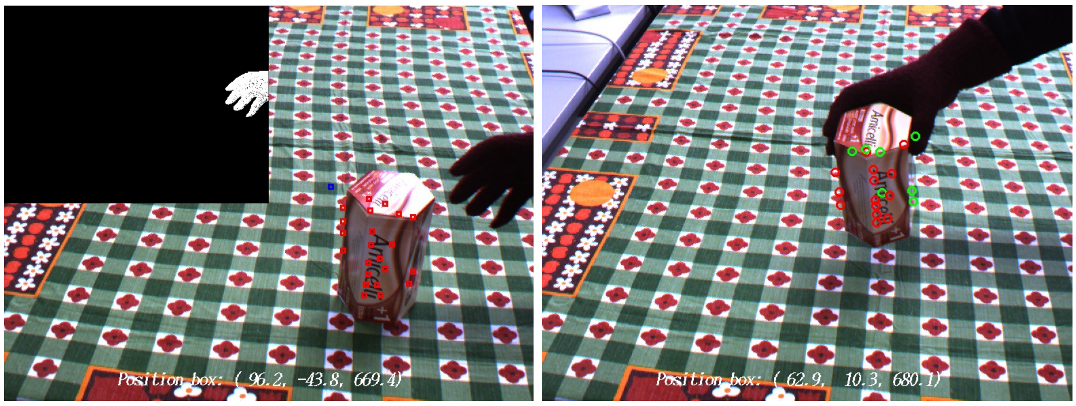

Position box: (96.2, -43.8, 669.4)
Position box: (62.9, 10.3, 680.1)

Figure 3.5: Contact detection and object tracking [1]. *Left*: The tracking is initialized after the contact detection between the hand and the region of interest. The small red boxes are the valid features. The blue ones are deleted features, which are not on the box. The initialization of the features is processed within a bounding box around the detected object. Afterward, we check for each feature, whether it is on the object itself using the pixel-wise determination of the object (Section 3.1.1.2). The top left corner of the image shows the position of the hand (blob-detection), when the contact occurs. *Right*: Example of features during the tracking. The tracked features are shown in red, the assumed positions of the lost features are, resp., drawn in green.

The tracking stops at the end of the manipulation. Afterward, the type of the object is determined. The object's type forms the key to find the corresponding object type in the

system's knowledge. It fulfills two purposes. First, known properties stored in the Atlas can be assigned to the current object. Second, new properties can be integrated in the existing knowledge.
In order to find out, which type of object is observed, we apply an object registration. The object is, therefore, recognized based on its shape.
The determination of the object's type takes place after the manipulation. It is much more efficient to analyze just the type of the manipulated object, instead of analyzing all objects in the entire scene before.

3.1.1.5 Representation of Object Knowledge

The recorded trajectory of the manipulated object is used for the determination of the manipulation-relevant object knowledge. The choice of the knowledge which is extracted and the way of representing it are important for the future abilities of the system. For example, they determine the efficiency in the detection of new information. The representation of the manipulation-relevant object knowledge influences the system's ability to understand the intention of the human manipulation. Moreover, the ability to re-use knowledge is also determined by the way of representing the knowledge. Therefore, we present the description and explanation of the representation in detail in Section 3.1.2.

It should be mentioned here, that the approach presented in this Section 3.1.1.1 built a basis for the work in [171].

3.1.2 Representation of Manipulation-Relevant Object Properties and Functionalities

The representation of the manipulation-relevant object knowledge consists of two main components: the *Object Container* and the *Functionality Map*. They are described next, followed by the presentation of the knowledge extraction from a trajectory. It should be pointed out, that the representation of the manipulation-relevant object knowledge is independent of the method used for the trajectory acquisition.

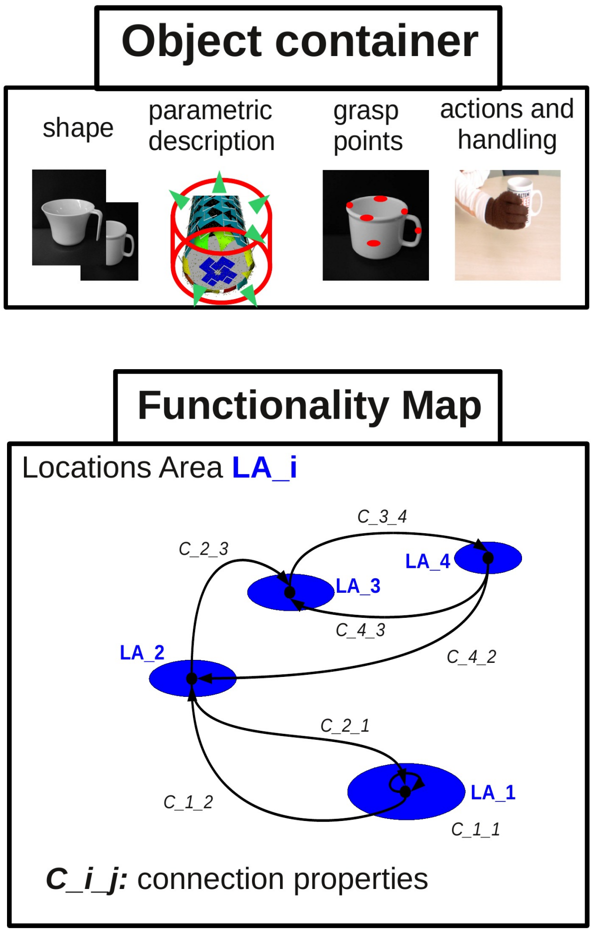


Figure 3.6: Object Container and Functionality Map [2]. *Top:* The Object Container consists of the object properties. *Bottom:* The Functionality Map is the abstract representation of the manipulation-relevant functionalities of the environment. It consists of the *Location Areas*, between which actions are performed. The properties of these actions are stored in the connections between the Location Areas.

An overview of the system is given in Fig. 3.6. It illustrates the Object Container comprising the object properties and the Functionality Map containing the manipulation-relevant actions.

3.1.2.1 Manipulation-Relevant Object Properties

The first component, which is specified, is the Object Container. It stores the object properties, which are directly related to the object itself.

As already motivated, a general knowledge of the object properties is of interest. Hence, simple records of x,y,z-coordinates along a trajectory shall not be listed, but the abstract handling properties. We consider the orientation, the maximal allowed acceleration, the mass and the center of gravity as important in this work. Some of these properties are not observable with a pure vision-based system or a pure tracking system, which provides a 6 DoF-trajectory. Therefore, the Atlas, which contains the "experience" (*a-priori information*), is used to obtain this information.

The handling properties themselves are constraints, which limit the handling possibilities of the object. For example, if an object is kept upright all the time, the object-orientation does put a constraint on the manipulation in this context.

3.1.2.2 Functionality Map of the Environment

The Functionality Map is the second component of the representation. It contains the functionalities of the object related to the environment.

The first element of the Functionality Map are the *Location Areas*. These areas are the locations in 3D space, where a manipulation sequence can start or end. Location *Areas* are explicitly defined in contrast to single locations. An object is usually placed in a certain area and not on one certain coordinate in space.

The connections between Location Areas are the second element of the Functionality Map. A connection exists between two Location Areas, if an action has been performed directly between both areas without visiting another Location Area meanwhile. It is important to consider, that a connection is directed. Therefore, a connection from A to B is different from the connection in its opposite direction from B to A. The connection itself consists of the manipulation properties of the action performed on this connection. Of course, different manipulation alternatives can occur. Moreover, the system needs to store the different properties for each object separately. Two exemplary Functionality Maps can be seen in Fig. 3.7.

The properties, which are stored in the Functionality Map, are the following:

- **pushed object vs. lifted object** - An object can be manipulated by lifting or by pushing it. A pushed object needs just to be pushed in the desired direction, whereas lifting an object requires much more effort (e.g., knowledge about the way of grasping, the object's weight).

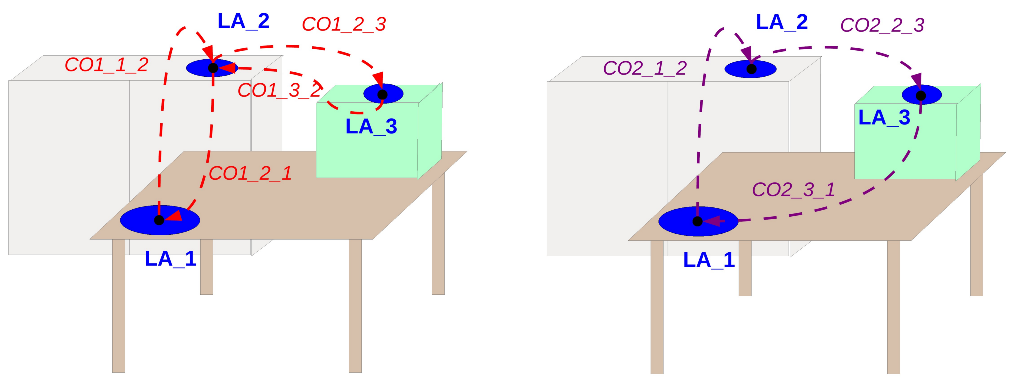


Figure 3.7: Functionality Maps for two exemplary objects [2]. The figure illustrates the three identified Location Areas in the environment as well as the connections between them. The first Location Area is on the table, the second one on the gray cupboard and the third one on the green box on the table. The arrows show the connections between the Location Areas. As it can be seen, different connections can exist, depending on the object. For example, there is a connection from Location Area 3 to 1 for the the object on the right, whereas the Functionality Map of the object on the left does not have such a connection.

- **arbitrary movement vs. constrained trajectory** - The trajectory between two Location Areas has either an arbitrary shape or it represents a constrained trajectory. A constrained trajectory connects the Location Areas in a direct manner while avoiding detours. In contrast, an arbitrary movement does not have such a directed shape. Consequently, the constrained trajectory sets a constraint on the possible trajectories, whereas an arbitrary movement does not.

- **connection relevance** - The connection relevance shows the probability of a manipulation-relevant property, based on the number of observed actions.

- **velocity constraints during pick-up** - Three phases defining an action are used: the pick-up, the transportation and the placement phase. The maximal speed during the pick-up phase is stored in the Functionality Map, since it is an indicator of the difficulty to pick up an object.

- **grasp taxonomy** - The grasp type is mainly important for the pick-up and placement phase of the manipulation and is not part of this work. The grasp taxonomy considered for the system is summarized in [172].

- **approach vector** - The approach vector is, similarly to the grasp type, mainly important for the pick-up and placement phase of the manipulation and is not part of this work. The approach vector is the direction, from which the object is grasped in the object-centric perspective.

The Location Areas and their connection properties can be transferred to a similar environment, since just the corresponding areas in the similar environment need to be identified. Then, the known functionalities can be applied. Hence, the existing knowledge can be mapped to the current environment (Working Memory).
Furthermore, the Location Areas can be used for a segmentation of action sequences in the workflow. At each Location Area, a new sequence starts. Moreover, the Location Areas can work as handover areas for objects. An object can be delivered to somebody else, there.

A Functionality Map of one object can formally be seen as a graph. It consists of a pair $(\mathcal{V}, \mathcal{E})$ with a finite set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. $\mathcal{V}$ contains all Location Areas and $\mathcal{E}$ all connections between the Location Areas. As already described, the connections and, therefore, the edges are directed. Multiple edges can exist, since different kinds of manipulation properties can be observed (e.g., pushed vs. lifted object). A manipulation can also lead from an initial Location Area to itself again. Consequently, the graph contains loops as well. Furthermore, the graph is non-complete, since not all Location Areas have to be connected. It is possible, that the graph is disconnected: The system might observe demonstrations for one object between two different, non-overlapping subsets of Locations Areas. Of course, a strongly connected graph is desirable for a large range of possible paths in future planning of manipulations. However, a disconnected graph or weakly connected graph can be built up during observations.

The assignment of the properties to the Object Container or the Functionality Map depends on the kind of the property. A property, which is related to the functionality in the environment, is assigned to the Functionality Map. For example, the velocity constraints during the pick-up is part of the Functionality Map, since it depends on the environment of the object (e.g., obstacles). In contrast, a property which is directly related to the object and its state is a part of the Object Container. The maximal allowed acceleration for an object is an example for such a property (e.g., no high accelerations for a cup filled with coffee).

3.1.2.3 Knowledge Extraction

The proposed Object Container and Functionality Map need to be filled with information. Hence, the knowledge extraction is described, next. The scene can be observed with any system providing a 6 DoF-trajectory of the manipulated objects.

Object Container The properties which shall be extracted for the Object Container are the maximal acceleration value and the orientation of the object during the manipulation.

Maximal Acceleration The maximal acceleration value is estimated by computing the change of two consecutive speed values. Similarly, the speed is approximated by the difference between two consecutive positions per time unit.

Orientation The orientation is determined from the observed 6 DoF-trajectory. The object orientation can change around three axes. One axis is the vertical axis of the object. It is parallel to the normal of the plane and running through the middle of the object at the initial position. The other two axes are the horizontal axes of the object. They are parallel to the plane at the initial position of the manipulation. Just the orientation change around both horizontal axes (i.e., changes of the first two components of the orientation $\boldsymbol{o}$: o_1 and o_2) are of interest for the constraints in the manipulation task. The aim is the distinction of a motion with rotation from a motion without rotation. Hidden Markov Models (HMMs) [21] are used for the classification, because of their ability of generalization. They are statistical classifiers, which use an observation sequence for the estimation of the underlying state-sequence. Moreover, they take into account knowledge of the past (previous state) in the sequential input. Here, discrete HMM with $\boldsymbol{\lambda} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\Pi})$ are chosen. They comprise a transition probability matrix $\boldsymbol{A}$, an observation symbol probability distribution matrix $\boldsymbol{B}$ and an initial state distribution $\boldsymbol{\Pi}$.

As described, the HMMs receive an observation sequence as input. Hence, we convert the original sequence of observed changes around both horizontal axes (o_1 and o_2) into such an observation sequence. First, a pre-processing takes place. We apply an overlapping window of 400 ms with a 200 ms overlap (according to [173]) on the original sequence. Then, the angles between the axes of the current coordinate frame and the axes of the initial coordinate frame at the beginning of the manipulation are measured for each window. It is possible, that different amounts of angles occur for different objects depending on the object and the way of recording its trajectory. The relative amount of change is needed for each object. Hence, the angles are normalized for each object with its maximum angle occurring in all movements of the object. Now, we have a sequence of the relative changes around both horizontal angles. The application of the window ensures, that it contains changes at a visible, but still very small size.

After this pre-processing, we want to convert the sequence of the relative changes into the desired observation sequence to use the HMMs. Hence, we need a codebook, which allows us to assign each element of the sequence of the relative changes to, resp., one corresponding symbol. This means, that the very granular number of each element is represented by a symbol. It reflects a characteristic area of granular numbers. The resulting sequence of symbols forms the desired observation sequence. In order to build up the desired codebook, we cluster the elements of the sequence of the relative changes independently of their time of occurrence. Each cluster refers, then, to one characteristic area of granular numbers. The K-means algorithm [174] is used, therefore. It aims to create K cluster $\boldsymbol{\mu}_k$ in a manner, such that the distance between an element and its corresponding cluster is minimized. In our application, an element consists of a pair (o'_1, o'_2) which contains the relative change around the horizontal axes after the pre-processing. We label each element with $\boldsymbol{o}'_n = (o'_1, o'_2)^T$ and the total number of elements with N. Moreover, we introduce a binary variable with $r_{nk} \in \{0, 1\}$ which equals 1, if the n-th element is assigned to cluster k. It is 0, otherwise. The following term (Eq. 9.1 in [175]) is minimized in the K-means algorithm:

$$\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|\boldsymbol{o}'_n - \boldsymbol{\mu}_k\|^2 \tag{3.10}$$

The resulting cluster are 2-dimensional. They form the core of the desired rotation information codebook.

The next step is the buildup of two HMMs, in order to classify the motions as ones which contain a rotation ($\boldsymbol{\lambda}_r$) or as rotation-free ones ($\boldsymbol{\lambda}_{noR}$). For each HMM, the transition and emission probabilities need to be calculated. We use the maximum likelihood estimation for this purpose. The estimator receives training sequences $\boldsymbol{o}_{training}$. It estimates the parameters $\boldsymbol{\lambda} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\Pi})$ of the HMM in a manner, such that the probability to observe the training sequences given the HMM ($p(\boldsymbol{o}_{training}|\boldsymbol{\lambda})$) is maximized [175]. The required training sequences are labeled as motions which contain a rotation or as rotation-free ones.

For the evaluation, the system receives test sequences, which are pre-processed as described above. The corresponding symbols in each codebook are assigned by the k-nearest-neighbors-method. The k nearest neighbors of each element (o'_1, o'_2) are determined. The element gets, then, assigned to the class to which most of the k nearest neighbors belong (see, e.g., [175]). To evaluate the classification performance of the trained HMMs, the maximum log likelihood log $\mathrm{P}(\boldsymbol{o}_{test}|\boldsymbol{\lambda}_i)$ of a given model $\boldsymbol{\lambda}_i$ is computed for each test sequence with observations $\boldsymbol{o}_{test}$ similarly to [23]:

$$\boldsymbol{\lambda}_r^* = \arg\max[\log\ \mathrm{P}(\boldsymbol{o}_{test}|\boldsymbol{\lambda}_{noR}), \log\ \mathrm{P}(\boldsymbol{o}_{test}|\boldsymbol{\lambda}_r)]\ . \tag{3.11}$$

Functionality Map The Functionality Map, which relates the object's functionality and its environment, is built next.

Location Areas The possible Location Areas have to be determined, first. Therefore, the available trajectories are split up in single sequences, which consist of the object movements between two consecutive stops. The x,y,z-coordinates do not change at such a stop in the trajectory. The collected 3D-points of the stops are clustered. Similarly to before, the k-nearest-neighbors-method is used for the clustering. The resulting cluster-centers are the centers of the Location Areas. If objects of different heights are involved, the values of their heights can differ at the stop points on the same surface. Then, a projection on the corresponding plane makes the clustering more convenient.
It is possible, that the system detects two Location Areas, which coincide in fact. However, they could randomly appear as two. These Location Areas can be fused, since they are close to each other and have the same connection properties.

Connection Properties The next step is the determination of the connections between the detected Location Areas as well as the corresponding properties of the connections. This is done for each object. The properties, used in this work, are the distinction of a pushed vs. a lifted object, the differentiation of an arbitrary movement vs. a constrained trajectory, the velocity constraints during the pick-up phase and the connection relevance of a property on a certain connection. If possible, the grasp type of the manipulation is determined.

Pushed Object vs. Lifted Object: An object is pushed, if it is in contact with its supporting plane during the entire manipulation. The computed normal vector $\boldsymbol{n}_r^*$ (see Section 3.1.1.1) is used for this purpose. Eqs. 3.1 and 3.2 can be rewritten as

$$\mathrm{P}_r : \rho_1 k \cdot x + \rho_2 k \cdot y + \rho_3 k \cdot z = d_r. \tag{3.12}$$

Using the already mentioned $k = \frac{d_r}{B}$, it becomes clear, that the normal vector $\boldsymbol{n}_r^*$ can be used as well:

$$\rho_1 \cdot x + \rho_2 \cdot y + \rho_3 \cdot z = \frac{d_r}{k} \tag{3.13}$$

$$\Leftrightarrow \boldsymbol{n}_r^* \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = B. \tag{3.14}$$

The center of an object placed on a table is, of course, above the table plane. Therefore, the plane is translated by the object's height h_{obj} in the direction of $\boldsymbol{n}_r^*$:

$$\boldsymbol{n}_r^* \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = B + h_{obj} = h_{push}. \tag{3.15}$$

In order to determine an object as pushed, the trajectory of the manipulated object has to stay within the translated plane. Hence, if the 3D-positions of a trajectory fulfill Eq. 3.15, the object is pushed. Otherwise it is lifted.

Arbitrary Movement vs. Constrained Trajectory: A Principle Component Analysis PCA (with rescaling) is performed for the distinction of an arbitrary movement and a constrained trajectory. The PCA is done on a 4.8 s window with a 2.4 s overlap. The resulting principal components are normalized. Now, the arbitrary movement is determined. It has no main direction of motion. The movement is relatively large in all directions. Therefore, the third component of the PCA is of especial interest. It shows the direction of the smallest motion. If this motion has a high amplitude, the entire motion has a relatively high amplitude in all directions, since even the direction of the smallest motion is high. Consequently, it is an arbitrary movement. The smallest motion is defined as "high" in two cases. The first case is a comparison with the main direction of motion (= the first PCA-component): If the magnitude of the first and the third component are relatively "close" to each other, there is hardly any main direction of the movement, and the movement is arbitrary. "Close" means, that the component of the smallest movement is multiplied with a factor (multiplication factor *arbitrary-movement*), which determines, how many times the component of the largest movement is maximally allowed to be larger than the component of the smallest movement. The second case of a “high” smallest motion is occurring, when the third component is higher than a threshold (*arbitrary-movement-threshold*). The *arbitrary-movement-threshold* has to be chosen in the magnitude of the third PCA-component of the arbitrary movements. If all the described criteria are not met, the direction of the smallest motion is not high, and the movement is a constrained trajectory.

Velocity constraints during the pick-up: The pick-up phase is defined manually with 50 samples from the initial position. As already described, the speed is computed for two consecutive samples.

Connection relevance: The connection relevance can be determined easily by dividing the number of occurrences of a certain property on a connection by the number of all movements on this connection.

Grasp Type: The grasp type is determined by manual labeling.

The extracted manipulation properties are compared with the already perceived knowledge in the Functionality Map of the corresponding object. If a new property is observed between two Location Areas, a new connection is established.

3.2 Representation of Object Relations in the Environment for Dexterous Manipulations

The just described representation of manipulation properties and actions provides very general knowledge about manipulations. We want to go a step further as well. In the case of dexterous manipulations, a more detailed representation is required. Sophisticated manipulation capabilities are an important step towards daily use suitability of robots. We focus on advanced knowledge about dexterous manipulation. The requirements on the physical capabilities of the involved robotic hand and arm are separate issue and, therefore, not considered here.
The desired representation of the knowledge has to enable a successful reuse even if the environment changes. Hence, it should be possible to build up alternative procedures to achieve a desired goal of a task. Moreover, the knowledge representation has to be independent of the robot to allow a usage of the knowledge on different robotic systems. The representation should be easy to use, since a human should be able to command the robot without programming the exact path. At the same time, it has to be possible to extract the required knowledge for the representation from observation. To conclude, a representation at an abstract level is required rather than, e.g., a record of observations from demonstrations.

The aim of a task is usually a desired goal state of objects in the environment. The goal requires, then, a modification of the current state of the environment. In general, a state could be related to a physical state of an object. For example, a cup can be empty or it can be filled with coffee. Hence, its physical state (weight) is different. Moreover, the handling properties of an object can differ (e.g., tilting of a cup filled with coffee should be avoided). A representation of such general properties has been motivated in Section 3.1. As described, we focus here on more dexterous tasks, which require a much more detailed knowledge about the goal state in the environment. The representation of such tasks should be flexible enough to describe predefined fixed paths, if desired. At the same time, less limited paths have to be represented appropriately as well. It should be possible to change those paths, if there are other actors or objects in the environment, which affect the scope of action of the robot.

Possible applications are daily life manipulations as well as specialized tasks. An example of such a specialized task are medical applications (robot-assisted minimally invasive surgery). Knot-tying is an example of a demanding task in this area. This requires very complex manipulations of the involved threads. Such manipulations do usually not take place in the same pre-determined environment. The environment can change each time. Hence, the manipulation path has to be adapted. Moreover, it should be possible to transfer knowledge from one medical robot to another. This could be done by a direct transfer of the knowledge base or by an observation of a robot demonstrating the tasks.

We use the perspective of *contact states* as base of the abstract representation here. We describe the environment and its changes as contact states and contact states changes. Such a contact state contains not only the information about the existence of a contact, but also about the type of contact. Therefore, we introduce appropriate relations between objects. These can describe the type of contact in more detail, if complex contact like knots are desired. The abstraction of knowledge in contact states allows flexibility in the task execution through alternative paths. Moreover, the representation is independent of the robot. Hence, a transfer from one robot to another is easily possible. A further important aspect is, that the contact state perspective enables a clearly arranged representation with a wide range of descriptions from simple to complex contacts.

An illustrative example is shown in Fig. 3.8. If somebody wants to drink water, he could ask for a cup with water. The goal state of the contact is, hence, water in the cup. Of course, one could directly fill water from the water conduit into the cup. The usage of a tool like a can inbetween is another possibility: The can could be filled with water and the water can be filled from the can into the cup. Both ways (with and without the can) lead to the desired final contact state. The advantages or disadvantages of the tool usage can depend on further *external factors* like efficiency (reduced efficiency due to an additional tool) or further tasks (re-filling of the cup at another place).

We define a *tool* as appliance to change the contact states in the environment to the desired ones. The robot itself is defined as an *active object* which executes the required manipulations. All other objects which cannot perform actions themselves are *non-active objects*. Tools are, in general, also non-active objects, which are used actively to change the contact states. For example, if somebody wants to drink water from a cup, we can consider a can as tool to fill water in the cup.

It is possible to depict very simple manipulations in the contact state perspective. For example, an object can be transported from one location to another. The aim contact state is, then, the contact of the object with its final location. Hence, the *Functionality Map* (Section 3.1) can be easily used. Moreover, the contact state perspective has advantages concerning the execution of transportation tasks. For example, box A has to stand on box B on a desired table T. Currently, both boxes are placed separately on another table. Our perspective of contact states allows to build up of two alternatives: Box B can be transported to T and placed there. Afterward, A can be transported and put on B. Alternatively, A can be placed on B, first, and, then, both boxes can be transported and placed on the table T. The final choice of the procedure depends on external factors (e.g., heavier load, but just one transportation in the second procedure).

To conclude, we present a representation of the environment for dexterous manipulations in a contact state based perspective. The representation is chosen in a manner, such that (1) it extends existing work with respect to a detailed information representation for dexterous manipulations, (2) it is clearly structured to enable an easy usage, (3) it is reusable even in changing environments, and (4) path alternatives can be build up if feasible.

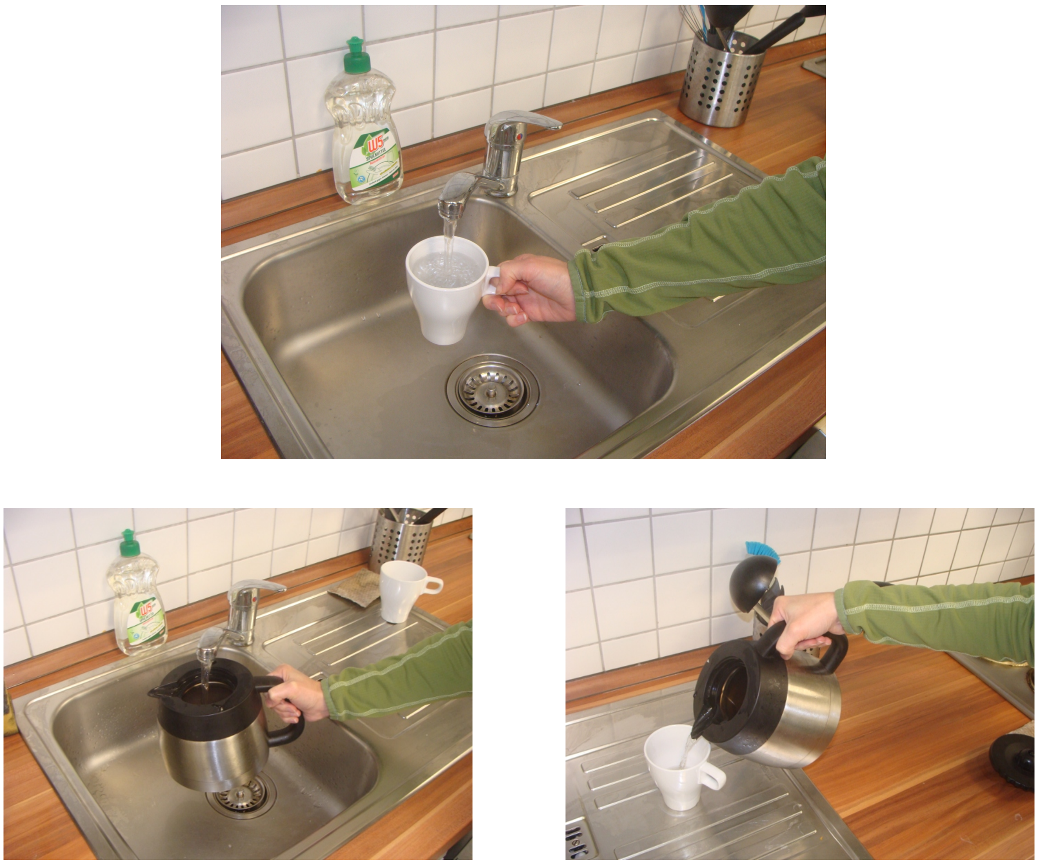


Figure 3.8: Contact state based representation of object relations [7]. The desired contact state is water in the cup. Hence, one could directly fill water into the cup (top figure). Of course, one could use a tool, e.g., a can inbetween to achieve the desired contact state (bottom row). The final choice depends on several factors like efficiency (usage of an additional tool requires time capacities) or further tasks (e.g., refilling of the cup).

In the following, the contact state and the roles of objects are defined for the representation of the environment for dexterous manipulations. The composition and usage of the contact state knowledge are described afterward.

3.2.1 Contact State Definition

We define a contact state as a special function with respect to the involved objects and properties of the contact:

```
contact(objects, contact_properties)
```

The involved objects are a set of keys referring to information in the system's knowledge (e.g., Atlas). They could be implemented with an index pointing to the corresponding object in the knowledge.
The contact properties can be described in different ways. This enables a flexible usage under different requirements. For example, a set of contact points can strictly describe the contact state. Another, much richer and more complex possibility are relations. Relations have the advantage, that they can describe the contact properties in a more abstract and less restricted manner. We define a relation in contact properties by a relation attribute and the involved objects:

```
relation: [relation_attribute, objects]
```

Such relations allow alternative procedures, since the path is described by the *properties* of the dexterous manipulation. Motion-restrictions or efficiency requirements can be considered for the final choice of the path. For example, a search for alternative paths can be processed (e.g., due to additional obstacles in the environment). If, e.g., an object A should be placed on object B, the relation would be defined as follows:

```
relation: [over, (a,b)]
```

As we can see, the goal state of the manipulation is defined. Hence, we are not limited to one path. All paths fulfilling the desired property can be applied.
Of course, much more complex descriptions of the contact can be necessary. We use, then, a set of properties to define the contact:

```
contact_properties := {point_set, relation, property_set}
```

Such a set of properties, which can, in turn, contain a set of properties. This allows to structure the properties. The property sets can contain sets of points, sets of relations or further variables defined the type of contact. We could, e.g., consider a cut on an organ in a medical scenario (see also simulations in Section 7.2.1). A cutting tool (the knife) and the organ are the involved objects. The cut is described by the cut properties:

```
contact({knife, organ},
    cut_properties)
```

Such a cut cannot be chosen arbitrarily in an operation (e.g., to excise a tumor). Here, it has to be performed accurately along a certain path, which is represented by a set of successive points. Moreover, the cut has a certain depth, which defines how deep the cut has to be.

```
cut_properties = {cut_points, depth}
```

The depth of the cut could be the same for all points or it could be specified by a set representing the respective depth for each point of the cut. One needs to consider that the property "depth" implies knowledge about the space (depth with respect to a certain direction). Such implicit knowledge needs to be considered and, hence, included in the implementation.

As the term property "set“ already indicates, the properties of the contact can be reordered arbitrarily within the set. In contrast, a relation is strictly defined, since the adjective describes the relation of the objects to each other. Hence, the order of the involved objects matters in the relation.

The usage of relations or fixed points depends on the application. There can be tasks, for which a path of fixed points is necessary as for a precise cut along a line. For other tasks, several or even many paths can be used to achieve the desired goal state, even if the manipulation is dexterous. For example, if a knot has to be tied at shoes, the procedure is not limited to one fixed path. Instead, the complex contact state of the involved threads can be described by relations (as we will also see later on in the Experiments).

3.2.2 Object Role Definition

The objects involved in the task can be active, like a robot with a gripper (*active objects*). Other objects are not active themselves, as, e.g., a cup (*non-active objects*). Of course, it is possible to use non-active objects through an active object. We call these non-active objects *tools*. The role of the objects can depend on the current task. For example, a can can be used as a tool to fill a cup with water. If the task is the contact state "water in the cup“, we are done. Otherwise, the cup can, in turn, be a tool, e.g., to drink water.

The contact state perspective allows the composition of alternatives in a task. One or more tools can be included in the task flow, if desired. We have already illustrated such an example (water drinking, see beginning of Section 3.2 and Fig. 3.8). The final choice of a procedure depends on external factors as, e.g., the reusage of a tool. Our definition of contact states and the subsequent definition of the object role enables the build up of alternatives if desired and appropriate.

3.2.3 Composition and Usage of the Contact State Knowledge

The described contact state based knowledge representation can be composed through different procedures. One procedure is the contact state definition by humans. It is not necessary to program the entire robot. Just the above descriptions need to be filled out. A second procedure is the observation of demonstrations by humans or other robots. The contact states in the environment have to be observed, then. Hence, we are independent of the demonstrator (its physical build-up, its capabilities, its knowledge representation, etc.).

In order to achieve possible paths which can fulfill the task in a scenario, we need to evaluate the contact state description. If the task allows just a path of fixed points, the path is predetermined. It can be executed straight forward. If the properties are described by a relation, the relation has to be evaluated first, before a path can be created. As we described, a contact property can contain sets of properties. An element of these properties can, in turn, consist of a set of properties. Therefore, we need to consider several levels of properties. We start with one of the properties at the lowest level. This means, that these properties are not defined through another property set.

If the contact properties are defined by a set, the execution order of the elements in the set is not defined in advance. This could, e.g., be determined by a real experiment or a physical model, which simulates the execution of the relations in different orders. The results of such a simulation, respectively, experiment show, which execution orders lead to the desired contact state. It is possible, that one, several or all possible orders lead to the goal.

The objects themselves can already determine the area of possible paths. One object or parts of an object (e.g., a thread) can be fixed to a certain position. The task has to be processed within the reachable region of the (partly) fixed object. The movable object(parts) can be moved to achieve the desired goal. Moreover, the movable object(parts) allow to provide alternative paths. If, e.g., a knot has to be tied at a shoe, the possible intersection point of the threads is limited to the intersection area of both threads. During the knot-tying execution, the intersection point is flexible within this intersection area. At the end, the threads are tightened and the position of their intersection is much more restricted to a smaller region.

The proposed contact state based representation is not limited to pre-defined paths. The objects can be described relative to other, possibly movable objects. Hence, the final path can be chosen under consideration of temporary constraints (e.g., other organs as obstacles). Different approaches can be used to build up the final path. We simply take a path which fulfills the desired properties and constraints, here. In future work, the choice of the final points can, e.g., be optimized as suggested in Section 4.2. Another possibility is the usage of Dynamic Movement Primitives (DMPs) [29], [30].

Chapter 4

Path Optimization for Abstractly Represented Tasks with Respect to Efficient Actuation

In the analysis of manipulator structures, the potential of path optimization is of interest, as we motivated already. Hence, an appropriate path optimization method with respect to efficient actuation needs to be developed. We described already, that manipulations are, most of the times, not restricted to a fixed path (e.g., Section 1.1). Variations are possible. The robotic system can make use of some of these varieties to simplify the actuation to improve the efficiency of the generated motion. Nevertheless, the human's intention behind the manipulation may not change. But how should the path look like to support efficient actuation? Fig. 4.1 illustrates a scenario with different path configurations.

First, we provide an analysis tool which uses the abstraction of the human actions to compare different path types with respect to efficient motion profiles. This work is published in [3]. Afterward, we investigate in an advanced concept for path optimization with respect to efficient actuation. The optimized path needs to enable the easiest and most efficient actuation. Hence, each joint of the robot should move slowly and smoothly. Abrupt turnarounds of the acceleration should be avoided. Furthermore, energy can be saved and the strain on the hardware can be reduced. The work is published in [5].

We introduce the *Elastic Power Path* to optimize the path with respect to efficiency in actuation. In analogy to an elastic band, the Elastic Power Path has a certain elasticity. When an elastic band is stretched, additional energy is required. In the context of efficiency, we want to minimize this additional energy. The less stretched the band, the closer desired optimum is. This depends, of course, on the elasticity of the band. Analogously to an elastic band, the Elastic Power Path can be stretched and relaxed. Its elasticity depends on the actuation of the robot along the path. The more efficient the actuation, the closer the Elastic Power Path is

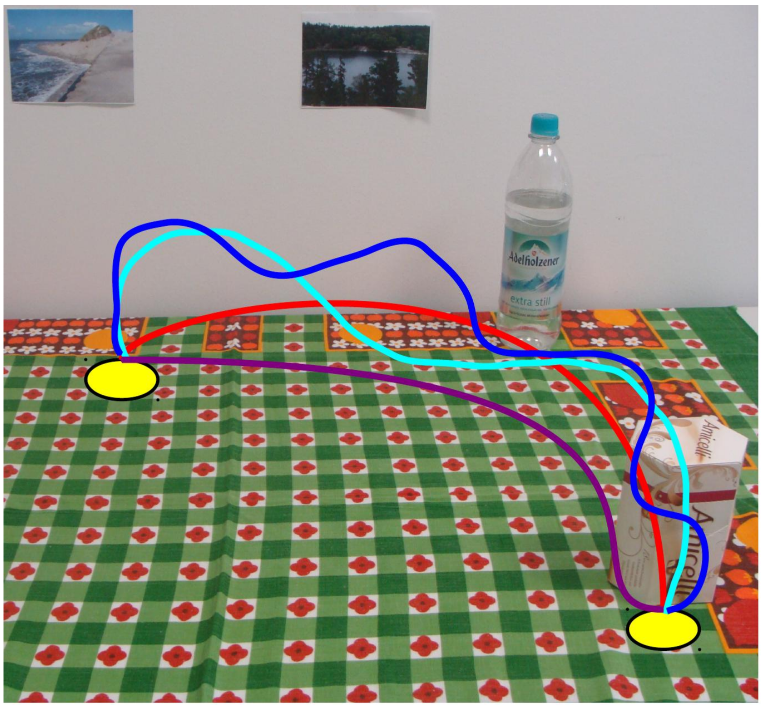


Figure 4.1: Path optimization: Different path configurations [5]. The Figure shows different path configurations for a manipulation between two areas (yellow). Which path is the most efficient one with respect to actuation?

to its optimum. Our aim is an optimal path with respect to efficient actuation. We measure the required power in the style of a hiker climbing up and down hills. The required power is increasing, if more hills have to be climbed or if higher hills have to be climbed. The speed and acceleration profiles play an important role regarding the power. In the context of path optimization for a manipulator, we focus on the speed and acceleration of the joints. The less often the direction of the acceleration is reversed and the smaller its magnitude, the less power is needed.

It is important to point out, that we do not analyze all possible configurations of the robot in the entire workspace. The concept of the Elastic Power Path allows an efficient optimization for the desired task: Just promising configurations of the robot are considered during the optimization. Moreover, the optimization does not depend on a specific manipulator structure. For example, it is not necessary to find appropriate joints in the manipulator structure for each (sub-)motion.

We build on the concept of Functionality Maps (Section 3.1) which represent observed manipulation tasks in an abstract manner. Different properties of manipulation tasks are stored there, as, e.g., the type of manipulation or characteristic areas, which are start or end points of manipulations.

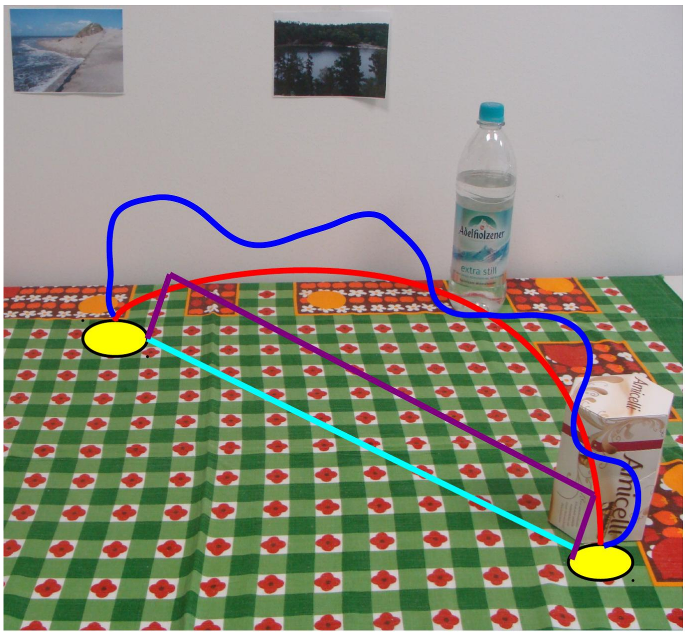


Figure 4.2: Paths with different properties for a manipulation between two areas [5]. The Figure shows paths with different properties (e.g., motion shapes) for a manipulation between two areas (yellow). Which path should be chosen to achieve efficient actuation during the manipulation?

The concept of the Elastic Power Path allows to optimize paths (1) with respect to efficient actuation (2) in task-specific contexts (3) independently of the manipulator's structure.

The path configuration framework and the path optimization approach are presented next. Both are grounded on abstractly represented tasks and consider efficient actuation of the manipulator.

4.1 Path Configuration

As described, we provide an analysis tool which uses the abstraction of the human actions to compare different path types with respect to efficient motion profiles. We evaluate paths with respect to the efficiency of actuation for a given robot. Which modifications of the path influence the robot's actuation? How can we make use of the freedom in path planning, which accompanies the abstract representation of tasks?

The paths, which we want to compare, can be distinguished between the following three categories:

- **basic motion shape** - Four different basic motion shapes (bs) are considered here: a *line (bs 1)*, a *half circle (bs 2)*, a *wiggly line (bs 3)* and a *half quadrilateral (bs 4)*. The latter three are positioned upright over basic motion shape 1 along the vertical axis in the room (see Fig. 4.2).

- **compression and elongation** - The basic motion shapes 2-4 can be compressed or elongated by a factor e along the vertical axis in the room.
- **bias** - The basic motion shapes 2-4 of the path can be biased. The path is, then, turned around the axis of basic motion shape 1 by an angle β.

The robot's actuation properties for a certain path can depend on the placement of its base. Therefore, we process the evaluation for different positions of the base.

Our experiments will show, that the path properties influence the actuation in a not necessarily intuitive manner (see Section 7.4.1). Therefore, it is worth to further analyze the path properties with respect to efficient actuation. Hence, a path optimization approach is presented next.

4.2 Path Optimization

We want to optimize a path for a manipulator with respect to efficient actuation. A clear definition about the used variables and definitions is important. Therefore, we give a short overview of them.

As in the previous chapter, the robotic system is described in the DH-convention [164] in the form shown in [9]. The manipulator system consists of D joints. Each joint j contributes a certain linear velocity vector $\boldsymbol{v}_j$ to the overall velocity $\boldsymbol{v}$ of the end-effector:

$$\boldsymbol{v} = \sum_j \boldsymbol{v}_j \tag{4.1}$$

All vectors are given in a global coordinate frame, if not labeled otherwise. The linear velocity $\boldsymbol{v}_j$ of a rotational joint j can be computed by

$$\boldsymbol{v}_j = \boldsymbol{\omega}_j \times (\boldsymbol{p}_{ee} - \boldsymbol{p}_j) \tag{4.2}$$

with $\boldsymbol{p}_{ee}$ as the position of the end-effector, $\boldsymbol{p}_j$ as the position of joint j and $\boldsymbol{\omega}_j$ as the rotational velocity caused by joint j:

$$\boldsymbol{\omega}_j = \dot{\theta}_j \cdot \hat{\boldsymbol{Z}}_j \tag{4.3}$$

with $\hat{\boldsymbol{Z}}_j$ as 3-dimensional unit vector of the z-axis of joint j and $\dot{\theta}_j$ as magnitude of the angular rate (see, e.g., [9]). The acceleration is labeled with $\triangle\boldsymbol{v}_j$, indicating its meaning as difference between two consecutive speed values.

4.2.1 Elasticity of the Path

Our basic idea can be illustrated through an elastic band with linear elasticity. The elasticity can be described through Hooke's law, similarly to a linear spring:

$$F = -k \cdot x \tag{4.4}$$

with x as the displacement from the equilibrium position, k as the spring constant and F as the resulting force.

We use the elasticity of the path as illustration of our idea, therefore, we assume that the displacement is within the elastic range. As motivated at the beginning, we want to optimize the path with respect to efficient actuation. Hence, the spring constant k depends on the efficiency of the system's actuation. The displacement x describes the distance between the current configuration of the path points and their ideal configuration.

Hook's law of elasticity can be seen as optimization function, which optimizes the system's configuration with respect to energy. In contrast to a real elastic band, the path optimization function is much more complex. First, the elasticity depends on the efficiency of the actuation. Many local minima can occur in the optimization function. Second, the ideal configuration of the path points, resp., the displacement x is unknown. Therefore, the force F cannot be computed directly. This results in a chicken-and-egg problem: The displacement x is unknown, consequently, the force F cannot be computed and vice versa.

4.2.2 The Elastic Power Path

Similarly to Hook's law, we want to optimize the path configuration of a manipulator system with respect to energy. In our context, the energy is consumed by the manipulator to perform the desired task. The manipulator spends the energy to move its joints. Hence, joint movements at smooth and low speed are desirable. This requires, in turn, a smooth acceleration profile. The acceleration should not only be smooth, but it should also have a low magnitude. Moreover, zero crossings should be avoided, since a change of the acceleration sign refers to a large energy loss. The movement in the previous direction has to be decreased or even stopped (braking), while a movement into the other direction has to be built up. Moreover, braking means, that energy is necessary to stop the movement in the previous direction, which resulted absurdly from an earlier investment of energy.

We model the required energy in the style of a hiker climbing up and down hills. Hence, we compute the power the hiker would need to climb up and down all desired hills. In general, the power P is computed through

$$P = \frac{\triangle E}{\triangle t} \tag{4.5}$$

with time step $\triangle t$ and the difference between the initial and final kinetic Energy $\triangle E$ as

$$\triangle E = E_{final} - E_{initial} = \frac{1}{2}m(v + \triangle v)^2 - \frac{1}{2}mv^2. \tag{4.6}$$

We are interested in the derivative of the change of the kinetic energy $\frac{d}{dt} \triangle E$, in order to minimize the change of the kinetic energy $\triangle E$:

$$\frac{\frac{d}{dt} \triangle E}{\triangle t} = m \cdot \frac{v \cdot \triangle v + 0.5 \cdot (\triangle v)^2}{\triangle t}. \tag{4.7}$$

In our case, the velocity $\boldsymbol{v}$ is the linear velocity of the end-effector. The relevant acceleration is $\triangle \boldsymbol{v}_{j,i}$, which is the maximal acceleration contributed by joint j within $\triangle \boldsymbol{t}_{j,i}$. $\triangle \boldsymbol{t}_{j,i}$ is the required time to climb up and down hill i. The hiker starts and ends his/ her tours at consecutive zero points in the acceleration profile ($\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$). Each $\triangle \boldsymbol{v}_{j,i}$ reflects, then, one peak in the acceleration curve. If the acceleration curve proceeds below zero-level, it is mirrored on the zero-level to ensure that all peaks are counted equally later. It is important to point out, that the zero points do not change. Fig. 4.3 illustrates this procedure.

The term $\frac{\frac{d}{dt} \triangle \boldsymbol{E}_{j,i}}{\triangle \boldsymbol{t}_{j,i}}$ can be computed for each hill i in the acceleration profile of joint j. However, the relevant part of $\frac{\frac{d}{dt} \triangle \boldsymbol{E}_{j,i}}{\triangle \boldsymbol{t}_{j,i}}$ is the fraction in our case, since m is always constant. Hence, just a change of a variable in the fraction enables a change of the magnitude of the result. m works just as a scaling factor. Consequently, we want to minimize the following Objective Function O_j for each joint j along all zero points $\boldsymbol{p}_i$ on the path:

$$\mathrm{O}_j = \sum_i \frac{2 \cdot |\boldsymbol{v}_{j,i}| \cdot | \triangle \boldsymbol{v}_{j,i}| + (| \triangle \boldsymbol{v}_{j,i}|)^2}{\triangle \boldsymbol{t}_{j,i}}. \tag{4.8}$$

We use absolute lengths due to the described mirroring of parts of the acceleration-curve on the zero-level to create the hills. The absolute lengths of $\boldsymbol{v}_{j,i}$ enable an equal counting of all peaks. The original fraction has been multiplied with a factor of two for esthetic reasons. The sum of all $\triangle \boldsymbol{t}_{j,i}$ is, of course, not changing during the optimization.

The overall Objective Function O is the sum of the Objective Functions O_j for each joint j:

$$\mathrm{O} = \sum_j \mathrm{O}_j = \sum_j \sum_i \frac{2 \cdot |\boldsymbol{v}_{j,i}| \cdot | \triangle \boldsymbol{v}_{j,i}| + (| \triangle \boldsymbol{v}_{j,i}|)^2}{\triangle \boldsymbol{t}_i}. \tag{4.9}$$

4.2.3 Optimization of the Path

Our aim is the optimization of the path in the context of an abstractly represented task. As already mentioned in the beginning of the Chapter, we want to build on the concept of the Functionality Maps (see Section 3.1.2). Such a Map contains, e.g., the characteristic places of manipulations (Location Areas) and information about the type of the manipulation. Here, we make use of the Location Areas and the distinction between a manipulation of a pushed object on a table and a manipulation of a lifted object. These characteristic properties have to be preserved during the optimization. For example, the path of a pushed object should be a path of a pushed object after the optimization of the path, too. Similarly, the path of the lifted object should still refer to a path of a lifted object.

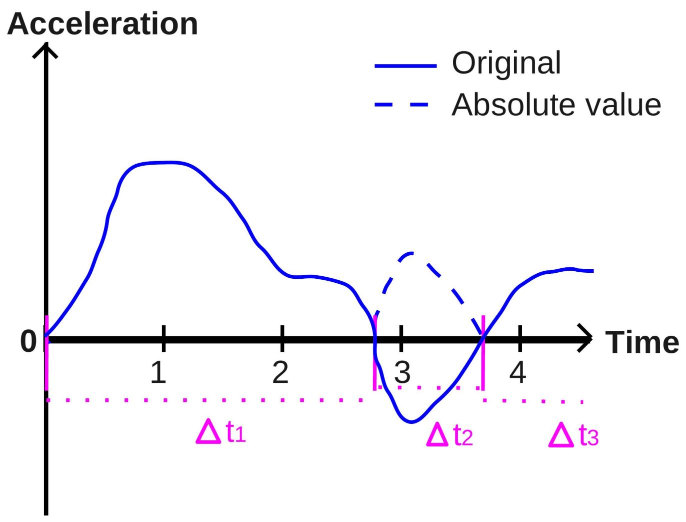


Figure 4.3: Visualization of the Objective Function in the acceleration profile [5]. The figure shows the original acceleration profile in blue. The negative values in the curve are mirrored at zero-level (dashed blue curve). Each zero point (vertical magenta lines) forms a border of a $\triangle \boldsymbol{t}_i$. The black numbers are the original time steps. They refer to the time step x, when the corresponding point $\boldsymbol{p}_x$ is reached along the path. It is clearly visible, that the original time steps do not necessarily form the borders of $\triangle \boldsymbol{t}_i$ in the Objective Function O (Eq. 4.9).

Hence, we limit the possible configuration of the path. Of course, the start and end point of a path are hardly allowed to change. Moreover, points close to the start, resp., end point should lead to the start, resp., end point. The closer a point to the start or end point, the less freedom should be available for the path configuration. Therefore, we introduce a sphere S_x around each point $\boldsymbol{p}_x$ of the path. The sphere comprises the area, within which the point $\boldsymbol{p}_x$ is allowed to be placed during the optimization. The radius r_x of the sphere S_x depends on the position of $\boldsymbol{p}_x$ within the path. The further away $\boldsymbol{p}_x$ from the start and end point, the larger r_x.

If the original points $\boldsymbol{p}_x$ are uniformly distributed along the path, the radius r_x can be computed as follows:

$$r_x = \begin{cases} C \cdot x & \text{if } x \leq \frac{X}{2}, \\ C \cdot (X - x + 1) & \text{otherwise.} \end{cases} \tag{4.10}$$

C is a constant basic distance, X refers to the overall number of points along the path and x is the index of $\boldsymbol{p}_x$. The index x is increasing along the path and it starts with $x = 1$ at the start point. Fig. 4.4 illustrates the spheres along a path. The initial points on the half circle are just chosen to ensure a path for a lifted object. The final points $\boldsymbol{p}_x$ do not need to stay on a circle, since they can be placed arbitrarily within each sphere S_x. In the case of a pushed object, we have to ensure, that each point stays at the original height above the plane on which the object is pushed. Therefore, the sphere S_x is reduced to a circle at the desired height above the plane.

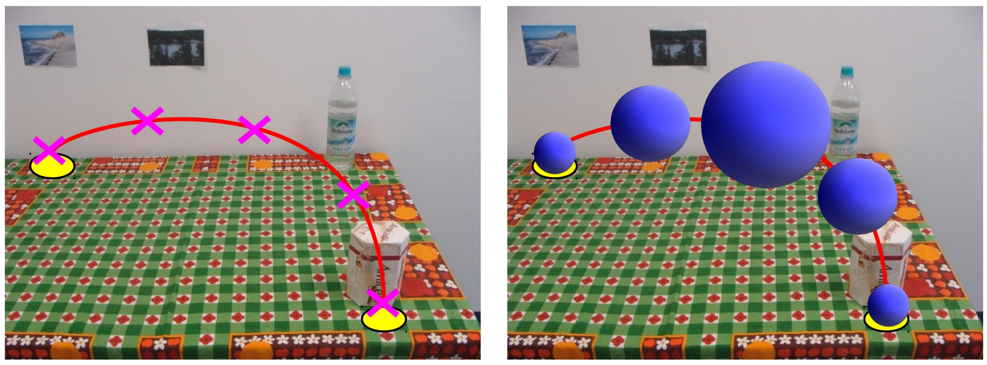


Figure 4.4: Path optimization spheres [5]. The original points of the path are depicted on the *left* (magenta crosses). The corresponding spheres are visualized on the *right* in blue. Each point is just allowed to move within its corresponding sphere.

The settings of the parameters introduced in this Chapter are further discussed in the experiments (Section 7.4).

Chapter 5

Structure Analysis of Manipulators

The aim of this thesis is an *analysis of manipulator structures* in task-specific contexts. We want to compare the capabilities of different robots with respect to *efficient actuation.* The comparison is processed (1) within a robot in the case of joint failures and (2) between robots with or without joint failures. It is important to point out, that the analysis can be processed *independently of the structure* of the manipulator. The results have to be comparable between different manipulator structures. Therefore, an abstract representation of the robot's dynamic capabilities is necessary. We introduce the *Maneuverability Volume* and the *Spinning Pencil* for this purpose. The Maneuverability Volume shows, how efficiently the end-effector can be moved to any other position. The Spinning Pencil reflects the robot's capability to change its end-effector orientation efficiently. The work partly is published in [4].

Today, robots are able to perform different tasks as, e.g., work in the kitchen. The structure of the used robots can be very different, reaching from, e.g., a simple three DoF manipulator to complex systems like humanoid robots. Hence, the robot's capabilities can vary strongly. Some of the robots might have extensive capabilities, whereas others have just a limited ones. The efficiency during the desired motion can be different as well. In an ideal case, the robot moves smoothly and slowly. The workload should be distributed equally among all joints.

Our aim is the comparison of different manipulator structures with respect to efficient actuation. We want to use *one* approach which leads to a result that is *independent* of the structure of the robot. At the same time, the change of the manipulator's capabilities should be analyzable if one or several of its joints fail. Is the robot still able to reach the desired positions at all? Which capabilities of the manipulator remain at which quality and which ones are lost?

Hence, we are interested in an abstraction of the robot's capabilities, which enables an *inter-robot* and *intra-robot* comparison of the efficiency in actuation. The possibly high dimensionality of the manipulator structure has to be reduced to a representation at a manageable

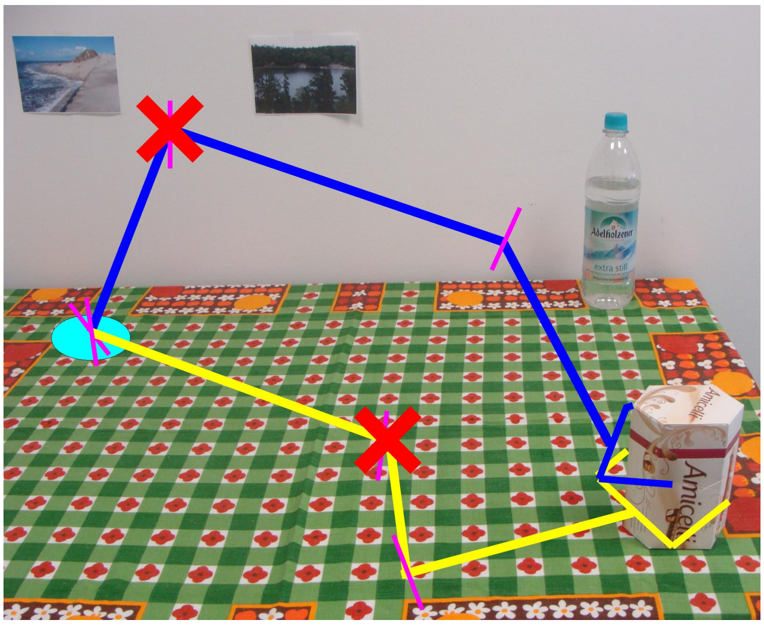


Figure 5.1: Analysis of manipulator structures under joint failure [4]. (light blue circle: robot's base; magenta straight lines on robots: joint axes symbols): What happens, if, e.g., the red-marked second joint of the blue manipulator fails? Is the robot still able to move the box as a task might require? If yes, which effort is necessary? Can the robot still turn the object? How would the yellow robot be affected under a similar joint failure? Would it keep better capabilities than the first one?

dimensionality. Fig. 5.1 illustrates an exemplary scene with two manipulators, which are supposed to perform the same task under joint failure.

We introduce the *Maneuverability Volume* and the *Spinning Pencil* as abstract representation of the robot's capabilities for the inter-robot and intra-robot comparison.

The Maneuverability Volume is a parallelepiped, which represents the robot's capabilities to move efficiently into a set of directions. The starting point of our idea is the following: If a joint is moved by a fixed angular rate, it contributes a certain velocity to the end-effector, which is represented by a *base velocity vector*. The base velocity vectors of all joints can be combined to the desired overall velocity of the end-effector. Of course, it is desirable, that the base velocity vectors point into significantly different directions to enable an efficient motion into a large set of directions. Hence, the vectors should span a large volume to cover a large area. Moreover, the vectors are spanned by the fixed angular rate, which is the same for all joints. Therefore, the magnitudes of the base velocity vectors allow an easy comparison of the efficiency of different joints. A joint which contributes a vector of a large magnitude is more efficient than another joint contributing a vector of a smaller magnitude.

The robot's capability to change its end-effector position efficiently is represented by the described Maneuverability Volume. We want to be able to separately analyze the robot's capabilities to change the end-effector orientation efficiently. In an ideal case, the robot is able to turn its end-effector without changing its position. Otherwise, a change of orientation requires also a correction of the position. For example, the robot could need to rotate an object without changing the position (e.g., a jug needs to be turned to fill coffee into a cup). We can imagine the axis of the desired rotation as the rotation-axis of a spinning top. If an undesired change in position accompanies the change in orientation, we need to model it. We take the change in position of the end-effector as the spinning top's diameter. A spinning top in form of a spinning pencil is desirable, since the pencil reflects the advantageous thin shape of the spinning top. Hence, we represent the robot's capability to change its end-effector orientation by the Spinning Pencil.

We are interested in the analysis of the manipulator structure in a task-specific context. The representation of the task is important to us, since it is the basis on which we work. We want to use an *efficient*, *abstract* representation of a task. Therefore, we develop the concept of Functionality Maps and Location Areas (Section 3.1) further. We make use of the Location Areas as representations of positions, where actions typically start or end. In some areas, only simple tasks are performed. These tasks can even be performed by a robot with limited capabilities. In contrast, complex manipulations can take place at other areas. Consequently, the task determines the areas where manipulations are performed and the capabilities which are required from the robot. For example, a Location Area in a cupboard is used to place an object mainly from one direction, which is the front. In contrast, complex manipulations can be performed at a Location Area on a table. The robot needs better capabilities on the table than at the cupboard. The capabilities which are required by the desired, abstract task need to be integrated into the analysis.

Moreover, we are interested in the potential of path optimization in the context of structure analysis under joint failure. We want to make use of the described path optimization approach (see Section 4.2) for this purpose. Such an optimization can allow to compensate a joint failure at least partly. Hence, we evaluate the potential of path optimization for different robots under the failure of one or several joints. Fig. 5.2 illustrates such a situation.

To sum up, we provide an inter- and intra-robot comparison (1) applicable to manipulators of any structure, (2) with respect to efficient actuation, (3) in task-specific contexts, (4) under the consideration of joint failures (5) including the potential of path optimization.

In this chapter, we present the *analysis of manipulator structures* with respect to *efficient actuation*. It allows the comparison of robots' capabilities. Furthermore, the *potential of path optimization* is analyzed in this context.

As before, we describe the robotic system in the DH-convention [164] in the form shown in [9], the orientation of the robot's end-effector is described as Z-Y-X Euler angles [9], and the overall number of joints is D.

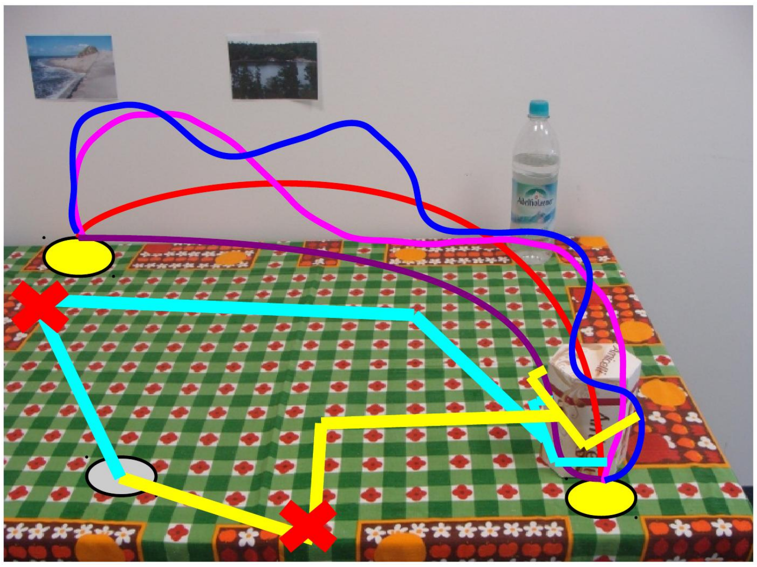


Figure 5.2: Analysis of manipulator structures under joint failure and path optimization (light gray circle: robot's base; magenta straight lines on robots: joint axes symbols): Different path configurations are shown for a manipulation between two areas (yellow). Which path is the most efficient one with respect to actuation? Can an appropriate choice of the path at least partly compensate a joint's failure, as, e.g., the failure of the second joint of the cyan robot?

5.1 Maneuverability Volume

At a (reachable) position, the robot can move its end-effector into a larger or a smaller set of directions depending on the design and the current configuration. The content of the set depends on the motion of each joint, since each joint j contributes a certain linear velocity vector $\boldsymbol{v}_j$ to the overall velocity $\boldsymbol{v}$ of the end-effector. As in Section 4.2, the overall velocity of the end-effector $\boldsymbol{v}$, the linear velocity vector $\boldsymbol{v}_j$ contributed by a rotational joint j and the rotational velocity $\boldsymbol{\omega}_j$ caused by joint j are defined as follows:

$$\boldsymbol{v} = \sum_j \boldsymbol{v}_j \tag{5.1}$$

$$\boldsymbol{v}_j = \boldsymbol{\omega}_j \times (\boldsymbol{p}_{ee} - \boldsymbol{p}_j) \tag{5.2}$$

with $\boldsymbol{p}_{ee}$ as the position of the end-effector, $\boldsymbol{p}_j$ as the position of joint j and $\boldsymbol{\omega}_j$ as the rotational velocity caused by joint j:

$$\boldsymbol{\omega}_j = \dot{\theta}_j \cdot \hat{\boldsymbol{Z}}_j \tag{5.3}$$

with $\hat{\boldsymbol{Z}}_j$ as 3-dimensional unit vector of the z-axis of joint j and $\dot{\theta}_j$ as magnitude of the angular rate (see, e.g., [9]).

If we consider a fixed angular rate $\dot{\theta}_{fix}$ for each joint j, the magnitude of the rotational velocity $\boldsymbol{\omega}_{j,fix}$ is the same for all joints, since $\hat{\boldsymbol{Z}}_j$ is a unit vector. We call the resulting linear velocity *base velocity* $\boldsymbol{d}_j$ of a joint j. It depends on the current configuration:

$$\boldsymbol{d}_j = (\dot{\theta}_{fix} \cdot \hat{\boldsymbol{Z}}_j) \times (\boldsymbol{p}_{ee} - \boldsymbol{p}_j) \tag{5.4}$$

or

$$\boldsymbol{d}_j = \boldsymbol{\omega}_{j,fix} \times (\boldsymbol{p}_{ee} - \boldsymbol{p}_j) \tag{5.5}$$

A desired end-effector speed can be combined by the base velocities:

$$\boldsymbol{v} = \sum_j \boldsymbol{v}_j = \sum_j (g_j \cdot \boldsymbol{d}_j) \tag{5.6}$$

with a scalar g_j. Of course, it depends strongly on the vectors of the base velocity, whether and at which effort a desired end-effector speed can be achieved. If low joint speeds are desired, the base velocity vectors should have a large magnitude (resulting in a small effort g_j in Eq. 5.6). Moreover, the vectors $\boldsymbol{d}_j$ (and their counterparts, pointing in the opposite direction) should be perpendicular to each other. A combination of the vectors $\boldsymbol{d}_j$ can, then, efficiently reach any direction in space. At the same time, the robot is further away from a singular configuration. Such a singularity would be reached, if at least two vectors point into the same direction.
We discuss the settings of the parameters introduced in this Chapter (e.g., $\dot{\theta}_{fix}$) in detail in the experiments (Section 7.5).

Three base velocity vectors span a volume (parallelepiped), which increases, when the vectors are rather perpendicular to each other and when the magnitude of the vectors is enlarged. A parallelepiped can be computed by a triple product V:

$$V = trip(\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3) = (\boldsymbol{d}_1 \times \boldsymbol{d}_2) \cdot \boldsymbol{d}_3 \tag{5.7}$$

The parallelepiped K of three base velocity vectors is, therefore:

$$K = trip(\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3) \tag{5.8}$$

An advantageous parallelepiped has a large value K.

Moreover, an evaluation of the maneuverability independently of the length of the base velocity vectors $\boldsymbol{d}_j$ is interesting. It allows to ensure, that the base velocity vectors are perpendicular to each other as good as possible. First, a normalization is executed: $\hat{\boldsymbol{d}}_l = \frac{\boldsymbol{d}_l}{\|\boldsymbol{d}_l\|}$. Afterward, just the angles between the vectors are considered in the parallelepiped L:

$$L = trip(\hat{\boldsymbol{d}}_1, \hat{\boldsymbol{d}}_2, \hat{\boldsymbol{d}}_3) \tag{5.9}$$

The combination of both parallelepiped K and L gives us the desired measurement of maneuverability, which we call the *Maneuverability Volume mV*:

$$mV = \omega_k \cdot trip(\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3) + \omega_l \cdot trip(\hat{\boldsymbol{d}}_1, \hat{\boldsymbol{d}}_2, \hat{\boldsymbol{d}}_3) \tag{5.10}$$

with the weightings ω_k and ω_l to enable different priorities of the desired properties. An exemplary parallelepiped K is illustrated in Fig. 5.3.

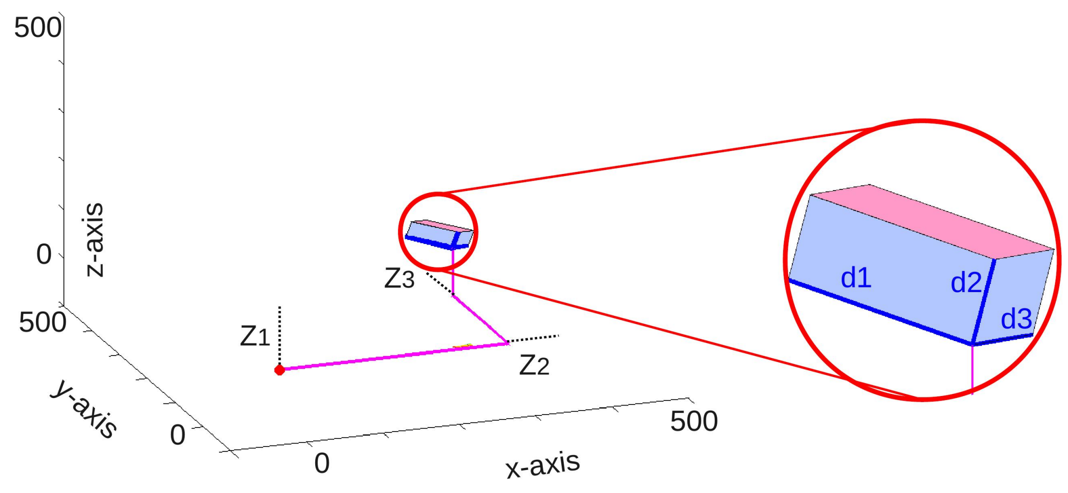


Figure 5.3: Parallelepiped (part K of the Maneuverability Volume) [4]. A robot with three rotational joints is drawn in magenta (unit: mm). Its joints can turn around the black dotted z_j-axes. Each joint contributes one of the blue base velocity vectors $\boldsymbol{d}_j$, when it moves with the speed $\dot{\boldsymbol{\theta}}_{fix}$. E.g., the largest vector $\boldsymbol{d}_1$ is contributed by joint 1. The three blue base velocity vectors $\boldsymbol{d}_j$ span the light blue/ light red parallelepiped. The red dot is the robot's base.

The Maneuverability Volume can be spanned by three vectors. If the robot has more than three joints, we combine the base velocity vectors $\boldsymbol{d}_j$ to three final velocity vectors $\boldsymbol{d}_{j,f}$. We just need to check all combinations of the vectors $\boldsymbol{d}_j$ to the final $\boldsymbol{d}_{j,f}$ to find the most advantageous Maneuverability Volume. For example, six base velocity vectors $\boldsymbol{d}_j$ should be combined to three final velocity vectors. One final velocity vector $\boldsymbol{d}_{1,f}$ can consist of one, two, three or four different vectors $\boldsymbol{d}_j$, so that the remaining two final velocity vectors $\boldsymbol{d}_{j,f}$ consist of at least one vector $\boldsymbol{d}_j$. Each vector $\boldsymbol{d}_j$ can just be assigned to one of the final velocity vectors $\boldsymbol{d}_{j,f}$. Consequently, our analysis takes place in the 3D-space, where the manipulations takes actually place. A projection to another space is not necessary.

In contrast to the traditional approach with the determinant of a Jacobian, we are include in our Maneuverability Volume (1) an additional emphasis on the desired 90° angle between the base velocity vectors and (2) a new extension to manipulators with more than three DoF.

5.2 Spinning Pencil

The next step is the analysis of a manipulator's capability to change its end-effector orientation efficiently. It is desirable, that the orientation can be changed around any axis at the position of the end-effector, while the position of the end-effector is not changing. For example, a jug needs to be turned around a certain axis to fill a cup with coffee, while the center of the jug should stay at its original position. As already described in the at the beginning of the Chapter,

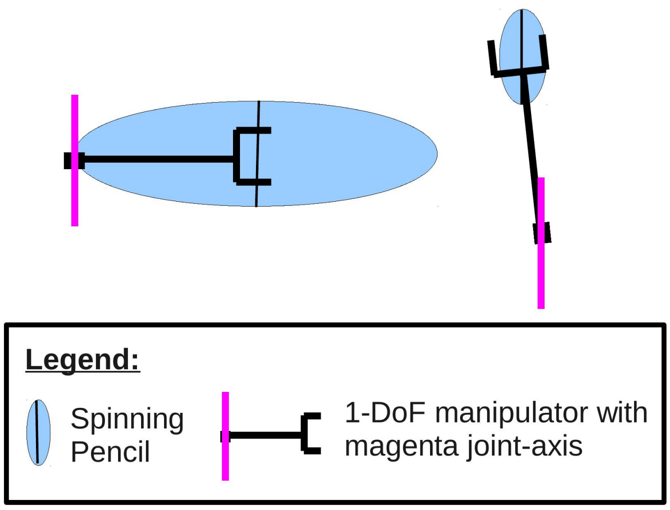


Figure 5.4: Spinning Pencil [4]. Two different 1-DoF manipulators are illustrated. The thin black line in each blue Spinning Pencil shows the length and the orientation of the pencil. Both pencils have the same length and point into the same direction. The different magnitudes of the diameters are due to the fact, that the end-effector on the left conquers a much larger distance to change its orientation than the end-effector on the right.

we imagine a spinning top, which reflects these properties: The spinning top rotates around a certain axis. In the case of a manipulator, the end-effector can turn around one (or more) of its joint axis. Depending on the robot's configuration, a change of the orientation can automatically and unpreventably result in an undesired position change. The distance between the position of the end-effector before and after the rotation is chosen as the diameter of the spinning top. A small diameter is, of course, desirable. Hence, we introduce the *Spinning Pencil*, since the pencil reflects the small diameter. Fig. 5.4 illustrates the principle of the Spinning Pencil.

We want to analyze the robot's capabilities to change the orientation of its end-effector efficiently, while keeping the desired position $\boldsymbol{p}_{ee}$. Therefore, we turn joint j by an angle γ. In an ideal case, the end-effector stays at $\boldsymbol{p}_{ee}$ independently of the magnitude of γ. Otherwise, the end-effector moves to a new position $\boldsymbol{p}_{n,j}$. The euclidean distance between the old position $\boldsymbol{p}_{ee}$ and the new one $p_{n,j}$ is labeled with $e_j = \|\boldsymbol{p}_{n,j} - \boldsymbol{p}_{ee}\|$. We compute the mean μ_e and the variance σ_e of all e_j (all joints j), which should be small in an advantageous Spinning Pencil.

Of course, the robot should be able to change the end-effector's orientation around any arbitrary axis running through the original position $\boldsymbol{p}_{ee}$. However, the robot's capability to turn depends on the z-axes of its joints. Hence, z-axes which point nearly in the same direction provide almost the same axis for the change of orientation. The more different the orientation of the z-axes, the more axes are provided for the orientation change. In the ideal case, the z-axes are perpendicular to each other. Therefore, we compute the normalized z-axis ${}^{ee}\hat{\boldsymbol{Z}}_j$ of each joint j in the frame of the end-effector. Afterward, we analyze the angles between the z-axes ${}^{ee}\hat{\boldsymbol{Z}}_j$.

Once again, we want to process the analysis in 3D space, where the manipulations actually take place. Since the manipulator can be redundant, it might be necessary to analyze more than three z-axes. Hence, we analyze all possible combinations of z-axes. We compute the determinant of three normalized axes, since it reflects the angles between the axes: If all three axes are perpendicular to each other, the determinant reaches its maximum of one. The smaller the determinant, the less perpendicular the axes are to each other. Therefore, we take a combination c of three z-axes ${}^{ee}\hat{\boldsymbol{Z}}_j$ and compute the determinant $s_{1,c}$:

$$s_{1,c} = det({}^{ee}\hat{\boldsymbol{Z}}_{c1}, {}^{ee}\hat{\boldsymbol{Z}}_{c2}, {}^{ee}\hat{\boldsymbol{Z}}_{c3}) \tag{5.11}$$

with $c1$, etc., as labeling of the three chosen z-axes ${}^{ee}\hat{\boldsymbol{Z}}_j$. The determinant $s_{1,c}$ is computed for all $n_1 = \binom{D}{3}$ combinations. All determinants $s_{1,c}$ are summed up and normalized with the overall number of combinations n_1:

$$s_1 = \frac{\sum_c s_{1,c}}{n_1} \tag{5.12}$$

It is possible, that at least some z-axes lie in a plane. In this case, the determinant in Eq. 5.11 becomes zero. Therefore, we analyze the angle between two z-axes additionally. The dot product is used for this purpose. Similarly to $s_{1,c}$, we compute the dot product $s_{2,c}$ for all combinations of two z-axes:

$$s_{2,c} = {}^{ee}\hat{\boldsymbol{Z}}_{c1} \cdot {}^{ee}\hat{\boldsymbol{Z}}_{c2} \tag{5.13}$$

The results of $s_{2,c}$ are projected into the range of 0-90°. The final s_2 consists of the sum of all $s_{2,c}$, which is normalized with the total number of combinations $n_2 = \binom{D}{2}$:

$$s_2 = \frac{\sum_c s_{2,c}}{n_2} \tag{5.14}$$

The values of s_1 and s_2 should be large in an advantageous case. In contrast, the mean μ_e and the variance σ_e should be minimized. Therefore, we combine all parts of the Spinning Pencil sP as follows:

$$sP = \omega_1 \cdot s_1 + \omega_2 \cdot s_2 + \omega_3 \cdot \frac{1.0}{\mu_e + \sigma_e} \tag{5.15}$$

with ω_i as weighting of the respective property. All terms of the sP should be large in an advantageous case. The third term $\frac{1.0}{\mu_e+\sigma_e}$ is increasing, when the mean μ_e and the variance σ_e are decreasing as desired.

In contrast to traditional approaches with the Jacobian, we consider also (1) rotation axes lying in a plane and (2) the undesired effect of a position change which can accompany a change in orientation.

5.3 Maneuverability Analysis under Joint-Failures

Next, we want to analyze the remaining maneuverability of a manipulator, if one or more of its joints fail. First, we compute the original maneuverability. We combine the Maneuverability Volume and the Spinning Pencil in the *Maneuverability Analysis* MA_l at Location Area l:

$$MA_l = \omega_m \cdot mV + \omega_s \cdot sP \tag{5.16}$$

with a weighting ω_m and ω_s to enable different preferences of both properties (sum of ω_m and ω_s equals 1.0).

After the computation of the original MA_l, the Maneuverability Analysis is processed for the manipulator with the broken joint. The manipulator with the broken joint has simply one DoF less than the original one. The DH-parameters need just to be changed around the broken joint: A direct transformation from the joint before the broken joint to the joint after the broken joint is needed. The joint angle θ_b of the broken joint j_b can be treated as a fixed parameter. We just need to process the above Maneuverability Analysis for the new manipulator, since it can be computed for manipulators with any number of joints. Therefore, the procedure can easily be extended to a manipulator with multiple broken joints.

5.4 Analysis of Multiple Location Areas

Up to now, we described the approach for one Location Area. Most of the times, the robot has to be able to work at multiple Location Areas. The requirements on the maneuverability depend on the type of Location Area. Therefore, we introduce a weighting ω_l of each Location Area l depending on the level of the required maneuverability. A high maneuverability requirement corresponds to a high weighting of the Location Area.

The final objective function MA contains the Maneuverability Analysis MA_l of each Location Areas l:

$$\mathrm{MA} = \sum_l \omega_l \cdot MA_l \tag{5.17}$$

with the weighting ω_l of Location Area l. The higher the final value MA, the better the maneuverability.

5.5 Structure Analysis under Path Optimization

We introduced already the Maneuverability Analysis under joint failure in the previous Sections. Now, we are interested in the potential of path optimization in this context. Hence, we apply the path optimization approach presented in Chapter 4 in the context of a manipulator structure analysis.

Regarding the exiting path optimization method, we need an extension with respect to (1) the inclusion of the orientation and (2) the application in the analysis. In the original

EPP (Chapter 4), each point along the path consisted just of its position. In our experiments here, we include the orientation, since we want to contribute the results to the original Maneuverability Analysis (Section 5.3). There, the orientation has been considered. Similarly to the original analysis in Section 5.3, the object is supposed to be kept upright during the manipulation (e.g., necessary for a cup filled with coffee). Hence, the new EPP-value contains, now, also a residual r. It is the summed absolute difference between the desired and the real orientation (α_d, α_r) of the end-effector of the manipulator. We include the difference for both horizontal orientations (angle α_a with a={1,2}), since the object can be kept upright, even if the vertical orientation (the third angle, α_3) changes:

$$r = \gamma_{EPP} \cdot \sum_{a=1}^{A=2} \|\alpha_{d,a} - \alpha_{r,a}\| \tag{5.18}$$

The variable γ_{EPP} allows a stronger penalization of undesired differences between the desired and the real orientations.
The residual r is added for each joint d at each point p_i in the Objective Function O (Eq. 4.9). The resulting, new EPP-value indicates, how good the efficiency can be kept, while the object has an upright orientation. Hence, we can compare the EPP-values (1) regarding improvements in case of joint failures and (2) between different robots. They show the changes, resp., differences in efficiency. The smaller the value is, the better the efficiency is.

The optimization is applied on all paths which the robot is supposed to perform. The result of the overall Objective Function O (Eq. 4.9) is computed for each path. We, then, compute the mean of all O:

$$\mathrm{O}_{mean} = \frac{\sum_{pa} \mathrm{O}}{AP} \tag{5.19}$$

The total number of paths is AP. Each single path is labeled with pa. Additionally, we select the maximum of all O:

$$\mathrm{O}_{max} = \max_{pa} \mathrm{O} \quad . \tag{5.20}$$

The mean O_{mean} gives us a general impression about the manipulator capabilities under joint failure and path optimization. The maximum O_{max} refers to the worst case, since very small values are desirable in the evaluation of the overall Objective Function O (Eq. 4.9). Both values are used to analyze the potential of path optimization in comparison to the basic Maneuverability Analysis without optimization. The basic Maneuverability Analysis itself is much more general, since it is less coupled to under-lying, possibly very constrained manipulations. This is twofold: One the one hand, it is more unaffiliated. On the other hand, a potential path optimization cannot be considered. Overall, one can use the application of the proposed path optimization to provide further results in the structure analysis.

Chapter 6

Analysis of Master-Slave Systems

In this Chapter, we want to indicate a further application area of the proposed structure analysis: Telemanipulation systems. In principle, a telemanipulation system consists typically of a master system and a slave system. A human user controls the slave through the master system. Hence, the human is moving the master system and the slave system is executing the movements. Both systems can be separated through a large distance. Such master-slave systems have a large number of applications, reaching from telemanipulation in, e.g., disaster operations to medical applications. The choice to deploy a certain master-slave system can significantly be influenced by the level of convenience in system usage. If one system is more convenient to use than another one, it has a clear advantage. Hence, an analysis of master-slave systems can provide useful information to the system designers and the decision-maker.

Different factors determine the level of convenience in system usage. Haptic feedback or appropriate system structures are examples of such factors. We focus on the structures similarly to the proposed analysis of serial chain manipulators before. The path optimization method is utilized to measure the "efficiency in actuation" for the human. The efficiency of the manipulation system itself can be seen secondary aim. The aspired analysis with respect to a convenient usage for the human user is much more interesting, since it can provide an important argument to deploy a system.

Further aspects are important in the analysis. For example, an appropriate model of the human's hand-arm-system is necessary to base the analysis on realistic motions of the human. Moreover, the human users do usually differ with respect to their anatomy and their behavior. Such differences have to be considered in the analysis. As it can be seen, both aspects are extensive topics themselves. We focus on the actual analysis of telemanipulation systems in this work. Both other components are implemented as simple basic versions to allow an illustration of the potential of telemanipulation systems analyses.

Of course, the question arises, how the human is usually planning and performing arm movements. In general, a lot of trajectories seem to be possible. This offers diverse possibilities to optimize the motion. Such an optimization can take place with respect to different criteria.

The first one is the optimization with respect to the hand trajectory (especially, minimum-jerk model by Flash and Hogan [176]). The second criteria is an optimization in the joint space (especially, minimum torque-change model by Uno *et al.* [177]). Dean and Brüwer [178] favor a mixture of both. The resulting joint velocity profiles of their study show a preference of smooth curves as long as possible. This indicates, that our path optimization method (Chapter 4) can be applied on the analysis of joint motions for a given path. Further criteria for optimization are discussed in [179]. Cost functions are also included there. More complex movements are considered in [180]. Discrete and rhythmic movements of a multi-joint human arm are simulated there. The underlying optimization takes place in joint space. The proposed optimization here is also processed in joint-space. The influence of different arm lengths has been analyzed in the optimization of two-joint arm movements in [181]. The results, there, indicate, that it is worth to consider the different anatomy of human users.
A drawback of the described work is the limitation to a few number of DoF for the human arm (often, two or three DoF). In contrast, our approach considers a model human arm with 7 DoF. An extension to more DoF is easily possible. Moreover, any desired given path can be analyzed with our proposed method. Hence, we can apply our analysis to complex movements (e.g., in the medical area).

To conclude, we provide a basic simulation in the context of the analysis of arbitrary telemanipulation systems. The previously developed methods need to be adapted appropriately to telemanipulation systems.

6.1 Structure Analysis: Autonomous vs. Non-autonomous Systems

In the previous chapters, the analysis of serial chain manipulators was presented. The robots are considered to act autonomously based on abstractly represented task knowledge. Now, we are interested in the analysis of master-slave systems. This application differs significantly, since the systems are controlled by a human user. Hence, important aspects are changing:

1. **Task knowledge**: The robot does usually not need (abstractly represented) task knowledge. It is directly controlled by a human user. Of course, one could discuss, whether the robot could profit from a partial knowledge in certain situations. Semi-autonomous behavior for at least some subsequent steps could be advantageous (e.g., [182]). If one has, e.g., to deal with a large time delay due to large distances between the master and the slave (e.g., [97]), a direct and fast "reactive behavior" to environmental influences could be desirable from the slave. We focus on entirely non-autonomous robots in this chapter, since this type of application has not been presented yet. Semi-autonomous systems could be considered later on. A combination of analyses of non-autonomous and autonomous robots could be applied, then.

2. **Path constraints**: The path of the executing slave is, most of the times, entirely fixed, since the master system commands the exact path directly. In our application, the paths are entirely fixed, since we focus on non-autonomous systems. Of course, one could argue, that a modification of the existing system could improve the convenience in the system usage. However, we are interested in the analysis of the original, unbiased usage of the system.

3. **"Systems" with potentials for optimization**: Three "systems" are involved during the execution of a task with a non-autonomous master-slave system. The first one is the executing slave. It just executes the commanded path. Hence, the path is fixed and it cannot be changed for optimization. Possible redundancy can offer the only optimization potential. The second involved system is the master. It is controlled by the human. Consequently, it does not provide potential for optimization except (rare) redundancy. The third "system" is the controlling human. The human's action is affected by (1) possible task constraints and (2) the workspace boundaries of the involved systems. However, he or she can choose the path freely within these two types of constraints.

To conclude, the only optimization potential is available on the human's side. Such an optimization can improve the convenience of the usage of the system. Of course, a convenient system usage is desirable for the human. It is an important aspect for the deployment and the acceptance of the system. Hence, the execution should be optimized with respect to the human user at first. The optimization regarding the actuation of the console and the executing robot plays just a secondary role here. In contrast to the previously presented Maneuverability Analysis, the perspective has changed: The human is in the focus now.

Fig. 6.1 illustrates a telemanipulation scene with three exemplary human arms (including different arm lengths and positionings of the shoulders).

6.2 Master-Slave Systems: Workspace Mapping

In comparison to the presented Maneuverability Analysis, the analysis of master-slave systems differs also with respect to the Location Areas. These characteristic areas of manipulations are determined in the workspace of the slave. The mapping into the master's workspace cannot be done straight forward. This is due to the different workspaces and workspace boundaries. The Location Areas can be projected to different positions in the master's workspace depending on the application, the user, the way of executing the desired task and the variations which can occur in repetitions of the task. Hence, we do not provide an analysis with respect to Location Areas.

However, the Elastic Power Path does not rely on Location Areas, but on possible paths. Hence, the concept can be utilized to analyze the efficiency. We simply take the Objective

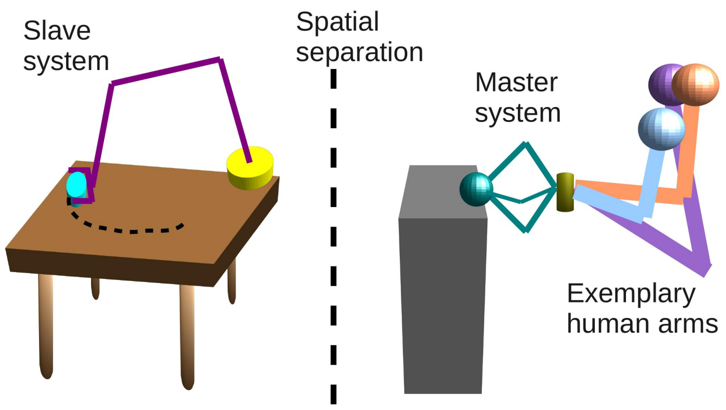


Figure 6.1: Exemplary scene with a master-slave system. The figure shows an exemplary slave system on the left (magenta slave with yellow base). It is executing a manipulation on the cyan object (symbolized by the dashed black line on the table). The manipulation is controlled by a human through an exemplary master system on the right (dark cyan master). The master and the slave system can be separated by a large distance. Three exemplary human's arms of different lengths and shoulder positionings (respective spheres) are illustrated.

Function O of the Elastic Power Path (see originally Eq. 4.9) to measure the efficiency along a path:

$$\mathrm{O} = \sum_j \sum_i \frac{2 \cdot |\boldsymbol{v}_{j,i}| \cdot | \triangle \boldsymbol{v}_{j,i}| + (| \triangle \boldsymbol{v}_{j,i}|)^2}{\triangle \boldsymbol{t}_i}. \tag{6.1}$$

In the analysis of master-slave systems, the human is in the focus as we motivated already. Hence, the joints j of the human's hand-arm "system" are relevant for the Objective Function O. Of course, one has to consider the workspace boundaries during the path execution. It might be necessary to reset the master system during the execution, since a workspace boundary is reached. This causes additional work to the human user, which has to be considered in the analysis. The concept of the Elastic Power Path allows the consideration of additional work, since the *entire* path in the workspace of the master can be analyzed using the Objective Function O. This includes also the path to reset the system. Therefore, the path consists of all hills i in the acceleration profile of this entire path. The velocities $\boldsymbol{v}_{j,i}$, resp., accelerations $\triangle\boldsymbol{v}_{j,i}$ along the entire path are considered. As in Eq. 4.9, $\triangle\boldsymbol{t}_{j,i}$ is the time required to climb up and down hill i.

6.3 Structure Analysis: Limits and Potentials

We have already seen, that the optimization potential is mainly available on the human's side of the system. A convenient usage of the system is important as we motivated already. The convenience can be supported by many aspects, like easy handling, advanced feedback from the executing system or comfortable, efficient usage of the system. We focus on the comfortable, efficient usage for the human. It has similarities to an efficient actuation of a serial chain manipulator. Smooth and slow motions are preferred by human users analogically to manipulators (see also beginning of the Chapter). Abrupt turnarounds in the acceleration are not desirable. If a reset is required during the execution of a task, the resulting additional movement has to be added as further, less desired work.

One way to optimize the "actuation" of the human user, is, of course, training of the human. A lot of work exists already in this field as the related work shows. Our focus is the *analysis* of the human's arm and hand motions during task execution. We consider the entire hand-arm-system of the human. Its motions can, of course, not only be influenced by the master system, but also from the positioning and the size of the human's hand and arm (see illustration in Fig. 6.1).

A possible long-term application is a system analysis before the actual system is built. Hence, we simulate our experiments. In this work here, we want to indicate the potential of the Elastic Power Path concept as a framework for the efficiency analysis of a human's motion during the work on a master-slave system.

In order to provide a sophisticated analysis tool, an advanced human hand-arm model would be necessary. However, just basic modules for arm, resp., hand detection exist (e.g., [183], [184], [185]). The build-up of an advanced model of the human hand-arm system is out of the focus of this thesis. Currently, there is ongoing work in this area (e.g., human arm model with redundancy [186], gesture recognition based on depth data [187]). A satisfying model is still missing. We use the currently available DH parameters for the dynamic model of the human arm described in [188].

Chapter 7

Experiments

The experiments are presented in this chapter. First, the extraction and representation of the underlying validation scenarios is shown. The performance of the required advanced estimation of inverse kinematics is demonstrated afterward. Experiments on the proposed path optimization are presented thereafter. The structure analysis follows, then. A further application of the structure analysis is shown in the last section of this chapter: The application on master-slave systems.

7.1 Validation Scenarios

First, two data sets are extracted from the observation of real human actions. The human manipulated objects between characteristic places. The observed sequences (seq.) are processed to build up an abstract task representation as described in Section 3.1.1. In order to show the capabilities of the proposed knowledge representation, the system is evaluated on data perceived with different recording devices. First, the results on data from an external tracking system (called "tracking data") are presented (Section 7.1.1, data set I). The tracking data provides directly the 6 DoF-trajectories of the tracked markers, which are placed on top of the manipulated objects. Afterward, the system is tested on vision data in Section 7.1.2 (data set II). An extensive description can be found in [189]. Fig. 7.1 illustrates both data sets.

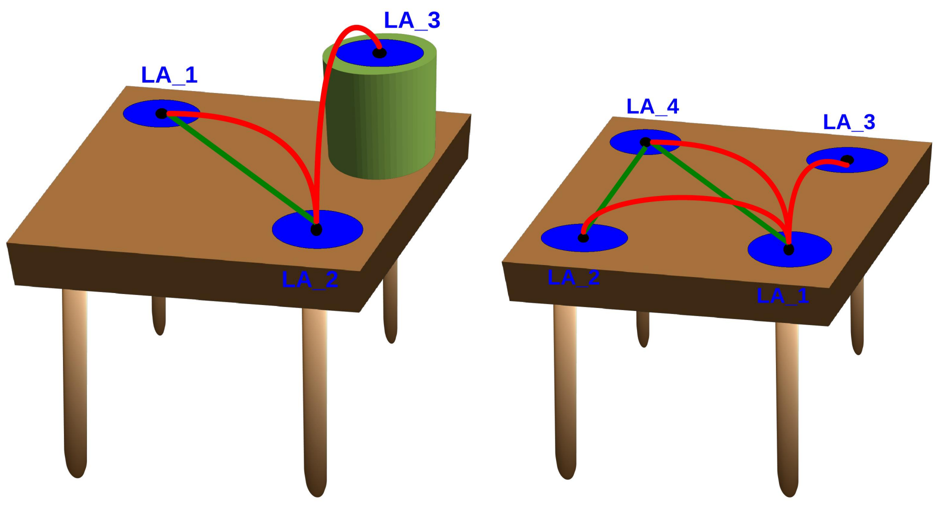


Figure 7.1: Data set I (left) and II (right) [2]. The lines/ curves between the blue Location Areas (**LA**) refer to trajectories of lifted objects (red) and trajectories of pushed objects (green).

7.1.1 Experiments on an External Tracking System

The tracking data is recorded with a marker-based IR tracking system[1] at 50 Hz. A basic pre-processing is performed, first. Each movement has to fulfill the requirement of an at least minimal motion ($> 0.005 m/s$). If the motion is slower than the threshold, it is deleted, since it does hardly provide any information at all. Moreover, we apply a smoothing of the trajectory with a 1.4 s moving-average-window against high-frequency noise, which can especially occur at the beginning of the movement. Afterward, the sequences vary between 4.3 s and 17.72 s, the average is 7.45 s. Examples of trajectories are shown in Fig. 7.2.

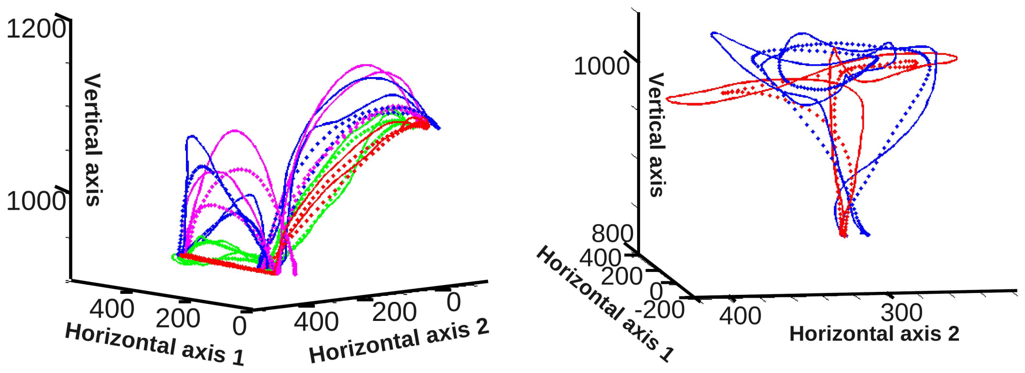


Figure 7.2: Trajectories of movements - tracking data (in mm). [2] The lines refer to the original movements. The dotted lines are the result after the basic pre-processing with the requirement of an at least minimal motion and the described smoothing. *Left:* Milk carton. The line movements are shown in red (without rotation) and in green (with rotation). The curve movements are drawn in magenta (without rotation) and in blue (with rotation). *Right:* Cup. The trajectories of the arbitrary movements are shown in red (without rotation) and in blue (with rotation).

The sequences are recorded with four different objects: a milk carton, a spoon, a cup and a vase. The cup is used in two ways: First, it is grasped at the handle. Then, the cup is grasped as a whole from the side, leading to five different functional objects for the test. For each object, there are 18 different actions of one person, shown in Table 7.1.

7.1.1.1 Object Container

In order to fill the Object Container with the relevant information, we need to extract the relevant properties from the observed manipulations. As a first step, we distinguish motions with rotations and rotations-free ones. Hence, we apply the HMMs as motivated in Section 3.1.2.3. The HMMs require an observation sequence as input. As described, the original sequence of

[1] Advanced Realtime Tracking system. Advanced Realtime Tracking GmbH, *url:* http://www.ar-tracking.de/

Table 7.1: Sequence properties - tracking data. *Left*: Description of the constrained trajectories and arbitrary movements. Constrained trajectories contain movements of pushed objects which are not lifted from a plane. The trajectories of the lifted objects can form a line or a curve. *Right*: Further description of the four constrained trajectories (seq. 1-4, 5-8, 9-12, 13-16): The object is first lifted from the initial position on the table to a higher position on a box. Then, it is moved back. Afterward, it is moved into the corner of the table, which is on the same level of height like the initial position, and moved back.

Seq.:	Movement	Rotation	Seq.:	Start Pos.	End Pos.
1-2	constr.: line		1	table-start	box
3-4	constr.: pushed		2	box	table-start
5-8	constr.: curve		3	table-start	corner
9-12	constr.: line	x	4	corner	table-start
13-16	constr.: curve	x			
17	arbitrary				
18	arbitrary	x			

observed changes around both horizontal axes is, therefore, converted into the desired observation sequence. For this purpose, we build up the desired rotation information codebook using training sequences (Section 3.1.2.3). The pre-processed elements of the training sequences need to be clustered. Each cluster symbolizes a characteristic range of input-values in the codebook, afterward. We set the initialization of the cluster, otherwise the results are not always deterministic, even though they look mostly very similar. The initialization values can chosen arbitrarily for this purpose. They just need to be in the range of 0.00 and 1.00, since the input values (normalized deviations of the angles) are in the same range. We simply take values between 0.00 and 1.00 with a step size of 0.10 and values between 0.00 and 0.10 with step size of 0.01. Resp., two values form a pair for the initialization of one cluster, since the clustering is performed in 2D. A change of the initial values (other combinations or/ and other values within 0.00 and 1.00) lead to very slightly different clusters which look mostly very similar to other clusters formed with the current initialization.

The rotation information codebook consists of 64 symbols. If we consider the range of the input-values (0.00 and 1.00) and assume an uniform distribution of the symbols, each symbol would represent a range of appropriately 0.02. This means, that it would represent 2 percent of the maximally range of 1 for each object. Hence, even small ranges could be represented.

Each HMM has ten states. A further evaluation of different number of states is not within the scope of the thesis. The current setting fulfills its purpose. The experiments show good results as we will see. The knn-assignment of a new value to a cluster is done with k=3. We just tested an assignment with k=5, which did not change the outcome. Similarly to before, the current setting fulfills its purpose: The experiments show good results.

Table 7.2: Statistical result of the classifications - tracking data. Statistical result of the rotation-classification and the classification of the (non-)arbitrary movements.

Property:	Accuracy	True positive rate	True negative rate
Rotation	80.0%	66.7%	93.3%
Arbitrary Movement	94.4%	80.0%	96.3%
Pushed Object	98.9%	100.0%	87.5%

We use the Matlab Statistics Toolbox for the HMM and the clustering with the K-means algorithm. The knn-classification is done with the Matlab Bioinformatics Toolbox .[1]

For the rotation classification, a leave-one-out cross validation is made. 42 of 45 of the motions without rotation are correctly labeled, and 30 of 45 motions with rotation are correctly classified. Therefore, the system performs definitely better than guessing the classification (ground truth: 50%), and performs quite well (see statistical measures in Table 7.2).

The statistical measures are computed with the following equations ($\# =$ number, $t_p =$ *true positives*, $t_n =$ *true negatives*, $f_p =$ *false positives*, $f_n =$ *false negatives*):

$$accuracy = \frac{\#t_p + \#t_n}{\#t_p + \#f_p + \#t_n + \#f_n} \tag{7.1}$$

$$true\ positive\ rate = \frac{\#t_p}{\#t_p + \#f_n} \tag{7.2}$$

$$true\ negative\ rate = \frac{\#t_n}{\#t_n + \#f_p} \tag{7.3}$$

The different statistical measures show the learned capabilities in detail. A basic measure is the accuracy, which gives the percentage of correctly classified sequences among all sequences. It does not take into account, that there are different numbers of sequences for each class. If most of the sequences belong to one "majority"-class (e.g., fictional example "majority": 80%), the system might just guess the "majority"-class for each classification task. This leads to a high accuracy (example "majority": 80%), even though the system does not learn the classification. Therefore, further statistical measures are necessary. The described scenario can be detected with the computation of the true positive rate and the true negative rate (example "majority": 100% and 0%), since it shows the rate of correct classifications for each class in a binary classification task.

The final result of the Object Container can be seen in Table. 7.3. In order to make the Object Container more convenient, acceleration classes are introduced. The number of acceleration classes is chosen to be three for illustration. Each class represents an approximately equal sized part of the achieved acceleration values within $0.003 - 0.01$.

[1]Matlab: Statistics Toolbox, Bioinformatics Toolbox.

Table 7.3: Result for the Object Container - tracking data. It shows the number of observations for each acceleration class and the rotation-classification. Legend: R = motion with rotation, Acc. class = acceleration class: acc. class 1 for $x < 0.006\ m/s^2$, acc. class 2 for $0.006\ m/s^2 <= x < 0.009\ m/s^2$ and acc. class 3 for $0.009\ m/s^2 <= x$.

Acc. class	1		2		3	
Object Tracking	no R	R	no R	R	no R	R
Milk	3	3	6	3	3	0
Spoon	3	1	3	4	4	3
Cup-handle	2	4	3	8	1	0
Cup	5	2	8	1	2	0
Vase	10	4	4	0	0	0

7.1.1.2 Functionality Map

The result of the computed Location Areas **LA**_i is shown in Fig. 7.3. We assume in this experiment, that the correct number of Location Areas (three) is known. Hence, we process the *k*-nearest-neighbors-method is processed with *k*=3 to determine the Location Areas. All three Location Areas are identified correctly.
If the number of Location Areas is not given, the system needs to be able to distinguish the clusters itself. The stop-points of the manipulations (start and end position of each sequence) need to be clustered for this purpose. One could, e.g., introduce a maximal distance between a stop-point and its cluster-center. If the point is further away from the center, it cannot be assigned to a Location Area. Therefore, a new cluster needs to be built. Of course, the choice of the magnitude of the maximal distance has a significant influence on the size and the number of Location Areas. Hence, the entire map can get affected. Moreover, the magnitude depends on the application area. In a kitchen scenario, Location Areas can be separated by meters (e.g., table and cupboard), whereas, the Location Areas in a medical application can lie within millimeters. This shows, that the detailed evaluation of the correct separation of Location Areas is a complex topic on its own. We just performed an exemplary run for the iterative separation in the experiments on the vision system (see Section 7.1.2).

For the classification as arbitrary movement in the Functionality Map, the multiplication factor *arbitrary-movement* for the third component and the *arbitrary-movement-threshold* need to be determined. They allow the distinction of the arbitrary movement and the constrained trajectory. As described, the arbitrary movement is relatively large in all directions. Therefore, the third component of its PCA is of especial interest, since it shows the direction of the smallest motion. If it is high, an arbitrary movement is detected. It is defined as “high” in two cases. In the first case, the magnitude of the first and the third component are relatively “close” to each other. Then, there is hardly any main direction of the movement. The multiplication factor

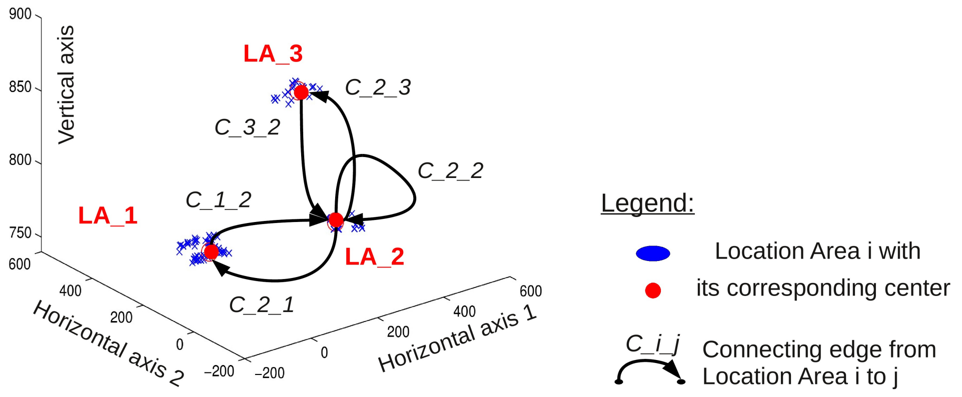


Figure 7.3: Result of the Location Areas - tracking data. [2] The blue crosses show the computed stop-points of all sequences (in mm). The black arrows are drawn to visualize the identified connections between the Location Areas.

arbitrary-movement determines in this case, how many times the component of the largest movement is maximally allowed to be larger than the component of the smallest movement. It is chosen in a manner, such that the maximal number of sequences is classified correctly. Hence, it is 15 in our experiments. In the second case, the third component is higher than the arbitrary-movement-threshold. It chosen in the magnitude of the third PCA-component of the arbitrary movements (0.06 in our experiments). Slight modifications (about $(\pm)$ $5 - 10\%$ of the current values) of the multiplication factor (arbitrary-movement) and the arbitrary-movement-threshold, resulted just in small decrease in the number of correctly classified sequences (about $5 - 10\%$).

Similarly to Section 7.1.1.1, we use the Matlab for the implementation (PCA: Matlab Statistics Toolbox, knn-classification: Matlab Bioinformatics Toolbox).[1]

The tracking data does not provide the normal vector of the table plane, which is required for the proposed distinction pushed vs. lifted object (see Section 3.1.2.3). Therefore, the height difference to the table is measured along the axis, which is known to be vertical to the table. The object is defined to be pushed, if its height difference to the table is not changing (± 5 mm).

The results of the classification of the (non-)arbitrary movements (at first without the distinction of a pushed or lifted object) is depicted in Table 7.2. 8 of 10 arbitrary movements are correctly labeled, and 77 of 80 constrained trajectories are classified correctly. This shows, that the system performs definitely better than guessing (ground truth: 11%). For the true positive rate (see Table 7.2), one has to consider, that there are just 10 arbitrary movements among all 90 sequences, leading to a significant influence of each of the two mislabeled arbitrary movements. The third PCA-component is not high enough to achieve a correct classification in both cases. There are some non-arbitrary movements which have third PCA-components at

[1]Matlab: Statistics Toolbox, Bioinformatics Toolbox.

the same magnitude. This holds, similarly for the three misclassified constrained trajectories: They contain (part of) an arbitrary movement, since the magnitude of the corresponding third PCA-components is relatively high. The classification of the pushed vs. the lifted object is successful for all sequences except one spoon-sequence, resulting in 98.9% correctly classified sequences. The spoon was, apparently, not lifted very high. Hence, it was misclassified as pushed object.

The results of the Functionality Maps are shown in Fig. 7.4. The best (= completely correct concerning the kind of movement) results are achieved for the milk and the cup, whereas the cup-handle result is the worst classification concerning the kind of movement. The misclassified arbitrary movement (red self-loop **LA**_2) and the misclassified constrained trajectory (magenta connection from **LA**_1 to **LA**_2) can be seen in Fig. 7.4. The result for the other two objects show just one misclassification concerning the kind of movement. The results show, that the system is able to deal with some misclassifications, since it achieves correct high connection relevances for all objects except the cup-handle.
The velocity constraints during the pick-up vary between 0.01 m/s and 0.24 m/s. All velocity constraints of the spoon and the vase are relatively low (max. 0.09 m/s). Both objects require more attention during the pick-up. Due to its shape, the spoon is relatively difficult to pick up. The vase needs to be picked up carefully, since it is fragile.

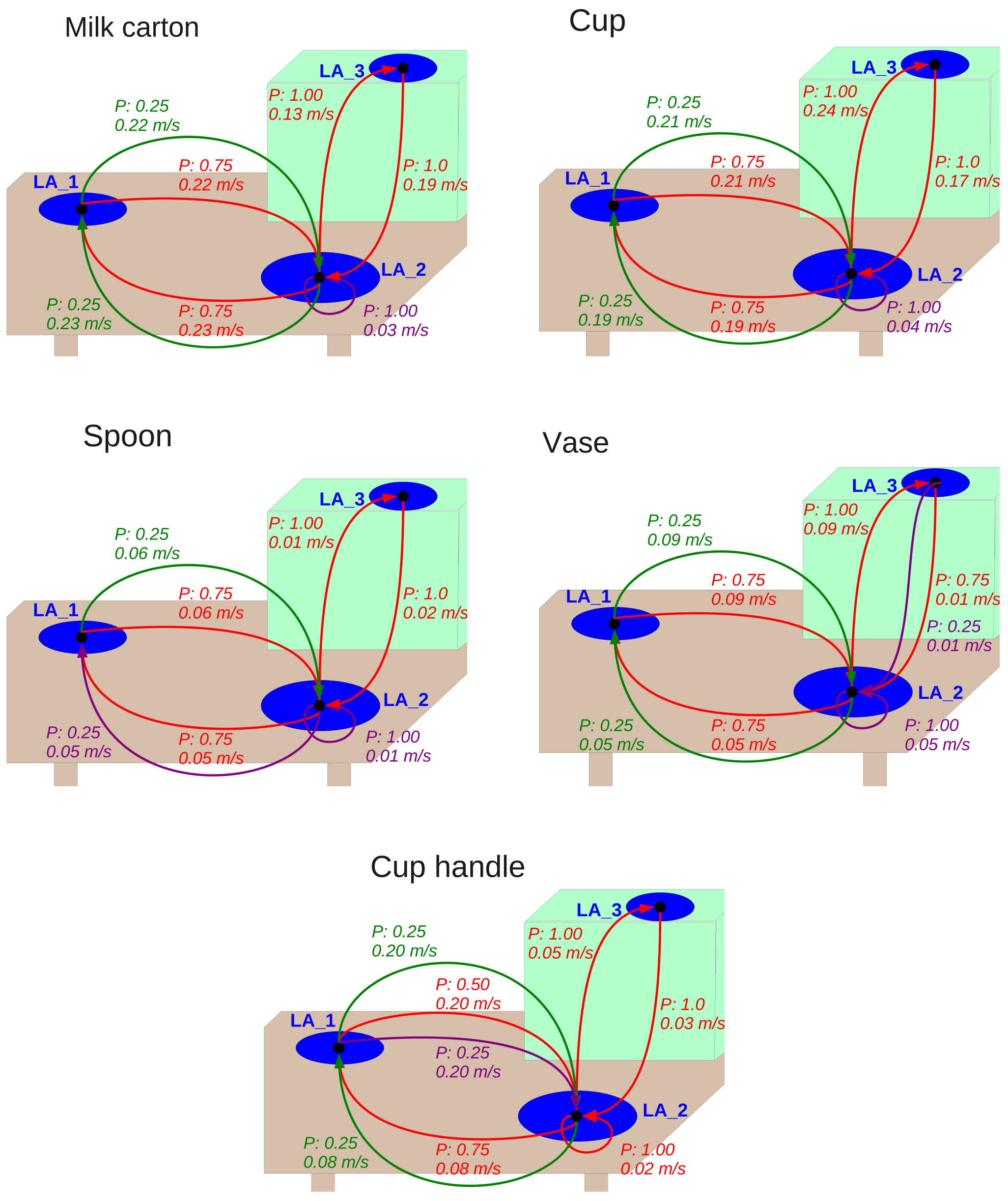


Figure 7.4: Result of the Functionality Map - tracking data. [2] The Functionality Maps for the tracking data are shown. The Location Areas are identified correctly in two corners of the table and on the green box on the table. The connections are illustrated as arrows with their corresponding relevance (probability "P"): Red arrow = constrained trajectory, green arrow = pushed object, magenta arrow = arbitrary movement.

7.1.2 Experiments on a Vision System

After the experiments on tracking data in Section 7.1.1, the experiment on vision data is evaluated. The implementation is done in C/C++. The vision data is recorded with a Firewire Marlin FO46C camera at 30 Hz and an image size of 640x480 pixel (width x height). OpenCV, XVision, extended KLT [27] and V-GPS [169] are used for the desired trajectory acquisition (see Section 3.1.1 and 7.1.2.3). The algorithms are running on a Linux system. The C++ Implementation of Hidden Markov Model by Dekang Lin [1] is (slightly modified) used for the implementation of HMMs. The PCA, the K-means algorithm and the knn-classification are done with OpenCV. The determination of the object type is done by manual labeling. The registration of rigid known objects considered for the vision system is described in [190]. In principle, one could consider, e.g., deformed or even unknown objects. Our focus is, however, the recognition of known objects to use them as keys to identify the corresponding object type in the system's knowledge.

The properties of the real human actions are listed in Table 7.4. The actions are performed between four locations at the corners of a table. The region, in which the manipulations take place, has a size of approximately 45 x 30 cm (width x depth). All sequences, described in Table 7.4, are performed with four different objects, leading to an overall number of 40 sequences. The labeling of the sequences can be found in Table 7.5. An exemplary sequence for each object is shown in Fig. 7.5.

At first, the object has to be identified on which the manipulation is performed. This is essential, since it enables the observation of the object. Moreover, the quality of the results influences the achievable performance of the following steps. Hence, it is worth to evaluate the results of this first step as well as the following tracking in detail.

[1] Copyright (C) 2003 Dekang Lin, lindek@cs.ualberta.ca, *url:* http://webdocs.cs.ualberta.ca/ lindek/hmm.htm .

Table 7.4: Sequence properties - vision data. At first, the object is pushed from the bottom right (br) to the top left corner (tl). Then, it is pushed further to the bottom left corner (bl). Afterward, the object is raised and moved straight to the initial position at the bottom right corner. There, an arbitrary movement is performed, leading to the same position again. A constrained trajectory to the top left corner and back follows. Then, four movements with rotation are done. They lead from the bottom right corner to the top right corner (tr) and back, to the bottom left corner and back to the initial position. These ten movements are performed with four objects, labeled with increasing numbers (see Table 7.5).

Seq.:	Movement	Rotation	Start Pos.	End Pos.
1	constr.: push		br	tl
2	constr.: push		tl	bl
3	constr.: curve		bl	br
4	arbitrary		br	br
5	constr.: curve		br	tl
6	constr.: curve		tl	br
7	constr.: curve	x	br	tr
8	constr.: curve	x	tr	br
9	constr.: curve	x	br	bl
10	constr.: curve	x	bl	br

Table 7.5: Sequence labeling - vision data. The ten described movements in Table 7.4 are performed with four objects, labeled with increasing numbers.

Obj.:	Seq. interval
1	1 -10
2	11 -20
3	21 -30
4	31 -40

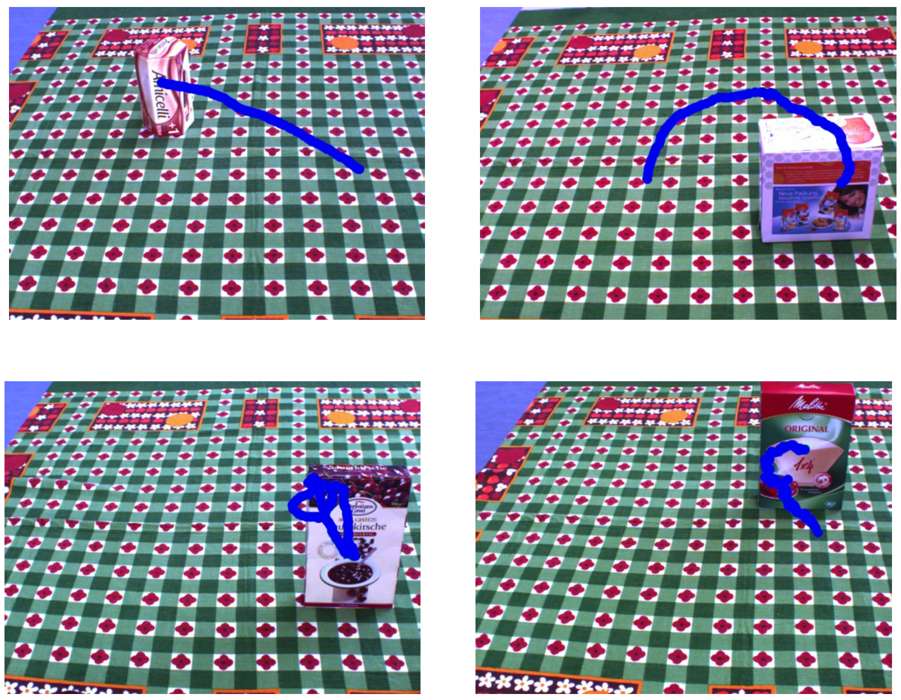


Figure 7.5: Trajectories of movements - vision data. (Top left: [2].) The figure shows examples of the trajectories, which are recorded with the vision system.

7.1.2.1 Clustering of Object Candidates on a Table

For the identification of the possible object candidates, the plane subtraction procedure described in Section 3.1.1.2 is applied. The candidate is correctly identified for all sequences. Fig. 7.6 shows the candidates for the sequences shown in Fig. 7.5.

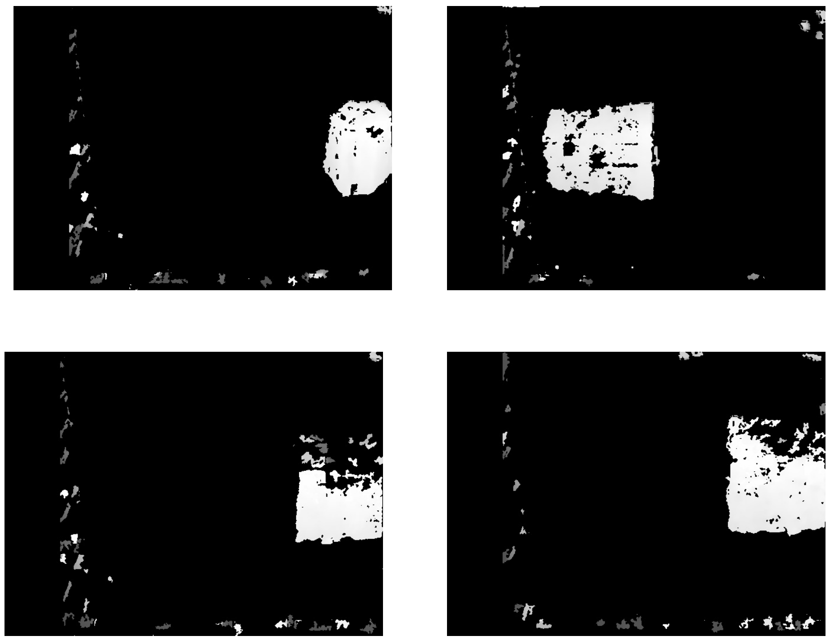


Figure 7.6: Object Candidates - vision data. The remaining objects are depicted after the plane subtraction in the disparity map. The identified objects correspond to the sequences shown in Fig. 7.5.

The sizes of the object candidates in the image vary within each object. This is due to several reasons. First, the sequences start at different positions and, therefore, the objects are placed at different distances from the observing camera system (seq. 2, 6 and 8 for each object; see exemplary comparison in Fig. A.1). Second, the disparity is not computable for some parts of the object, due to reflections on the object's surface (e.g., seq. 14 in Fig. A.2) or due to ambiguous structures which make the search of correspondences more difficult (e.g., seq. 21 in Fig A.2). This results also in non-connected parts of the object (e.g., seq. 14 in Fig. A.2) and in smaller object candidates in general (e.g., seq. 21 in Fig. A.2).

7.1.2.2 Candidate Selection

After the detection of the object candidates, one object is selected as region of interest (see Section 3.1.1.3). A selection occurs as soon as the hand of the human demonstrator is in contact with an object candidate.

The original outer bounding box R_o around the detected objects is reduced by b=20 pixel from each side to avoid false positive alerts for a contact caused by, e.g., accidental contacts at the borders. As described, if a sufficient number n_c of contacts candidates has been seen in a row, the object is selected as region of interest (ROI). We start the experiments with $n_c = 3$. A contact is detected for all sequences. The hand is correctly identified for all sequences of object 1 and 2. Too early false positive alarms occur for three sequences of object 3 (seq. 23, 25, 27; see, e.g., Fig. A.3) and four sequences of object 4 (seq. 31, 34, 37, 38). These objects contain similar colors like the glove, which is just used for the color detection of the hand (blob-detection). We use the described basic blob-detection method (Section 3.1.1.3) to detect the human's hand. Of course, one could even further modify the basic method. However, this version fulfills its purpose, in principle. Modifications of the involved parameters (e.g., n_c) do not improve the result of 82.5% correctly identified contacts. This holds also for the application of a further modification of the basic method: In order to detect a hand as such, the hand has to consist of a minimal number of pixel. Experiments with different numbers did not improve the original result of 82.5% correctly identified contacts. Here, the false positive alarms result just in the effect, that the tracking of the object starts, before the object is manipulated.

7.1.2.3 Parsing of Human Action

After the contact detection between the hand and the object, the tracking of the object is initialized and the manipulation of the object is observed.

OpenCV, XVision, extended KLT [27] and V-GPS [169] are used for the trajectory acquisition. The extended KLT is a tracking algorithm, which fulfills the requirements motivated in Section 3.1.1.4. It allows the extraction and the tracking of 2D-features on the object. The number of tracked features influences the quality of the trajectory acquisition. The more feature are tracked stably during the manipulation, the more information is available to determine the trajectory. The more information can be included in the computation of the trajectory, the more precise the result can get. However, the objects can have different sizes and different colored patterns on the surface. Both properties have a significant influence on the quality of the tracking. A small object with hardly any patterns will be much more difficult to track than a large one with a lot of structures on its surface. In the worst case, the object does not provide any pattern on a smooth, reflecting surface. The tracking applied here would fail in such a case. However, we focus on daily-life objects which provide enough patterns and can be tracked with the described method.

As described, the number of tracked features has a significant influence on the quality of the trajectory acquisition. Therefore, we examine it further. We compare the average number of (1) the features at the start, (2) the features at the end, (3) the permanently tracked features, (4) the average number of features during the manipulation and (5) the average size of the ROI (Table 7.6). It is interesting to see, that the average number of features at the start is a bit higher for object 1 than for object 2, even though object 1 is the smaller one. It

provides more structure to the feature detector. Object 3 is among the bigger objects and it is very structured. This leads to a high number of features at initialization. Moreover, it is not surprising, that a large amount of these features can be tracked during the manipulation, a lot of them permanently. For object 1, the number of features at the end, the number of permanently tracked features and the average number of features is small. Especially in comparison to object 2, the result is noticeable, since both object have approximately the same number of features at the beginning. The position of the detected features influences this result, because object 1 has a relatively large structured surface on top of the object, on which features are found at the beginning. These features get easily lost, since the object is grasped on top of the object, and the features disappear.
Of course, the number of tracked features depends on the speed of the observed motion. If the features move out of the search region of the tracker (Section 3.1.1.4), they cannot be found any more. Hence, a fast motion requires a high frame rate. Then, the features move just within the search region in the subsequent image. In our case, the frame rate is not very high (see first part of Section 7.1.2), since both cameras run all the time (stereo). Hence, the recorded movements are performed at low speed.
The tracking stops at the end of the manipulation. It is assumed to be reached, when the features are not moving any more.

Table 7.6: Parsing of human action in average - entire system. The size of the ROI is compared with the average number of features (start, end, permanently, in average during the manipulation) for each object.

Obj. :	∅ Size ROI	∅ # features:			
		start	*end*	*permanently*	*∅ seq.*
1	14100,0	16,4	7,9	1,7	10,4
2	22841,0	13,7	12,5	6,1	13,3
3	21309,7	31,4	24,7	12,0	25,8
4	27069,2	19,7	12,5	6,0	14,7

At the end of the sequence, the objects are placed in the same horizontal orientation as at the beginning of the manipulation. Therefore, the *remaining angle* at the end of each sequence should be zero for the both horizontal axes. Consequently, the remaining angle can be seen as an error measurement for the computation of the object orientation during the manipulation. Five intervals are built for the remaining angles ($[0;5]$, $]5;10]$, $]10;15]$, $]15;20]$, $]20;[)$. Each remaining angle is assigned to one of them. The result in Table 7.7 shows, that we achieve a good performance with 85% of the remaining angles below 10 degree. There are just three real outliers with 24.54 degrees (seq. 6) and 23.47 resp. 71.02 degrees (both horizontal angles in seq. 18) as remaining angle. In both sequences, a "jump" occurs during the second part of the

image sequence, since the recording device switches the storing device. This leads to features, which are not correctly redetected or reinitialized. Hence, the orientation is not computed correctly.
However, the remaining angle gives just an impression about the precision in the tracking results. In our experiments, we are, especially, interested in the analysis of characteristic manipulation properties. If the orientation is not computed very precisely, it *can* be possible to extract the relevant properties, nevertheless. For example, if the object is rotated and a small remaining angle occurs, the curve of the varying horizontal orientation gets shifted after a while. However, the characterizing *variations* of the horizontal orientation can still be visible. Similarly, for an object which is not rotated, the magnitude of its horizontal orientation should get shifted, but variations like for a rotation should not occur in the profile of the horizontal orientation. This is also confirmed by our results. Both sequences with the significant remaining angles (described above) are classified correctly as motions without, resp, with rotation.

Table 7.7: Remaining angles after manipulation - vision data. After the manipulation, the object is placed on the table with the same horizontal orientation as at the initial position. Consequently, the angle, which remains at the end of the manipulation, gives an impression about the accumulated error in the observation. For each sequence, the absolute value of the remaining angle is given in degree. Then, these values are assigned to one of five different intervals.

Interval:	$[0;5]$	$]5;10]$	$]10;15]$	$]15;20]$	$]20;[$
	50	18	4	5	3

Fig. 7.7 illustrates the computation of the orientation of the objects. The normal vector is used for the illustration of the object orientation. The normal vector is rotated and translated like the object during the manipulation. It can be seen clearly, that object 4 has been rotated (Fig. 7.7, bottom right), whereas the other objects are kept in the same vertical orientation as at the initial position.

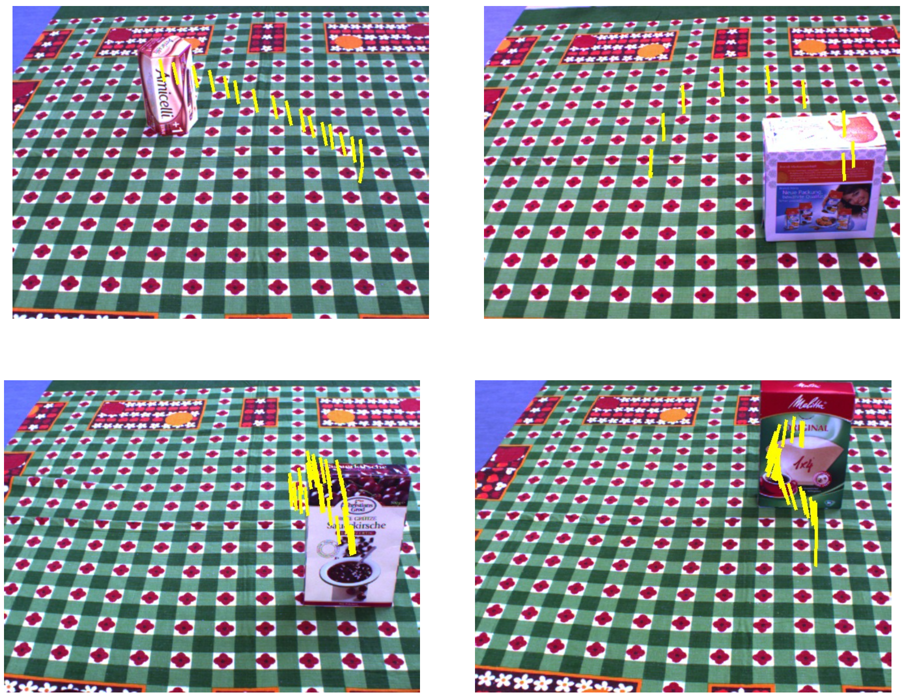


Figure 7.7: Development of the object's orientation along trajectories - vision data. The figure shows the change of the object's orientation during the manipulation. It is clearly visible, that object 4 (bottom right) has been rotated, whereas the other three objects are kept in the same vertical orientation.

7.1.2.4 Representation of Manipulation-Relevant Object Knowledge and Functionality in the Environment

After the acquisition of the trajectory of each sequence, the sequences are analyzed to build up the Object Container and the Functionality Map. Similarly to the tracking data, a basic pre-processing is performed (a minimal motion $> 0.01\ /sample$, 140 sample moving-average-window). Furthermore, the first and last 20 samples are cut of in order to deal with the arbitrary motions at the beginning and at the end of the sequences. The values of the angles are smoothed along each dimension separately. The initialization and threshold values are set as in the experiment with the tracking data, except for the *arbitrary-movement-threshold* (0.005 for the vision data). This could indicate, that the chosen values could be applied in more cases, maybe in general. However, a very large number of experiments, also in different application areas would be necessary to generalize these values. The smaller arbitrary-movement-threshold for the vision data is due to the fact, that the magnitude of the third PCA-component of the arbitrary movements is much smaller here. Possibly, this is due to the movements at low speed (see Section 7.1.2.3).

Object Container For the Object Container, the maximal acceleration within each sequence is computed, as well as the appearance of rotation during the manipulation.

The appearance of rotation, resp., no rotation is correctly identified for 31 of 40 sequences. Eight sequences are mislabeled as sequences with rotations. All of them show, that one or both horizontal angles vary during the manipulation. The variations are not as strong as for most of the sequences with rotation, but it is still visible (see, e.g., Fig. A.4 for object 1). Seq. 17 is the only sequence, which is mislabeled as sequence without rotation. The variations are relatively small in comparison to the other sequences with rotation. The statistical measures in Table 7.8 shows the accuracy, the true negative rate and the true positive rate of the classifications.

Table 7.8: Statistical result of the classifications - vision data. Statistical result of the rotation-classification, the classification of the pushed vs. lifted objects and the classification of the (non-)arbitrary movements.

Property:	Accuracy	True positive rate	True negative rate
Rotation	77.5%	93.8%	66.7%
Arbitrary Movement	80.6%	100.0%	78.6%
Pushed Object	95.0%	87.5%	96.9%

The entire result of the Object Container is shown in Table 7.9. The acceleration classes have the same intervals as in Section 7.1.1. The choice of the intervals in Section 7.1.1 seems to be appropriate for illustration, since none of the classes is empty in both experiments.

Table 7.9: Result for the Object Container - vision data. The table shows the number of observations for each acceleration class and the rotation-classification. Legend: R = motion with rotation, Acc. class = acceleration class: acc. class 1 for $x < 0.006\ m/s^2$, acc. class 2 for $0.006\ m/s^2 <= x < 0.009\ m/s^2$ and acc. class 3 for $0.009\ m/s^2 <= x$.

Acc. class	1		2		3	
	no R	R	no R	R	no R	R
object 1	1	1	0	1	2	5
object 2	3	0	0	0	4	3
object 3	2	1	1	1	1	4
object 4	0	2	2	2	1	3

Functionality Map Further properties of the sequences need to be analyzed for the buildup of the Functionality Maps. One important property is the correct assignment of the start and end position to the corresponding Location Areas. The Location Areas can be seen as link between the abstract knowledge in the Functionality Map and the current scene in the real world. The Location Areas themselves are determined successfully. The assignment is successful for 77 of 80 positions (96.3%). The misclassifications occur for the end positions of sequence 8, 21 and 33. The z-components (the depth) of the end positions are closer to other Location Areas for these three sequences.

We also perform an exemplary run for an iterative separation of the Location Areas. It is based on a maximal distance between a Location Area and a just observed start/ end position. If the distance between each of the Location Areas and the current start/ end position is larger than the maximal distance, a new Location Area is created. The critical point is the determination of an appropriate magnitude of the maximal distance. Here, we just wanted to illustrate the possibility to apply an iterative separation. For simplification, we have chosen half of the distance between the start and end position of the first sequence (bottom right and the top left Location Area). All four Location Areas are detected properly in this case.

After the determination of the Location Areas, we analyze the manipulations observed in-between. As the statistical measures in Table 7.8 show, the result of the distinction between a pushed object and a lifted object is remarkable. There is just one sequence mislabeled as pushed object, and one sequence mislabeled as lifted object.
The classification as arbitrary movement or as movement with a constrained trajectory is performed with a true positive rate of 100.0%. Consequently, no arbitrary movement is mislabeled as non-arbitrary movement. Six sequences are misclassified as arbitrary movements instead of movements with constrained trajectory. These movements contain small parts with an arbitrary shape.

The Functionality Maps of object 1 and 4 have, resp., just one wrong assignment of an end location. Everything else is correct (see object 1 in Fig. 7.8). The Functionality Map of object 2 suffers mainly from sequences misclassified as arbitrary movements (see Fig. 7.9). Apparently, these sequences of object 2 contain parts of arbitrary movements. Possibly, parts of the sequences appear as arbitrary due to the relatively low frame rate in this experiment (see first part of Section 7.1.2). Intermediate, smooth points might got lost.

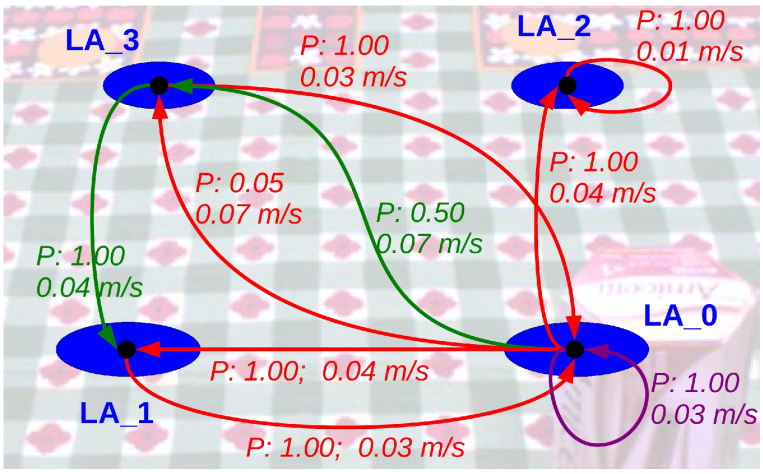


Figure 7.8: Functionality Map: object 1 - vision data. [2] The wrong assignment of an end location is visible at Location Area 2 (incorrect self-loop). Everything else is correct.

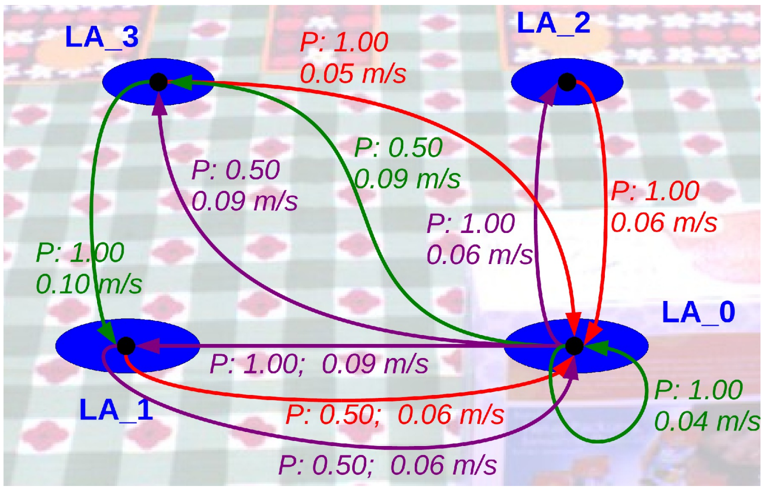


Figure 7.9: Functionality Map: object 2 - vision data. [2] The classification for object 2 suffers mainly from sequences misclassified as arbitrary movements (magenta arrows: arbitrary movements, red arrows: constrained trajectories, green arrows: pushed objects).

Hence, two of the four Functionality Maps achieve very good results (object 1 and 4).

One Functionality Map (object 2) suffers from a couple of misclassifications regarding the distinction of arbitrary and non-arbitrary movements. The Functionality Map of object 3 has two misclassified connection properties and one wrong assignment of an end location. However, one sequence of object 3 can be seen a outlier, since its connection property, as well as the assignment of its end location, are wrong. Besides the second misclassified connection property, the map is correct.
Fig. A.6 - A.9 in the Appendix show the entire result of the Functionality Maps for each of the four objects. An ideal solution of the Functionality Map is depicted in Fig. A.5.

At the end, the kind of grasp is determined manually. The corresponding classes are described in [172]. All used grasps are power grasps with an abducted position of the thumb. Fig. 7.10 illustrates two exemplary grasps.

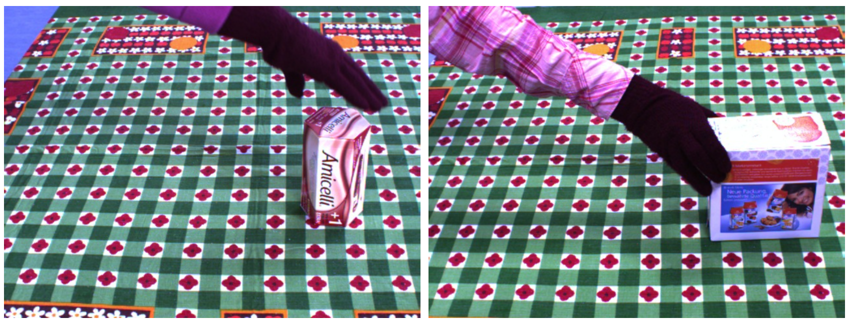


Figure 7.10: Grasp types - vision data. [2] The power grasp with an abducted position of the thumb is shown exemplary for object 1 (seq. 7) and object 2 (seq. 14). The grasps of the other objects look similarly. They are also power grasps with an abducted position of the thumb.

7.2 Contact State-Based Representation of the Environment for Dexterous Manipulations

In this part of the experiments, we want to illustrate the advantageous options of the proposed contact state-based representation of dexterous manipulations. We build up three scenarios, which require detailed knowledge about the corresponding manipulation. Two scenarios are in the context of medical applications. One further scenario can also be applied in the medical context: knot-tying. We describe the properties of the knot as for a "normal“ knot, which is usually used by humans to tie their shoes. Such a scenario illustrates a possible non-medical, daily-life application.

Our scenarios focus on a top-down approach for the desired tasks. This means, that the task knowledge is described in the contact states and in relations (e.g., filled in by a human user). As already mentioned in Section 3.2.3, a bottom-up approach could be applied as well.

We evaluate the system with simulations. Real-world applications would require many advanced sensing methods to illustrate our approach (e.g., check of the depth of a cut in scenario I, thread detection at the beginning of scenario II). This is not the focus of this thesis.
The implementation is done in C/C++. We simulate a robotic system which is supposed to reach desired points along paths. The robotic system is described in the DH-convention suggested by Denavit and Hartenberg [164] in the form shown in [9]. We simulate an arm of the MiroSurge system from DLR with a gripper, since its DH-parameters are available in [104], [191], [192]. A schematic diagram is shown in Fig. 7.11. We just consider the DH-parameters here. Hence, e.g., joint limits are not taken into account. This allows us to illustrate the principle of our representation, since we can create motions independently of the magnitude of the motions' step size. In general, any medical robot could be used in the scenarios, since the approach is independent of the robot.
Moreover, we do not use uniformly distributed points along the path. The points are chosen in a manner, such that the shape of the path can be illustrated. The robot's end-effector has to pass an entry point to the human body. This entry point has to be considered during the entire procedure.
The inverse kinematics is estimated through the proposed method in Chapter 2. This way of estimating the inverse kinematics allows us the estimation of several alternative joint configuration sequences. We are neither limited to pre-defined start configurations of the robot nor to possible local minima along the path. However, the larger the number of parameters which are supposed to be estimated, the longer it will take. In order to speed up the computation, we split up the estimation in two parts in this experiment: First, we estimate the first seven DoF (outside the body), until we reach the desired point roughly. Then, we estimate the last two DoF (inside the body). Hence, the search space is reduced. Of course, it is possible, that we do not achieve the best solution for the inverse kinematics and, subsequently, for joint speeds,

etc.. However, this procedure is applied in all scenarios in this experiment. The conditions are, therefore, the same for all runs.

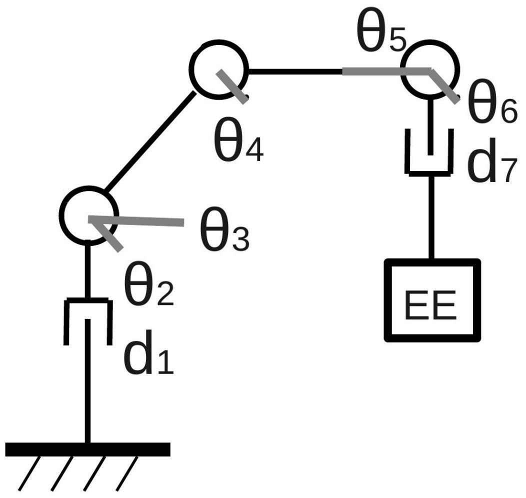


Figure 7.11: Principle, schematic diagram of the MiroSurge system from DLR. The first seven DoF of the simulated MiroSurge system are illustrated. These DoF are outside the body. Each circle symbolizes one, resp., two rotational joints. The gray lines coming out of a circle refer to, resp., one joint axis. The symbols of the translational joints are bracket-shaped. The "EE"-box (bottom right) symbolizes the end of the visualized manipulator.

In our simulations, we want to illustrate the potential of the proposed contact state definition in Section 3.2. If possible (e.g., no path of fixed points), we make use of the defined relation in contact properties. Both definitions are characterizing the representation. They are shown below, again:

```
contact(objects, contact_properties)
```

```
relation: [relation_attribute, objects]
```

We do not implement a real experiment or a physical model to verify, e.g., the execution order of the single components of the description. We choose one or, if possible, several paths manually in a manner, such that they fulfill the required properties. This allows us to illustrate, that different paths can fulfill the requirements of a desired manipulation described by the proposed representation.

7.2.1 Scenario I: Cut

The first scenario is a simple cut on an organ during an operation. Such a cut is performed with a knife. In Section 3.2.1, we defined already such a contact as follows:

```
contact({knife, organ},
    cut_properties)
```

The **cut_properties** define, how the cut on the organ is performed. A cut cannot be chosen arbitrarily in an operation (e.g., to excise a tumor). Here, it has to be performed accurately along a certain path, which is represented by a set of successive points. Of course, a cut has a certain depth, which defines how deep the cut has to be.

```
cut_properties = {cut_points, depth}
```

The depth of the cut could be the same for all points (single value) or it could be specified by a set representing the respective depth for each point of the cut (vector). We simply use a single value in our simulation. Usually, the magnitude of the depth is given by the specific application (e.g., size of tumor). Moreover, the property "depth“ implies knowledge about the space: It has to be defined with respect to the direction of the depth. For simplicity, we use the vertical orientation in the room as direction of the depth in our simulations. The direction of the depth is important in this application, since it sets implicitly the orientation of the knife during the cut.

The path consists of fixed points. Hence, it is not possible to find alternatives. However, we can optimize the configurations of the robot along this path, e.g., with respect to efficiency in actuation. Therefore, we evaluate different robot configuration sets in our simulations. In general, the cut could consist of fixed points arranged in any configuration, as long as the configuration reflects the cut appropriately. We assume a line of fixed points to illustrate the principle.

Table 7.10: Results for scenario I (cut). The maximal and the average speed of all joints is computed for each configuration set. The sets with the smallest maximal speed peak and the one with the minimal average speed are depicted.

Configuration set:	Magnitude of the speed peak:	Average speed:
Smallest maximal speed peak:	0.15 rad per time unit	0.05 rad per time unit
Minimal average speed:	0.16 rad per time unit	0.04 rad per time unit

The maximal and the average speed of all joints are computed for each configuration set. The set which has the smallest maximal speed peak and the set with the minimal average speed are of special interest, since they have the most desirable properties with respect to efficient actuation. Table 7.10 shows the average speed and the speed peak for both sets. The configuration set which has the smallest maximal speed peak along the path has a peak at about 0.15 rad per time unit and an average speed of 0.05 rad per time unit. The configuration set with the minimal average speed (0.04 rad per time unit) has a slightly higher maximal speed peak at 0.16 rad per time unit. Hence, the differences regarding the speed peak and the average speed are relatively small in this experiment.

7.2.2 Scenario II: Knot-tying

The second scenario is a knot-tying scenario. The goal of the scenario is a knot, which connects the involved threads. Consequently, the contact of the knot-tying scenario consists of a set of threads and certain properties, which define the type of contact (the knot).

```
contact(thread_set, knot_properties)
```

Theoretically, more than two threads could be knotted. We focus here on two threads, since it shows the basic principle.

```
thread_set = {t_1(fp_1, l_1),
              t_2(fp_2, l_2)}
```

Each thread *t_i* has its respective fix point *fp_i* and a length *l_i*. The properties of the knot are defined by two sets of properties. The knot has a desired intersection point *ip* and the knot should be tightened at this intersection point.

```
knot_properties = { intersection_point ip,
    tightened_knot at ip }
```

The intersection point is the point of contact. There, the knot is further characterized by the properties of the type of knot. A knot of a simple, daily-life shoe tying scenario is used here:

```
intersection_point ip = {
  [over_behind, (t_{1, lower}, t_2, fp_1)],
  [under, (t_{1, upper}, t_2)],
  [under, (t_{1, upper}, t_{1, lower})] }
```

with

```
[over_behind, a, b, c)]
```

as description of putting a over and behind b. The variable c can be seen as perspective to define "behind" an object. In an implementation, one needs to know the gravity vector to determine the meaning of "over". The check of the property "behind" could be done in two steps. First, a has to be further away from c than b. Second, in the perspective of c, a and b should lie in approximately the same direction. The exact definition of the same direction depends on the application. In our case, it is defined very generous: The points are configured in a manner, such that the angle between the vectors $\boldsymbol{ca}$ and $\boldsymbol{cb}$ is smaller than $50°$.
As described in Section 3.2.3 and at the beginning of Section 7.2, a real-world application would require many advanced sensing methods, which are not in the focus of this thesis. We just want to mention here, that one could, also, consider a simplification to observe the desired goal states in a demonstration: One could record the trace of the end-effector of a demonstrating manipulator. This could simplify the thread detection and tracking under possible occlusions in this scenario.

A knot is tightened through a distance maximization for each thread tip and the intersection point, respectively, the fixed points.

```
tightened_knot = {
  [maximize distance, (t_{1, tip}, ip)],
  [maximize distance, (t_{2, tip}, ip)],
  [maximize distance, (t_{1, tip}, fp_1)],
  [maximize distance, (t_{2, tip}, fp_2)],
  tolerance t_k}
```

The tolerance *t_k* increases the area around the positions, at which the distance is defined to be maximized. Of course, it depends on the application, on the available objects and the environment. An application could require a knot at a certain position along the threads (e.g., esthetic reasons for a knot on a shoe). The tolerance *t_k* would be relatively small in such a case. If, e.g., an obstacle blocks the areas, where the distance is maximized exactly, we need a sufficient tolerance to ensure, that an intersection point can be created at all. In general, the larger the magnitude of the tolerance is, the more areas exist to build the intersection point. However, if its magnitude is chosen too large, the goal, the *tightened* knot, cannot be ensured any more. For simplicity, we have chosen a quarter of the magnitude of the distance between the fix points of the threads. It is relatively generous, since we are not focused on a very specific position of the knot.
The maximization itself could be implemented as follows: Around the intersection point, resp., the fixed points, a sphere is build. Each sphere has a radius at the magnitude of the respectively maximized distance described above (thread tip - intersection / fixed point). The intersections of all four spheres form the areas, where the intersection point can be placed. If no intersection exists, we need to make use of the tolerance: It has to be large enough, such that an area exists, which includes all four spheres.

As described in Section 3.2.3, the execution order of the elements in the set is not defined in advance, if the contact properties are defined by a set. The order could, e.g., be determined by a real experiment or a physical model, which simulates the execution of the relations in different orders. The results of such a simulation, respectively, experiment show, which execution orders lead to the desired contact state. It is possible, that one, several or all possible orders lead to the goal.
In our experiments, we evaluate the contact state description manually. First, the intersection point is of interest, since it is at the lowest level in the property description. A simulation of the possible manipulation orders would, e.g., show, that, if the last relation (move the upper part of *t_1* under its lower part) is perform first, there will be no long-term effect, since further actions of *t_1* will destroy the effect. Therefore, it has to be processed later. If the upper part of *t_1* is put under *t_2* first, it will be impossible to move the lower part of *t_1* over *t_2* stably (see Fig. 7.12 in comparison to Fig. 7.13). Hence, the relations have to be achieved in increasing order. Consequently, we cannot get alternatives regarding the execution order of the relations.

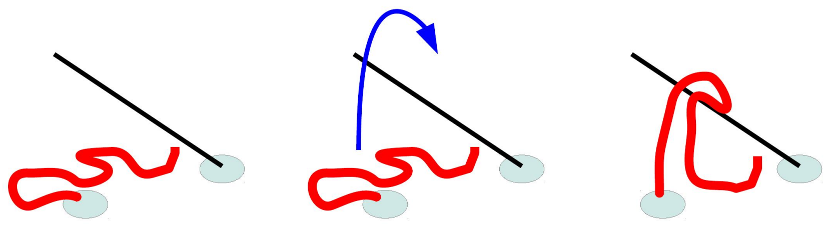


Figure 7.12: Exemplary non-working execution order in the knot-tying scenario. The top of thread t_1 (red) is moved under thread t_2 (black), first. However, it will be impossible to move the lower part of t_1 over t_2 stably (at least after the tightening with the distance maximization). The working execution order is depicted in Fig. 7.13.

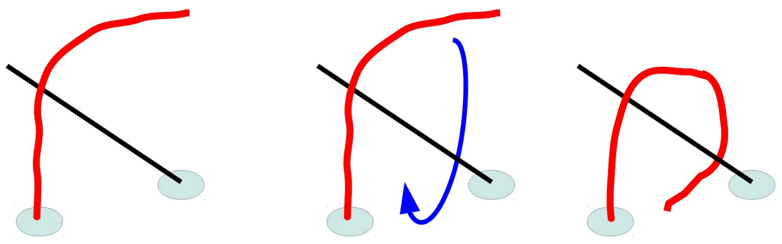


Figure 7.13: Exemplary working execution order in the knot-tying scenario. If the lower part of t_1 (red) is moved over t_2 (black), first, the top of thread t_1 can moved under thread t_2, stably.

We see in the definition of the thread set, that two threads are involved. At least one of them has to be moved to change the contact states. We assume here for simplicity, that we use two manipulators. Each manipulator moves just one of the threads. Thread t_1 should be moved over and under t_2. This means, that a lift of t_1 and t_2 is necessary. t_2 can be lifted before or during the procedure (last possibility: before t_1 should be moved under t_2). Hence, these two alternatives are possible. We consider the first one here.

We can sum up, that the execution of the property **tightened_knot** is straight forward, but not fixed to a pre-defined path.

In our simulation of the scenario, one path is created without further constraints. A second path is built under the consideration of an obstacle - another organ. This requires a partial shift of the first (original) path (see also Fig. 7.14). The start positions are the same for both paths, whereas the intermediate points are shifted away from the organ. The end points are just slightly moved to achieve an end position, which is similar to the original one, while avoiding the obstacle. The magnitude of the shift is limited with respect to the task: If the task is not fulfilled any more, the shift cannot be applied. The tolerance t_k determines here also the freedom within which the path can be changed. It allows to a limited tolerance with respect to the tightening of the knot. Hence, the path is less restricted and can be shifted.

The advantage of our representation is clearly visible here: Only the pick-up points of the threads are fixed. The final execution of the desired manipulation is not limited to a fixed path. A possible path can, e.g., be shifted easily to avoid an obstacle.

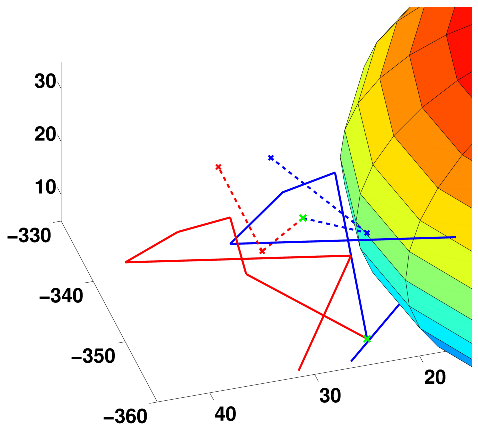


Figure 7.14: Original and shifted path in the knot-tying scenario [7] (unit: mm). The original path (blue) cannot be applied, when the colored organ is present, since both are intersecting. In contrast, the shifted path (red) is collision-free. The dashed lines illustrate the path of the supporting manipulator (lifting one thread, tightening). The green crosses indicate the pick-up points of the threads.

Table 7.11: Results for scenario II (knot-tying). The maximal and the average speed of all joints is computed for both paths and both manipulators.

Path and manipulator (man.):	Magnitude of the speed peak:	Average speed:
Original path, main man.:	0.7 rad per time unit	3.1 rad per time unit
Shifted path, main man.:	0.6 rad per time unit	3.1 rad per time unit
Original path, supporting man.:	0.1 rad per time unit	0.3 rad per time unit
Shifted path, supporting man.:	0.1 rad per time unit	0.5 rad per time unit

Table 7.11 shows the results of the simulation of the know-tying scenario. The differences between the original and the shifted path are small regarding the average speed and the maximal speed peak (average: about 0.7 and 0.6 rad per time unit; peak: both 3.1 rad per time unit). The large speed peaks are due a defined movement, which required a significant change of the end-effector's orientation (when the tip of thread t_1 needs to be moved from behind t_2 under t_2 to the front). The movements have been defined manually. Possibly, a usage of a physical simulation plus an advanced path optimization could avoid such movements.

The average speed of the supporting manipulator (lifting the thread, tightening) is the same in the original and the shifted path (0.1 rad per time unit). The peaks in the speed profile differ (0.3, respectively, 0.5 rad per time unit). To conclude, the shift of the path influences the efficiency of the manipulation just partly. The obstacle avoidance can be performed without disadvantages regarding the efficiency of the main manipulator. The speed peaks in the profile of the supporting manipulator differ.

7.2.3 Scenario III: Suturing

The third scenario describes a suturing during an operation. A suturing is performed between at least two objects, which have to be in contact at certain points afterwards. The contact suturing consists of the tissues, which have to get in contact under consideration of the suturing properties.

```
contact(suturing_tissues,
           suturing_properties)
```

We consider two objects o_1, o_2 for our suturing scenario, which have a respective stiffness st_i. Moreover, an additional tool ("**supporting_item**") is introduced to keep the objects o_1, o_2 in contact.

```
suturing_tissues = {o_1(st_1),o_2(st_2),
                       supporting_item}
```

Theoretically, the objects can be set in contact in any way. For example, the end-effectors of a two-armed robot could move the objects together. If the end-effector unhand its contact to the respective object, the objects might jump back into their original positions without contact. Hence, another tool (supporting_item) like a thread is needed to keep the objects in a long-term contact. Theoretically, one could bind the thread around both objects. If the thread cannot be put around the objects, the thread can connect both objects through a stitch. We want to achieve the contact of the suturing through stitches along a desired contact line. A sufficient number of **suturing_points** has to ensure this contact.

```
suturing_properties = {
  suturing_points,
  stitch_properties,
  tightening_suturing}
```

The suturing points are described by vectors. The stitches at these points are described through certain **stitch_properties**. For example, a simple direct stitch from one tissue to the other can be chosen. A stitch in form of a cross is more demanding, but more stable. We choose a simple stitch for our experiments. This means, that the ingoing and outgoing penetration point are placed in a manner, such that (1) they are next to the respective suturing point, (2) each of them is located on respective one object, (3) they are mirrored on each other with respect to the suturing point and (4) the vector from the ingoing to the outgoing point points into the desired penetration direction. However, depending on the requirements of the respective application, the penetration direction can be more or less restricted. A tolerance t_s, penetration reflects the magnitude of the restriction. It can, e.g., be the maximally allowed angle between the vector from the ingoing to the outgoing point points and the penetration direction. Similarly, the definition of *next* to a respective suturing point depends on the application. We introduce the tolerance t_s, simple as maximally allowed distance to the respective suturing point. The more precise and the smaller the stitch has to be, the smaller both tolerances are.

```
stitch_properties = {
  {simple_stitch},
  {penetration_direction},
  tolerance t_{s, simple},
  tolerance t_{s, penetration},
  depth d_s}
```

In our simulations, we choose generous values for the tolerances, since we just want to determine one possible path of the stitch to illustrate the principle (t_s, simple = 50 mm, t_s, penetration = ± 30°). The depth d_s is the maximal depth, which the stitch is supposed to reach. We set it to 40 mm in our simulation.

Moreover, a tightening is required for each suturing point i.

```
tightening_suturing = {
  [maximize distance, (t_{1, tip}, suturing_outgoing_penetration_point(i))],
  tolerance t_s}
```

It is important to point out, that the suturing_outgoing_penetration_point(i) is defined implicitly through the suturing points and the type of stitch (see paragraph above). Similarly to before, we use a tolerance for the tightening to allow, e.g., the consideration of obstacles. In our implementation, we considered a very small tolerance t_s with a quarter of the distance between the ingoing to the outgoing point at each stitch, since we do only assume one obstacle next to an ingoing point. It hardly affects the areas for the distance maximization in the tightening operation.

The contact of the suturing scenario illustrates our concept of the role of the involved objects. In order to perform the desired stitch, a needle is necessary. The needle and the thread are non-active objects, which are used as tools. The thread is the additional tool (the supporting item) to keep the objects in contact. The needle is a further tool, which is necessary during the execution. The tissues which have to get in contact are non-active. The two-armed robot is the active object, which performs the task.

In our simulation, we assume that the tissues are soft and deformable. No handover of the needle is required. We treat the needle as elongation of the manipulator as long as it is grasped

by the end-effector. The path of the end-effector is restricted at the respective stitch points. However, it is less restricted in the approach phase and in the after-stitch phase. The closer the path point to the stitch point is in these phases, the more restricted it is. Hence, we use a cone-shaped path restriction in the approach and after-stitch phase. The cone's height points towards the ingoing penetration direction. The ingoing penetration direction depends on the depth d_s and the shape of the stitch within the tissue. For simplicity, we choose a isosceles trapezoid as shape of the stitch.
In the approach and after-stitch phase with the cone-shaped path restriction, we represent the cone by points on the cone's cover (see Fig. 7.15). The orientation is adapted step-wise to the desired orientation. If restrictions have to be considered, alternatives can be chosen within the cone and the step-wise adaption of the orientation. For example, other organs should not get injured by the needle tip. In order to decrease injury avoidance, one could choose the path with the maximal distance to other organs. The organs could be represented through virtual fixtures [193], [194]. The alternative paths in cone-shape are similar to principles in [83]. Parts of our path are fixed (stitch points), whereas others are more flexible (approach phase).

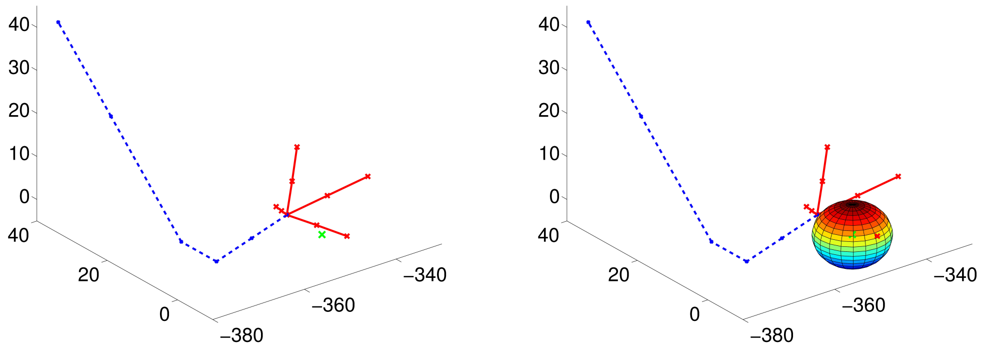


Figure 7.15: Path of the suturing scenario [7]. *Left:* The blue path depicts the fixed part of the path in the scenario (unit: mm). The red part shows alternative paths in the approach phase, which is less restricted. The green cross indicates the center of an obstacle (e.g., another organ). *Right:* The entire obstacle is illustrated.

First, the configuration set with minimal average speed (0.21 rad per time unit; peak: 1.26 rad per time unit) is determined for the fixed part of the path (the stitch points). Its first configuration along the fixed path is the desired configuration at the end of the approach phase. We focus on the analysis of the possible approach paths. Each of them leads to the first fixed point in the stitch. In our simulations, four paths form the described cone (see Fig. 7.15, left). They are alternative paths, which want to analyze further. Hence, we just use the first configuration of the fixed part of the path (the stitch points) as a constraint: It has to be reached at the end of the approach phase. The rest of the fixed path is not analyzed further, since there are no alternatives.

Table 7.12: Results for scenario III (suturing). The maximal and the average speed of all joints is computed for all four paths reflecting the cone (see Fig. 7.15). The path with the maximal distance to the organ is the top path. The left path has the smallest average speed and the smallest speed peak.

Path:	Magnitude of the speed peak:	Average speed:
Top:	1.2 rad per time unit	2.9 rad per time unit
Left:	1.1 rad per time unit	2.6 rad per time unit

Table 7.12 depicts both of the most interesting results: The result for the path with the maximal distance to the organ (top path), and the path which has the smallest average speed and the smallest speed peak (left path). In order to check the distance between the path and the organ, we check the distance between each path point and the center of the organ. If the distance is smaller than the radius of the organ, the path intersects the organ. The system analyses correctly, that the bottom path intersects the organ. Hence, it cannot be used. The other three paths can be used, since they do not intersect the organ. The top path is furthest away from the organ. Fig. 7.15 illustrates these results clearly. Both paths at the side have the same distance to the organ. The path on the left has the smallest average speed (1.1 rad per time unit) and the smallest speed peak of all four paths (2.6 rad per time unit). The average speed of the top path is slightly worse with 1.2 rad per time unit. Its speed peak at 2.9 rad per time unit is just a bit higher. The relatively large speed peaks occur for the last two joints (close to the end-effector). Possibly, this is due to the split estimation of inverse kinematics (described at the beginning of Section 7.2). Both joints are used for the fine adaption to the desired position. The required fine motions during the manipulation are, probably, performed by them. The same holds for the estimation of the inverse kinematics along the fixed part of the path (the stitch points). The speed peak is also relatively high, there.
To sum up, the path furthest away from the organ leads just to a slightly worse efficiency of the manipulator than the path for which the manipulator's efficiency is best.

7.3 Estimation of Inverse Kinematics of Arbitrary Serial Chain Manipulators and Human-Like Robotic Hands

In the first Sections of this Chapter, we presented the experiments on abstract representations of manipulation tasks. In order to be able to analyze robots' manipulation capabilities with respect to desired tasks, we need to be able to estimate the inverse kinematics. If forms the basis to allow a robot's manipulation at all. Hence, we presented such an estimation approach in Chapter 2. Now, we test it on the data sets extracted from observation of real human actions (Section 7.1). The robot has to grasp an object at a certain position and to move it to another position in an upright orientation. The robot's base positions are shown in Fig. 7.16. The orientation of the base is chosen in a manner, such that the joint axis of the first joint is perpendicular to the expected main plane of manipulation (e.g., table plane). Each path consists of $N = 20$ consecutive points.

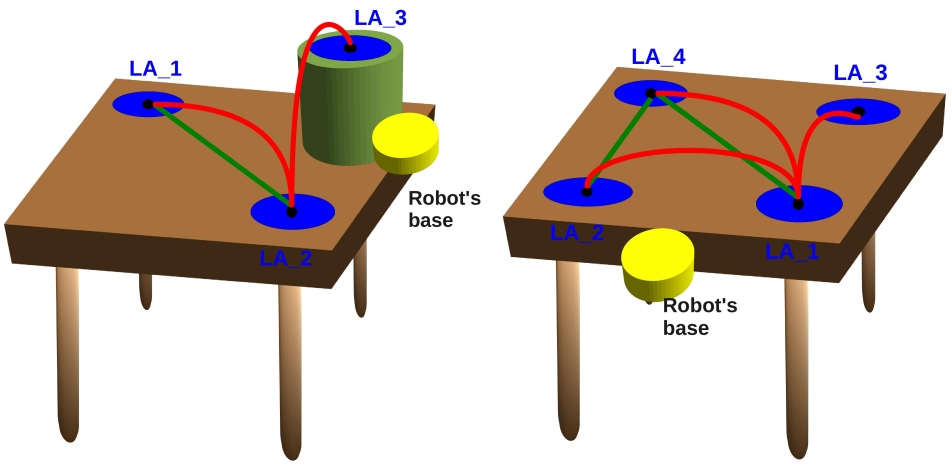


Figure 7.16: Data set I (left) and II (right) with the robots' base positions [4]. Similarly to Fig. 7.1, the lines/ curves between the blue Location Areas (**LA**) refer to trajectories of lifted objects (red) and trajectories of pushed objects (green). Additionally, the robots' base positions are shown in yellow.

7.3.1 Implementation

The implementation is done in C/C++. The stochastic approach for global minimization was presented in [26]. We use the implementation by Oliver Ruepp [195]. The simulated serial chain manipulator is described in Table 7.13. It consists of six rotational joints perpendicular to each other. We are also interested in the estimation of the inverse kinematics of a human-like robotic hand. The hand is supposed to be moved in a manner, such that the tips of the fingers and the thumb are able to reach desired grasp points on an object. The parameters of the human-like robotic hand are described in Table 7.14. It consists of a thumb and four fingers with three DoF each. Moreover, the hand has one additional DoF between the root of the thumb and the roots of the fingers. For simplification, this DoF is modeled as the fourth joint of the thumb. Each finger joint has a range from 0 to 120 degree. Fig. 2.3 illustrates the corresponding hand.

Table 7.13: Parameters of the serial chain robot (angles in degree, length in mm, EE = transformation to the end-effector).

joint i:	α_{i-1}:	a_{i-1}:	d_i:	θ_i:
1	0	0	0	θ_1
2-6	90	0	300	θ_i
EE	0	0	0	0

Table 7.14: Parameters of the human-like robotic hand (similarly to Table 7.13) with t=thumb and f_F = finger F. Joint 1 of the thumb is the joint closest to the tip of the thumb. The first joint of each finger is its corresponding root. B_F describes the transformation from the end of the thumb at the carpus to the base of finger F.

	joint i:	α_{i-1}:	a_{i-1}:	d_i:	θ_i:
t	1-3	0	20	0	$\theta_{t,i}$
t	4	0	40	0	$\theta_{t,4}$
t	EE	0	0	0	0
f_F	1-3	0	20	0	$\theta_{f_F,i}$
f_F	EE	0	20	0	0
B_1	EE	0	0	0	0
B_2	EE	0	0	20	0
B_3	EE	0	0	40	0
B_4	EE	0	0	60	0

The robot is supposed to reach the points along the desired paths. We introduce a tolerance around the each of these points. If the end-effector cannot exactly be moved on the point, but, it can reach the point within a tolerance t_h, the point is still defined to be hit. We set the tolerance for such a hit to $t_h = 20$. The manipulator can, then, easily reach all points on the paths. Often, several manipulator configurations exist as solutions. Hence, alternative configurations can be found as well. Then, the final configuration for a task execution can be chosen among these alternatives. For example, efficiency criteria can be considered for such a choice. In general, the smaller the tolerance is, the less configurations exist to reach the

desired points. On the one hand, the end-effector is closer to the desired points with any of the (estimated) solutions. On the other hand, there are less solutions and, therefore, less alternative configurations. In the worst case, no solution exists.
The variable γ (Eq. 2.3) works as penalization factor, if perspective solutions are close to each other. The higher the factor, the higher the penalization of known configurations. It is more probable, then, that the estimated solutions are significantly more different. In our case, the solutions need to differ, but, we do not search for very separated solutions. Hence, we set the factor to $\frac{1}{3}$ to penalize configurations smoothly, when they are close to each other. Two configurations are treated as different, if the euclidean distance $e_{\boldsymbol{c}}$ in between is at least u (Eq. 2.3). The larger the minimal distance u is, the further away the solutions need to be from each other. We choose a relatively small minimal distance with $u = 0.2$ rad. It allows us to find solutions which differ. At the same time, we are able to estimate a larger number of solutions. The scaling $\alpha = 300$ is large, since the range of the second term (error with respect to the desired orientation, in rad) in Eq. 2.4 is much smaller than the first one (error with respect to the desired position, in mm; workspace: 45 x 30 cm (Section 7.1.2)). If we want to ensure a very accurate estimation of the position of the end-effector, we need to put an emphasis on the first term. The scaling α has to be decreased, then. Similarly, if a very accurate estimation of the end-effector's orientation is required, the scaling needs to be increased. The search for further solutions is stopped, if either the result of the corresponding objective function is higher than $2.5 \cdot t_h$ or a maximal number of iterations is reached (200 in our experiments). If the result of the corresponding objective function is higher than $2.5 \cdot t_h$, we assume, that it is very unlikely to find a solution with a result smaller than the tolerance t_h. The maximal number of iterations is never reached in the experiments. It has been chosen in a magnitude, that it is never reached. However, depending on the application, it could be a good tool to speed up the estimation, since the estimation could be stopped earlier. If possible solutions get lost due to the maximal number of iterations, a further examination of the quality and the necessity of these solutions would be needed. For the estimation of inverse kinematics along entire paths, we apply the concept of adaptive tunneling (Section 2.3). We want to ensure, that the manipulator configurations are really close to each other along the path. Hence, we choose a very small initial search space around the previous configuration of the manipulator ($l_c = 0.01$). Furthermore, we set $m = 0.2$, in order to increase the search space just in very small steps to ensure smooth motions along a path (see Eq. 2.5-2.7). The larger both values are chosen, the faster the search space is enlarged. On the one hand, a solution can be found faster. On the other hand, the estimated solution can be further away from its ancestor, since it is determined in a much larger search space.

7.3.2 Results

We analyze the manipulation tasks for each data set. Moreover, we repeat each experiment five times to check the stability of the experiments.

At first, we estimate the inverse kinematics of the human-like hand. An exemplary virtual shut grasp is depicted in Fig. 7.17. Nearly all of the grasps are done successfully. There is just one grasp among all experiments and repetitions, which could not be applied in reality: Since we do not have implemented a collision detection yet, the fingers are intersecting themselves in form of a loop. A collision detection could be added to the objective function in Section 2.2 as penalization of undesired intersections. Once the hand has successfully grasped the object, we just need to position the end-effector of the serial chain manipulator appropriately. It has to be positioned in a manner, such that the computed grasp can be performed. Some of the solutions are illustrated in Fig. 7.18.As it can be seen, the solutions differ significantly.

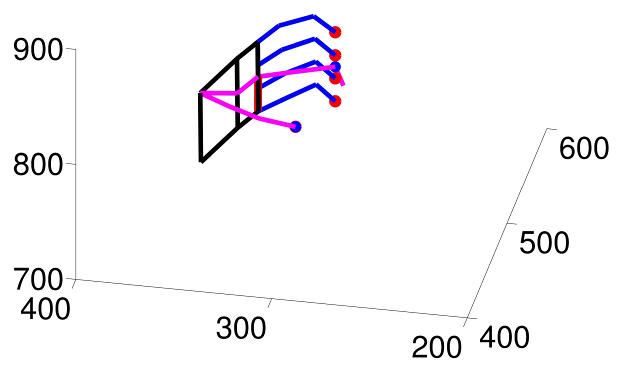


Figure 7.17: Exemplary virtual shut grasp, data set II [6]. The illustrated hand has a black palm, blue real fingers and a magenta model of the hand with the virtual finger, the thumb and the red orientation stick. The red dots are the aim positions of the real fingers, the blue one is the goal of the virtual finger. The black axes of the global coordinate frame give an impression about the magnitude of the hand (unit: mm). As it can be seen, the virtual finger does not exactly reach its aim position. However, the real fingers are achieving their aim positions precisely.

Afterward, the desired manipulation is performed. We focus on the serial chain manipulator. The hand needs just to stay in the known grasp position to hold the object. The desired manipulation consists of a 3D trajectory, on which the object has to be kept upright (e.g., a cup with coffee). We repeat the process of adaptive tunneling for different start configurations, if the start position can be reached with different configurations.
Fig. 7.19 shows the average and maximum of, resp., the joint speed, the angle around the horizontal axes and the residual of the position for data set II. In an ideal case, all three values are low. A low joint speed is a sign for smooth motions along the path. The angle around the horizontal axes should be close to zero, since the object is, then, kept upright as desired. A low residual in position means, of course, that the desired position is achieved accurately. The best result is shown on the left, the worst on the right. Even the worst result has very desirable low values. This shows clearly, that the direct optimization of the angle around the horizontal axes and the residual in position (see Eq. 2.4) succeeds. The desired low joint speeds are achieved

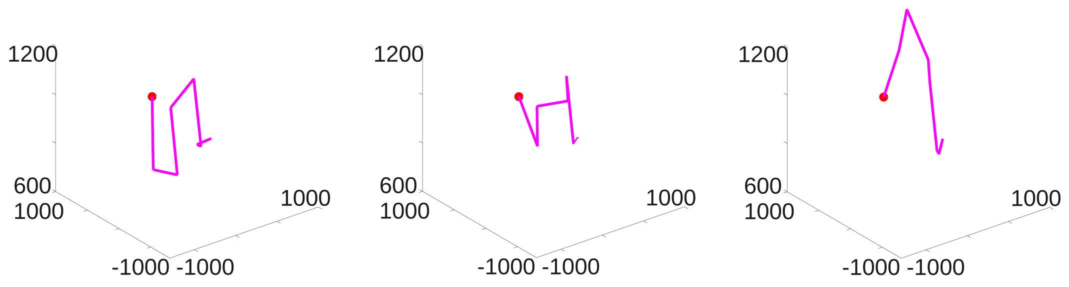


Figure 7.18: Exemplary manipulator configurations [6]. Configurations of the magenta manipulator with the red base for the same position of the end-effector are drawn (data set I). For illustration, a stick stub was added to the end-effector instead of the hand to show its orientation more clearly. The black axes of the global coordinate frame are given as orientation between the subfigures (unit: mm).

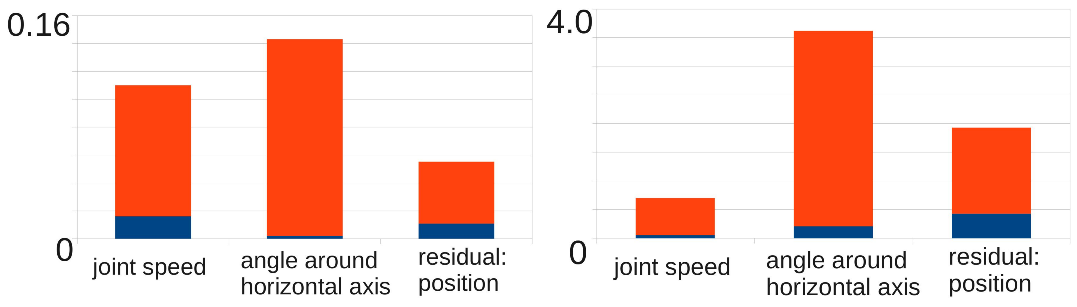


Figure 7.19: Statistical results of the adaptive tunneling [6]. Statistical results of the best (*left*) and the worst (*right*) results of adaptive tunneling for data set II. Each figure shows the blue mean and the orange maximum of the joint speed (in rad/ time unit), the angle around the horizontal axes (in degree) and the residual of the position (in 10 mm).

indirectly through the limitation of the search space in the adaptive tunneling method. The results for data set I are even better. Fig. 7.20 shows an exemplary development of joint speeds. On the right, we see, that one trajectory is done at an extremely low speed, but in a zig-zag-pattern. This can be explained through the application of the Stochastic Optimization: We search for the solution in a very small search space at the beginning. Within this space, the solution is picked arbitrarily. The joint speed of the trajectory in Fig. 7.20 is so small, that the solution is already found within this extremely small search space. Some trajectories have a peak in the development of the joint speed, but even this peak is very low with 0.5 rad. Fig. 7.21 illustrates some consecutive configurations of the manipulator along the trajectory. As we can see, its motions are smooth.

Concerning the computation time and the stability of the solutions, we achieve satisfying results. The estimation of the inverse kinematics of one single point takes between 15 and 60 seconds, depending on the number of possible solutions (between 20 and 80 solutions for a

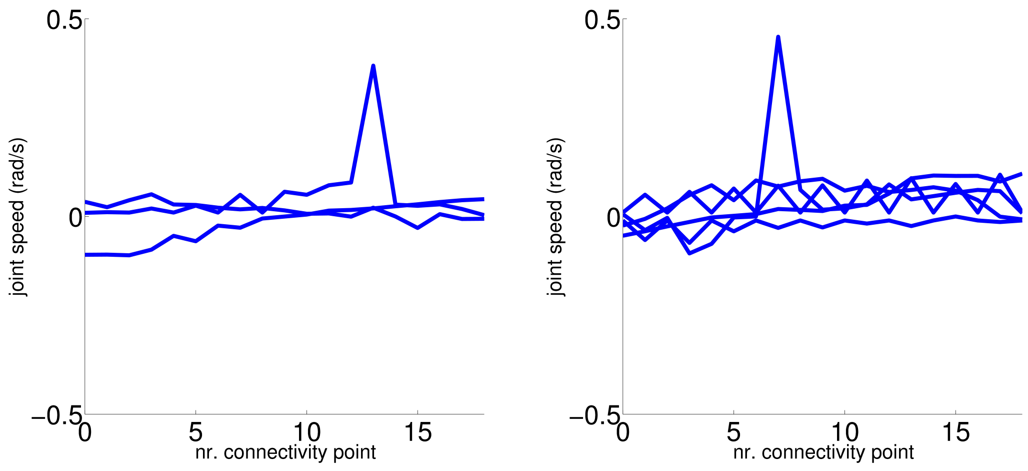


Figure 7.20: Exemplary joint speeds along the trajectories [6]. The x-axis can be seen as the time axis, if 1 s is needed to move to a consecutive point. *Left:* Speed of joint 1 in data set I. *Right:* Speed of joint 4 in data set II.

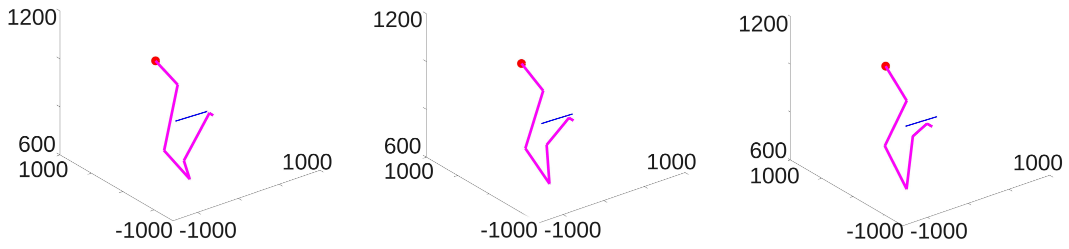


Figure 7.21: Consecutive configurations of the manipulator along a trajectory [6]. The robot is depicted at position 1, 5, 10 (data set I) on the blue trajectory. Illustration similarly to Fig. 7.18 (unit: mm).

point in our experiments). The adaptive tunneling is started with the three best solutions of the inverse kinematics of the start positions, the best one according to the statistical values in Fig. 7.19 is chosen at the end. The adaptive tunneling for the entire trajectory is done in 45-90 seconds. The inverse kinematics of the entire human like robotic hand is estimated within 5-10 seconds. We tested several local optimization methods (e.g., Matlab Optimization Toolbox: simplex search method, Trust-Region Dogleg method, Levenberg-Marquardt method) on the same data sets, but all of them got stuck in local minima. It was not possible to find the inverse kinematics for *all* desired points with any of these methods. This forced us to search for a *global* estimation of inverse kinematics. The small peaks in the speed profiles (see, e.g., Fig. 7.20) are negligible, if one considers, that it was not possible to find solutions with another method.

The number of solutions is stable for the inverse kinematics of a single point in data set I with just two outliers (just about 25 solutions found instead of about 40). For data set II, the estimation of the inverse kinematics seems to be more difficult for two places: About 50 solutions are found in ca. 50% of the repetitions, while just the half of this amount is determined in the other repetitions. The amount of possible solutions depends also on the definition of "different solutions", e.g., the tolerance t_h and the minimal distance u between two different configurations. Moreover, the number of possible solutions can differ, depending on the real distance between two configurations. Nevertheless, the large amount of solutions in our experiments shows, that we find many possible solutions, even in the worst case. As described, the three best results of the inverse kinematics of the start position are evaluated before the adaptive tunneling. The best result is taken at the end. A third of the other solutions have single higher outliers in the joint speed about 1.5 - 2.5 rad/ time unit, the other values are approximately the same. The repetitions show similar results, just one outlier occurs in one repetition of one trajectory for the joint speeds (about 3 rad/ time unit). The estimation of the inverse kinematics of the hand is very stable, there is just one outlier among all repetitions (residual of 10 mm for a finger; normally a residual about 1.5 mm), which is still very small.

7.4 Path Optimization for Abstractly Represented Tasks with Respect to Efficient Actuation

In this Section, we present the experiments on the proposed path optimization (Section 4). In the first part of this Section, the experiments on the path *configuration* with respect to efficient actuation are processed. These experiments indicate, whether there is a potential to improve the efficiency. The experiments on the proposed path *optimization* are shown afterward.

7.4.1 Path Configuration

The experiments on the path configuration method are performed on simulated and real world data. We use simulated scenes for the first scenario A and real scenes for the second scenario B. The first simulated scene (Scene I, scenario A) has simply two Location Areas on a table (distance between both Location Areas: 400 mm). The second simulated scene (Scene II, scenario A) consists of three Location Areas, which are placed on the corners of a quadrilateral (side length of the quadrilateral: 400 mm). Both scenes are simple test scenes. They form a basic test frame for the path configuration method. In the real world scenario (scenario B), we use the Location Areas of the Functionality Maps extracted in Section 7.1 (illustrated in Fig. 7.1). The paths between the Location Areas can be applied to manipulate objects. Each path consists of 20 points.
The factor e ranges from 0.5 to 2.0 with a step size of 0.5. The angle β is chosen from the set {-45°, 0°, +45°}. The wiggly line is built up on the half quadrilateral with a perpendicular deviation from the half quadrilateral of 20 mm (see also Section 4.1).

A simple 3-DoF robot with three rotational joints perpendicular to each other is simulated. All links have a length of 400 mm (DH-Parameter: d_1, d_2, a_3). An end-effector position is defined to be reached, when the distance between the desired and the real position is below 50 mm (tolerance t_h in Section 7.3).

The position of the robot's base is varied on circles and rectangles, which are spanned up around the mean of all Location Areas. The circles and rectangles have three different sizes (Fig. 7.22) and three different heights (table height and above). In experiment (A) scene I, the radius of the circle is chosen within the set {400 mm, 500 mm, 600 mm} and the half of the side length of the rectangle has a value of the set {300 mm, 400 mm, 500 mm}. In all other experiments, these values are increased by 200 mm (radius: {400 mm, 600 mm, 800 mm}; half of the side length of the rectangle: {300 mm, 500 mm, 700 mm}). The height is chosen within the set {0 mm, 200 mm, 400 mm}. Experiment (A) scene I consists just of two Location Areas, hence, smaller values are sufficient to place the circles and rectangles out of the region of the Location Areas. In the other experiments, three or four Location Areas are used. Therefore, larger values are necessary to span the circles and rectangle outside of the Location Areas. We choose six equally distributed points on each circle and eight points on each rectangle. Hence, a large number of different base positions ((6+8)x3x3=126) are distributed around the Location

Areas, where the manipulations take place. All of them are used in the experiments on the path configuration.

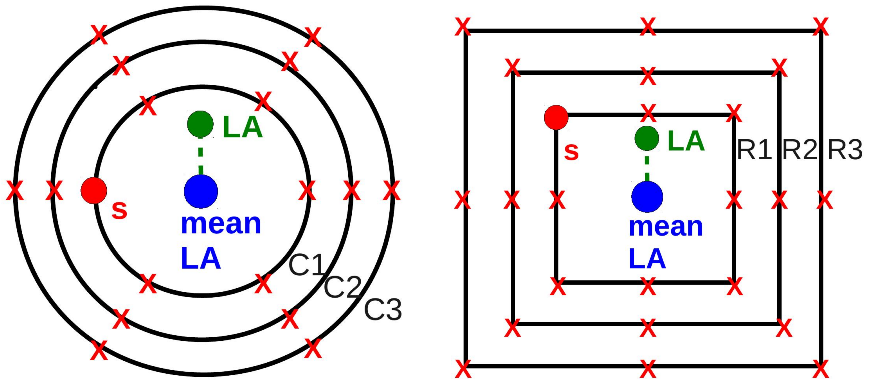


Figure 7.22: Possible positions of the robot's base in relation to the Location Areas (LA). The red crosses show the different base positions on circles and rectangles spanned up around the mean of all Location Areas.

We analyze the properties of the 100 best results for each scene. This large number should give a good overview of the preferred properties for the path configuration. The best results are defined as the paths, for which the maximally required joint speeds are the lowest. The maximal change of the joint speed (maximal acceleration) is very small in all scenes of the 100 best results (below $7 \cdot 10^{-5}$ rad/s; average joint speed for comparison: 0.004 - 0.038 rad/s). Therefore, the maximal change is not included in the definition of the best results here. In general, we just evaluate paths, which can be reached by the manipulator (within the tolerance of 50 mm). Fig. 7.23 shows the ratio of each basic motion shape in each scenario. It is surprising, that a line, which is the shortest connection between two points, is hardly among the best results in the scenarios. The line seems to be a demanding motion shape. Even the wiggly line and the half quadrilateral are significantly more often among the 100 best results than the straight line. The half circle is the most favorite motion shape. In general, compressed paths are preferred, which lead to relatively short connections (compression/ elongation factor $e = 0.5$ in 80% of the best 100 results). At first, this seems to be quite logical. However, the line (the shortest connection) is hardly among the best 100 results. Apparently, the path of a straight line is more demanding to reach than a compressed path in one of the other three shapes.

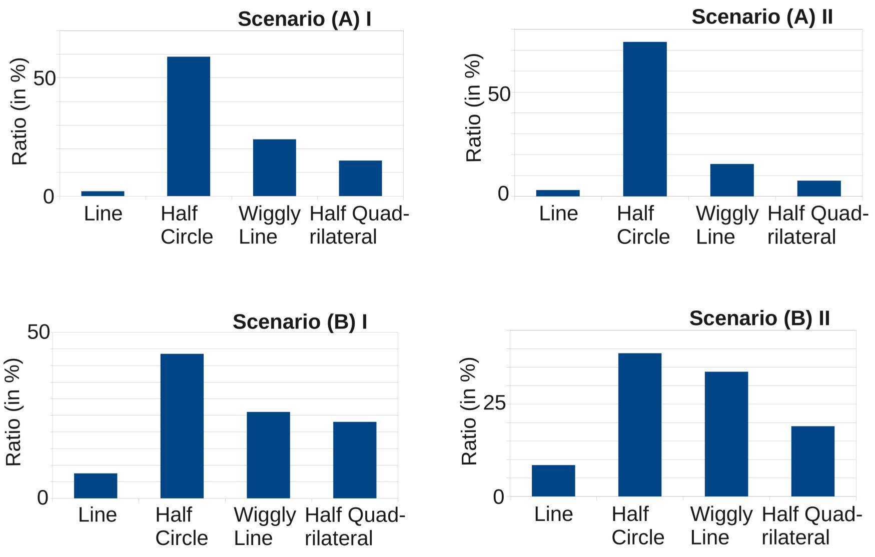


Figure 7.23: Ratio of each basic motion shape (among the 100 best results) [3].

Fig. 7.24 illustrates the most favorite motion shapes and compression/ elongation factors of the paths in general. The figure visualizes clearly, that the shortest connection, the line, is hardly among the best results. The compressed paths are preferred, especially the curve shaped one.

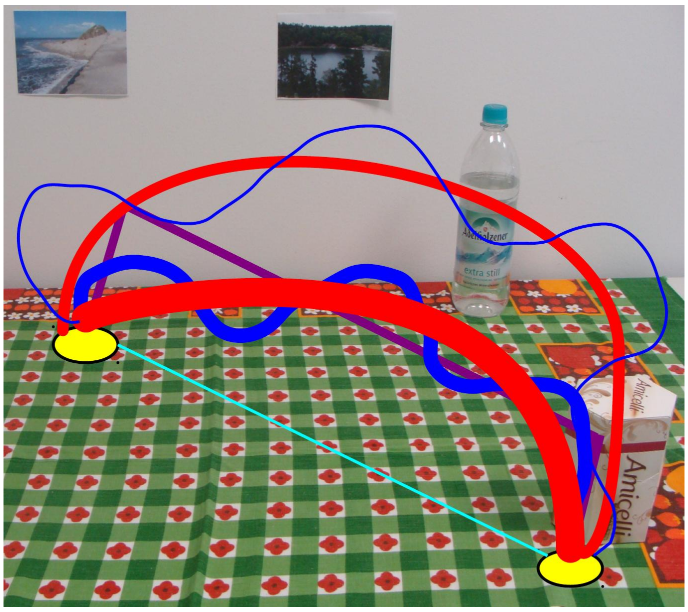


Figure 7.24: Illustration of the preferable basic motion shapes and compression/ elongation factors among all scenarios [3]. The thicker a line/ curve, the higher its preference. As it can be seen, the half circle is the most favorite motion shape. Moreover, it is visible, that a small compression/ elongation factor is preferred.

The 100 best results do not show one general value preferred as bias (angle β). For example, in scene I of experiment (A), the positions, which are further away from the mean Location Area are preferred. A further analysis of the relation between the angle β and the chosen positions shows, that β is pushing the trajectory away from the chosen base most of the times. In scene II of the same experiment, this effect is less strong, but still visible: Smaller distances are preferred here and the β is often pulling the trajectory towards the robot's base. For experiment (B), a similar observation can be done, even though the effect is less significant. It seems to be used as a fine adaption of the distance between the base and the desired points of the end-effector. The path can be "pushed away" or "pulled" towards the robot's base, in order to enable more efficient actuation.

An additional result of our work are the preferable areas of the robot's base with respect to efficient actuation. We analyze the base position of the 100 best results in this experiment. Most of the areas are on the side of the desired path at table height. A base placement at table height is preferred in three out of four scenes (Fig. 7.25). Just in scenario A, Scene I, the

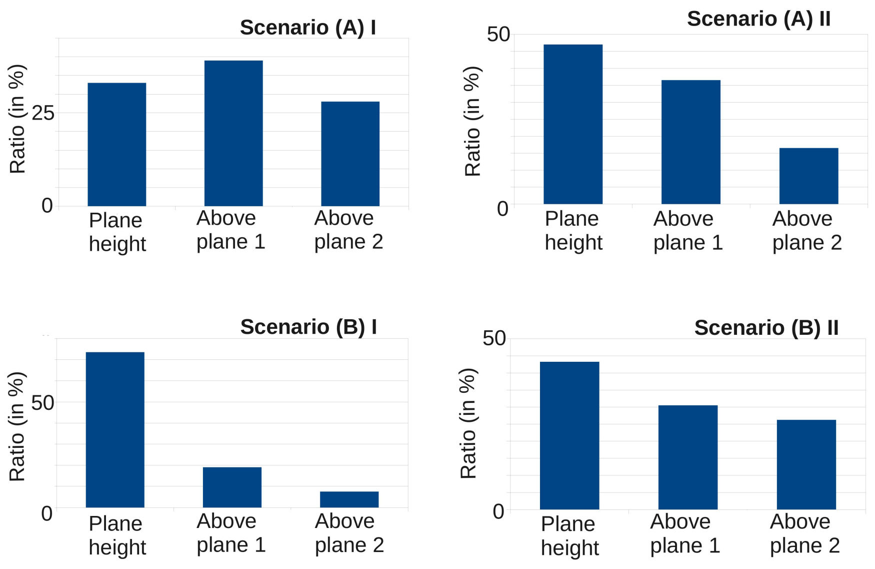


Figure 7.25: Ratio of height of the robot's base [3]. "Plane height" is the height at the table plane. "Above plane x" refers to a height "x" above the table. The higher "x", the further away the height from the table.

base positions of the 100 best results are approximately equally distributed among the three different heights. The height of the base does not seem to influence the efficiency of actuation in this scene.

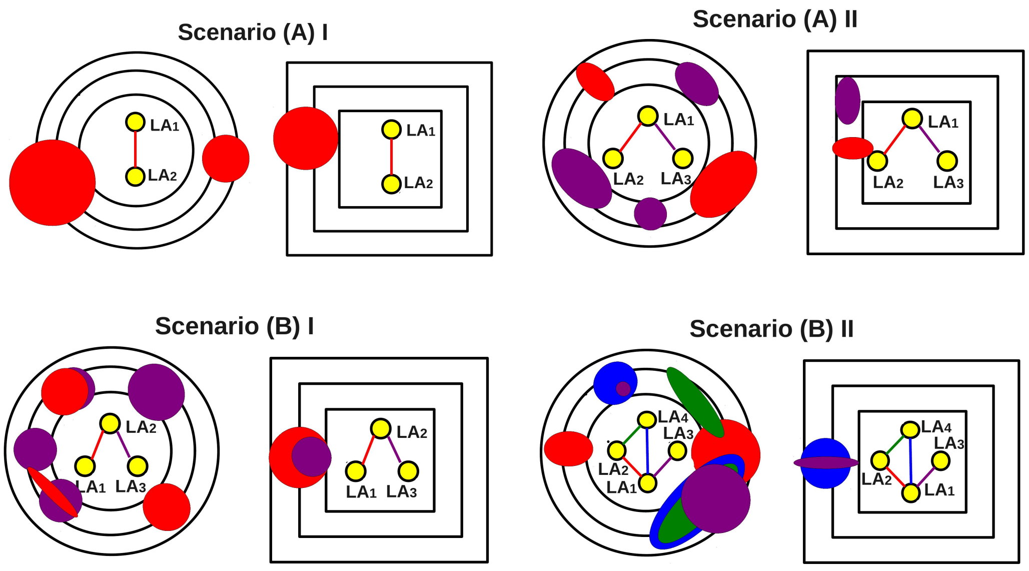


Figure 7.26: Exemplary preferred positions of the robot's base in relation to the Location Areas (LA) [3]. The color of the preferred regions (red, magenta, blue and green circles, resp., ellipsoids) for the robot's base refers to the corresponding path in the same color.

Fig. 7.26 shows, that base positions on the side of the desired path are preferred. Apparently, the manipulator can work more efficiently, if the manipulation area is on its side. If the manipulator is placed behind a start, resp., end position, the manipulation area is further away. The manipulator's capabilities probably decrease and, hence, the efficiency decreases as well. It is interesting to see, that the base positions on the rectangles are hardly among the 100 best results. The base positions on the rectangles do not seem to allow an advantageous positioning of the base in comparison to the positions on the circles.

Our experiments show, that the intuitive solution does not necessarily agree with the results optimizing for efficiency. Therefore, it is worth to further analyze the path properties with respect to efficient actuation. Hence, the proposed path *optimization* method is presented in the next Section.

7.4.2 Path Optimization

After the promising results of the path configuration method, the experiments on the optimization of the path follow. The experimental setup and the implementation are described first. The results are shown afterward.

7.4.2.1 Experimental Setup and Implementation

The simulated manipulator consists of three rotational joints perpendicular to each other ($D = 3$). All links have a length of 300 mm (DH-Parameter: d_1, d_2, a_3)

During the optimization, the configuration of the path points is changed. However, the path points need to be arranged in a manner, such that characteristic properties of the original manipulation are still kept. Therefore, we limit the possible change of a path point x to a sphere S_x with radius r_x (see Section 4.2.3). We allow each joint j to change a point on the path partially within the sphere. Hence, the part within which, resp., one joint is allowed to change a path point is limited to r_x/D (D is the number of joints as before). The Jacobian is used to transfer the part r_x/D to the configuration space. This transferred part gives us the new range, within which the respective joint can be moved. The optimization is, afterward, processed within the new ranges.
The radius r_x of each sphere depends on the position of the corresponding path point x within the path. The closer the point to the start of end point (Location Areas), the more limited it is (see Eq. 4.10). This allows to ensure, that the path leads to the desired start and end point. The magnitude of the radius depends on a constant basic distance C (see Eq. 4.10), which scales the radius. It is chosen as $C = D \cdot 10.0$ mm. Hence, the constant basic distance is 10.0 mm per joint. A definition in dependence of the number of joints is reasonable, since each joint is allowed to change partially with r_x/D (see above). The choice of 10.0 mm as constant basic distance is per joint is large enough to allow a large search space is enabled for the points in the middle of the path. At the same time, it is sufficiently small to ensure, that the path leads to the desired start and end point.

The path consists of $N = 5$ points. We choose a relatively small number of points, since we want to optimize the overall path in a very short computation time. Intermediate points can be computed after the optimization by a simple partial movement of the joints if desired.

In order to keep the paths of pushed objects on the original height h_o above the plane, we add a residual R_x for each point p_x to the Objective Function O:

$$R_x = \begin{cases} 0 & \text{if } \|h_o - h_x\| \leq t_{h,h}, \\ A \cdot \|h_o - h_x\| & \text{otherwise.} \end{cases} \tag{7.4}$$

h_x is the current height of the point p_x above the plane. We introduce a tolerance $t_{h,h}$ around the desired height h_o. The tolerance has to be chosen sufficiently small to ensure, that the path refers still to a path of a pushed object. At the same time, a small tolerance allows a limited freedom to configure the path points in an advantageous manner.

The placement of points outside the region of the tolerance is penalized with a factor A.
We use the normal vector of the corresponding table plane to compute the desired and current height of the points. Of course, $R_x = 0$ for objects which are not pushed during the manipulation.

The estimation of the inverse kinematics is done through the approach presented in Section 2. The underlying stochastic approach for global minimization in [26] is also used for the optimization of the path. The original path points are uniformly distributed along a line (if the object is pushed) or along a half circle which is positioned upright above the table (if the object is lifted). We estimate the three best configurations for the start point to position the manipulator as close as possible to the desired start point. Afterward, we estimate the inverse kinematics along the entire paths (see adaptive tunneling method in Section 2.3) for all three configurations. The resulting three configuration sets are optimized with the proposed method. The best one is taken at the end.

Similarly to the already presented experiments (e.g., Section 7.3), we introduce a tolerance around each desired point. If the manipulator can reach a point within the tolerance, the point is defined to be reached. In this experiment, just the grasp has to be extremely accurate (tolerance below 5 mm). Once the object has been grasped, we can extend the tolerance. The points after the start point are considered as reached by a certain configuration, if the distance of the real end-effector position to its desired position is smaller than a tolerance t_h (defined similarly in Section 7.3). It is chosen in a manner, such that all path points can be reached by the manipulator. The points of data set I are reached with a tolerance t_h of 50.0 mm. For data set II, the tolerance has to be extended to 75.0 mm to ensure, that all end-effector positions are within the tolerance.
In our experiments, we want to compare the efficiency in actuation of the robot along the original path and along the optimized path. In order to show the performance of our system, we need to determine configurations at the original points, which should already be advantageous with respect to actuation and not arbitrarily. Hence, we determine just an arbitrary start configuration of the manipulator and search for consecutive configurations which are close to their corresponding ancestors along the path (see adaptive tunneling in Section 2.3).

The tolerance $t_{h,h}$ for the allowed height above the table is set to 5.0 mm. It is relatively small in comparison to the entire manipulation area (45 x 30 cm (Section 7.1.2). This ensures, that the property “pushed object” is preserved. Similarly to the tolerance t_h, tolerance $t_{h,h}$ had to be extended to 25.0 mm for data set II, to ensure, that all end-effector positions are within the tolerance.

The placement of points outside the region of the tolerance is penalized with a factor of $A =$ 10.0 (Eq. 7.4). The results of the experiments show, that the points are not placed outside the allowed region.

7.4.2.2 Results

The overall results of the optimization are depicted in Fig. 7.27 and 7.28. They visualize clearly, that nearly all of the initial values of the Objective Function (Eq. 4.9) have been decreased as desired. This shows, that a modification of the path configuration leads a significant improvement of the efficiency in actuation in nearly all cases.

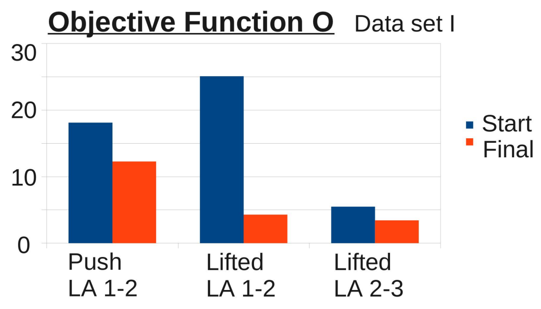


Figure 7.27: Result of the Objective Function, data set I [5]. The figure shows clearly, that the values of the Objective Function decrease from their blue initial values to the orange final values after the optimization (push/ lifted = pushed/ lifted object along the path, **LA** = Location Area).

More details of the results are shown in Fig. 7.29-7.32. They show the change of the acceleration peaks for each single joint along the path. An advantageous change of such a peak means, that it is either reduced or it is merged with another peak. This refers to an efficiency improvement in actuation, since the required accelerations are either reduced or a change in the acceleration direction has been eliminated (see Section 4.2.2).
A look at the change of the acceleration peaks is most of the times enough to see the improvement after the optimization clearly (see, e.g., Fig. 7.29). Just the optimization of the path of the pushed object between Location Area (**LA**) 1 and 2 requires an additional look at the change of the joint speed to see the improvement (Fig. 7.30). The acceleration peak is shifted to another time step, where the corresponding joint speed is significantly lower. The resulting speed profile is more smooth, then. Sometimes, an acceleration peak might be increased slightly to enable a significant reduction of another peak as, e.g., in Fig. 7.31.

In the experiments on the paths of the pushed objects in data set II, we can see, that the paths are not only optimized with respect to actuation. The distances between the desired and the real height above the table are also improved (see Fig. 7.32 and 7.33).

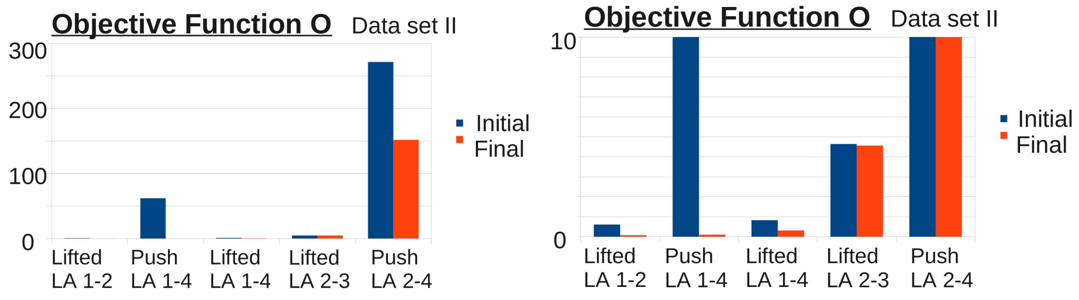


Figure 7.28: Result of the Objective Function, data set II [5]. The figure on the left shows the same results as the figure on the right. However, the figure on the right has a smaller data range on the vertical axis for a better illustration of the smaller values. As is can be seen, most of the values can be significantly reduced. This shows, that the modification of most of the paths has a strong effect. Just one path (lifted object between **LA** 2 and 3) can hardly be improved with respect to actuation.

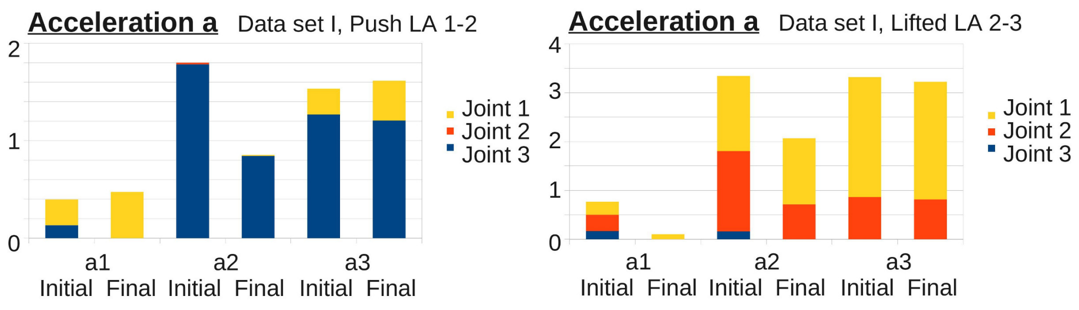
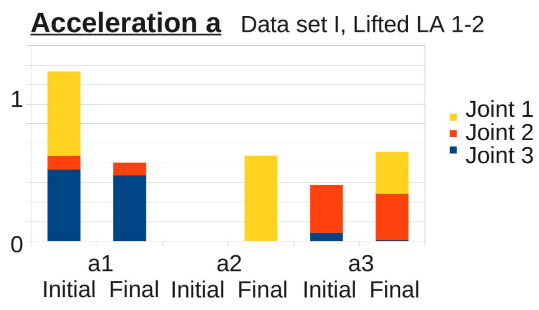


Figure 7.29: Acceleration values per joint and per acceleration time step a_x along the path [5]. The acceleration time step a_x (in rad/s^2) refers to the time between point p_x and point p_{x+2}. For each acceleration time step a_x, the acceleration values at the beginning and the end of the optimization are shown (initial, resp., final values). The depicted acceleration values are the $\triangle v_{j,i}$ values which are relevant for the final Objective Function O (Eq. 4.9) at their time of occurrence x. Hence, the peaks of the hills in the acceleration profile are shown. The acceleration peaks of the path for the pushed object between Location Area (**LA**) 1 and 2 show a significant reduction of the acceleration of the third joint at a_2 (*left* figure). Moreover, the blue peak of the acceleration at a_1 is not existing any more after the optimization. This means, that the original hill is merged with another hill, in this case the hill with the the peak at a_2. If hills have been merged, an undesired zero crossing has disappeared. Such a merge of hills is one of the desired aims of our approach. Some of the depicted acceleration values are very small, so that they are hardly visible (e.g., joint 2 on the left). Very desirable results are shown on the *right*: Most of the peaks are clearly reduced.



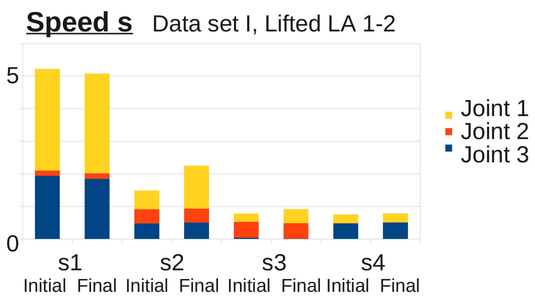


Figure 7.30: Acceleration values per joint and per acceleration time step a_x (*left*), and the corresponding speed values (*right*) [5]. The left figure shows the acceleration values similarly to Fig. 7.29 (unit: rad/s^2). The improvement of the peaks of the acceleration becomes just clear in the comprehensive survey with the speed values. Instead of an acceleration peak of the first joint at the beginning at a_1, the peak is shifted to a_2. This is useful, since the corresponding speed value s_2 is significantly smaller than s_1 (resp. the yellow final values). Hence, the result of $|v_{j,i}| \cdot |\triangle v_{j,i}|$ in Eq. 4.9 has a smaller magnitude than the original one at initialization.

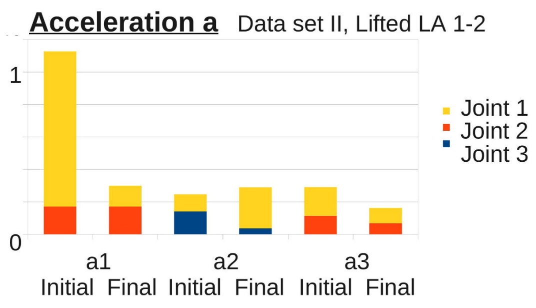


Figure 7.31: Acceleration values per joint and per acceleration time step a_x along the path (similarly to Fig. 7.29) [5]. Especially the peak of the joint 1 for the lifted object between **LA** 1 and 2 is significantly decreased at a_1 (unit: rad/s^2). Therefore, the slightly increasing peak of the same joint at a_2 is an acceptable price to pay and can be easily compensated in the overall result of the Objective Function. All acceleration peaks of the other paths of lifted objects in data set II are clearly reduced, too (not depicted).

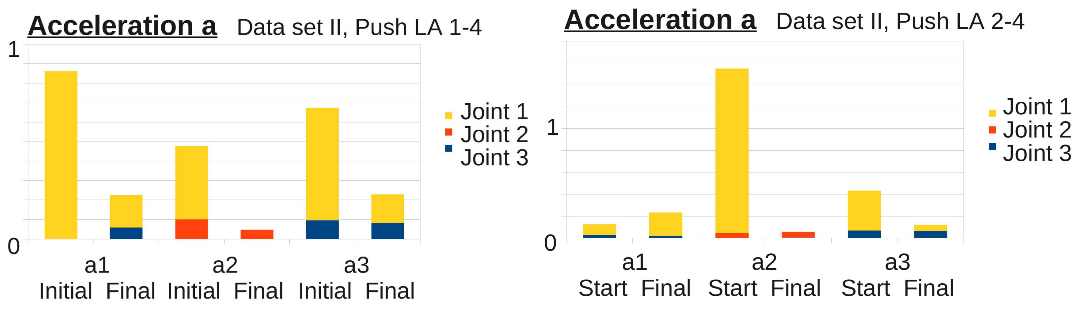


Figure 7.32: Acceleration values per joint and per acceleration time step a_x along the path (similarly to Fig. 7.29) [5]. As it can be seen, nearly all of the acceleration peaks along the paths of pushed objects in data set II are significantly reduced (unit: rad/s^2). Additionally, the new points are also better positioned with respect to the table plane than the original points (see Fig. 7.33).

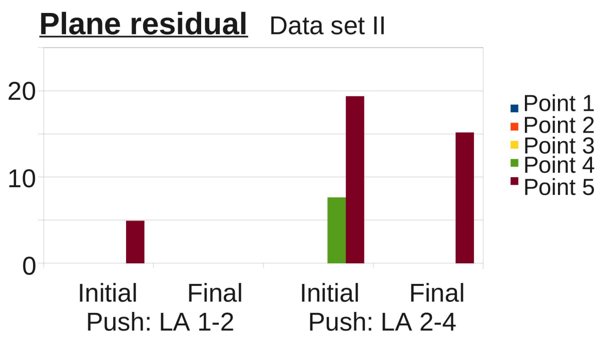


Figure 7.33: Remaining residual with respect to the table [5]. The residual (in mm) is shown for each point along the paths of pushed objects in data set II. As it can be seen, the proposed optimization method reduces also the remaining residuals along the path. As described, the residuals are included in the Objective Function. This explains the large magnitude of the values of the Objective Function in Fig. 7.28 for the paths of the pushed objects in comparison to the paths of lifted objects.

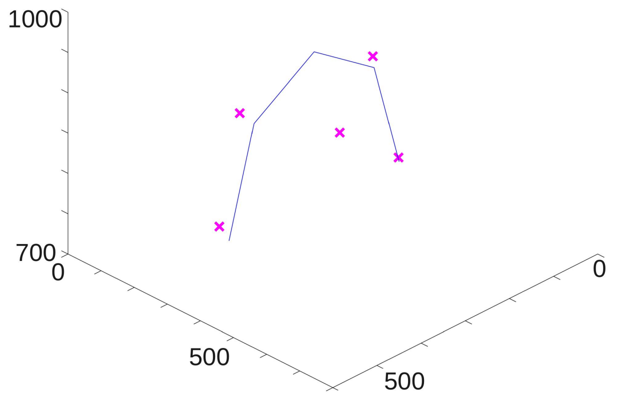


Figure 7.34: Exemplary original path in comparison to the new points [5]. The original path is drawn in blue in comparison to the new magenta points along the path of a lifted object **LA** 1-2, data set I (unit: mm). The possibility to change the position of the middle point at a large extent is clearly used here.

The change of the path configuration itself is illustrated in Fig. 7.34 for a path of data set I. It is clearly visible, that the position of the points in the middle of the path has been changed within a large radius (see Section 4.2.3).

The runtime of the method and the stability of the results are good. We want to achieve very accurate start configurations with respect to the desired start position in 3D. Hence, we accept a relatively long runtime for the estimation of a start configuration with about 20 seconds. We use the stochastic optimization approach to be able to compute several different start configurations and to be able to apply the implementation independently of the build-up of the robot. In general, any other procedure to estimate/ compute the inverse kinematics could be used as well. The optimization of the path itself took less than one second per path. We optimize the paths of the three best start configurations and take the best result after the optimization (Section 7.4.2.1). About the half of the other two results is just slightly worse than the corresponding best result (until +1 in the result of the overall Objective Function). The other half has clearly worse results (between +10 and +30 in the result of the overall Objective Function). A repetition of the experiments might lead to slightly varying results, since we use the stochastic optimization approach to estimate the inverse kinematics. We repeat the experiments three times, hardly any change of the results is observable.

To sum up, we achieve very desirable results in our experiments, since the efficiency in actuation is significantly improved most of the times.

7.5 Structure Analysis of Manipulators

After the extraction of the validation scenarios (Section 7.1) and the experiments on the required components (estimation of inverse kinematics in Section 7.3, path optimization method in Section 7.4), we analyze manipulator structures as proposed in Section 5. After a short description of the experimental setup and the implementation, the results of the analyses are presented. First, the general Maneuverability Analysis is processed, before the analysis with path optimization follows.

7.5.1 Experimental Setup and Implementation

The context of our experiments is the described manipulation task (Section 7.1). We analyze the robot's capabilities with respect to this task under the failure of none, one or multiple joints.

The Maneuverability Analysis is processed for two different 6-DoF manipulators. The first one is an test robot, which has simply consecutive rotational joints perpendicular to each other and all links have a length of 300 mm (DH-Parameter: d_i, a_5). The other manipulator is a widely known Unimation PUMA 560. The robot's base positions are shown in Fig. 7.16. The orientation of the base is chosen in a manner, such that the joint axis of the first joint is perpendicular to the expected main plane of manipulation (e.g., table plane).

We use a fixed angular rate of $\dot{\theta}_{fix} = 1.0$ rad (Eq. 5.4) to compute the Maneuverability Volume. The magnitude has been chosen arbitrarily for illustration. It is large enough to move the joint in a clearly visible way ($\frac{1}{6}$ turn) without causing very large volumes. Moreover, we focus on the parallelepiped K of the Maneuverability Volume (Eq. 5.10). The parallelepiped L puts just an additional emphasis on *perpendicular* base velocity vectors. Therefore, K is prioritized with a weighting of $\omega_k = 0.8$ in contrast to $\omega_l = 0.2$. For the Spinning Pencil (Section 5.2), the angle γ is set to 180°, considering the maximally crossed distance in the turn of a joint in the worst case. The weighting of the different components of the Spinning Pencil in Eq. 5.15 is set to $\omega_1 = \frac{1}{12}$, $\omega_2 = \frac{1}{12}$ and $\omega_3 = \frac{5}{6}$. This leads to an equal weighting of the first two terms and an emphasis on the third term in the equation. Both of the first two terms show how perpendicular the joint axes are to each other (in 2D and 3D). The third term is important to us, since it supports the desired *thin* Spinning Pencil. We set a strong priority on the Maneuverability Volume in the Maneuverability Analysis MA_l with $\omega_m = 0.75$ and $\omega_s = 0.25$ (Eq. 5.16), since we prioritize the capability to move the object in the manipulation experiment.

Concerning the weighting of the Location Areas, we choose the following priorities. Characteristic area 2 in data set I is chosen as the most important area with a weighting of 0.5. It is located on the table and can, therefore, be used for more, and also more complicated manipulations than the other Location Areas. These act as places of storage and have a weighting factor of 0.25. The weighting of the Location Areas in data set II is related to the number of in- and out-coming trajectories. The more in- and out-coming trajectories, the higher the weighting.

7.5.2 Results: Maneuverability Analysis

The end-effector needs to reach certain positions in order to perform a desired task. We define a position as reached, if the remaining distance between the desired and the real position is smaller than a tolerance t_h (defined similarly in Section 7.3). We use a city block distance $p = p_p + \alpha \cdot p_o$ with the distance p_p in 3D position (unit: mm) and the distance p_o in orientation (unit: rad) to measure the remaining distance. The variable α is used to compensate the different scales of p_p and p_o ($\alpha = 300$ showed good results in our experiments). The tolerance t_h is set to 25 in the experiments with no, resp., one broken joint. A higher tolerance $t_h = 150$ is necessary in the experiments with two or three broken joints, in order to be able to reach the desired positions at all before the remaining maneuverability can be analyzed. This shows already, that the failure of multiple joints influences the capabilities of the robots significantly, since the robot is often not even able to move its end-effector to a desired position. If several configurations are possible to reach such a desired position, we analyze all of them and take the most advantageous one according to the Maneuverability Analysis. Hence, our experiments are not limited to the analysis of a single configuration.

At first, the Maneuverability Analysis is processed for the entirely working robots in comparison to the robots with one broken joint. As the results in Fig. 7.35 and 7.36 show, each robot has joints, which are essential to reach the desired positions. In contrast, a failure of another joint might hardly affect the performance. It is interesting, that joint 4 of the test robot is essential for both tasks. Moreover, a failure of one of the last three joints of the PUMA hardly affects the performance as in both data sets. This can be explained through the physical structure of the PUMA (see description below Fig. 7.35). In the following, we focus on the discussion of data set II, since both robot's have approximately the same magnitude of the results in the Maneuverability Analysis, what enables a better comparison of the joint failures. An evaluation of data set I would be done similarly.

As already mentioned, the tolerance for a hit is enlarged for the experiments with two or three broken joints. This is important for the discussion of the results for the robots with two broken joints in Fig. 7.37. For example, Fig. 7.36 shows, that the PUMA does not reach its desired position if joint 1 fails. Therefore, the PUMA should not be able to reach its desired positions if joint 1 and a further joint are broken. Due to the necessary extension of the tolerance, the PUMA is able to reach its desired position if, e.g., joint 1 and 4 are broken. Fig. 7.37 shows the joints, which are essential under the extended tolerance.

Of course, we expect, that a higher number of working joints increases the magnitude of the result in the Maneuverability Analysis. The comparison of the experiments with a different number of broken joints confirms this widely (see Fig. 7.38). One has to consider the larger tolerance for a hit in the experiments with two or three broken joints. Therefore, it is possible, that the robot can reach points in the additional tolerance with new configurations. The new configurations can be advantageous with respect to the Maneuverability Analysis. Hence, the

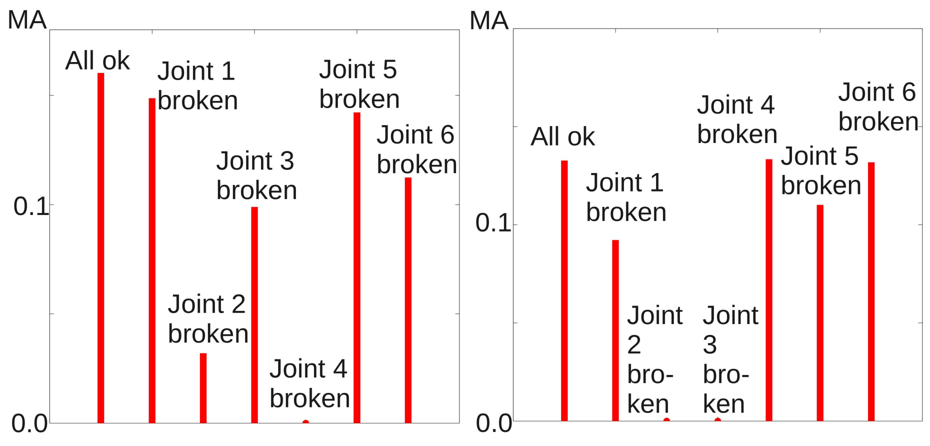


Figure 7.35: Complete manipulator vs. manipulator with one broken joint for the test robot (*left*) and the PUMA (*right*), data set I [4]. As it can be seen, there are essential joints: If these joints fail, the robot cannot reach the desired positions any more, since the Maneuverability Analysis results in $MA = 0$ (joint 4 of the test robot; joint 2, resp., 3 of the PUMA). The failure of other joints can reduce the maneuverability (e.g., joint 3 of the test robot), whereas the failure of some others hardly affects the maneuverability in our requirements (e.g., joint 4 of the PUMA). We have put an emphasis on the capability to move the end-effector rather than to turn it in our requirements. Therefore, it is quite consequential from the PUMA's structure, that the failure of one of the last three joints of the PUMA reduces the robot's maneuverability less significantly than a failure of one of the first three joints. The last three joints are significantly closer to the end-effector than the first three (see link lengths of the PUMA). Hence, the first three joints can be used to bridge a large gap to a desired position, whereas the last three joints can hardly perform this. It becomes clear, that especially joint 2 and 3 are essential for the PUMA to reach the desired positions in data set I.

result of the Maneuverability Analysis of the test robot with two broken joints can be better than with one broken joint.

Fig. 7.39-7.43 show some of the resulting Maneuverability Volumes and Spinning Pencils in a simplified, but more illustrative manner. We just show the parallelepiped K as Maneuverability Volume. The remaining parallelepiped L puts an additional emphasis on the angles between the edges, which are already illustrated in K. The illustration of the Spinning Pencil is different from Fig. 5.4: Our aim is an intuitive, compact interpretation of the figures. Hence, we draw the joint axis of each joint. We wish that a large pencil corresponds to a large value of the Spinning Pencil. Therefore, we set the length of the pencil to $\frac{\beta}{e_j}$. A disadvantageous large distance e_j results, then, in a short pencil. We use $\beta = 50000$. It allows a sufficiently large visualization of the Spinning Pencil as the Fig. 7.39-7.43 show.

The change of mV and sP in the case of the failure of one, resp., three joints of the test robot becomes clear in Fig. 7.39-7.41. The robot is loosing its capabilities, especially at the prioritized Location Areas at the bottom. Hence, its capability to perform the desired manipulation task

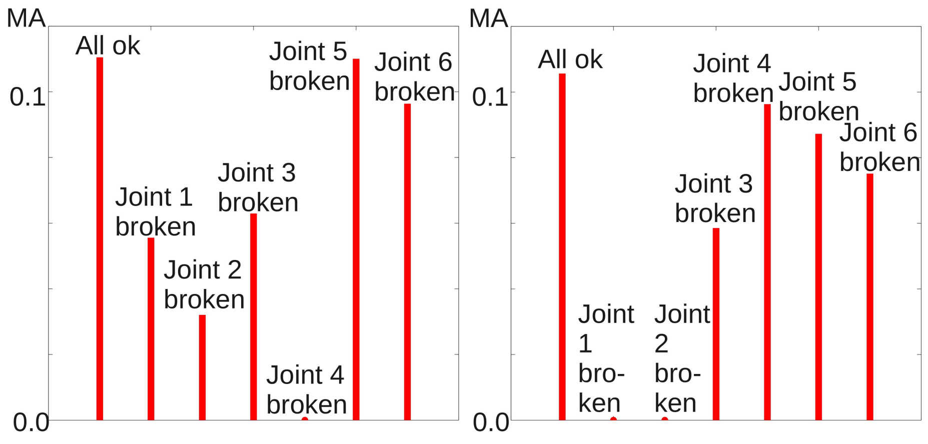


Figure 7.36: Complete manipulator vs. manipulator with one broken joint for the test robot (*left*) and the PUMA (*right*), data set II [4]. Similarly to Fig. 7.35, the essential joints for the task become visible (Maneuverability Analysis MA).

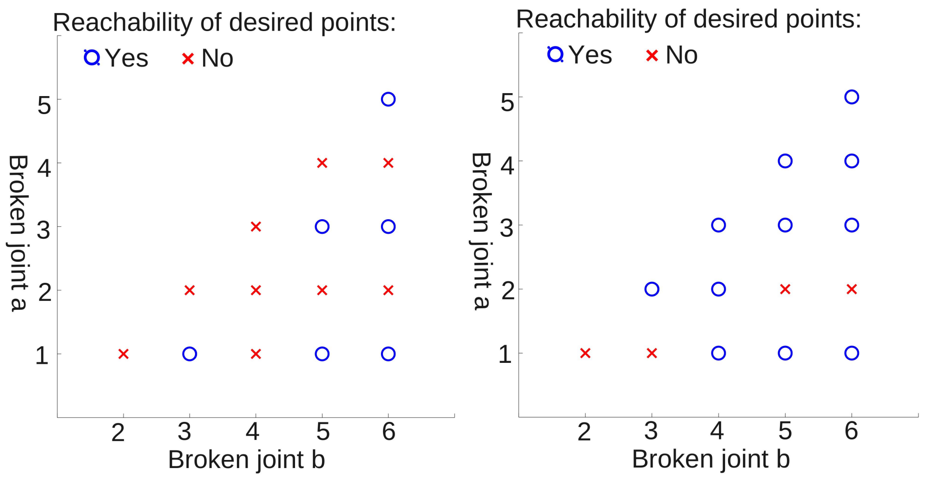


Figure 7.37: Capability of the test robot (*left*) and the PUMA (*right*) to reach the desired positions with two broken joints [4]. As it can be seen, joint 2 and 4 is very essential for the test robot to reach the positions (even under the extended tolerance).

is significantly affected.

Furthermore, Fig. 7.41 shows the additional tolerance, which the robot needs to reach the positions at all: The blue circles in Fig. 7.39 and 7.40 symbolize the original tolerance. For the purpose of a better illustration, these circles are larger than the original tolerance. The circles in Fig. 7.41 are proportionally enlarged according to the additional tolerance, which the robot needs to be able to reach the desired position. Consequently, not only the robot's capabilities are less advantageous, but, also, the robot get problems to reach the desired positions at all.

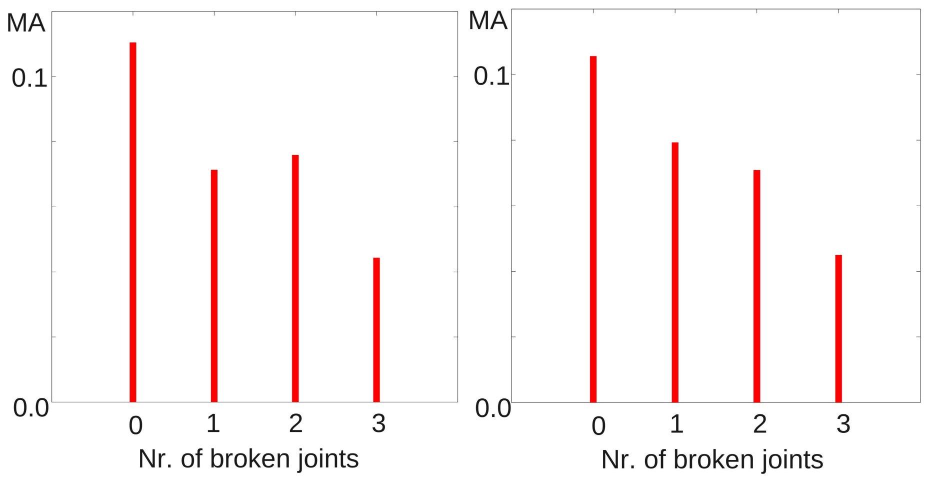


Figure 7.38: Average result of the Maneuverability Analysis (MA) with respect to the number of broken joints for the test robot (*left*) and the PUMA (*right*) [4]. Most of the times, a higher number of broken joints leads to a reduced maneuverability as expected. The higher value of the Maneuverability Analysis for the test robot with two broken joints can be explained through the higher tolerance for the hit of a desired position in the experiments with two or three broken joints.

However, this is not surprising, if one considers, that a failure of three joints reduces the number of DoF by the half in this case. The results clearly show the reduced capabilities and the required additional tolerance.

Fig. 7.42 and 7.43 illustrate the results of the PUMA similarly to the previous figures. Since the detail of both figures is the same as for the figures of the test robot, the larger magnitudes of the simplified mV and sP for the PUMA become visible. Nevertheless, the final results of both entirely working robots in Fig. 7.36 are nearly the same for both robots. This due to several reasons. First, multiple parts of the PUMA's Spinning Pencils point (nearly) into the same direction. Second, we put an emphasis on mV, which is not as large for the PUMA as its Spinning Pencils. Third, the large Maneuverability Volumes at the top are located at a less important Location Areas. Therefore, they influence the final result less significantly.

We repeat the experiments with none, one and two broken joints a second time to check the repeatability of the results. The Stochastic Optimizer in the estimation of the inverse kinematics can lead to slightly different configurations at the desired positions. The repetition of the experiments show, that these differences affect the magnitude of the results of the Maneuverability Analysis just slightly with an average difference of 0.005, including five outliers with a difference higher than 0.02

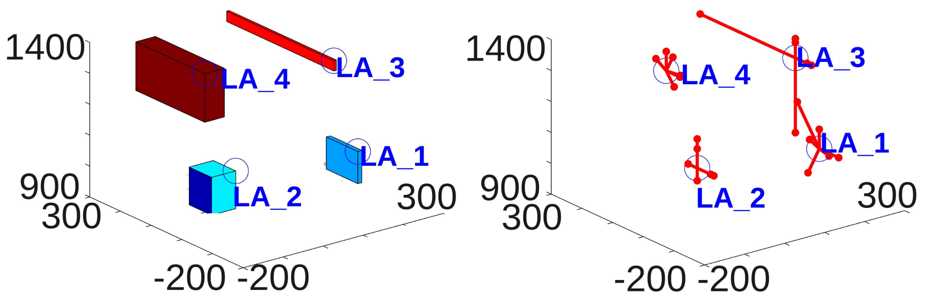


Figure 7.39: Simplified Maneuverability Volume mV (*left*) and Spinning Pencil sP (*right*) for the test robot when all joints are working [4]. The Location Area at the top right has low priority, therefore, the disadvantageous mV and sP at **LA_3** does hardly carry weight (unit in left and right subfigure: mm). The corresponding mV has just one long side and the sP is just spanned up in a plane.

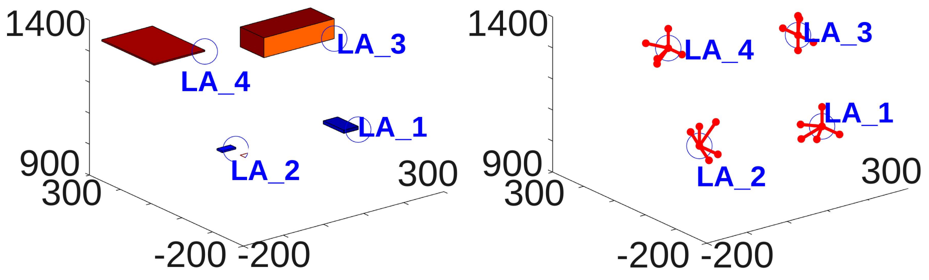


Figure 7.40: Simplified mV (*left*) and sP (*right*) for the test robot with broken joint 1 [4]. At the important Location Areas at the bottom, the robot can just reach strongly disadvantageous mV (unit in left and right subfigure: mm). Even though the sP is not significantly affected by the joint failure, the bad mV at the bottom reduces the possible maneuverability significantly as the statistics in Fig. 7.36 show.

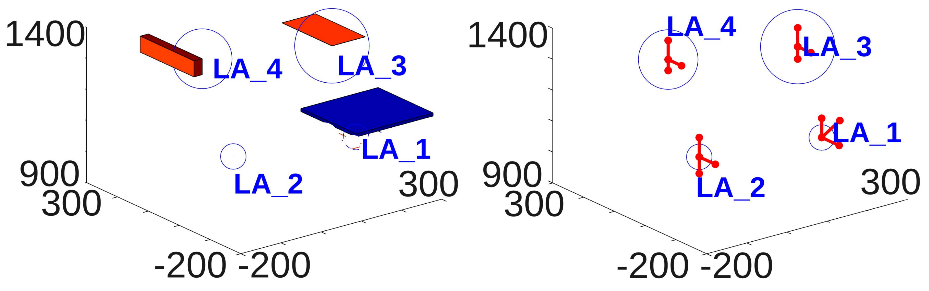


Figure 7.41: Simplified mV (*left*) and sP (*right*) for the test robot with broken joint 4, 5 and 6 [4]. As the larger blue circles at the top Location Areas already indicate, the robot has problems to reach the desired positions. At the same time, the mV are reduced, especially at the bottom left, and the sP are getting smaller (unit in left and right subfigure: mm).

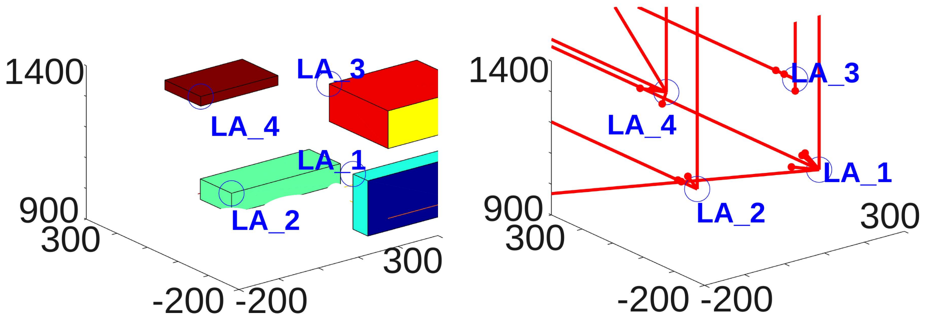


Figure 7.42: Simplified mV (*left*) and sP (*right*) for the PUMA when all joints are working [4]. The detail of Fig. 7.39 and this figure is the same to enable an easier comparison, even though the mV and sP do not fit in this figure any more (unit in left and right subfigure: mm).

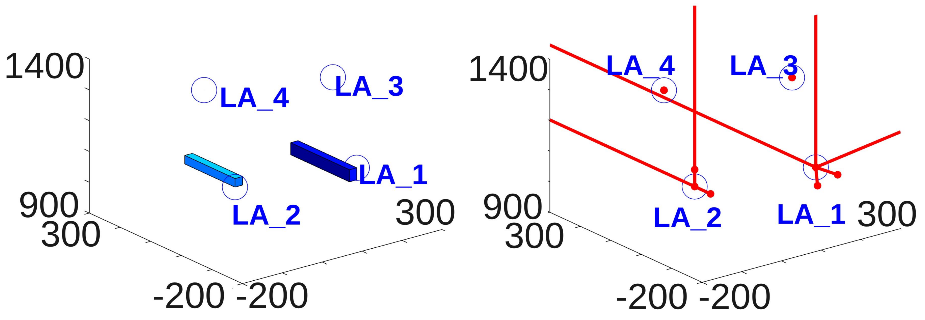


Figure 7.43: Simplified mV (*left*) and sP (*right*) for the PUMA with broken joint 3 [4]. The robot is not able to reach the less important Location Areas at the top any more, but it can position its end-effector at the prioritized areas at the bottom. The sP is still relatively advantageous there (unit in left and right subfigure: mm).

7.5.3 Results: Maneuverability Analysis under Path Optimization

In this part of our experiments, we perform a maneuverability analysis under the proposed path optimization approach for no, one or two broken joints. We start with the analysis of the failure of none and one joint. The points are considered to be reached by a certain configuration, if the distance of the real end-effector position to its desired position is smaller than a tolerance t_h. In the case of no failure, the points are considered to be reached within a tolerance $t_h = 75.0$ (defined similarly in Section 7.3). In comparison to the smaller tolerance in the experiments in Section 7.5.2 and 7.4.2, one needs to consider, that (1) all points along the path need to be reached and (2) the appropriate position as well as the upright orientation have to be achieved. The tolerance is extended to 150 (resp., 250) for the failure of one joint to allow to reach all Location Areas (resp., original points along the path).
In contrast to the experiments in Section 7.5.2, we consider also the desired orientation along a path (upright object orientation for transportation). It is important to us to ensure, that possible objects in the end-effector can be kept upright during the manipulation. Hence, we choose a relatively large penalization $\gamma_{EPP} = 10$ (Eq. 5.18) to avoid undesired changes of the orientation.

Fig. 7.44 illustrates the mean and the maximal EPP values. Hence, we show the average and the worst performance. Here, it is important to consider, that a result is better, the smaller the EPP value is. As it can be seen, several EPP-values are very high in comparison to others. This is due to the paths of pushed objects: They require an object movement on the plane, which is an additional constraint in comparison to a path of a lifted object. The more limited the set of possible path points is, the more difficult it is to reach desired points, especially in the case of a broken joint. The EPP-values of the paths of lifted objects (hence, without the pushed objects) are depicted in Fig. 7.45 for data set II. They are significantly smaller.

Most of the essential joints are similar to the Maneuverability Analysis without path optimization: E.g., joint 2 is important for the test robot. However, in the analysis with the path optimization, the test robot does not suffer from the loss of joint 4. One has to consider the higher tolerance for a hit in comparison to the experiments without path optimization (250 along the path vs. 25 at Location Areas before). Hence, the extended tolerance seems to allow a better performance in the case of a failure of joint 4. The joints mainly essential for the pushed objects become visible in the comparison of Figure 7.44 and 7.45. Joint 5 in the test robot is such an example.

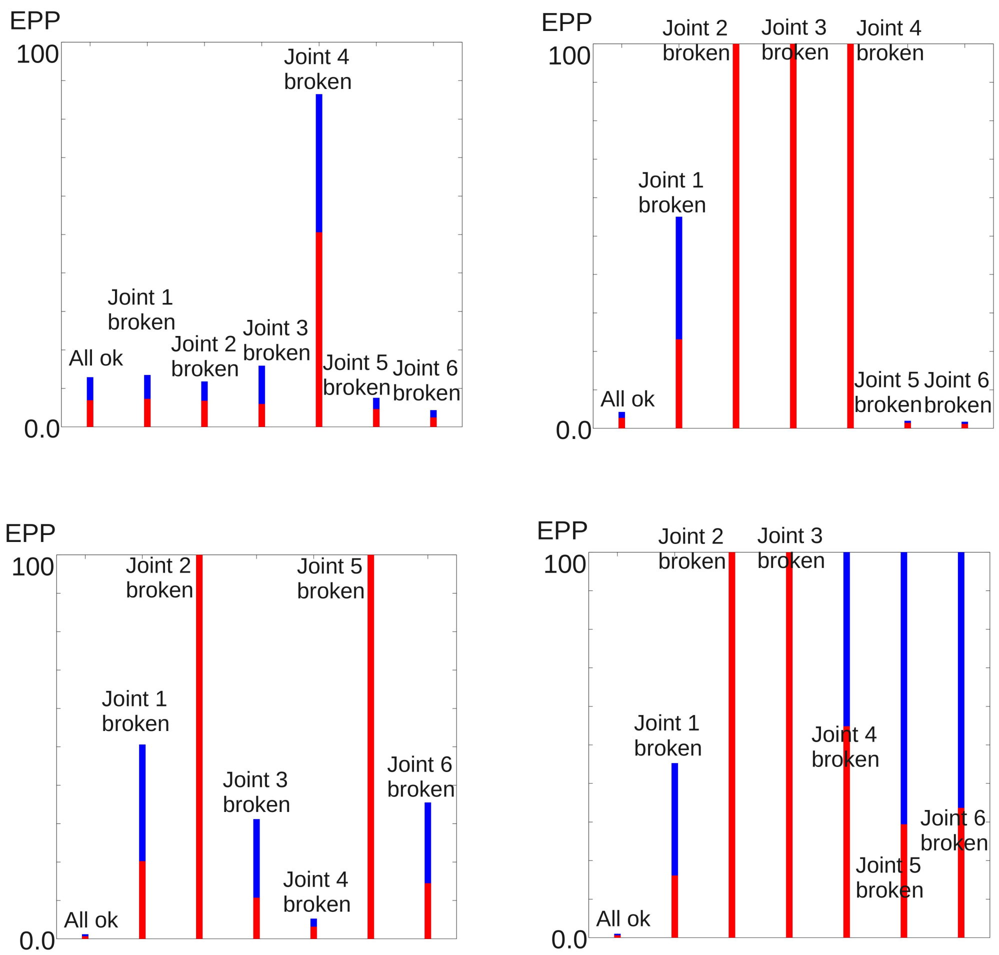


Figure 7.44: Complete manipulator vs. manipulator with one broken joint for the test robot (*left*) and the PUMA (*right*), data set I (*top*) and data set II (*bottom*). The mean of the EPP-values of all paths is shown in red, the maximal EPP-value of all paths is depicted in blue (resp., per robot and data set). Fig. 7.45 shows the EPP-values without paths of pushed objects, which cause some high EPP-values in this Figure here (EPP: Elastic Power Path).

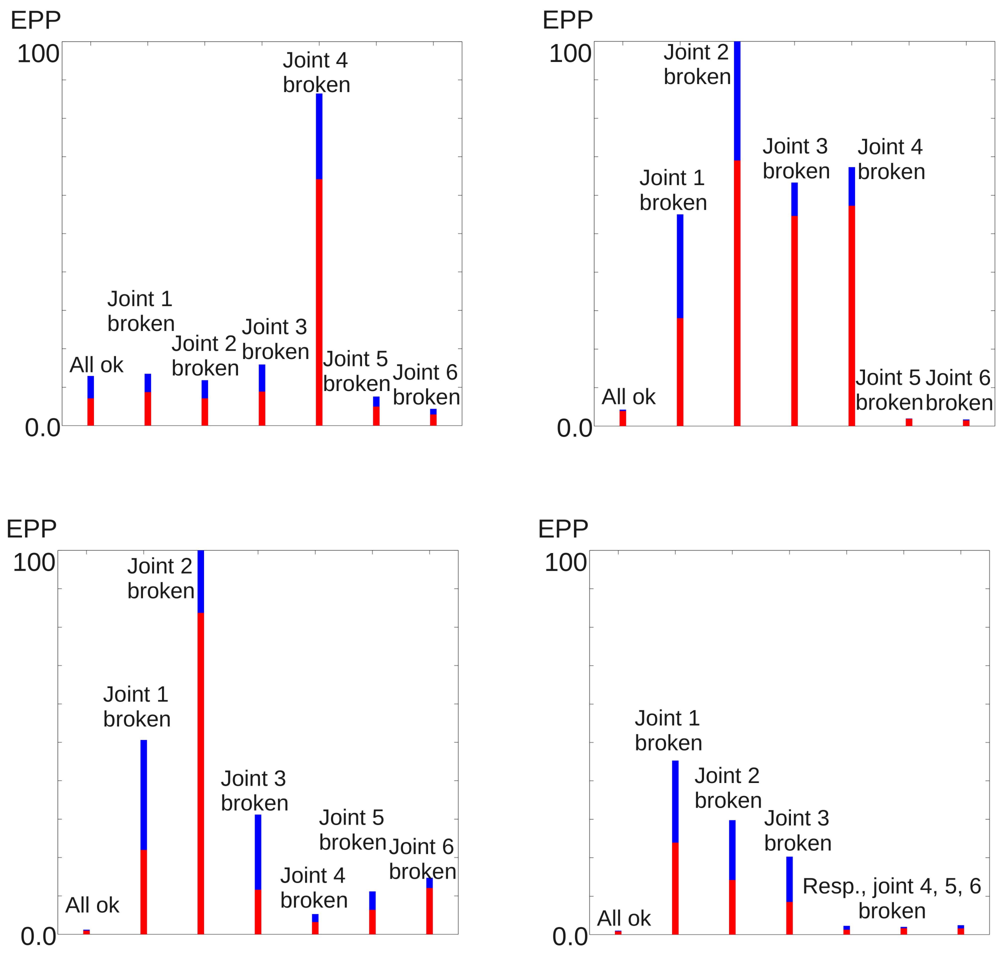


Figure 7.45: Complete manipulator vs. manipulator with one broken joint for the test robot (*left*) and the PUMA (*right*), data set I (*top*) and data set II (*bottom*), without paths of pushed objects. Most of the results are similar to the maneuverability analysis without path optimization: E.g., in data set II, joint 2 is important for the test robot. However, it robot does not suffer from the loss of joint 4 in data set II. One has to consider the higher tolerance for a hit in comparison to the experiments without path optimization (EPP: Elastic Power Path).

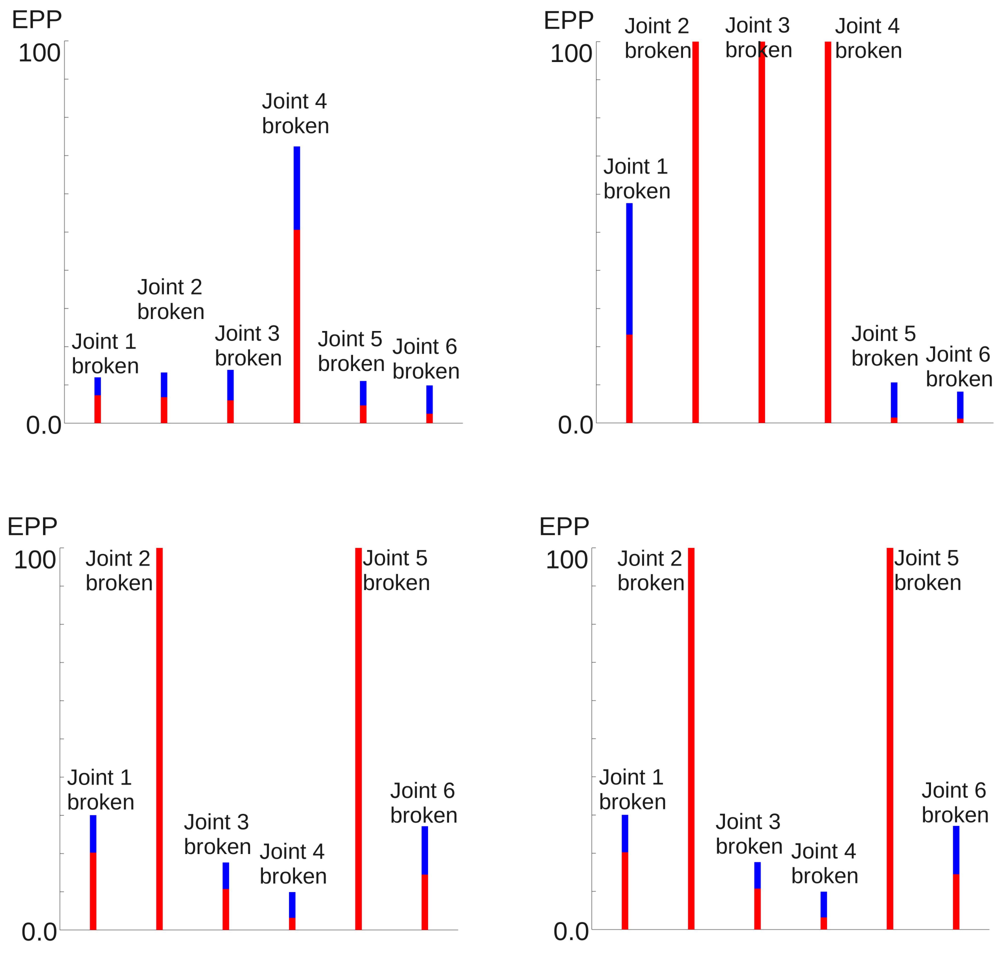


Figure 7.46: Comparison of the EPP-values before and after the optimization for the test robot (*left*) and the PUMA (*right*), data set I (*top*) and data set II (*bottom*). The improvement due to path optimization (start: blue; final: red) becomes visible for most of the cases. Some EPP-values are not within the scaling any more (EPP: Elastic Power Path). This is, again, due to the paths of pushed objects. Hence, Fig. 7.47 shows the improvement of the result without the paths of pushed objects. However, the improvement of the EPP-values of all paths (including paths of pushed objects) is successful (also for the non-visible parts).

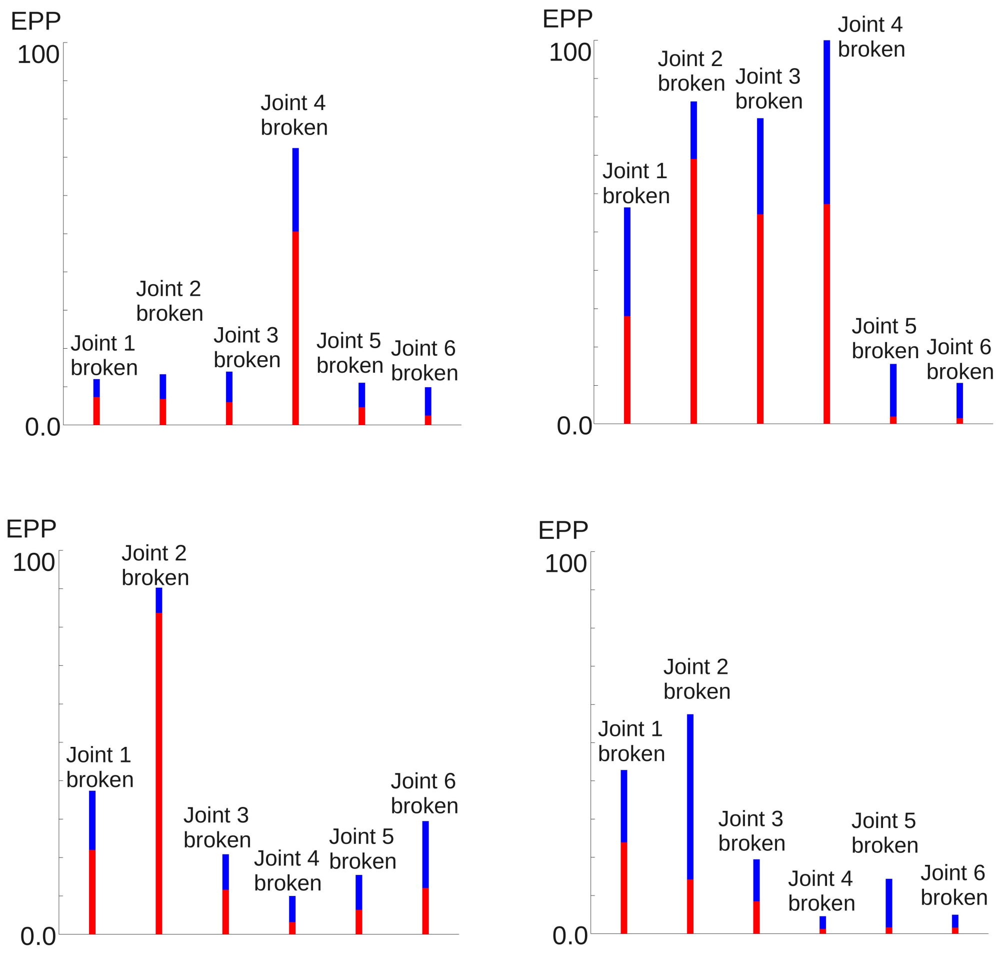


Figure 7.47: Comparison of the mean EPP-values before and after the optimization for the test robot (*left*) and the PUMA (*right*), data set I (*top*) and data set II (*bottom*), without paths of pushed objects. Similarly to Fig. 7.46, but without paths of pushed objects (start: blue; final: red). The effect of the path optimization itself is clearly visible: The EPP-values have been reduced in all cases of single joint failures (EPP: Elastic Power Path).

Fig. 7.47 further illustrates the improvement after the path optimization in the case of one joint failure. All EPP-values have been improved. Fig. 7.47 shows the results of the paths of lifted objects (without paths of pushed objects). The evaluation of the EPP-values of all paths (including paths of pushed objects) shows a similar reduction of the EPP-values (Fig. 7.46). Fig. 7.47 shows a significant improvement after the optimization for a failure joint 1, resp., 3 of the PUMA. Hence, it becomes visible, that the path optimization is able to partially compensate lost capabilities (see Fig. 7.35).

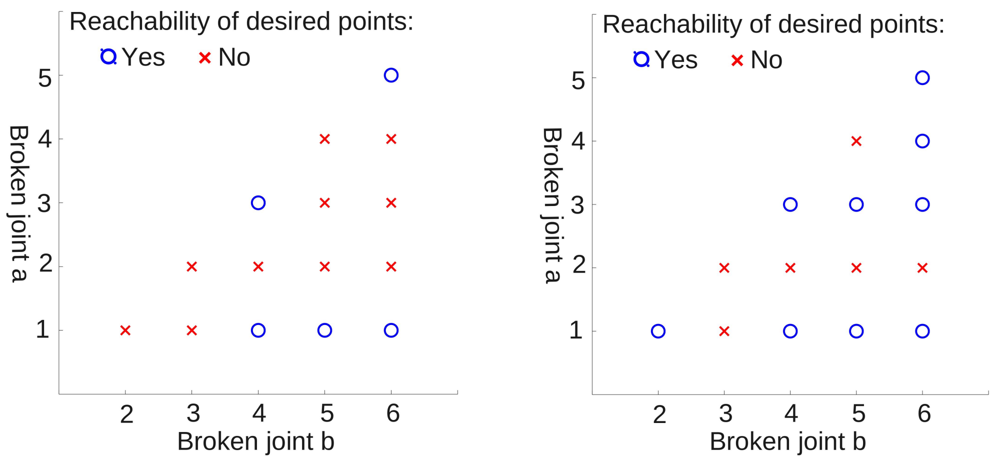


Figure 7.48: Capability of the test robot to reach the desired positions with two broken joints before (*left*) and after (*right*) the optimization, data set II. As it can be seen, the path optimization allows to reach the desired positions which were not reachable before the optimization in four cases.

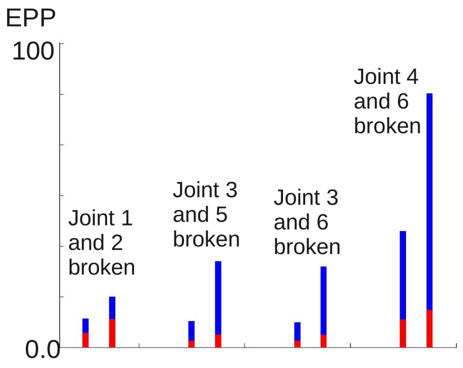


Figure 7.49: Comparison of the EPP-values before and after the optimization, test robot, data set II. The EPP-values (start: blue; final: red) of the four cases, in which desired positions are just reachable after the optimization (see Fig. 7.48). The mean and maximal EPP-value are depicted (resp., left and right for each case). The improvement due to the optimization is clearly visible (EPP: Elastic Power Path).

Exemplary results of the experiments with two joint failures are shown in Fig. 7.48 and 7.49. The tolerance for a hit is set to 250 (Location Area), resp., 350 (along the path) to allow a larger set of paths which can be reached at all. Fig. 7.48 shows the combinations for which the robot is capable to perform the desired manipulations. A manipulation is defined to be performable, if the EPP is below 100. Most of the paths of lifted objects achieved an EPP-value below 100 in case of no and just one joint failure (see, e.g., Fig. 7.45). Hence, 100 seems to be a reasonable border. After the path optimization (Fig. 7.48, right), the robot is

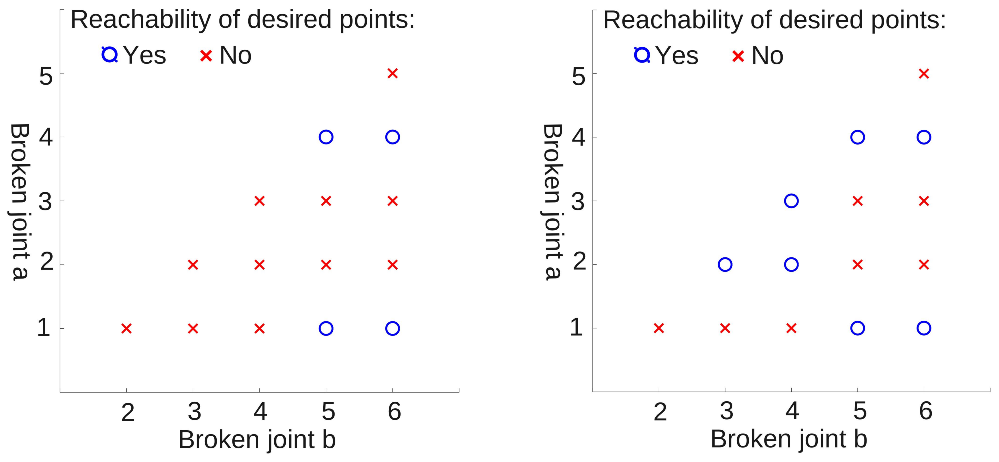


Figure 7.50: Capability of the PUMA to reach the desired positions with two broken joints before (*left*) and after (*right*) the optimization, data set II. Similarly to the test robot, the path optimization allows the PUMA to reach desired positions which were not reachable before the optimization (three cases).

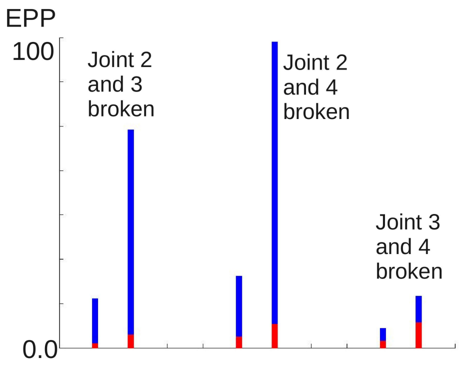


Figure 7.51: Comparison of the EPP-values before and after the optimization, PUMA, data set II (similarly to Fig. 7.49). The EPP-values (start: blue; final: red; mean: left; maximum: right) of the three cases, in which desired positions are just reachable after the optimization (see Fig. 7.50). The EPP-values are significantly improved in all three cases (EPP: Elastic Power Path).

able to reach paths which were not reachable before (Fig. 7.48, left). The improvement of these paths is shown in Fig. 7.49. The results of data set I look similarly. The optimization for the PUMA does not achieve such an improvement for two joint failures in data set I. However, the improvements for data set II are significantly as Fig. 7.49 and 7.50 show (three cases).

7.6 Analysis of Master-Slave Systems

In this part of the experiments, we indicate a further application area for the proposed analysis: Master-slave systems. We perform the experiments on three different scenarios of medical applications (MIS). The first scenario is a simple cut, the second one is knot-tying and the third one is a suturing task. They are further described in Section 3.2 and Section 7.2.
The cut does not provide any alternative paths, it is fixed. The path of the knot-tying scenario is less constrained and can be shifted. A second manipulator is necessary to support the main manipulator to ty the knot. In the suturing scenario, the path is partly fixed: The path of the stitch phase in the object is fixed. Just the approach phase is less constrained. Hence, alternative paths are possible there.

The simulated console is built up similarly to a traditional base with a parallel kinematics structure of the “delta” family [196]. The simulated links of the console have a length of, resp., 200 mm. A handle with 3 DoF is attached to the traditional base. Its three joint axes are perpendicular to each other and they intersect at the human’s wrist. It allows to change the orientation during teleoperations. The console is inspired by the sigma.7 haptic interface for MiroSurge [197]. The simulated path of the MiroSurge system from DLR with a gripper [104], [191], [192] is the same as in Section 7.2. In principle, any console could be simulated in the experiments. We have chosen a traditional base with a parallel kinematics structure (a console of the delta family). In order to allow a change of the orientation, we attach the handle to the base.
The inverse kinematics of the console is estimated similarly to the method in Chapter 2. The desired position of the end-effector is estimated for each serial chain of the parallel kinematics structure separately. We use the attached handle to move the end-effector in the desired orientation. Hence, we estimate the inverse kinematics for the orientation just with the handle.

The human’s behavior is assumed to be convenient with respect to smooth and slow motions (see also Chapter 6). Furthermore, “inconvenient” elbow positions close to or above the shoulder are considered as undesired. We compare the convenience of alternative paths if available. Furthermore, the convenience is also examined for different users who have different, arbitrarily chosen arm lengths (resp., 250 mm, 300 mm, 350 mm from the hand to the elbow, resp., from the elbow to the shoulder). Additionally, different positionings of the user with respect to height are tested. The basic height is 250 mm above the center of the console. The changed positionings are, resp., + 50 mmm, + 100 mmm above the basic height. The, resp., arm lengths are 300 mm (intermediate length before). We could also interpret the height of the positioning as follows. The height of one positioning (300 mm) has the same magnitude as the arm length (intermediate length before). The height of the other two positionings is ± 50 mm.
To sum up, five different experiments are performed in total (three different arm length; one of the lengths with two further, different positionings).

The five experiments are performed on each of the three scenarios. For each experiment in one scenario, the paths of five different start configurations are evaluated. If available, alternatives are also evaluated.

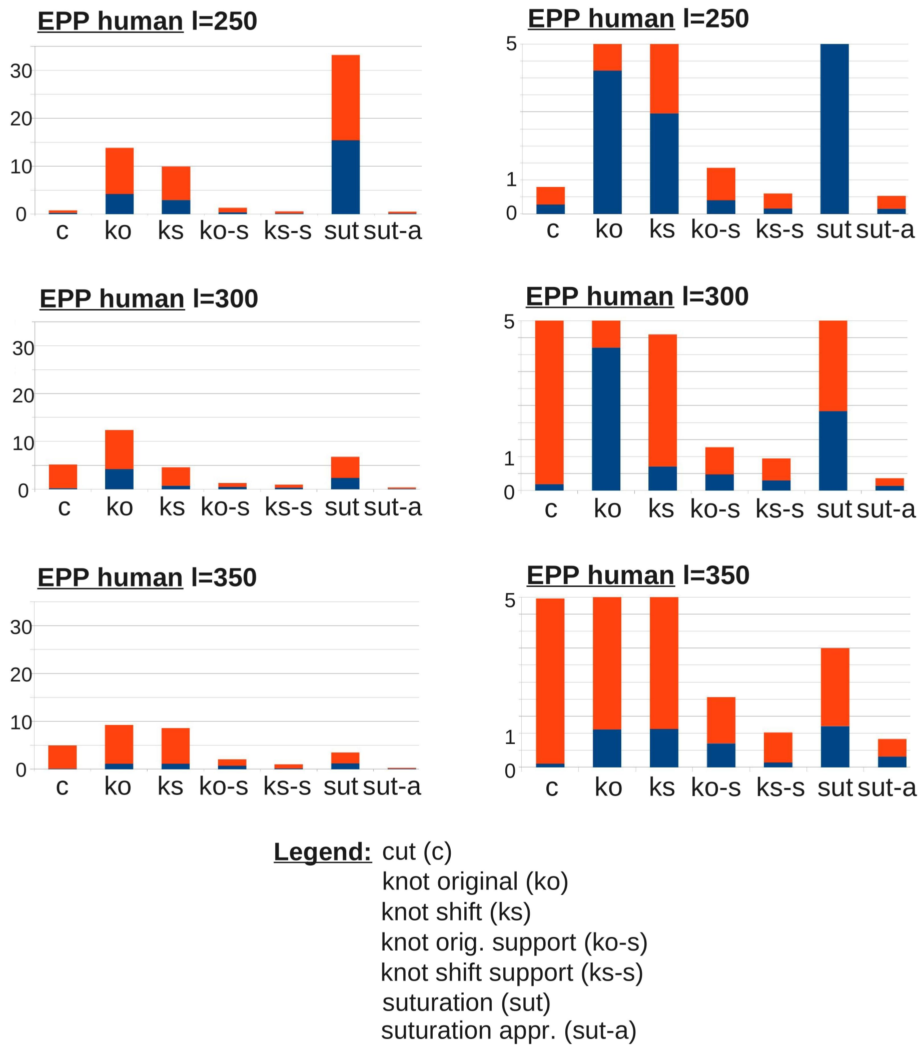


Figure 7.52: EPP human, different arm lengths. The results differ clearly between the users with different arm lengths. Here, the best results are achieved for the user with the longest arm lengths (EPP: Elastic Power Path).

The overall results are shown in Fig. 7.52 and 7.53. The minimal and the average result of the EPP-evaluation is respectively drawn (blue, resp., orange). The minimal EPP shows the best possible performance in each experiment. The average EPP gives an indication about the convenience for a set of different, alternative configuration sequences.
The comparison of the EPP values for the three different arm lengths (Fig. 7.52) shows, that

the longest one (l=350 mm) achieves the smallest minimal and average EPP values for most of the paths. Most of the times, the results of the arm lengths of l=300 mm are still better than the arm lengths of l=250 mm. To conclude, the usage is more convenient for a user with longer arms in our exemplary experiments.
The positioning of the user can significantly influence the level of convenience for the system usage as the experiments show. As described, the position of the shoulders with respect to height has been changed (+50 mm, +100 mm) for the arm lengths of l=300 mm. An increase of 50 mm improves the convenience for most of the paths, whereas an increase of 100 mm improves some EPP values while others worsen. The increase of the position by +50 mm achieves even better results than the original positioning of the user with the longest arm lengths of l=350 mm (arm lengths in the experiment with the changed positioning: l=300 mm).
If one considers just the results of the experiment with the different arm lengths (longer arms, higher convenience), the question arises, whether the convenience is, in general, higher for users with longer arms. This could be due to leverage. However, the results of the different user positioning show, that the positioning has also a significant influence on the convenience. The question is, whether an appropriate positioning is more important than the arm lengths of the user. In order to be able to answer this question, a large user study and a sophisticated analysis tool would be required (Section 6.3). This is, however, out of the focus of this thesis. We just want to indicate the potential of the Elastic Power Path concept as a framework for the efficiency analysis of a human's motion during the work on a master-slave system.

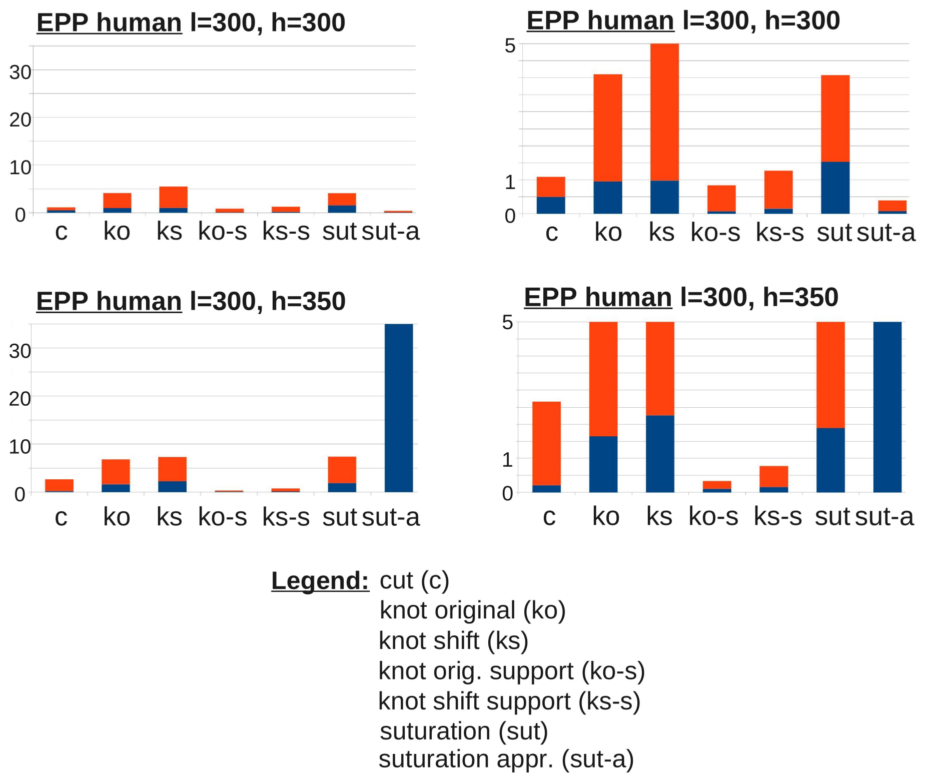


Figure 7.53: EPP human, different heights. The figure shows the results for different height positionings of the user (arm length of, resp., 300 mm; original height h = 250 in Fig. 7.52). The large EPP-value in the suturing approach in the last experiment is due to an inconvenient, high elbow position (EPP: Elastic Power Path).

We have seen in the results of the knot tying scenario (see Fig. 7.52), that the efficiency along the shifted path is slightly worse than along the original path for the first experiment (original positioning, arm lengths of, resp., 250 mm). The underlying reasons can partially be visualized by the trace of the elbow movement. If just the change of the elbow height is drawn, the differences between both paths become clear: The elbow height is changed much more often along the shifted path (see Fig. 7.54). Fig. 7.55 and Fig. 7.56 illustrate the corresponding configuration sequence for the knot tying scenario of the original, resp., shifted path.
Similarly, Fig. 7.57 and Fig. 7.58 show the configuration sequence for the suturing scenario with arm lengths of 250 mm, resp., 300 mm. The arm lengths with, resp., 300 mm are more efficient, since they do not require significant vertical movements like in the experiments with arm lengths of, resp., 250 mm. The advantages of the arm lengths of 300 mm becomes also visible in the corresponding speed profiles in Fig. 7.59 and Fig. 7.60. Most of the ("human joint") speeds are significantly lower for the arm lengths of 300 mm. There are still peaks existing (e.g., joint 1 or 7), but many peaks are eliminated and the curve is more smooth than before.

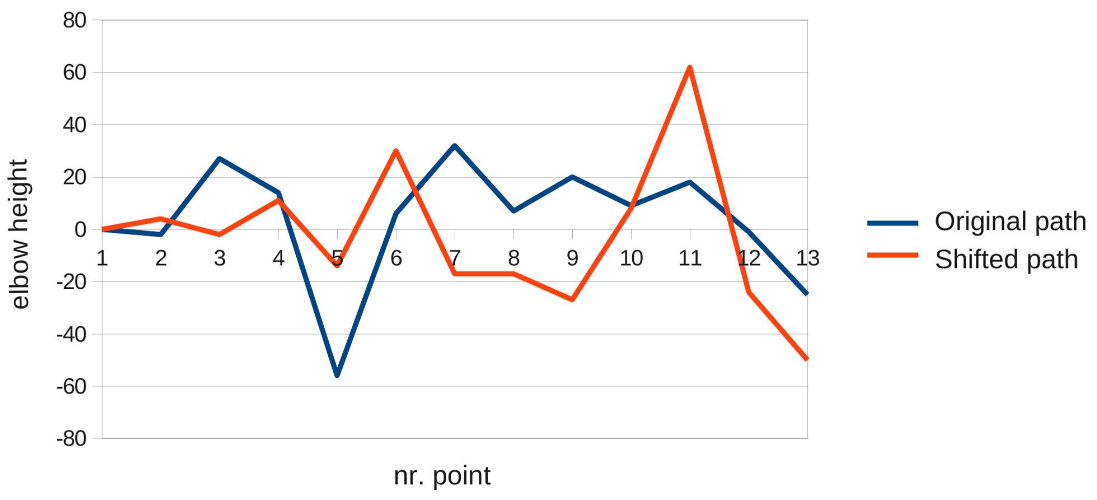


Figure 7.54: Change of the elbow height in the knot tying scenario. The change of the elbow height with respect to its previous position is depicted (unit: mm). The elbow height is clearly changed much more often along the shifted path (see, e.g., magnitude of the values and number of zero crossings).

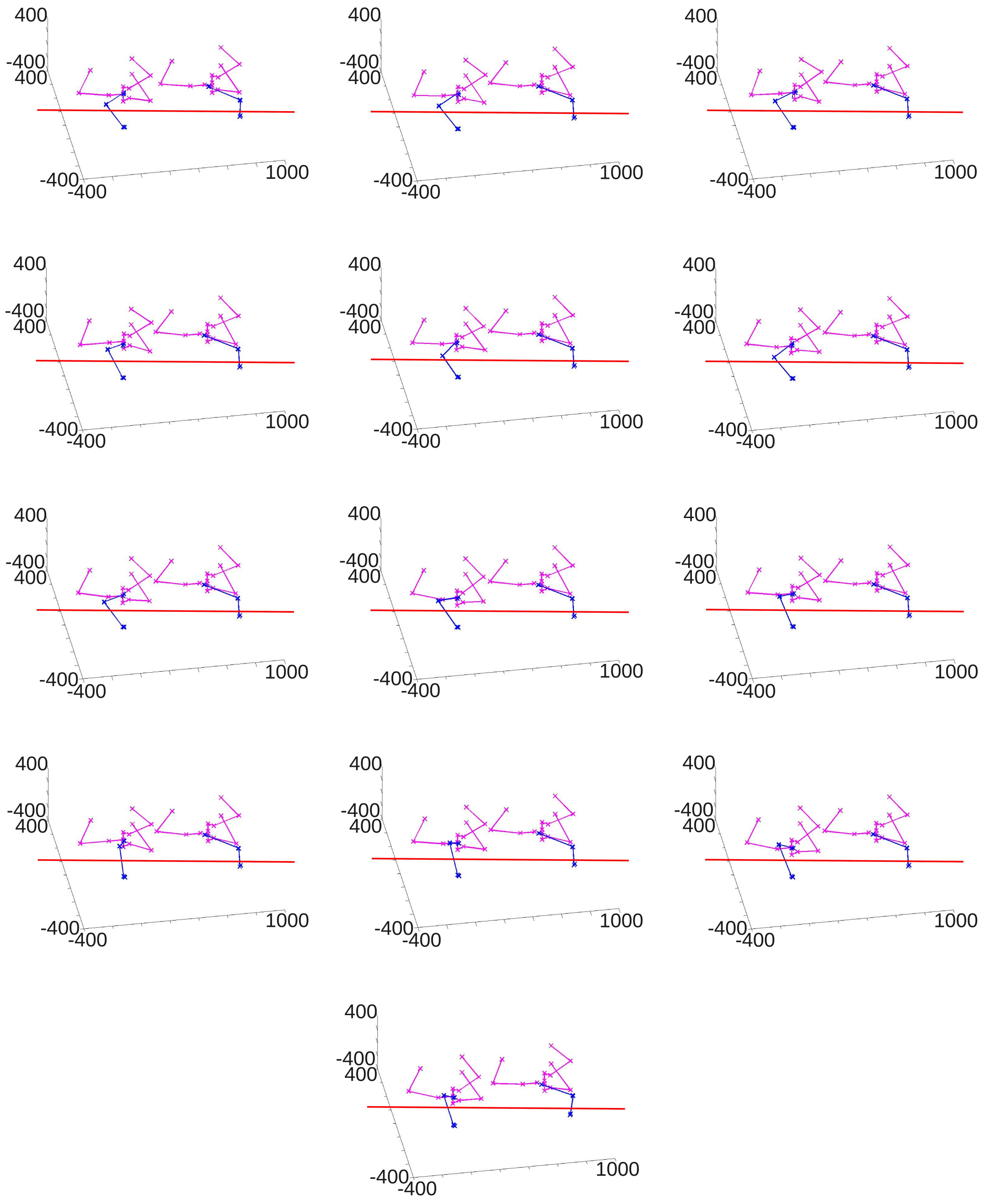


Figure 7.55: Console and arm configurations for the knot tying scenario. Both arms are drawn in blue, the respective consoles in magenta ("delta" family [196], buildup inspired by the sigma.7 haptic interface for MiroSurge as described at the beginning of the Section). The Figure visualizes the configuration sequence along the original path, for which the change of the elbow height depicted in Fig. 7.54. The red line is used as a reference height to achieve a better illustration of the elbow movement. As it can be seen, the left elbow moves upwards in the first four steps (unit: mm). Then, it is moved down in the fifth step. Afterwards, it moves again upwards until step eleven, where starts to move downwards again. The right arm performs just the supporting steps in knot tying (see Section 7.2.2).

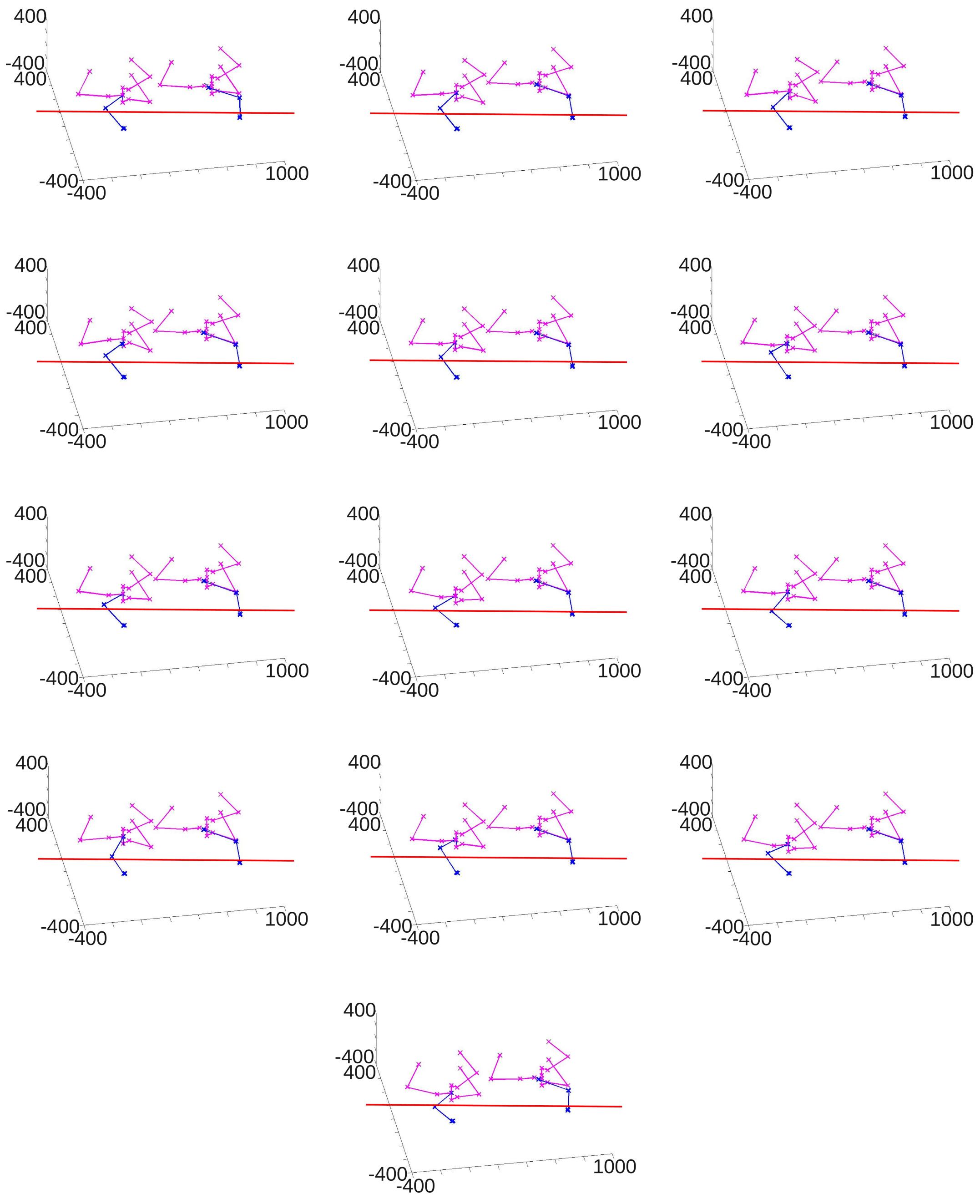


Figure 7.56: Console and arm configurations for the knot tying scenario, shifted path. Similarly to Fig. 7.55, the arms are drawn in blue, the respective consoles in magenta (unit: mm). The Figure visualizes the configuration sequence along the shifted path, for which the change of the elbow height depicted in Fig. 7.54. The slightly worse efficiency (see Fig. 7.52) is visible in the left elbow movement. It changes its motion upwards, resp., downwards in most of the steps.

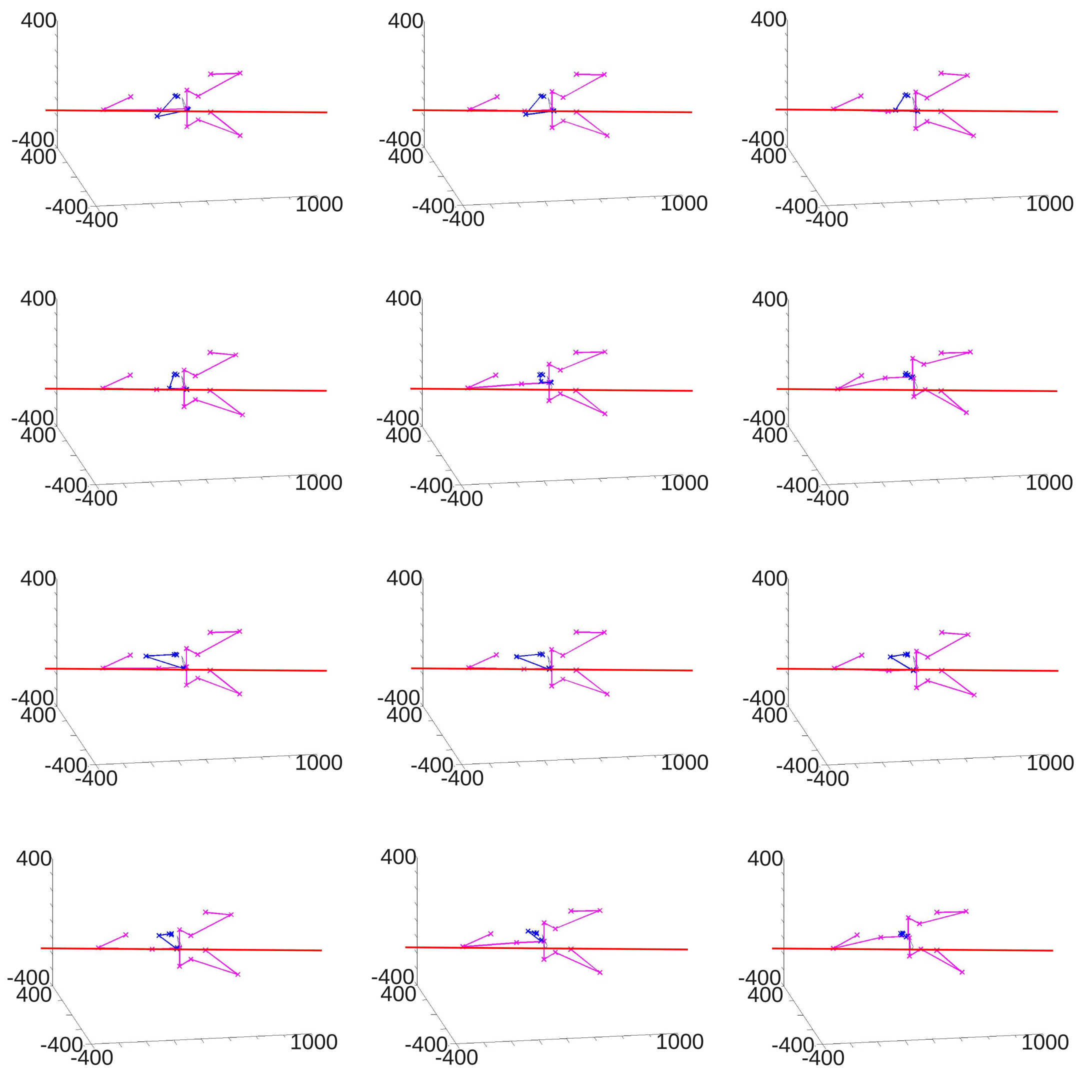


Figure 7.57: Console and arm configurations for the suturing scenario, arm length of respectively 250 mm. The configurations of the left arm are shown in blue, the configurations of the robot are drawn in magenta (unit: mm). As it can be seen, the elbow moves not only in the vertical direction, but also in the horizontal plane. Sometimes, the elbow is even overlaps with the entire arm. The configurations with arm lengths of respectively 300 mm are more advantageous, since they do not have an additional horizontal movement (see Fig. 7.58).

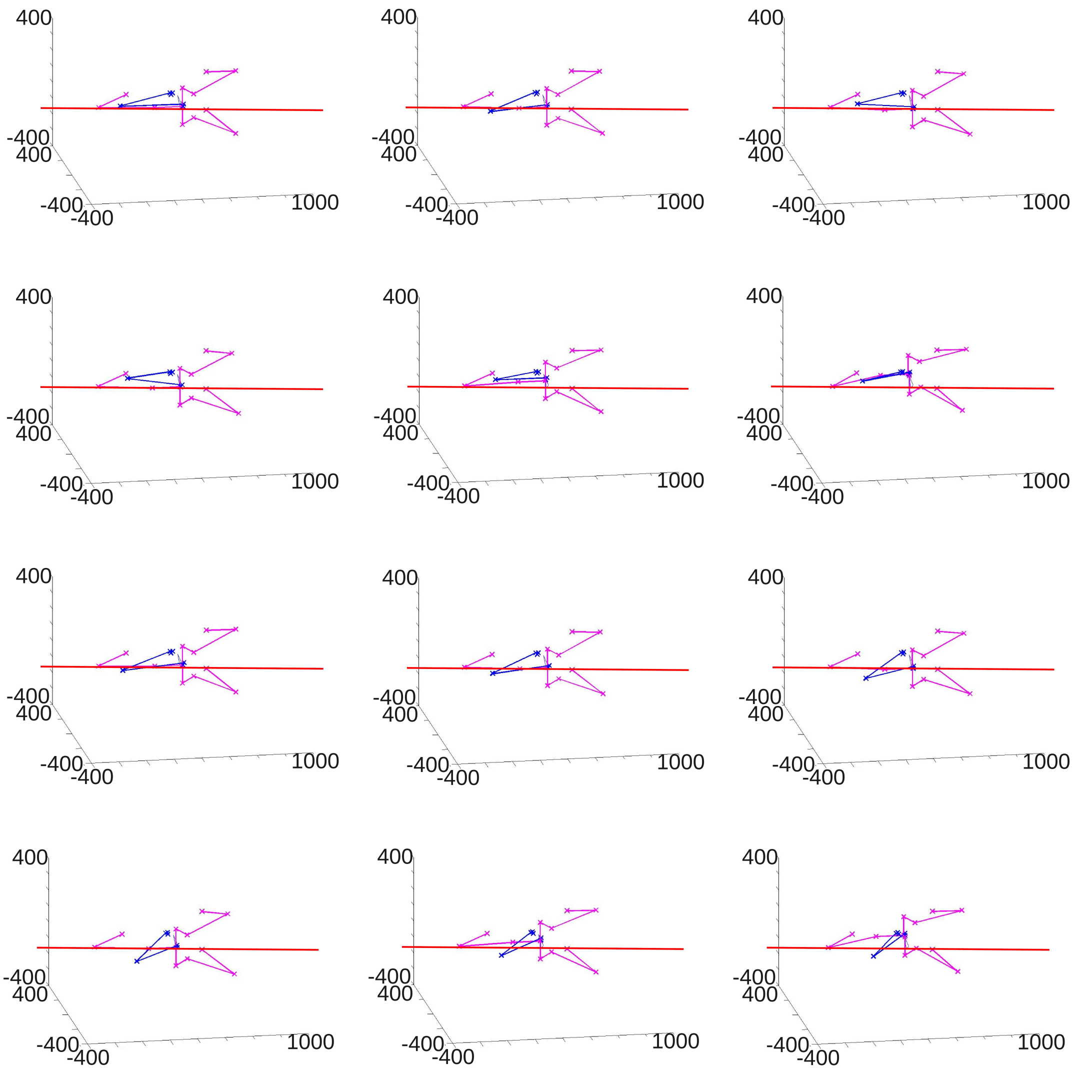


Figure 7.58: Console and arm configurations for the suturing scenario, arm length of respectively 300 mm. The configurations of the left arm are shown in blue, the configurations of the robot are drawn in magenta (unit: mm). The elbow is mainly changing its position with respect to height. This is more advantageous with respect to efficiency than the elbow movements in the suturing scenario with an arm length of respectively 250 mm. There, the movements have an additional horizontal movement (see Fig. 7.57). The more advantageous movements here are reflected in the evaluation in Fig. 7.52. The respective speed profiles in Fig. 7.59 and 7.60 confirm this.

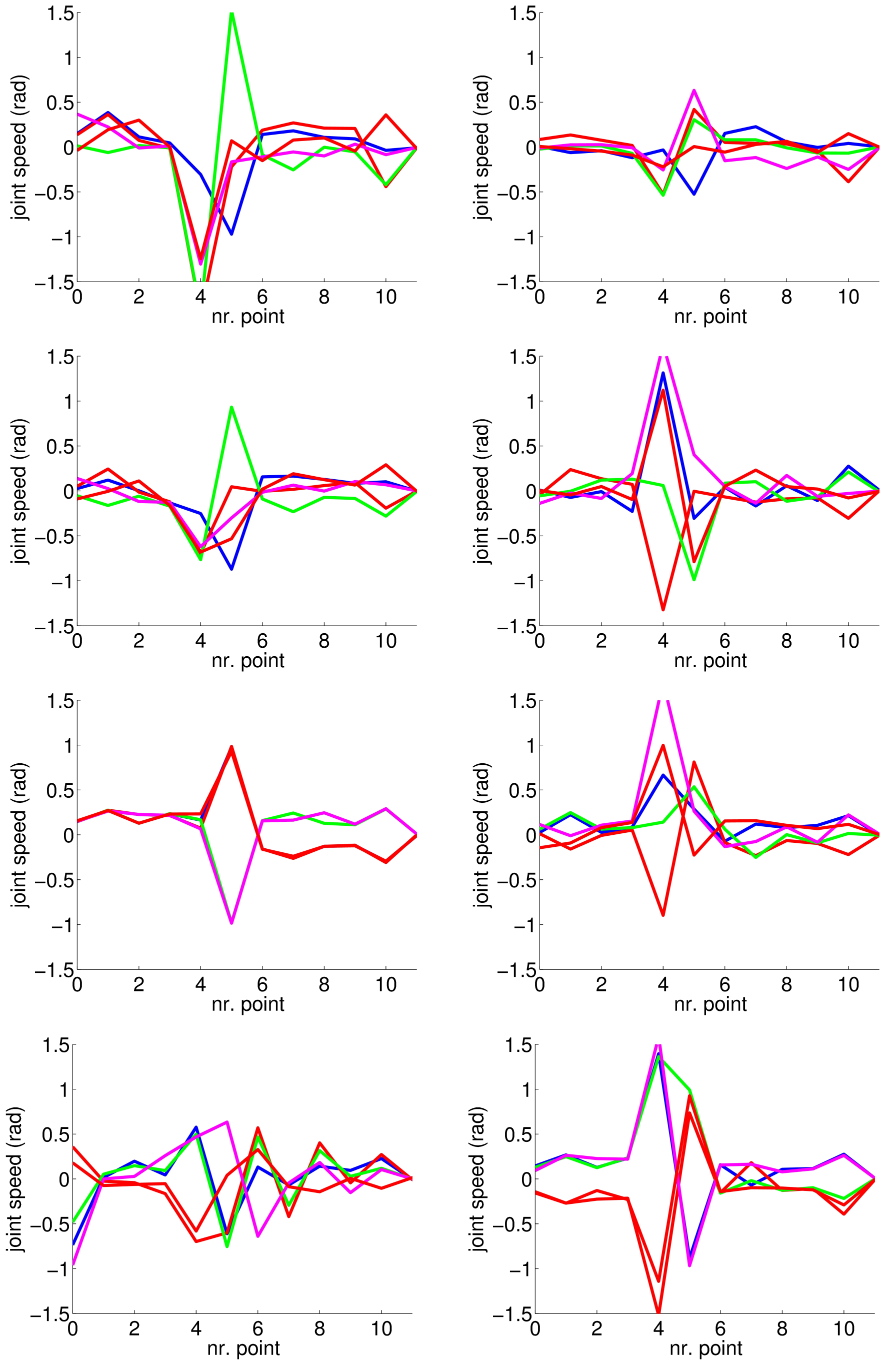


Figure 7.59: Speed profile of the human's motion, suturing scenario, arm length of respectively 250 mm. Each subfigure depicts the speeds of one joint. Each color within a subfigure refers to one of the five experiments.

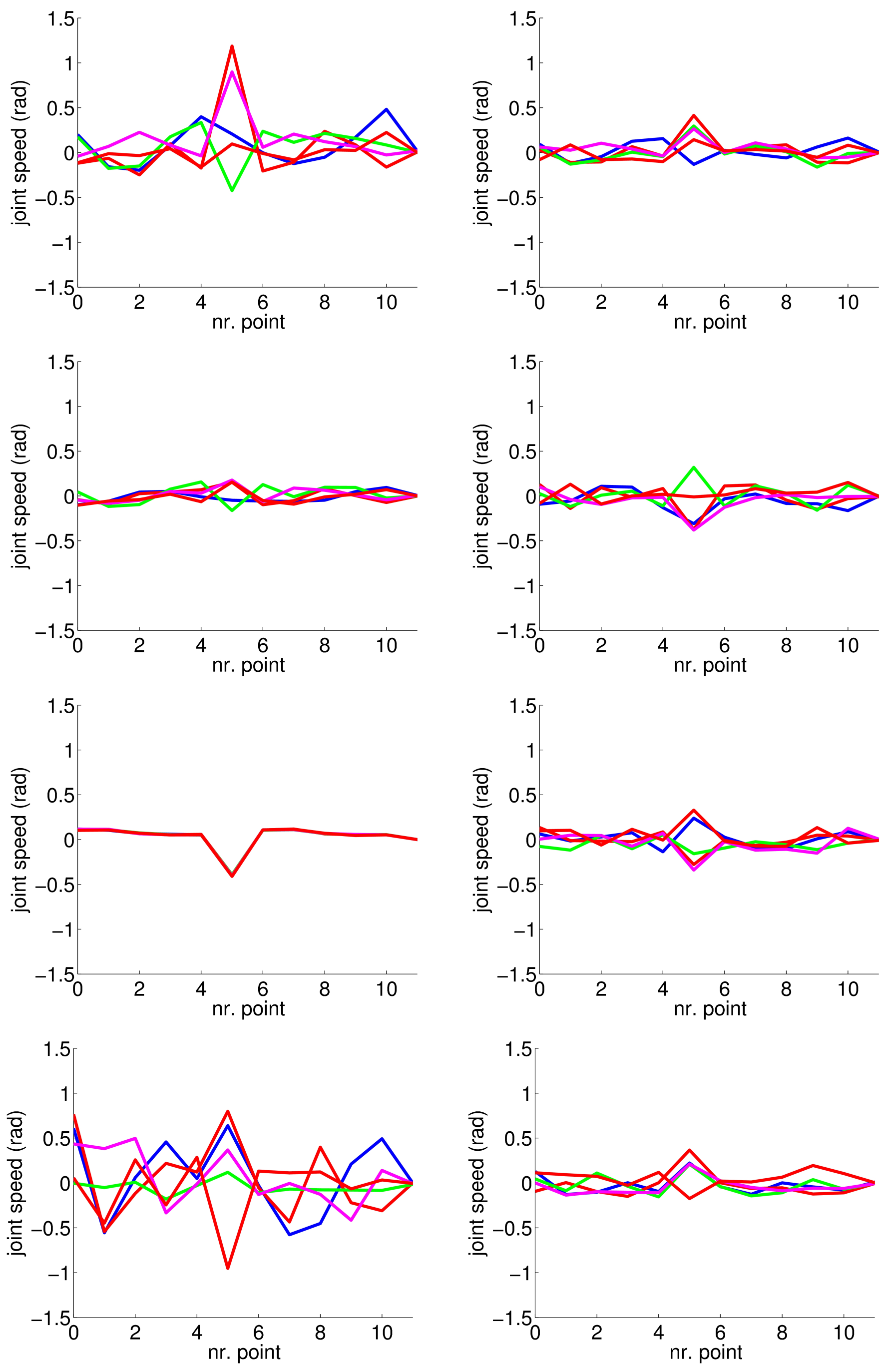


Figure 7.60: Speed profile of the human's motion, suturing scenario, arm length of respectively 300 mm. Similarly to Fig. 7.59, each subfigure depicts the speeds of one joint for the five experiments (resp., one color per experiment). In comparison to Fig. 7.59, most of the speeds are significantly lower. There are still peaks existing (e.g., joint 1 or 7), but many peaks are eliminated and the curve is more smooth than before.

We compared also the EPP-results of the *console* for the different scenarios and experiments. “Alternative” configurations of the console do hardly exist (e.g., differences about 10^{-5} rad, which is extremely small). This is not surprising due to the structure of the console. Moreover, it is quite consequential, that the results of the Objective Function of the EPP do also hardly change for these “alternative” configurations (e.g., differences about 10^{-4}). Furthermore, there are hardly any changes in the EPP values for alternative paths as, e.g., for the original and the shifted path in the knot tying scenario.

To sum up, the results show the influence of the different arm lengths and the user positioning. Therefore, the consideration of arm lengths of the expected users as well as an appropriate positioning are important. Furthermore, a good model of the human user would be of enormous benefit to provide a sophisticated analysis. Additionally, the current simulation includes just a simulation of the user’s motion. The “inverse kinematics“ of the human is estimated similarly to the inverse kinematics of serial chain manipulators in Chapter 2 and Section 7.3. The evaluation on real human actions could provide further aspects and insights. A large user study would be required for an analysis at a sophisticated level. This is, however, out of the focus of this thesis. To conclude, the current simulations can only indicate the potential of structure analyses for master-slave systems.

Chapter 8

Conclusion and Future Work

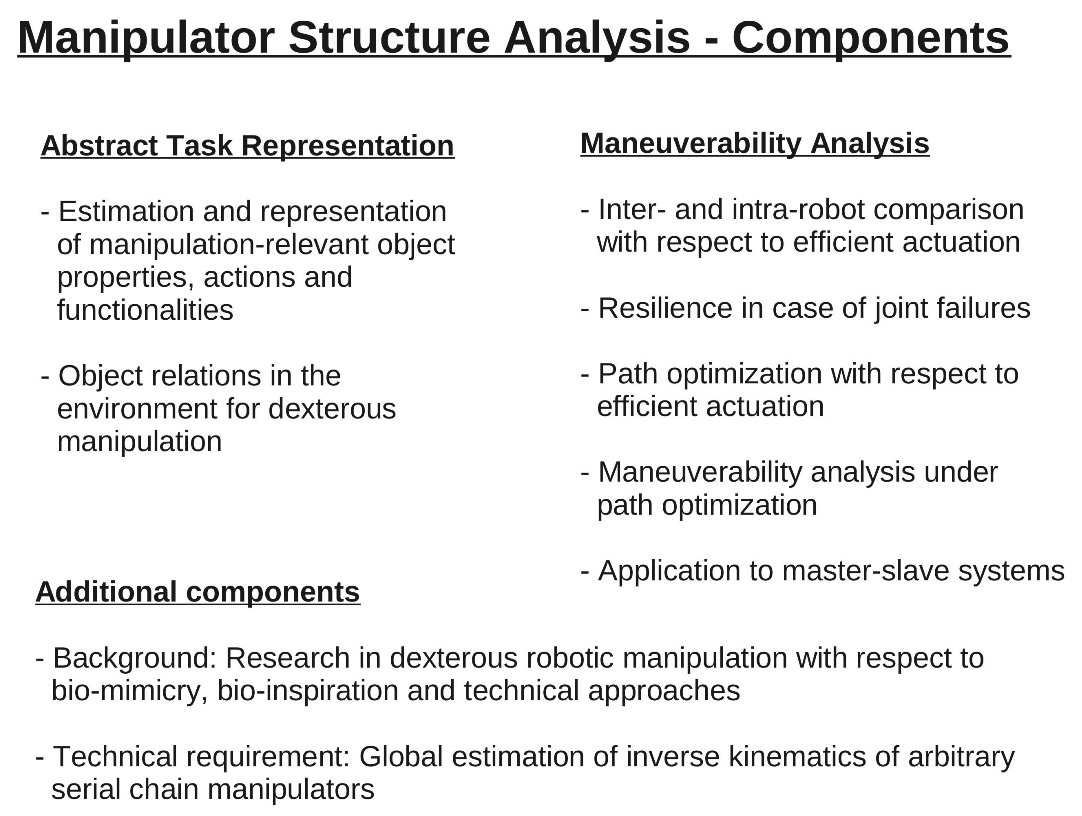
In this thesis, serial chain manipulator structures have been analyzed in the context of abstractly represented tasks. Possible joint failures and the potential of path optimization have been examined. A conclusion and a discussion of the presented work follow next. The thesis closes with possible directions for future work.

8.1 Conclusion and Discussion

All components of the proposed maneuverability analysis in the context of abstractly represented tasks are, once again, shown in Fig. 8.1. We shortly review the abstract representation of tasks, first. Afterward, a short summary of the manipulation analysis follows, before the additional components are briefly revised. At the end of this Section, the components of the task-specific structure analysis of manipulators and their interplay are discussed.

8.1.1 Summary and Conclusion

The abstract representation of tasks forms the first of the two main parts. The proposed estimation and representation of manipulation-relevant object properties and actions enables a focused observation of human actions. The framework allows to deal with strong variations in actions performed by a human operator. Only changes of characteristic properties require an update of the information in the internal representation. The proposed descriptors consist of an Object Container and a Functionality Map spanning typical object locations in a graph. They allow a close monitoring of changes in a physical state of the object or its function in the environment, since the detection of really new information is enabled. This is an important contribution towards robotic systems, which act in a more autonomous manner. The object-centric perspective allows the system to acquire knowledge efficiently and, moreover, to re-use the knowledge in similar situations. Furthermore, the experiments on external tracking data and vision data show, that the system can derive the knowledge from different sources.



Manipulator Structure Analysis - Components

Abstract Task Representation

- Estimation and representation of manipulation-relevant object properties, actions and functionalities
- Object relations in the environment for dexterous manipulation

Maneuverability Analysis

- Inter- and intra-robot comparison with respect to efficient actuation
- Resilience in case of joint failures
- Path optimization with respect to efficient actuation
- Maneuverability analysis under path optimization
- Application to master-slave systems

Additional components

- Background: Research in dexterous robotic manipulation with respect to bio-mimicry, bio-inspiration and technical approaches
- Technical requirement: Global estimation of inverse kinematics of arbitrary serial chain manipulators

Figure 8.1: Maneuverability analysis for abstractly represented tasks - Components. The Figure shows the components for both main parts and the additionally required components as already in Fig. 1.3. All components are presented in this thesis.

More complex dexterous manipulations are represented in the form of object relations in the environment based on a contact state perspective. This allows not only an easy handling of complex dexterous manipulations, but also a reusage and an adaption in changing environments if feasible. Our scenarios illustrate the advantages. The paths are adapted if necessary due to obstacles (e.g., other organs in medical applications). Moreover, efficiency criteria can be included in the final choice of the path. The usage of different robots is possible, since the representation is robot-independent.

After the summary of the first main part of the manipulator structure analysis, we shortly review the second main part: The structure analysis itself.
It comprises also the analysis of the potential of path optimization with respect to efficient actuation. The context of abstractly represented tasks is utilized to optimize path configurations in general. The introduced concept of the Elastic Power Path reflects the desired elasticity of the path as well as the aim to minimize the necessary energy. The experiments show, that the number and the magnitude of the peaks in the acceleration profile can be reduced significantly

to improve the efficiency. Hence, the system is able to make use of the freedom in path planning which accompanies the abstract representation of tasks.
The abstract task representation forms the context of the actual analysis of the maneuverability of different, arbitrary manipulators. We introduce the Maneuverability Volume and the Spinning Pencil to abstractly represent the robot's capabilities with respect to efficiency. Possible joint failures are also considered in the analysis. The already described Elastic Power Path concept allows to optimize paths which is of special interest in this context.
The entire structure analysis provides an inter-robot and an intra-robot comparison. The results of our experiments show, (1) whether the specified task areas can be reached, (2) the essential joints for a desired task, (3) the illustrative comparison of the capabilities of robots, (4) the magnitude of the reduction of the original maneuverability under joint failure and (5) the potential of path optimization.
The analysis of master-slave systems indicates a further application area of the proposed approach. It differs from the analysis of autonomous serial chain manipulators with respect to its human-centered perspective. Advanced experiments would require a sophisticated model of the human user and his/ her behavior, which is out of the focus of this thesis. However, our basic experiments point out possible potentials. The results indicate, that it would be worth to investigate further into this direction, if a a sophisticated model of the human user and his/ her behavior is available.

Besides the two main parts of the thesis, two additional components are provided. The first one is the overview of the current state of the art in robotic grasping and manipulation. It shows, that many successful approaches exist for a lot of aspects of the overall problem. Both discussed perspectives, bio-mimicry/ bio-inspiration and technical approaches, make important contributions. The entire grasping and manipulation capabilities and abilities (including independence of humans and knowledge about, e.g., physical and handling properties of unknown objects) in any environment (including the capability to interact with any human) have not been achieved yet by a robotic system. The missing or incomplete components comprise limited sensing capabilities and a further independence of human teachers in unstructured environments.
The work presented here, contributes mainly to technical approaches, since the major part of this thesis has an object-centric or robot-centric perspective. The object-centric perspective is applied for the estimation and representation of knowledge relevant for (dexterous) manipulations, since the objects and their properties, resp., relations are in the focus. The representations contribute towards autonomous systems which acquire knowledge efficiently and which allow a reusage of the knowledge in similar situations. The robot-centric perspective forms the base for the path optimization and the manipulator structure analysis. Of course, the robot is the object of interest, since the robot's capabilities are optimized, resp., analyzed with respect to its own actuation. The developed analysis of the task-specific suitability of a robotic system has not been presented yet. The human-centric perspective is only used in the additional analysis of master-slave systems. The human is the one who controls the system. Moreover, the perceived

level of convenience is an important aspect in the deployment of the system. Therefore, it is reasonable to focus on the human, there.
The overview of the current state of the art in robotic grasping and manipulation forms the first additional component in this work. The second additional component is an important technical background: The advanced concept for the estimation of the inverse kinematics of arbitrary serial chain manipulators. Only this type of concept allows the exhaustive manipulator structure analysis. The separate experiments on this component achieve good results. A large number of solutions is found for the inverse kinematics at single points. The estimation of the inverse kinematics along entire paths provides consecutive configurations which are close to each other as desired. Moreover, we apply the estimation on a human-like robotic hand successfully. The corresponding experiments with the efficiently reduced model of a complex, arbitrary hand show very accurate results for the estimation of the inverse kinematics.

8.1.2 Discussion of the Concepts

The components of the task-specific structure analysis of manipulators and their interplay are discussed in the following. The first main part of the thesis, the knowledge representation, consists of two components: The first one focuses on general manipulation-relevant properties. The second one deals with more dexterous manipulations. This allows the consideration of the specific requirements of both types of manipulations.
The second main part of the thesis, the structure analysis itself, is essentially based on two concepts: The Maneuverability Analysis and the Elastic Power Path. The Maneuverability Analysis provides a more general analysis of the manipulator capabilities. It is based on the characteristic areas of manipulation, the Location Areas. In contrast, the Elastic Power Path is built up on the possible paths and their properties. This concept is more specific to the actual accomplishment, since we analyze the potential of path optimization with respect to efficiency. Hence, the possible paths need to be included in this type of analysis. A further application of the Elastic Power Path are setups which do not or do hardly allow to determine the characteristic Location Areas in an appropriate manner. To sum up, both presented concepts are developed to consider different types of analyses and applications.

All components of the presented analysis are shown in Fig. 8.1. They are interconnected. In general, the interfaces are built up in a manner, such that components can be substituted as long as they fulfill the respective requirements. For example, the general task description has to be chosen in a manner, such that it allows to make use of, e.g., the Location Areas as characteristic manipulation areas. Overall, the components work at an abstract level of knowledge. This means, they can be applied to any serial chain manipulator and to any task, since they are independent of the robot structure and the task. Hence, a substitution is only reasonable, if these independences are kept. However, the single components are not only applicable in the provided context as we have already shown in our experiments. In the following section, further applications are discussed as future work.

8.2 Future Work

After the discussion of the presented task-specific structure analysis, we indicate possible directions of future work.

8.2.1 Extension of Single Components

The estimation and representation of manipulation-relevant object properties, actions and functionanlities in the environment could be extended as stand-alone part with a focus on more unknown situations and environments. For example, the transfer of knowledge could be examined further in this context.

The path optimization approach with the Elastic Power Path concept could be extended with respect to further properties stored in the Functionality Map. They can be included in the optimization. Moreover, the influence of obstacle avoidance could be analyzed. Both, the inclusion of further properties and the consideration of obstacles, will reduce the set of possible paths. An analysis of the limits of path optimization in different applications (e.g., manipulations of daily-life objects vs. medical applications) could be interesting in this context.
The optimization could also be applied to more complex dexterous manipulations. The representation of the object relations in the environment based on a contact state perspective can allow different paths to fulfill a task, too. The final choice of the path could be determined based on the proposed path optimization approach. Currently, a limited, discrete set of paths is considered. This set could be extended to a continuous one, if the developed path optimization would be applied.

In the experiments on master-slave systems, an appropriate model of the human's hand arm system would be important for an advanced analysis. The evaluation on real human actions could provide further aspects and insights. A large user study would be required for an analysis at a sophisticated level.
The experiments indicate, that the positioning of the user has a significant influence on efficiency. A development of a positioning procedure could be of advantage. An interesting question could be, whether, resp., under which conditions a general positioning procedure has a larger influence on the convenience than a very focused task-specific system development. The integration of a positioning procedure into planning in teleoperation systems (e.g., [198]) could be another issue.

8.2.2 Further Extensions and Applications of the Entire Analysis of Manipulators

We focused on the analysis of *manipulator structures* with respect to *efficient control*. In future work, the inclusion of other criteria relevant for the choice of a robot, e.g., available sensors and interaction capabilities (see also Chapter 1), could provide further important information.
For example, one could analyze the available sensors with respect to a desired task to answer

the following questions: Are the available sensors sufficient to fulfill the task? Is the robot, e.g., able to detect an object? Does one of the available robots have more appropriate sensors, which allow this robot to perform the task at a better quality in comparison to other robots (e.g., regarding runtime, accuracy)?
In the context of interactions, one could consider communication and intention recognition capabilities as well as physical interaction capabilities. Is the robot, e.g., able to recognize a changed intention of a human during a cooperation? Can the human, for example, command the robot by verbal communication? What happens, if the robot hits the human accidentally? Is it able to recognize such a hit? If yes, is it able to react?

The *motion* capabilities of a manipulator structure are analyzed in this thesis. As described, we do not consider the mass, mass-distribution or other physical properties of the robot, since they are extensive topics on their own. However, the forces acting on a robot are influenced by these properties. An analysis with respect to forces and torques (including forces and torques the robot can apply), could provide further interesting arguments to deploy one or another robot.

The structure analysis developed in this thesis can be used as a base for the design of robots. Currently, different platforms can be compared with respect to their suitability for desired tasks. In future work, the proposed concepts can be used as components towards a general robot design with the determination of all design parameters, e.g., all Denavit-Hartenberg parameters. Such a design faces the large dimensionality as a new challenge, especially, in advanced robotic systems.

Moreover, the proposed analysis can be extended to manipulators with translational joints and more complex systems like humanoid robots. Another extension could be mobile robots and walking robots. In the case of mobile robots, serial chain manipulators can be attached to mobile platforms and the entire system can be analyzed. Of course, new aspects occur due to the mobile platform. It has to be moved appropriately. Currently, the manipulator's base is fixed at one position. In the area of mobile manipulation, several approaches have already been presented. For example, Yamamoto and Yun [199] developed a control algorithm for the coordinated locomotion and manipulation of a mobile manipulator considering the effect of *dynamic interaction* [200]. Seraji [201] presented a *general approach* for motion control of a mobile base with a mounted manipulator arm. The *manipulability* is, e.g., kept in [202]. Bayle *et al.* [203] analyzed the *manipulability* of mobile manipulators based on the manipulability measure of [204], [86]. The extension of our approach to mobile manipulators can contribute a check for the *task-specific suitability* in this area.
Coordinated task executions between a human and a mobile manipulator could be examined as well (e.g., control scheme in [205]). Similarly to the analysis of master-slave systems, the convenient system usage for the human is an important aspect. However, it differs from master-slave systems, since the human and the executing robot interact directly without an intermediate "controlling" system like the master system. For this kind of interaction, the robot has to be

able to understand the human's intention during each step of the interaction as we described already at the beginning of this Section. Moreover, several mobile manipulators could cooperate to perform a task (e.g., decentralized control of cooperating mobile manipulators [206]). Such a cooperation could provide further interesting aspects for an analysis. In the case of a decentralized control, the robot needs to be able to understand the intention of its robotic teammates, similarly to an interaction with humans.

Humans can also control robotic systems in a more "direct" manner via, e.g., EEG maps [207] or a neural interface system [208]. Electromyography measurements [209] are another way to control a robotic system. The analysis of such systems requires the consideration of the new, more "direct" ways of controlling.

A further application area are parallel, possibly open-chain robots. A robotic hand is such an example. However, further aspects need to be considered here. Advanced sensing modalities can provide an important feedback. Hence, the quality of the sensing capabilities is crucial in this area.

Appendix A

Further Information about the Vision Data

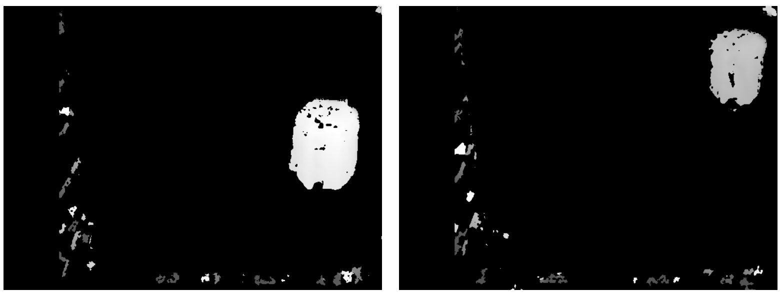


Figure A.1: Object Candidates: Different positions - vision data. The figure illustrates the different sizes of object candidates, depending on their distance to the observer system (see Section 7.1.2.1). As it can be expected, an object closer to the observer appears to have a bigger size in pixel than the same object placed further away.

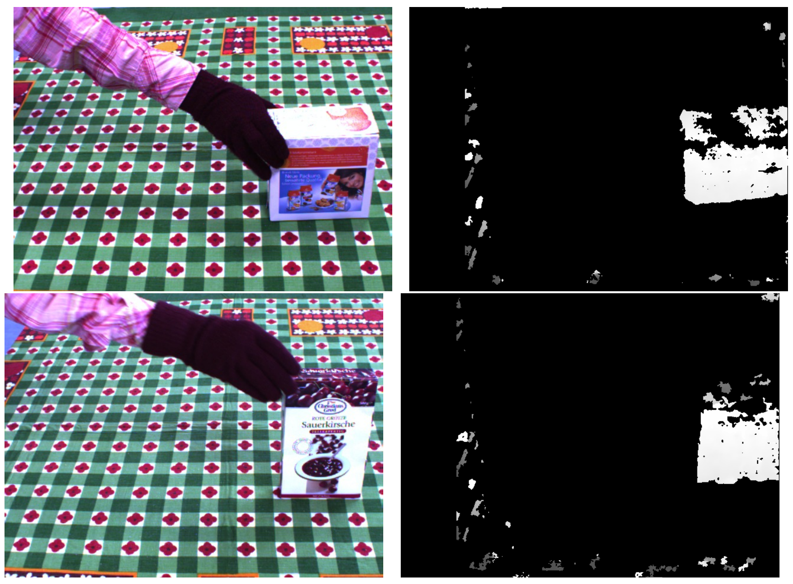


Figure A.2: Object Candidates: Smaller objects candidates in the image - vision data. Some object candidates cover smaller regions in the image than expected (see Section 7.1.2.1). This is due to the fact, that the disparity is not computable for some parts of the object. The figure depicts two reasons. First, there are reflections on the top of the object in the top left, leading to non-connected parts of the object in the disparity map on the top right. Second, the top of the object in the bottom left contains ambiguous structure, resulting in missing disparity information on the bottom right.

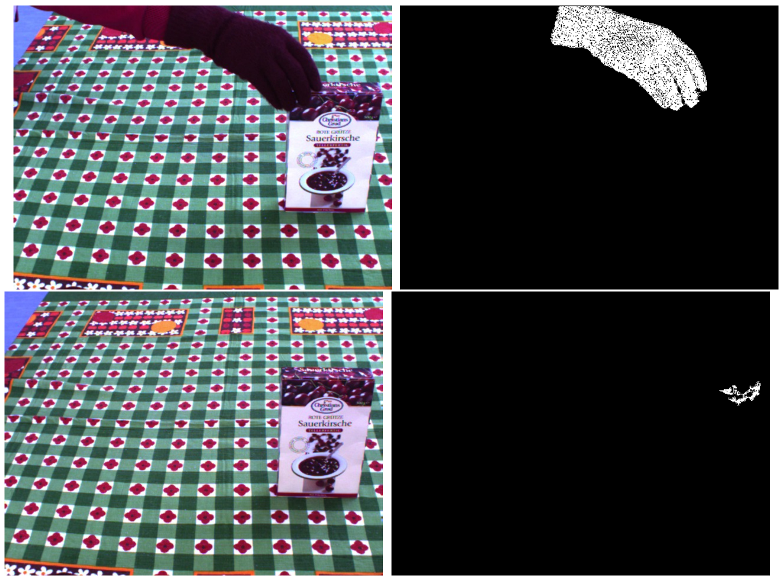
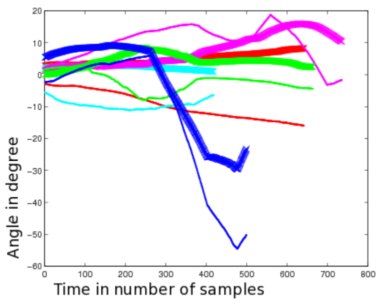


Figure A.3: Blob detection hand - vision data. The figure shows one exemplary correct identified hand and one example of a false positive detection of the contact between the hand and the object candidate (see Section 7.1.2.2). The false positive alarm is due to similar colors, appearing on the object and the glove, which is just used for the purpose of the blob detection.



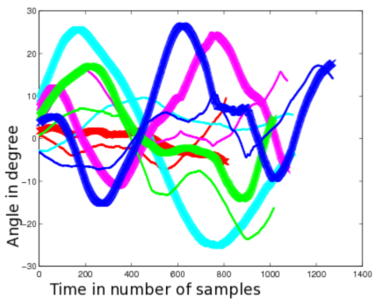


Figure A.4: Development of the angles for both horizontal axes - vision data. The change of the angles along both horizontal axes is shown for the sequences of object 1 (see Section 7.1.2.4). *Left:* Seq. 1 (red), 2 (cyan), 3 (mag.), 4 (green), 5 (blue). *Right:* Seq. 6 (red), 7 (cyan), 8 (mag.), 9 (green), 10 (blue). The thick and thine lines refer, resp., to one of the horizontal axes. The strong variations of the rotated sequences (seq. 7-10, on the right) are clearly visible. The mislabeled sequences without rotation (seq. 3, 4, 5, on the left) have also variations, at least along one of the horizontal axes (thick line of seq. 3 and 4, both axis of seq. 5, on the left).

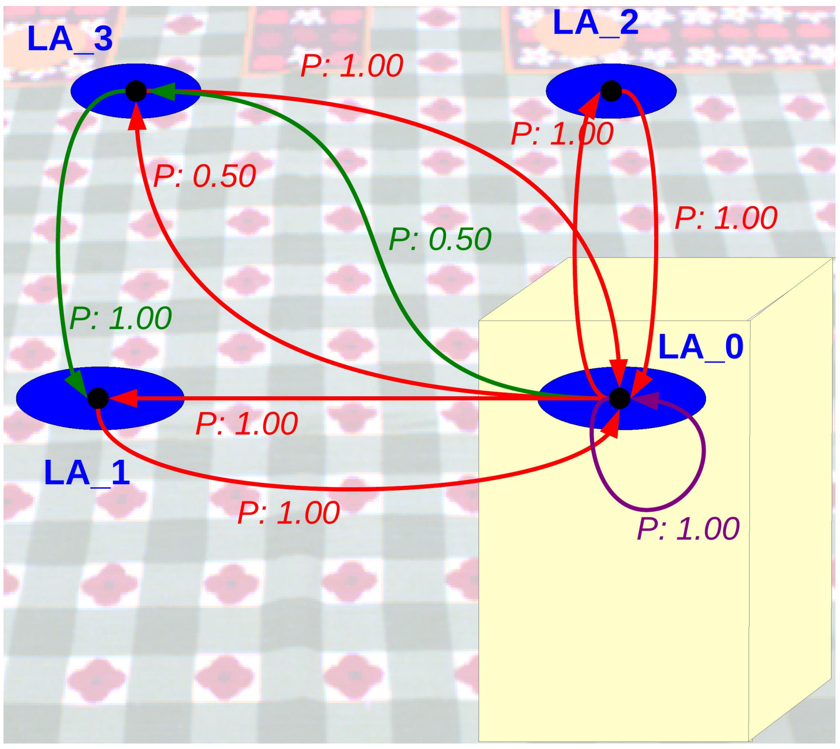


Figure A.5: Functionality Map: ideal solution - vision data. The resulting perfect Functionality Map, which should be achieved with the sequences (Type of all grasps: Power, thumb abd.) (see Section 7.1.2.4).

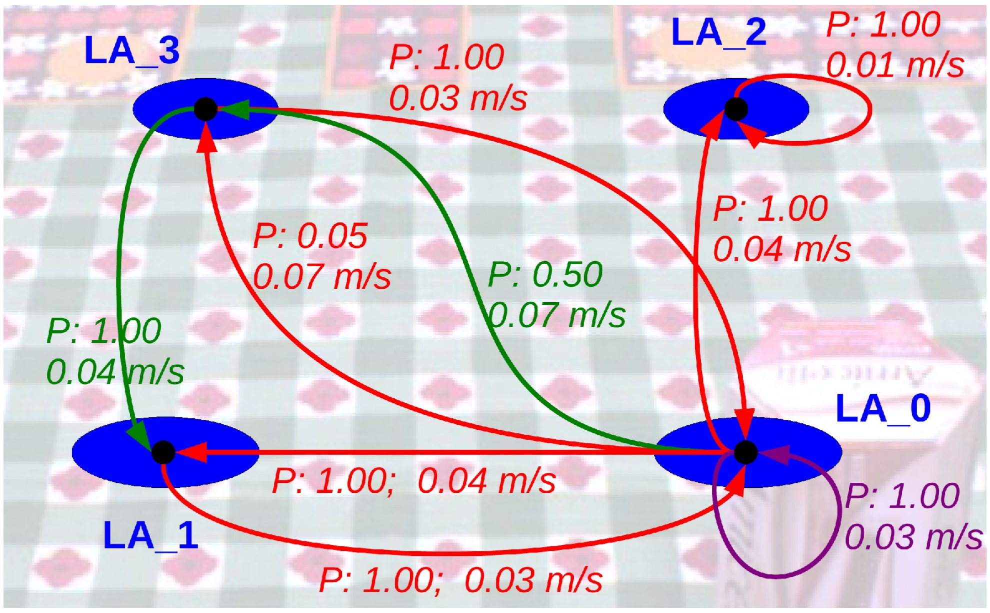


Figure A.6: Functionality Map: object 1 - vision data [2] (see Section 7.1.2.4). Except for the end location of self-loop at Location Area 2, all start and end locations are assigned to the correct Location Area. All of the connection properties are correctly identified. All types of grasps are correctly identified (Power, thumb abd.).

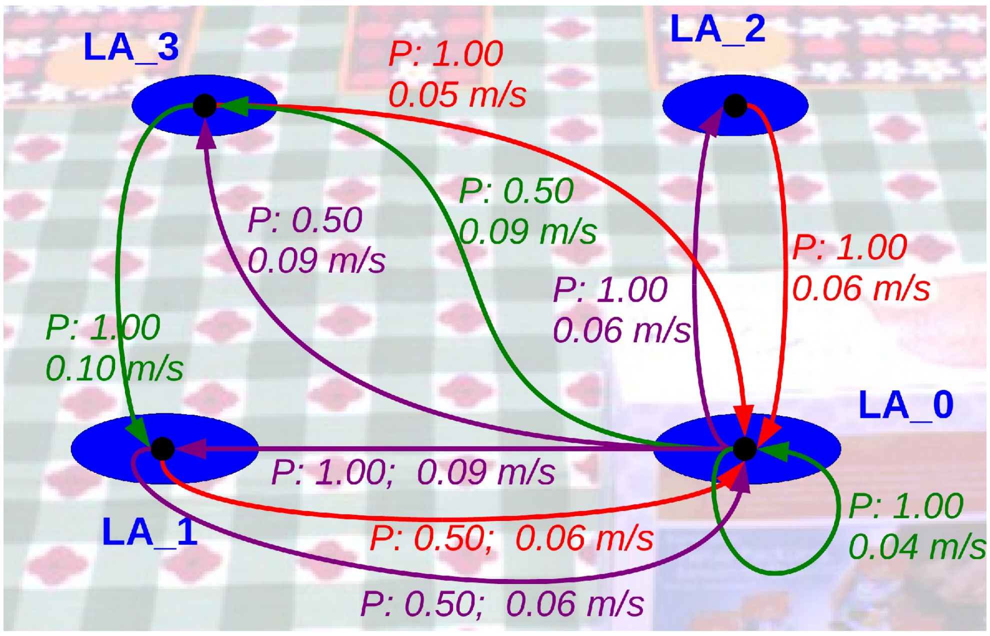


Figure A.7: Functionality Map: object 2 - vision data [2] (see Section 7.1.2.4). The misclassifications of the connection properties in the Functionality Map of object 2 become visible. The assignment to the Location Areas is done without any error. All types of grasps are correctly identified (Power, thumb abd.).

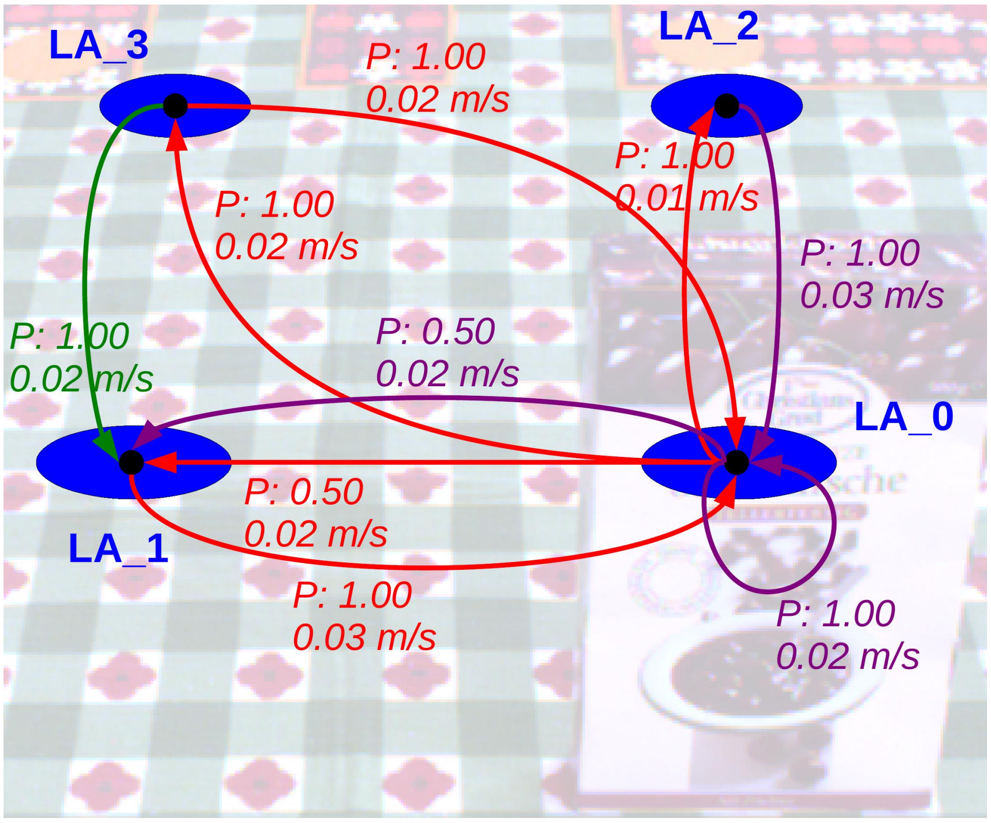


Figure A.8: Functionality Map: object 3 - vision data (see Section 7.1.2.4). The problem with seq. 21 (misclassified movement property and wrong assignment of the end location) can be observed in the connection of the arbitrary movement from Location Area 0 to Location Area 1. Besides one other misclassified connection property (arbitrary movement from Location Area 2 to Location Area 0), the Functionality Map is correct. All types of grasps are correctly identified (Power, thumb abd.).

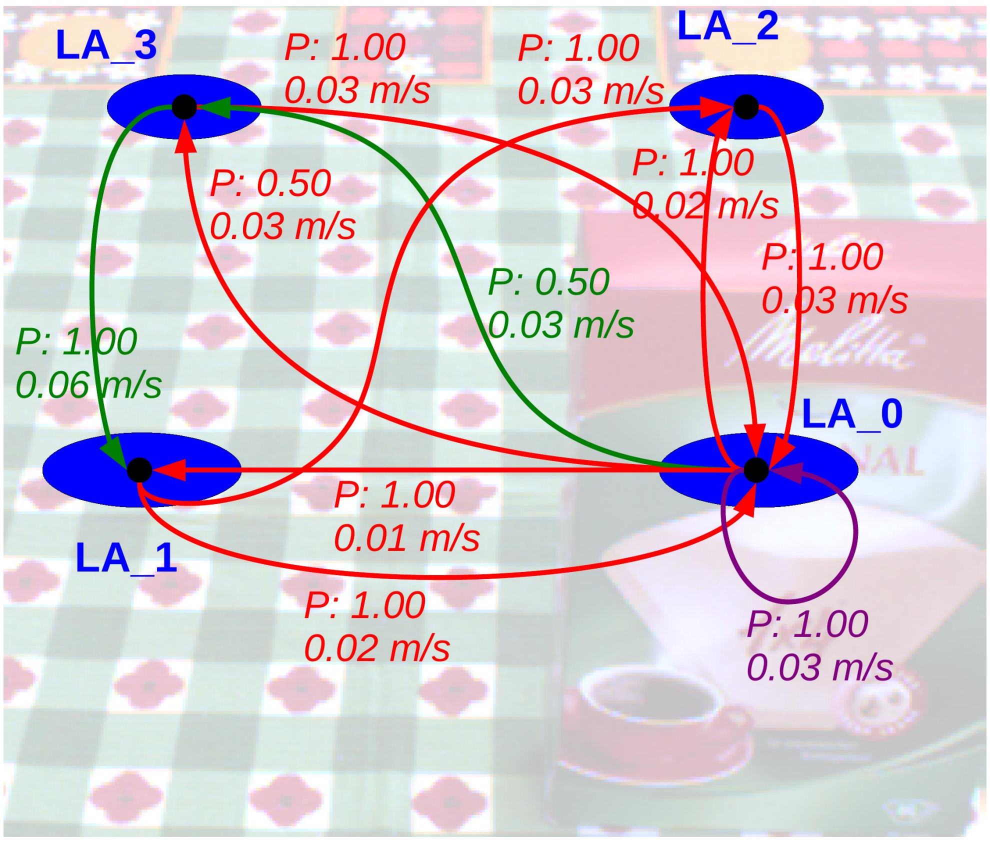


Figure A.9: Functionality Map: object 4 - vision data (see Section 7.1.2.4). The Functionality Map is nearly perfect, just the constrained connection from Location Area 1 to Location Area 2 should not exist. This is due to the wrong assignment of the end location, Location Area 0 is the proper one. All connection properties are correctly identified. All types of grasps are correctly identified (Power, thumb abd.).

Appendix B

Parameter Descriptions

B.1 Denavit Hartenberg Convention

The robotic systems in this thesis are described in the DH-convention suggested by Denavit and Hartenberg [164] in the form shown in [9].

The link frames are attached according to [9] as follows:

1. "Identify the joint axes and image infinite lines along them. For steps 2 through 5 below, consider two of these neighboring lines (at axes i and $i+1$)."
2. "Identify the common perpendicular between them, or point of intersection. At the point of intersection, or at the point where the common perpendicular meets the ith axis, assign the link-frame origin."
3. "Assign the $\hat{Z}_i$ axis pointing along the ith joint axis."
4. "Assign the $\hat{X}_i$ axis pointing along the common perpendicular, or, if the axes intersect, assign $\hat{X}_i$ to be normal to the plane containing the two axes."
5. "Assign the $\hat{Y}_i$ axis to complete a right-hand coordinate system."
6. "Assign 0 to match 1 when the first joint variable is zero. For N, choose an origin location and $\hat{X}_n$ direction freely, but generally so as to cause as many linkage parameters as possible to become zero."

The involved parameters describe the transformation between successive coordinate frames as described in [9]:

1. "a_i = the distance between $\hat{Z}_i$ and $\hat{Z}_{i+1}$ along $\hat{X}_i$"
2. "α_i = the angle between $\hat{Z}_i$ and $\hat{Z}_{i+1}$ measured about $\hat{X}_i$"
3. "d_i = the distance between $\hat{X}_{i-1}$ and $\hat{X}_i$ along $\hat{Z}_i$"
4. "θ_i = the angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about $\hat{Z}_i$"

The joint parameter is either θ_j (revolute joint) or d_i (prismatic joint). The total number of joints is D.

The resulting overall transformation matrix is ${}^{i-1}_{i}T$ (see [9]):

$$
{}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

.

B.2 Z-Y-X Euler angles

The orientation of the robot's end-effector is described as Z-Y-X Euler angles [9]. There, a rotation ${}^{A}_{B}R$ is described as follows:

"Rotate frame B first about $\hat{Z}_B$ by an angle α, then about $\hat{Y}_B$ by an angle β, and, finally, about $\hat{X}_B$ by an angle γ. In this representation, each rotation is performed about an axis of the moving system B rather than one of the fixed reference A."

The final rotation matrix is ${}^{A}_{B}R_{Z'Y'X'} = R_Z(\alpha)R_Y(\beta)R_X(\gamma)$:

$$
{}^{A}_{B}R_{Z'Y'X'} = \begin{bmatrix} cos\alpha\ cos\beta & cos\alpha\ sin\beta\ sin\gamma - \ sin\alpha\ cos\gamma & cos\alpha\ sin\beta\ cos\gamma + \ sin\alpha\ sin\gamma \\ sin\alpha\ cos\beta & sin\alpha\ sin\beta\ sin\gamma + \ cos\alpha\ cos\gamma & sin\alpha\ sin\beta\ cos\gamma - \ cos\alpha\ sin\gamma \\ -\ sin\beta & cos\beta\ sin\gamma & cos\beta\ cos\gamma \end{bmatrix}
$$

B.3 General Parameters

Table B.1: General Parameters.

Parameter	Description
D	Total number of joints
j	Joint j within the total number of joints D
θ_j	Joint angle θ_j at joint j
$\boldsymbol{v}_j$	Linear velocity of joint j
$\boldsymbol{\omega}_j$	Rotational velocity of joint j
N	Total number of points on a path
$\boldsymbol{p}_i$	Point $\boldsymbol{p}_i$ along a path

Mathematical Notation

Table B.2: Explanation of the mathematical notation.

Symbols/ Examples:	Description:
x, X	Scalar variables are italic lower or upper case letters
$\boldsymbol{a}$	Vectors are bold, italic lower case letters
$\overrightarrow{\boldsymbol{p}_a\boldsymbol{p}_b}$	A vector between two points $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$ is denoted by an arrow
$\boldsymbol{X}$	Matrices are bold, italic, Latin upper case letters
LA	Characteristic places bold, regular, Latin upper case letters
	(here: "Location Areas")
$\mathrm{F}(x)$	Functions are usually regular lower or upper case letters
$\mathbf{contact}(o, p)$	The special function "**contact**" is written in bold, lower case letters
$\mathcal{S}$	Sets and intervals are represented by calligraphic literals
$\{\theta_j\}$	The brackets { and } denote a set of variables or vectors
$\|x\|$	Absolute value of x
$\|\boldsymbol{a}\|$	Length of vector $\boldsymbol{a}$
$\\|\boldsymbol{p}_a\boldsymbol{p}_b\\|$	Euclidean distance between two points $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$

List of Abbreviations

Table B.3: List of abbreviations which are used in this thesis.

Abbreviation:	Description:
DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V.
DMP	Dynamic Movement Primitives
DoF	Degree of Freedom
DH	Denavit Hartenberg
GPS	Global Positioning System
GRASP	Emergence of Cognitive Grasping through Introspection, Emulation and Surprise (EU-project, 7th Framework Program)
HMM	Hidden Markov Model
IEEE	Institute of Electrical and Electronics Engineers
KLT	Kanade-Lucas-Tomasi (feature tracker)
MA	Maneuverability Analysis
MIS	Minimally invasive surgery
mV	Maneuverability Volume
PCA	Principal Component Analysis
resp.	respectively
ROI	Region of Interest
RSJ	The Robotics Society of Japan
seq.	sequences
sP	Spinning Pencil
V-GPS	Visual GPS
vs.	versus

Author's Publications

[1] Susanne Petsch and Darius Burschka. **Estimation of Spatio-Temporal Object Properties for Manipulation Tasks from Observation of Humans**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 192–198, Anchorage, Alaska, USA, May 2010.

[2] Susanne Petsch and Darius Burschka. **Representation of Manipulation-Relevant Object Properties and Actions for Surprise-Driven Exploration**. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1221–1227, San Francisco, California, USA, September 2011.

[3] Susanne Petsch and Darius Burschka. **Path Configuration for Abstractly Represented Tasks with Respect to Efficient Control**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems: Workshop Beyond Grasping - Modern Approaches for Dynamic Manipulation*, Vilamoura, Algarve, Portugal, October 2012.

[4] Susanne Petsch and Darius Burschka. **Analysis of Manipulator Structures under Joint-Failure with Respect to Efficient Control in Task-Specific Contexts**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1955–1961, Karlsruhe, Germany, May 2013.

[5] Susanne Petsch and Darius Burschka. **Path Optimization for Abstractly Represented Tasks with Respect to Efficient Control**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1268–1273, Karlsruhe, Germany, May 2013.

[6] Susanne Petsch and Darius Burschka. **Estimation of Inverse Kinematics of Arbitrary Serial Chain Manipulators and Human-Like Robotic Hands**. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 728 – 733, Wollongong, Australia, July 2013.

[7] Susanne Petsch and Darius Burschka. **Contact State Based Representation of Object Relations in the Environment for Dexterous Manipulations**. In

IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pages 1051 – 1057, Wollongong, Australia, July 2013.

References

[8] Franziska Zacharias, Christoph Borst, and Gerhard Hirzinger. **Capturing Robot Workspace Structure: Representing Robot Capabilities**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3229 – 3236, 2007. 7

[9] John J. Craig. *Introduction to Robotics - Mechanics and Control*. Prentice Hall, 2005. 7, 34, 68, 75, 76, 110, 185, 186

[10] D. Pieper and B. Roth. **The Kinematics of Manipulators Under Computer Control**. In *Proceedings of the Second International Congress on Theory of Machines and Mechanisms*, **2**, pages 159–169, 1969. 7

[11] Haider Mohamed, Samer Yahya, M. Moghavvemi, and S.S. Yang. **A New Inverse Kinematics Method for Three Dimensional Redundant Manipulators**. In *ICCAS-SICE*, pages 1557 – 1562, 2009. 7

[12] Alain Liégeois. **Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms**. *IEEE Transactions on Systems, Man, and Cybernetics*, **7**(12):868–871, 1977. 7

[13] D. Tolani, A. Goswami, and N.I. Badler. **Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs**. *Graphical models*, **62**:353–388, 2000. 7

[14] Rainer Konietschke and Gerhard Hirzinger. **Inverse Kinematics with Closed Form Solutions for Highly Redundant Robotic Systems**. In *IEEE International Conference on Robotics and Automation*, pages 2945–2950, 2009. 7

[15] Rodney G. Roberts and Anthony A. Maciejewski. **Nearest Optimal Repeatable Control Strategies for Kinematically Redundant Manipulators**. *IEEE Transactions on Robotics and Automation*, **8**(3):327–337, 1992. 7

[16] Andrew A. Goldenberg, B. Benhabib, and Robert G. Fenton. **A Complete Generalized Solution to the Inverse Kinematics of Robots**. *IEEE Transactions on Robotics and Automation*, **1**(1):14–20, 1985. 7

[17] F. Pierrot, A. Fournier, and P. Dauchez. **Towards a Fully-Parallel 6 DOF Robot for High-Speed Applications**. In *IEEE International Conference on Robotics and Automation*, pages 1288–1293, 1991. 7

[18] Dominik Bertram, James Kuffner, Rüdiger Dillmann, and Tamim Asfour. **An Integrated Approach to Inverse Kinematics and Path Planning for Redundant Manipulators**. In *IEEE International Conference on Robotics and Automation*, pages 1874–1879, 2006. 7

[19] D. Lyons. **A Simple Set of Grasps for a Dexterous Hand**. In *IEEE International Conference on Robotics and Automation*, **2**, pages 588–593, 1985. 7, 33, 36

[20] Stuart Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach*. Prentice Hall, 2003. 8, 9

[21] Lawrence R. Rabiner. **A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition**. *IEEE*, **77**(2):257–286, 1989. 9, 55

[22] N. Padoy, D. Mateus, D. Weinland, M.-O. Berger, and N. Navab. **Workflow Monitoring based on 3D Motion Features**. In *Proceedings of the International Conference on Computer Vision Workshops, IEEE Workshop on Video-oriented Object and Event Classification*, 2009. 9, 12, 15

[23] Carol E. Reiley and Gregory D. Hager. **Task Versus Subtask Surgical Skill Evaluation of Robotic Minimally Invasive Surgery**. In *Medical Image Computing and Computer-Assisted Intervention -MICCAI 2009*, pages 435–442, 2009. 9, 12, 15, 56

[24] Sabine Webel, Yana Staykova, and Ulrich Bockholt. **Towards Workflow Acquisition of Assembly Skills using Hidden Markov Models**. In *Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics*, pages 841–846, 2009. 9

[25] Sanmohan and Volker Krüger. **Primitive Based Action Representation and Recognition**. In A.-B. Salber, J. Y. Hardeberg, and R. Jenssen, editors, *SCIA 2009*, **5575**, pages 31–40. Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, 2009. 9

[26] Chavdar Papazov and Darius Burschka. **Stochastic Global Optimization for Robust Point Set Registration**. *Computer Vision and Image Understanding*, **115**, December 2011. 9, 32, 121, 135

[27] Elmar Mair, Klaus Strobl, Michael Suppa, and Darius Burschka. **Efficient Camera-Based Pose Estimation for Real-Time Applications**. In *IEEE International Conference on Intelligent Robots and Systems*, 2009. 9, 98, 102

[28] Tamim Asfour, Pedram Azad, Florian Gyarfas, and Rüdiger Dillmann. **Imitation Learning of Dual-Arm Manipulation Tasks in Humanoid Robots**. *International Journal of Humanoid Robotics*, **5**(2):183–202, 2008. 9

[29] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. **Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots**. In *IEEE International Conference on Robotics and Automation*, pages 1398–1403, Washington, DC, USA, 2002. 9, 64

[30] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. **Learning and Generalization of Motor Skills by Learning from Demonstration**. In *IEEE International Conference on Robotics and Automation*, pages 763–768, Kobe, Japan, 2009. 9, 64

[31] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998. 9, 19

[32] Deepak Verma and Rajesh P. N. Rao. **Imitation Learning Using Graphical Models**. In J. N. Kok et al., editor, *ECML 2007*, **4701**, pages 757–764. Lecture Notes in Artificial Intelligence, Springer-Verlag Berlin Heidelberg, 2007. 9, 19

[33] Grazia Bombini, Nicola Di Mauro, Teresa M. A. Basile, Stefano Ferilli, and Floriana Esposito. **Relational Learning by Imitation**. In A. Håkansson et al., editor, *KES-AMSTA 2009*, **5559**, pages 273–282. Lecture Notes in Artificial Intelligence, Springer-Verlag Berlin Heidelberg, 2009. 9, 19

[34] Sylvain Calinon, Florent Guenter, and Aude Billard. **On Learning, Representing, and Generalizing a Task in a Humanoid Robot**. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, **37**(2):286–298, April 2007. 9

[35] Sylvain Calinon, F. D'halluin, Eric L. Sauser, Darwin G. Caldwell, and Aude Billard. **Learning and Reproduction Gestures by Imitation**. *IEEE Robotics and Automation Magazine*, **17**:44 – 54, 2010. 9, 19

[36] Koichi Ogawara, Jun Takamatsu, Hiroshi Kimura, and Katsushi Ikeuchi. **Generation of a Task Model by Integrating Multiple Observations of Human Demonstrations**. In *IEEE International Conference on Robotics and Automation*, pages 1545–1550, May 2002. 10

[37] Bart Jansen and Tony Belpaeme. **A Model for Inferring the Intention in Imitation Tasks**. In *The 15th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN'06*, pages 238–243, 2006. 10

[38] Michael Pardowitz, Raoul Zöllner, and Rüdiger Dillmann. **Learning Sequential Constraints of Tasks from User Demonstrations**. In *5th IEEE-RAS International Conference on Humanoid Robots*, pages 424–529, 2005. 10

[39] Michael Pardowitz, Steffen Knoop, Rüdiger Dillmann, and Raoul D. Zöllner. **Incremental Learning of Tasks From User Demonstrations, Past Experiences, and Vocal Comments**. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, **37**(2):322–332, April 2007. 10

[40] Hae Won Park and Ayanna M. Howard. **Understanding a Child's play for Robot Interaction by Sequencing Play Primitives using Hidden Markov Models**. In *IEEE International Conference on Robotics and Automation*, pages 170–177, Anchorage, Alaska, USA, 2010. 10

[41] Volker Krüger, Dennis L. Herzog, Sanmohan Baby, Aleš Ude, and Danica Kragic. **Learning Actions from Observations**. *IEEE Robotics and Automation Magazine*, pages 30–43, June 2010. 10, 19

[42] Zoran Duric, Jeffrey A. Fayman, and Ehud Rivlin. **Function from Motion**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **18**(6):579–591, June 1996. 10

[43] Raoul D. Zöllner and Rüdiger Dillmann. **Using Multiple Probabilistic Hypothesis for Programming One and Two Hand Manipulation by Demonstration**. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2926–2931, Las Vegas, Nevada, USA, 2003. 10

[44] Louise Stark, Kevin Bowyer, Adam Hoover, and Dmitry B. Goldgof. **Recognizing Object Function Through Reasoning About Partial Shape Descriptions and Dynamic Physical Properties**. In *Proceedings of the IEEE*, **84**, pages 1640–1656, 1996. 10

[45] Masakatsu Mitani, Mamoru Takaya, Atsuhiro Kojima, and Kunio Fukunaga. **Environment Recognition Based on Analysis of Human Actions for Mobile Robot** . In *The 18th International Conference on Pattern Recognition (IEEE)*, pages 782–786, 2006. 10

[46] Antonio Chella, Haris Dindo, and Ignazio Infantino. **Learning High-Level Tasks Through Imitation**. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3648–3654, 2006. 11

[47] M. Eich, M. Dabrowska, and F. Kirchner. **Semantic Labeling: Classification of 3D Entities Based on Spatial Feature Descriptors**. In *IEEE International Conference on Robotics and Automation: Workshop on Best Practice Algorithms in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, Alaska, USA, 2010. 11

[48] Rohan Paul and Paul Newman. **FAB-MAP 3D: Topological Mapping with Spatial and Visual Appearance**. In *IEEE International Conference on Robotics and Automation*, pages 2649–2656, Anchorage, Alaska, USA, 2010. 11

[49] Rainer Jäkel, Sven R. Schmidt-Rohr, Martin Lösch, and Rüdiger Dillmann. **Representation and Constrained Planning of Manipulation Strategies in the Context of Programming by Demonstration**. In *IEEE International Conference on Robotics and Automation*, pages 162–169, Anchorage, Alaska, USA, 2010. 11

[50] Nadeesha Ranasinghe and Wei-Min Shen. **Surprise-Based Learning for Developmental Robotics**. In *ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems*, pages 65–70, August 2008. 11

[51] Nadeesha Ranasinghe and Wei-Min Shen. **Surprise-Based Developmental Learning and Experimental Results on Robots**. In *IEEE 8th International Conference on Development and Learning*, pages 1–6, 2009. 11

[52] Masahito Yashima. **On Planning for Whole Arm Manipulation with Switching Contact Modes**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1596–1601, 2001. 11

[53] Masahito Yashima and Hideya Yamaguchi. **Dynamic Motion Planning Whole Arm Manipulation Systems Based on Switching Contact Modes**. In *IEEE International Conference on Robotics and Automation*, pages 2492–2499, 2002. 11

[54] Masahito Yashima. **Randomized Manipulation Planning Considering the Local Optimization of Contact Mode Sequence**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2914–2919, 2003. 11

[55] Masahito Yashima, Yoshikazu Shiina, and Hideya Yamaguchi. **Randomized Manipulation Planning for a Multi-Fingered Hand by Switching Contact Modes**. In *IEEE International Conference on Robotics and Automation*, pages 2689–2694, 2003. 11

[56] Tetsuyou Watanabe, Kensuke Harada, Tsuneo Yoshikawa, and Zhongwei Jiang. **Towards Whole Arm Manipulation by Contact State Transition**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5682–5687, 2006. 11

[57] G.F. Liu, J. Li, and Z.X. Li. **Coordinated Manipulation of Objects by Multi-Fingered Robotic Hand in Contact Space and Active Joint Space**. In *IEEE International Conference on Robotics and Automation*, pages 3743–3748, 2002. 11

[58] Kensuke Harada and Makoto Kaneko. **Whole Body Manipulation**. In *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, pages 190–195, 2003. 11

[59] Kenneth Salibury, William Townsend, Brian Eberman, and David DiPietro. **Preliminary Design of a Whole-Arm Manipulation System (WAMS)**. In *IEEE International Conference on Robotics and Automation*, pages 254–260, 1988. 11

[60] Toshiharu Mukai, Shinya Hirano, Morio Yoshida, Hiromichi Nakashima, Shijie Guo, and Yoshikazu Hayakawa. **Whole-Body Contact Manipulation Using Tactile Information for the Nursing-Care Assistant Robot RIBA**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2445–2451, San Francisco, California, USA, 2011. 11

[61] Yasumichi Aiyama and Kazuki Sato. **Kinematics Analysis of Manipulation of Environment-Contacting object with Free-Joint-Structure**. In *IEEE International Conference on Mechatronics and Automation*, pages 40–45, 2011. 11

[62] Yasumichi Aiyama, Toshinori Yasui, and Tamio Arai. **Planning of Object Motion by Graspless Manipulation using Contact State Transition Graph**. In *4th IEEE International Symposium on Assembly and Task Planning*, pages 184–189, 2001. 11

[63] Siddharta S. Srinivasa, Michael A. Erdmann, and Matthew T. Mason. **Control Synthesis for Dynamic Contact Manipulation**. In *IEEE International Conference on Robotics and Automation*, pages 2523–2528, 2005. 11

[64] Michael Pardowitz, Raoul Zöllner, and Rüdiger Dillmann. **Learning Sequential Constraints of Tasks From User Demonstrations**. In *5th IEEE-RAS International Conference on Humanoid Robots*, pages 424–429, 2005. 12

[65] Peter Kazanzides, Gabor Fichtinger, Gregory D. Hager, Allison M. Okamura, Louis L. Whitcomb, and Russell H. Taylor. **Surgical and Interventional Robotics: Core Concepts, Technology, and Design**. *IEEE Robotics and Automation Magazine*, **15**(2):122–130, 2008. 12, 15

[66] Gabor Fichtinger, Peter Kazanzides, Allison M. Okamura, Gregory D. Hager, Louis L. Whitcomb, and Russell H. Taylor. **Surgical and Interventional Robotics: Surgical CAD-CAM Systems**. *IEEE Robotics and Automation Magazine*, **15**(3):94–102, 2008. 12, 15

[67] Gregory D. Hager, Allison M. Okamura, Peter Kazanzides, Louis L. Whitcomb, Gabor Fichtinger, and Russell H. Taylor. **Surgical and Interventional Robotics: Surgical Assistance Systems**. *IEEE Robotics and Automation Magazine*, **15**(4):84–93, 2008. 12, 15

[68] Russell H. Taylor. **A Perspective on Medical Robotics**. *Proceedings of the IEEE*, **94**(9):1652–1664, 2006. 12, 15

[69] R. H. Taylor, A. Menciassi, G. Fichtinger, and P. Dario. *Handbook of Robotics, Part F: Field and Service Robotics, Chapter 51 Medical Robotics and Computer-Integrated Surgery*. Springer Verlag Berlin Heidelberg, 2008. 12, 15

[70] Jaydev P. Desai and Nicholas Ayache. **Editorial: Special Issue on Medical Robotics**. *The International Journal of Robotics Research*, **28**(9):1098–1100, 2009. 12, 15

[71] Darius Burschka and Oliver Ruepp. **Vision-Based Analysis of Conventional Surgical Procedures**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems: Workshop on Methods for Safer Surgical Robotics Procedures*, San Francisco, California, USA, 2011. 12, 15

[72] Nicolas Padoy, Tobias Blum, Seyed-Ahmad Ahmadi, Hubertus Feussner, Marie-Odile Berger, and Nassir Navab. **Statistical Modeling and Recognition of Surgical Workflow**. *Medical Image Analysis*, **16**(3):632–641, apr 2010. 12, 15

[73] Nicolas Padoy and Gregory D. Hager. **Human-Machine Collaborative Surgery Using Learned Models**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5285–5292, San Francisco, California, USA, 2011. 12, 15

[74] Rainer Konietschke, Andreas Tobergte, Carsten Preusche, Paolo Tripicchio, Emanuele Ruffaldi, Sabine Webel, and Ulrich Bockholt. **A Multimodal Training Platform for Minimally Invasive Robotic Surgery**. In *IEEE International Symposium on Robot and Human Interactive Communication*, pages 422–427, 2010. 12, 15

[75] Yixin Gao, Mert Sedef, Amod Jog, Peter Peng, Michael Choti, Gregory Hager, Jeff Berkley, and Rajesh Kumar. **Towards Validation of Robotic Surgery Training Assessment Across Training Platforms**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2539–2544, San Francisco, California, USA, 2011. 12, 15

[76] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion*. The MIT Press, 2005. 13

[77] Dmitry Berenson, Thierry Simeon, and Siddharta S. Srinivasa. **Addressing Cost-Space Chasms in Manipulation Planning**. In *IEEE International Conference on Robotics and Automation*, pages 4561–4568, Shanghai, China, 2011. 13

[78] Hao Ding, Georg Schnattinger, Benjamin Passenberg, and Olaf Stursberg. **Improving Motion of Robotic Manipulators by an Embedded Optimizer**. In *IEEE Conference on Automation Science and Engineering*, pages 204–209, 2010. 13

[79] Lian Guangyu, Sun Zengqi, and Mu Chundi. **Optimal Motion Planning Passing Through Kinematic Singularities for Robot Arms**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4349–4354, 2006. 13

[80] Stephen L. Smith, Jana Tumová, Calin Belta, and Daniela Rus. **Optimal Path Planning for Surveillance with Temporal Logic Constraints**. *International Journal of Robotics Research*, **30**(14):1695–1708, 2011. 13

[81] Igor Belousov, Claudia Esteves, Jean-Paul Laumond, and Etienne Ferre. **Motion Planning for the Large Space Manipulators with Complicated Dynamics**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2160–2166, 2005. 13

[82] T.A. Rieswijk, G.G. Brouwn, and G. Honderd. **A Robust and Efficient Approach for the Time Optimization of Path Constrained Motions of Robotic Manipulators Incorporating Actuator Torque and Jerk Constraints**. In *IEEE International Symposium on Intelligent Control*, pages 507–513, 1992. 13

[83] Oliver Brock and Oussama Khatib. **Executing Motion Plans for Robots with Many Degrees of Freedom in Dynamic Environments**. In *IEEE Conference on Robotics and Automation*, **1**, pages 1–6, 1998. 13, 119

[84] Oliver Brock and Oussama Khatib. **Elastic Strips: A Framework for Motion Generation in Human Environments**. *International Journal of Robotics Research*, **21**(12):1031–1052, 2011. 13

[85] Oussama Khatib. **Inertial Properties in Robotics Manipulation: An Object-Level Framework**. *International Journal of Robotics Research*, **14**(1):19–36, 1995. 13

[86] Tsuneo Yoshikawa. **Manipulability of Robotic Mechanisms**. *The International Journal of Robotics Research*, **4**(2), 1985. 13, 172

[87] Jin-Oh Kim and Pradeep K. Khosla. **Dexterity Measures for Design and Control of Manipulators**. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 758–763, Osaka, Japan, 1991. 13

[88] Christiaan J. J. Paredis and Pradeep K. Khosla. **Kinematic Design of Serial Link Manipulators From Task Specifications**. *The International Journal of Robotics Research*, **12**(274):273–287, 1993. 13

[89] Haruhiko Asada. **Dynamic Analysis and Design of Robot Manipulators Using Inertia Ellipsoids**. In *IEEE International Conference on Robotics and Automation*, pages 94 – 102, 1984. 14

[90] Jihong Lee. **A Study of the Manipulability Measures for Robot Manipulators**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1458–1465, 1997. 14

[91] Alan Bowling and Oussama Khatib. **The Dynamic Capability Equations: A New Tool for Analyzing Robotic Manipulator Performance**. *IEEE Transactions on Robotics*, **21**(1):115 – 123, 2005. 14

[92] Matei Ciocarlie, Corey Goldfeder, and Peter Allen. **Dimensionality Reduction for Hand-Independent Dexterous Robot Grasping**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3270–3275, San Diego, California, USA, 2007. 14, 19

[93] Javier Romero, Thomas Feix, Hedvig Kjellström, and Danica Kragic. **Spatio-Temporal Modeling of Grasping Actions**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2103–2108, Taipei, Taiwan, 2010. 14, 19

[94] Kok-Meng Lee and Dharman K. Shah. **Kinematic Analysis of a Three-Degrees-of-Freedom In-Parallel Actuated Manipulator**. *IEEE Journal of Robotics and Automation*, **4**(3), 1988. 14

[95] G. Niemeyer, C. Preusche, and G. Hirzinger. *Handbook of Robotics, Part D: Manipulation and Interfaces, Chapter 31 Telerobotics.* Springer Verlag Berlin Heidelberg, 2008. 14

[96] Yasuyoshi Yokokohji and Tsuneo Yoshikawa. **Bilateral Control of Master-Slave Manipulators for Ideal Kinesthetic Coupling - Formulation and Experiment**. *IEEE Transactions on Robotics and Automation*, **10**(5):605–620, 1994. 14

[97] Blake Hannaford, Laurie Wood, Douglas A. McAffee, and Haya Zak. **Performance Evaluation of a Six-Axis Generalized Force-Reflecting Teleoperator**. *IEEE Transactions on Systems, Man, and Cybernetics*, **21**(3):620–633, 1991. 14, 84

[98] Septimiu E. Salcudean, Ming Zhu, Wen-Hong Zhu, and Keyvan Hashtrudi-Zaad. **Transparent Bilateral Teleoperation under Position and Rate Control**. *The International Journal of Robotics Research*, **19**:1185–1202, 200. 14

[99] Yasuyoshi Yokokohji, Noreo Hosotani, and Tsuneo Yoshikawa. **Analysis of Maneuverability and Stability of Micro-Teleoperation Systems**. In *IEEE International Conference on Robotics and Automation*, **1**, pages 237–243, 1994. 14

[100] Dale A. Lawrence. **Stability and Transparency in Bilateral Teleoperation**. *IEEE Transactions on Robotics and Automation*, **9**(5):624–637, 1993. 14

[101] Keyvan Hashtrudi-Zaad and Septimiu E. Salcudean. **Tranparency in Time-Delayed Systems and the Effect of Local Force Feedback for Transparent Teleoperation**. *IEEE Transactions on Robotics and Automation*, **18**(1):108–114, 2002. 14

[102] Jaime Rubi, Angel Rubio, and Alejo Avello. **Involving the Operator in a Singularity Avoidance Strategy for a Redundant Slave Manipulator in a Teleoperated Application**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2973 – 2978, 2002. 14

[103] Dal-Yeon Hwang and Blake Hannaford. **Teleoperation Performance with a Kinematically Redundant Slave Robot**. *The International Journal of Robotics Research*, **17**(6):578–597, 1998. 14

[104] Ulrich Hagn, Tobias Ortmaier, Rainer Konietschke, Bernhard Kübler, Ulrich Seibold, Andreas Tobergte, Mathias Nickl, Stefan Jörg, and Gerhard Hirzinger. **Telemanipulator for Remote Minimally Invasive Surgery**. *IEEE Robotics and Automation Magazine*, **15**(4):28–38, 2008. 14, 110, 155

[105] Peter Berkelman and Ji Ma. **A Compact Modular Teleoperated Robotic System for Laparoscopic Surgery**. *The International Journal of Robotics Research*, **28**(9):1197–1215, 2009. 14

[106] Blake Hannaford. **Stability and Performance Tradeoffs in Bi-Lateral Telemanipulation**. In *IEEE International Conference on Robotics and Automation*, pages 1764–1767, 1989. 14

[107] Carol E. Reiley, Henry C. Lin, David D. Yuh, and Gregory D. Hager. **A Review of Methods for Objective Surgical Skill Evaluation**. *Surgical Endoscopy*, **25**(2):356–366, 2011. 15

[108] Riccardo Muradore, Davide Bresolin, Luca Geretti, Paolo Fiorini, and Tiziano Villa. **Robotic Surgery: Formal Verification of Plans**. *IEEE Robotics and Automation Magazine*, **18**(3):24–32, 2011. 15

[109] Ilana Nisky, Michael H. Hsieh, and Allison M. Okamura. **A Framework for Analysis of Surgeon Arm Posture Variability in Robot-Assisted Surgery**. In *IEEE International Conference on Robotics and Automation*, pages 245–251, Karlsruhe, Germany, 2013. 15

[110] Franziska Zacharias, Ian S. Howard, Thomas Hulin, and Gerhard Hirzinger. **Workspace Comparisons of Setup Configurations for Human-Robot Interaction**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3117 – 3122, Taipei, Taiwan, 2010. 17

[111] Allison M. Okamura, Niels Smaby, and Mark R. Cutkosky. **An Overview of Dexterous Manipulation**. In *IEEE International Conference on Robotics and Automation*, pages 255 – 262, 2000. 17

[112] Antonio Bicchi. **Hands for Dexterous Manipulation and Robust Grasping: A Difficult Road Toward Simplicity**. *IEEE Transactions on Robotics and Automation*, **16**(6):652 – 662, 2000. 17

[113] Schunk. **Schunk GmbH und Co. KG: Service Robotics**, 2012. http://www.schunk-modularrobots.com/SCHUNKModularRobotics.html. 17, 18, 21

[114] Yoshiaki Sakagami, Ryujin Watanabe, Chiaki Aoyama, Shinichi Matsunaga, Nobuo Higaki, and Kikuo Fujimura. **The Intelligent ASIMO: Aystem Overview and Integration**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2478 – 2483, 2002. 18

[115] Honda. **American Honda Motor Co. Inc.: ASIMO Specifications**, 2012. http://asimo.honda.com/asimo-specs/ , Access: 11-02-2012. 18

[116] ottobock. **Otto Bock Healthcare Products GmbH: Michelangelo**, 2012. http://www.living-with-michelangelo.com/gb/home/ , Access: 11-02-2012. 18

[117] Markus Grebenstein, Alin Albu-Schäffer, Thomas Bahls, Maxime Chalon, Oliver Eiberger, Werner Friedl, Robin Gruber, Sami Haddadin, Ulrich Hagn, Robert Haslinger, Hannes Höppner, Stefan Jörg, Mathias Nickl, Alexander Nothhelfer, Florian Petit, Josef Reill, Nikolaus Seitz, Thomas Wimböck, Sebastian Wolf, Tilo Wüsthoff, and Gerhard Hirzinger. **The DLR Hand Arm System**. In *IEEE International Conference on Robotics and Automation*, pages 3175 – 3182, Shanghai, China, 2011. 18, 25

[118] Maxime Chalon, Markus Grebenstein, Thomas Wimböck, and Gerhard Hirzinger. **The Thumb: Guidelines for a Robotic Design**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5886 – 5893, Taipei, Taiwan, 2010. 18

[119] Hugo Gravato Marques, Michael Jäntsch, Steffen Wittmeier, Owen Holland, Cristiano Alessandro, Alan Diamond, Max Lungarella, and Rob Knight. **ECCE1: The First of a Series of Anthropomimetic Musculoskeletal Upper Torsos**. In *IEEE-RAS International Conference on Humanoid Robots*, pages 391 – 396, 2010. 18

[120] Aaron M. Dollar and Robert D. Howe. **Simple, Robust Autonomous Grasping in Unstructured Environments**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4693–4700, 2007. 18, 21

[121] Lael U. Odhner and Aaron M. Dollar. **Dexterous Manipulation with Underactuated Elastic Hands**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5254–5260, Shanghai, China, 2011. 18, 25

[122] L.U. Odhner, Raymond R. Ma, and Aaron M. Dollar. **Precision Grasping and Manipulation of Small Objects from Flat Surfaces using Underactuated Fingers**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2830–2835, Saint Paul, Minnesota, USA, 2012. 18

[123] C. Rodloff, Raphael Deimel, and Oliver Brock. **Adaptive Pneumatic Control for Soft Robotics**. Ongoing Diploma Thesis, 2012. http://www.robotics.tu-berlin.de/?id=118710 , Access: 11-02-2012. 18

[124] Oliver Brock. **Beyond Grasping - Back to Square One**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems: Workshop Beyond Grasping - Modern Approaches for Dynamic Manipulation*, Vilamoura, Algarve, Portugal, 2012. 18

[125] Raphael Deimel and Oliver Brock. **A Compliant Hand Based on a Novel Pneumatic Actuator**. In *IEEE International Conference on Robotics and Automation*, pages 2039–2045, Karlsruhe, Germany, 2013. 18

[126] Sing Bing Kang and Katsushi Ikeuchi. **Toward Automatic Robot Instruction from Perception-Mapping Human Grasps to Manipulator Grasps**. *IEEE Transactions on Robotics and Automation*, **13**(1):81 – 95, 1997. 19

[127] M. A. Arbib, T. Iberall, and D. M. Lyons. **Coordinated Control Programs for Movements of the Hand**. *Hand function and the neocortex, Experimental Brain Research Supplemental 10, A. W. Goodwin and I. Darian-Smith, Eds.*, 1985. 19

[128] Javier Romero, Hedvig Kjellström, and Danica Kragic. **Modeling and Evaluation of Human-to-Robot Mapping of Grasps**. In *IEEE International Conference on Advanced Robotics*, pages 1–6, 2009. 19

[129] Heni Ben Amor, Oliver Kroemer, Ulrich Hillenbrand, Gerhard Neumann, and Jan Peters. **Generalization of Human Grasping for Multi-Fingered Robot Hands**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, 2012. 19

[130] Raymond R. Ma and Aaron M. Dollar. **On Dexterity and Dexterous Manipulation**. In *15th International Conference on Advanced Robotics*, pages 1 – 7, 2011. 19

[131] Franziska Zacharias, Daniel Leidner, Florian Schmidt, Christoph Borst, and Gerhard Hirzinger. **Exploiting Structure in Two-Armed Manipulation Tasks for Humanoid Robots**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5446 – 5452, Taipei, Taiwan, 2010. 19

[132] David Fischinger and Markus Vincze. **Empty the Basket - A Shape Based Learning Approach for Grasping Piles of Unknown Objects**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2051 – 2057, Vilamoura, Algarve, Portugal, 2012. 20

[133] Lillian Chang, Joshua R. Smith, and Dieter Fox. **Interactive Singulation of Objects from a Pile**. In *IEEE International Conference on Robotics and Automation*, pages 3875 – 3882, Saint Paul, Minnesota, USA, 2012. 20

[134] Oliver Birbach, Udo Frese, and Berthold Bauml. **Realtime Perception for Catching a Flying Ball with a Mobile Humanoid**. In *IEEE International Conference on Robotics and Automation*, pages 5955 – 5962, Shanghai, China, 2011. 20

[135] T. Asfour, K. Regenstein, P. Azad, J. Schröder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann. **ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control**. In *IEEE-RAS International Conference on Humanoid Robots*, pages 169 – 175, 2006. 20

[136] Erico Guizzo and Evan Ackerman. **The Rise of the Robot Worker**. *IEEE Spectrum*, **49**(10):34–41, 2012. 20

[137] DLR. **Dexterous Robot Hands - Data sheet of DLR Hand II**, 2013. http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-3802/6102_read-8922/ , Access: 01-09-2013. 21

[138] David Silvera Tawil, David Rye, and Mari Velonaki. **Interpretation of the Modality of Touch on an Artificial Arm Covered with an EIT-Based Sensitive Skin**. *The International Journal of Robotics Research*, **31**(13):1627–1641, 2012. 21, 26

[139] DLR. **Dexterous Robot Hands - Future Work / Versions**, 2013. http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-3802/6102_read-8915/ , Access: 01-09-2013. 21, 26

[140] Mario Prats, Steven Wieland, Tamim Asfour, Angel P. del Pobil, and Rüdiger Dillmann. **Compliant Interaction in Household Environments by the Armar-III Humanoid Robot**. In *IEEE-RAS International Conference on Humanoid Robots*, pages 475 – 480, 2008. 21

[141] Tatsuya Nomura, Tomohiro Suzuki, Takayuki Kanda, and Kensuke Kato. **Measurement of Anxiety toward Robots**. In *15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 372 – 377, 2006. 21

[142] Lars Oestreicher. **Cognitive, Social, Sociable or just Socially Acceptable Robots?** In *16th IEEE International Symposium on Robot and Human interactive Communication*, pages 558 – 563, 2007. 21

[143] Satoshi Koizumi, Takayuki Kanda, Masahiro Shiomi, Hiroshi Ishiguro, and Norihiro Hagita. **Preliminary Field Trial for Teleoperated Communication Robots**. In *15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 145 – 150, 2006. 21

[144] Neta Ezer, Arthur D. Fisk, and Wendy A. Rogers. **Attitudinal and Intentional Acceptance of Domestic Robots by Younger and Older Adults**. *Universal Access in Human-Computer Interaction, Intelligent and Ubiquitous Interaction Environments, Lecture Notes in Computer Science*, **5615**:39–48, 2009. 21

[145] Tatsuya Nomura, Takayuki Kanda, Tomohiro Suzuki, Jeonghye Han, Namin Shin, Jennifer Burke, and Kensuke Kato. **Implications on Humanoid Robots in Pedagogical Applications from Cross-Cultural Analysis between Japan, Korea, and the USA**. In *16th IEEE International Symposium on Robot and Human Interactive Communication*, pages 1051 – 1057, 2007. 21

[146] Hiroko Kamide, Yasushi Mae, Koji Kawabe, Satoshi Shigemi, and Tatsuo Arai. **Effect of Human Attributes and Type of Robots on Psychological Evaluation of Humanoids**. In *IEEE International Workshop on Advanced Robotics and its Social Impacts*, pages 40 – 45, 2012. 21

[147] Cynthia Breazeal. **Social Interactions in HRI: The Robot View**. *IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews*, **34**(2):181 – 186, may 2004. 21

[148] Takayuki Kanda, Takahiro Miyashita, Taku Osada, Yuji Haikawa, and Hiroshi Ishiguro. **Analysis of Humanoid Appearances in Human-Robot Interaction**. *IEEE Transactions on Robotics*, **24**(3):725 – 735, 2008. 21

[149] Hiroko Kamide, Yasushi Mae, Koji Kawabe, Satoshi Shigemi, and Tatsuo Arai. **A Psychological Scale for General Impressions of Humanoids**. In *IEEE International Conference on Robotics and Automation*, pages 4030 – 4037, Saint Paul, Minnesota, USA, 2012. 22

[150] Aelee Kim, Hyejin Kum, Ounjung Roh, Sangseok You, and Sukhan Lee. **Robot Gesture and User Acceptance of Information in Human-Robot Interaction**. In *7th ACM/IEEE International Conference on Human-Robot Interaction*, pages 279 – 280, 2012. 22

[151] G. Veruggio and F. Operto. *Handbook of Robotics, Part G: Human-Centered and Life-Like Robotics, Chapter 64 Roboethics: Social and Ethnical Implications of Robotics.* Springer Verlag Berlin Heidelberg, 2008. 22

[152] Reg G. Grant. *Flight - 100 Years of Aviation. German Translation by B. Schäfer ad T. Kriele: Fliegen - Die Geschichte der Luftfahrt.* Dorling Kindersley Limited, Dorling Kindersley Verlag GmbH, 2002, 2003. 22

[153] G. Bossow. *Die Geschichte der Luftfahrt.* GeraMond Verlag GmbH, 2009. 22

[154] Otto-Lilienthal-Museum-Anklam. **Time line of aviation**, 2012. http://www.lilienthal-museum.de/olma/e5.htm , Access: 11-14-2012. 22

[155] Bernd Lukasch. **To fly like a bird**, 2012. http://www.lilienthal-museum.de/olma/esoest.htm , Access: 11-14-2012. 22

[156] Bernd Lukasch. **From Lilienthal to the Wrights**, 2012. http://www.lilienthal-museum.de/olma/ewright.htm , Access: 11-14-2012. 22

[157] Alexis v. Croy. *Was Sie über das Fliegen wissen sollten.* Herbig Verlagsbuchhandlung GmbH, 2010. 22

[158] Dawna L. Rhoades. *Evolution of International Aviation.* Ashgate Publishing Limited, Ashgate Publishing Company, 2008 (2nd ed.). 22

[159] Karlhans Müller. *Das komplete Buch vom Fliegen.* Wolfgang Krüger Verlag, S. Fischer Verlag GmbH, 1981. 22

[160] Heni Ben Amor, Ashutosh Saxena, Oliver Kroemer, and Jan Peters. **Workshop Beyond Grasping - Modern Approaches for Dynamic Manipulation**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, 2012. 25

[161] IEEE/RSJ. **IEEE/RSJ International Conference on Intelligent Robots and Systems**. Vilamoura, Algarve, Portugal, 2012. 25

[162] Grodon Cheng, Etienne Burdet, Ravinder S. Dahiya, and Philipp Mittendorfer. **Research Frontiers in Electronic Skin Technology: Multi-functional Bendable and stretchable electronic skin for robots and beyond**. In *IEEE International Conference on Robotics and Automation*, 2013. 26

[163] Markus Grebenstein, Maxime Chalon, Werner Friedl, Sami Haddadin, Thomas Wimböck, Gerhard Hirzinger, and Roland Siegwart. **The Hand of the DLR Hand Arm System: Designed for Interaction**. *The International Journal of Robotics Research*, **31**(13):1531–1555, 2012. 26

[164] J. Denavit and R. S. Hartenberg. **A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices**. *Journal of Applied Mechanics*, pages 215–221, 1955. 34, 68, 75, 110, 185

[165] Guangqi Ye, Jason Corso, Darius Burschka, and Gregory D. Hager. **VICs: A Modular Vision-Based HCI Framework**. In *Proceedings of 3rd International Conference on Computer Vision Systems*, pages 257–267, 2003. 44

[166] Darius Burschka and Gregory Hager. **Scene Classification from Dense Disparity Maps in Indoor Environments**. In *16th International Conference on Pattern Recognition. Proceedings*, 2002. 44

[167] Darius Burschka and Gregory D. Hager. **Vision-Based 3D Scene Analysis for Driver Assistance**. In *IEEE International Conference on Robotics and Automation*, 2005. 44, 45

[168] I. Oikonomidis, N. Kyriazis, and A.A. Argyros. **Markerless and Efficient 26-DOF Hand Pose Recovery**. In *Asian Conference on Computer Vision*, pages 2926–2931, Queenstown, New Zealand, 2010. 46

[169] Darius Burschka and Gregory Hager. **V-GPS – Image-Based Control for 3D Guidance Systems**. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1789–1795, October 2003. 47, 48, 98, 102

[170] B. K. P. Horn. *Robot Vision*. MIT Press, 1986. 48

[171] Juan Carlos Ramirez and Darius Burschka. **Framework for Consistent Maintenance of Geometric Data and Abstract Task-Knowledge from Range Observations**. In *IEEE International Conference on Robotics and Biomimetics*, pages 963 – 969, 2011. 50

[172] Thomas Feix, Roland Pawlik, Heinz-Bodo Schmiedmayer, Javier Romero, and Danica Kragic. **The Generation of a Comprehensive Grasp Taxonomy**. In *Robotics, Science and Systems Conference: Workshop on Understanding the Human Hand for Advancing Robotic Manipulation, Poster Presentation*, June 2009. 53, 109

[173] M. Kawato. **Trajectory Formation in Arm Movements: Minimization Principles and Procedures**. In *Advances in Motor Learning and Control, ser. Human Kinetics, H. N. Zelaznik, Ed. Human Kinetics Publishers, Chanpaign Illinois*, pages 225–259, 1996. 55

[174] J.A. Hartigan and M.A. Wong. **A k-means Clustering Algorithm**. *Applied Statistics*, **8**(1):100–108, 1979. 55

[175] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 55, 56

[176] Tamar Flash and Neville Hogan. **The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model**. *The Journal of Neuroscience*, **5**(7):1688–1703, july 1985. 84

[177] Y. Uno, M. Kawato, and R. Suzuki. **Formation and Control of Optimal Trajectory in Human Multijoint Arm Movements**. *Biological Cybernetics*, **61**:89–101, 1989. 84

[178] Jeffrey Dean and Michael Brüwer. **Control of human arm movements in two dimensions: paths and joint control in avoiding simple linear obstacles**. *Experimental Brain Research*, **97**:497–514, 1994. 84

[179] W.L. Nelson. **Physical Principles for Economies of Skilled Movements**. *Biological Cybernetics*, **46**:135–147, 1983. 84

[180] Armin Biess, Mark Nagurka, and Tamar Flash. **Simulating discrete and rhythmic multi-joint human arm movements by optimization of nonlinear performance indices**. *Biological Cybernetics*, **95**:31–53, 2006. 84

[181] Emanuele Lindo Secco, Luca Valandro, Roberto Caimmi, Giovanni Magenes, and Benedetto Salvato. **Optimization of two-joint arm movements: a model technique or a result of natural selection?** *Biological Cybernetics*, **93**:288–306, 2005. 84

[182] Jur van den Berg, Stephen Miller, Daniel Duckworth, Humphrey Hu, Andrew Wan, Xiao-Yu Fu, Ken Goldberg, and Pieter Abbeel. **Superhuman Performance of Surgical Tasks by Robots using Iterative Learning from Human-Guided Demonstrations**. In *IEEE International Conference on Robotics and Automation*, pages 2074–2081, Anchorage, Alaska, USA, 2010. 84

[183] Evan A. Suma, Belinda Lange, Albert Rizzo, David Krum, and Mark Bolas. **Flexible Action and Articulated Skeleton Toolkit (FAAST)**. In *Proceedings of IEEE Virtual Reality Conference*, pages 247–248, 2011. 87

[184] ROS (Robot Operating System). **ROS Wiki**, 2013. http://www.ros.org/wiki/ , Access: 04-08-2013. 87

[185] ROS (Robot Operating System). **MIT Kinect Demos**, 2012. http://www.ros.org/wiki/mit-ros-pkg/KinectDemos , Access: 04-08-2013. 87

[186] Tadashi Kashima, Keisuke Yanagihara, and Masao Iwaseya. **Trajectory formation based on a human arm model with redundancy**. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 959 – 963, 2012. 87

[187] Lee Jaemin, Hironori Takimoto, Hitoshi Yamauchi, Akihiro Kanazawa, and Yasue Mitsukura. **A Robust Gesture Recognition Based on Depth Data**. In *19th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, pages 127 – 132, 2013. 87

[188] Hyunchul Kim, Zhi Li, Dejan Mulitinovic, and Jacob Rosen. **Resolving the Redundancy of a Seven DOF Wearable Robotic System Based on Kinematic and Dynamic Constraint**. In *IEEE International Conference on Robotics and Automation*, pages 305–310, Saint Paul, Minnesota, USA, 2012. 87

[189] Susanne Petsch. *Representation of Manipulation-Relevant Object Properties and Actions*. Master's thesis, Technische Universität München, 2011. Unpublished. 89

[190] Chavdar Papazov and Darius Burschka. **Stochastic Optimization for Rigid Point Set Registration**. In *In Proceedings of the 5th International Symposium on Visual Computing (ISVC'09)*, **5875**, pages 1043–1054. Lecture Notes in Computer Science, Springer Verlag, December 2009. 98

[191] Ulrich Seibold, Bernhard Kübler, and Gerhard Hirzinger. **Prototype of Instrument for Minimally Invasive Surgery with 6-Axis Force Sensing Capability**. In *IEEE International Conference on Robotics and Automation*, pages 496–501, 2005. 110, 155

[192] Sophie Thielmann, Ulrich Seibold, Robert Haslinger, Georg Passig, Thomas Bahls, Stefan Jörg, Mathias Nickl, Alexander Nothhelfer, Ulrich Hagn, and Gerhard Hirzinger. **MICA - A New Generation of Versatile Instruments in Robotic Surgery**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 871–878, Taipei, Taiwan, 2010. 110, 155

[193] Louis B. Rosenberg. **Virtual Fixtures: Perceptual Tools for Telerobotic Manipulation**. In *IEEE Virtual Reality Annual International Symposium*, pages 76–82, 1993. 119

[194] Ankur Kapoor, Ming Li, and Russell H. Taylor. **Constrained Control for Surgical Assistant Robots**. In *IEEE International Conference on Robotics and Automation*, pages 231–236, 2006. 119

[195] Oliver Ruepp. *Recovery of Structure and Motion from Monocular Images under Poor Lighting and Texture Conditions*. PhD thesis, Technische Universität München, 2012. 121

[196] Sébastien Grange, François Conti, Patrice Roullier, Patrick Helmer, and Charles Baur. **The Delta Haptic Device**. *Mechatronics 2001*, 2001. 155, 160

[197] Andreas Tobergte, Patrick Helmer, Ulrich Hagn, Patrice Roullier, Sophie Thielmann, Sébastien Grange, Alin Albu-Schäffer, François Conti, and Gerhard Hirzinger. **The sigma.7 Haptic Interface for MiroSurge: A New Bi-Manual Surgical Console**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2445–2451, San Francisco, California, USA, 2011. 155

[198] Andreas Tobergte, Rainer Konietschke, and Gerhard Hirzinger. **Planning and Control of a Teleoperation System for Research in Minimally Invasive Robotic Surgery**. In *IEEE International Conference on Robotics and Automation*, pages 4225–4232, 2009. 171

[199] Yoshio Yamamoto and Xiaoping Yun. **Coordinating Locomotion and Manipulation of a Mobile Manipulator**. *IEEE Transactions on Automated Control*, **39**(6):1326–1332, 1994. 172

[200] Yoshio Yamamoto and Xiaoping Yun. **Effect of Dynamic Interaction on Coordinated Control of Mobile Manipulation**. *IEEE Transactions on Robotics and Automation*, **12**(5):816–824, 1996. 172

[201] Homayoun Seraji. **A Unified Approach of Motion Control of Mobile Manipulators**. *The International Journal of Robotics Research*, **17**(2):106–118, 1998. 172

[202] K. Nagatani, T. Hirayama, A. Gofuku, and Y. Tanaka. **Motion Planning for Mobile Manipulator with Keeping Manipulability**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1663 – 1888, 2002. 172

[203] B. Bayle, J.-Y. Fourquet, and M. Renaud. **Manipulability Analysis for Mobile Manipulators**. In *IEEE International Conference on Robotics and Automation*, pages 1251–1256, 2001. 172

[204] T. Yoshikawa. **Analysis and Control of Robot Manipulators with Redundancy**. *Robotics Research: The First International Symposium*, 1984. 172

[205] Yoshio Yamamoto, Hiroshi Eda, and Xiaoping Yun. **Coordinated Task Execution of a Human and a Mobile Manipulator**. In *IEEE International Conference on Robotics and Automation*, pages 1006–1011, 1996. 172

[206] Thomas Sugar and Vijay Kumar. **Decentralized Control of Cooperating Mobile Manipulators**. In *IEEE International Conference on Robotics and Automation*, pages 2916–2921, 1998. 173

[207] Andres Ubeda, Eduardo Ianez, Javier Badesa, Ricardo Morales, Jose M. Azorin, and Nicolas Garcia. **Control Strategies of an Assistive Robot Using a Brain-Machine Interface**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3553 – 3558, Vilamoura, Algarve, Portugal, 2012. 173

[208] L. R. Hochberg, D. Bacher, B. Jarosiewicz, N. Y. Masse, J. D. Simeral, J. Vogel, S. Haddadin, J. Liu, S. S. Cash, P. van der Smagt, and J. P. Donoghue. **Reach and Grasp by People with Tetraplegia Using a Neurally Controlled Robotic Arm**. *Nature*, **485**:372–377, 2012. 173

[209] Arash Ajoudani, Nikos Tsagarakis, and Antonio Bicchi. **Tele-Impedance: Teleoperation with Impedance Regulation Using a Body-Machine Interface**. *The International Journal of Robotics Research*, **31**(13):1642–1656, 2012. 173