On-the-Fly Workspace Visualization for Redundant Manipulators

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Mirko Kunze

aus Berlin

Datum der mündlichen Prüfung:	19. Juli 2016
Erster Gutachter:	Prof. Dr.-Ing. Dr. h.c. Heinz Wörn
Zweiter Gutachter:	Prof. Dr.-Ing. Rüdiger Dillmann

Once we accept our limits, we go beyond them.

Albert Einstein

Acknowledgements

This thesis would not have been possible without the support and the influence of numerous people.

First and foremost, I express my gratitude to Prof. Dr.-Ing. Dr. h.c. Heinz Wörn for providing me with the opportunity to pursue this work and for supervising this thesis. I also want to thank Prof. Dr.-Ing. Rüdiger Dillmann for agreeing to be second reviewer of this thesis. Furthermore, I want to thank Dr. rer. nat. Jörg Raczkowsky, who leads the medical research group MeGI at the IAR-IPR, for his support throughout my time at the institute and for always having our back.

A big Thank You for your time and your feedback goes to all participants in the user study: Jan, Luzie, Dr. Jens, Dave, Nils, Jacky, Sam, David PJ, Dr. Andi, Jazz, Stephan, Christian, Dr. Jules, Prof. Dr.-Ing. habil. Björn Hein and Dr. (RUS) Ilshat.

For thorough proofreading, countless discussions and general support I send big props and kudos to Dr.-Ing. Peter Loepelmann, Dr.-Ing. Tim Beyl and especially Philip Nicolai, who was my brother in arms during the diss time.

Without a healthy work-procrastination balance, a project as big as a PhD thesis is hardly manageable. Therefore, I want to thank all my colleagues for the enjoyable atmosphere at the institute. I will particularly miss the traditional roundly after lunch.

I especially thank all the friends I made during my time in Karlsruhe for enriching my life and keeping me sane. Dr. Adrian, Dr. (RUS) Ilshat, Dr. Mirko DC, Escalez, Dr. Sibby and especially Dave deserve extra special mentioning here for bonus-close friendship.

I want to give credit also to (back then not yet Prof. Dr.-Ing.) Sami Haddadin, who supervised my bachelor's thesis at DLR in 2008, which was when and why I decided to pursue a career in robotics. And I wouldn't have decided to apply at the KIT if it wasn't for Valeria.

The place of honor is dedicated to my loving and caring parents Angelika and Marco and my Bro Oliver for supporting every step of my life and always making me feel home at home. Thanks also to Sophie and Roland who have become part of the family and care for Oli and mom so I could set out and play with robots.

Abstract

Within this thesis, different concepts for on-the-fly visualization of workspace boundaries for 7-DoF manipulators have been investigated. It is explored, which possibilities parallelization and the utilization of modern graphics cards can bring to workspace visualization through on-line computation of positioning limits for the current situation. The higher goal is to employ intuitive visualization as a substitute for the natural human feeling for reachability that humans possess for their own arms.

The workspace is considered to be the set of poses that can be accessed by the TCP, taking into account robot joint limits, link lengths and singularities. The novel distinction between the directly and the generally accessible workspace is made: The directly accessible workspace contains all points that are reachable for the tool center point from the current position in a linear motion or a rotation about a constant axis without reconfiguration. The generally accessible workspace contains all reachable poses, even if a reconfiguration is necessary to reach them.

The generally accessible workspace is obtained using rasterization of the Cartesian space and assessing the reachability of every node by analytical inverse kinematics computation. The directly accessible workspace is obtained via simultaneous virtual robot motion in all directions.

In order to reduce the complexity of this six-dimensional problem, it is split into the consideration of translation limits while keeping the orientation constant and rotation limits while keeping the TCP position constant.

Different visualization concepts are devised, based on the considerations above. The motion boundaries for the current robot pose are displayed as semi transparent barriers within a virtual environment.

All investigated concepts are exemplarily implemented for the DLR/KUKA LBR IV manipulator and consider its redundant kinematic structure using a novel cost function concept. The usability and perspicuity of the devised visualization concepts have been assessed in a user study. While the translational components received *good* to *excellent* ratings, the rotational components scored average but still proved to be effective in the tested scenario.

Possible applications for the proposed concepts are visual support for the manual operation of manipulators, fast workspace analyses in time-critical scenarios, assistance for robot or target placement and repositioning, interactive workspace exploration for design and comparison of robots and tools and determination of the motion tolerance around a trajectory.

Zusammenfassung

Die vorliegende Arbeit stellt verschiedene Konzepte zur Visualisierung des Arbeitsraumes von Roboterarmen mit sieben Freiheitsgraden zur Laufzeit vor.

Motivation

Wird ein Roboterarm manuell verfahren (im Gelenkraum, im Kartesischen Raum, durch Handkontakt oder Teleoperation), gelangt man häufig in Gelenkanschläge, Singularitäten oder anderweitig ungünstige Konfigurationen. Dies ist selbst für erfahrene Benutzer schwer zu erkennen, beziehungsweise zu vermeiden.

Im Rahmen dieser Arbeit wird untersucht, welche Möglichkeiten sich durch Parallelisierung und Verwendung moderner Grafikkarten für die Arbeitsraumanalyse eröffnen. Es werden Verfahren entwickelt, welche es ermöglichen, situationsspezifische Arbeitsraumanalysen zur Laufzeit durchzuführen.

Das höhere Ziel ist, durch visuelles Feedback das natürliche Gefühl über die Positionierfähigkeiten zu substituieren, das ein Mensch für den eigenen Arm hat.

Methoden

Alle Analysen können auf beliebige kinematische Strukturen angewendet werden und wurden im Rahmen dieser Arbeit am Beispiel des KUKA/DLR Leichtbauroboters IV realisiert. Die betrachtete Struktur weist sieben rotatorische Gelenke auf und ist somit kinematisch redundant, wodurch eine höhere Flexibilität erreicht wird.

Grundsätzlich wird der Arbeitsraum als Menge aller Posen betrachtet, die durch den Werkzeugmittelpunkt (auch TCP) unter Berücksichtigung der Werkzeugorientierung erreichbar sind. Hierbei werden Gelenkanschläge, Singularitäten und die Länge des Roboters berücksichtigt.

Zunächst wird die Unterscheidung zwischen prinzipiell erreichbarem und direkt erreichbarem Arbeitsraum eingeführt. Beim prinzipiell erreichbaren Arbeitsraum handelt es sich um die Menge aller Posen, die ein Manipulator mit dem TCP erreichen kann, auch wenn hierzu eine Rekonfiguration nötig ist. Der direkt erreichbare Arbeitsraum beschreibt hingegen die Menge aller Posen, die in einer direkten Bewegung (lineare Translation oder Rotation um eine feste Achse durch den TCP) aus der aktuellen Pose heraus erreichbar sind.

Zusätzlich wird zwischen translatorischem Arbeitsraum bei fester TCP-Orientierung und rotatorischem Arbeitsraum bei fester TCP-Position unterschieden, da die Visualisierung einer Grenzfläche im sechsdimensionalen Raum nur durch Dimensionsreduktion erreicht werden kann.

Für jede der vier resultierenden Kategorien – prinzipiell/direkt erreichbar bzw. rotatorische/translatorische Bewegungsgrenze – werden Visualisierungskonzepte entwickelt und vorgestellt. Zusätzlich wird ein Konzept für die Darstellung des prinzipiell erreichbaren Arbeitsraums für Translation vorgestellt, wobei jeder Punkt innerhalb des Arbeitsraumes in jeder Orientierung einer zuvor festgelegten Menge an Orientierungen erreichbar ist.

Die Berechnung der Arbeitsraumgrenzen erfolgt durch Lösung der inversen Kinematik parallelisiert auf der Grafikkarte. Im Falle der prinzipiell erreichbaren Arbeitsräume wird der Kartesische Raum in einem Gitter gerastert. An jedem Knoten wird die Erreichbarkeit analytisch geprüft, wobei die Redundanz des Roboters berücksichtigt wird. Anschließend wird mittels des Marching Cubes Algorithmus' die Hülle um alle erreichbaren Punkte gelegt.

Zur Berechnung des direkt erreichbaren Arbeitsraumes verfährt der Roboter im Falle der Translation den TCP virtuell in eine gerasterte Untermenge aller Raumrichtungen bis ein Gelenk anschlägt oder eine Singularität erreicht wird. Dort wird die Grenze gerendert. Im Falle der Rotation wird um eine Menge an Achsen gekippt, die senkrecht zur dominanten Werkzeugachse stehen und die Grenze wird auf eine Kugel um den TCP gezeichnet. Während der virtuellen Fahrten wird der Nullraumparameter des Roboters lokal optimiert, wofür ein neues Konzept zur Kombination verschiedener Kostenfunktionen entwickelt wurde.

Sämtliche im Rahmen dieser Arbeit entwickelten Konzepte wurden in C++ und OpenCL implementiert. Die Anbindung an den Roboter, an die Visualisierung und an die Eingabegeräte ist über das verbreitete Robot Operating System (ROS) umgesetzt. Dies ermöglicht die einfache Integration der entwickelten Visualisierungen in eine Vielzahl von Forschungsplattfomen, die ebenfalls auf ROS basieren. Die Darstellung selbst erfolgt über RViz, einem

Visualisierungsprogramm für die ROS-Umgebung. Zur Nutzereingabe sind ein Gamepad und eine 6D-Maus angebunden.

Ergebnisse

Alle Konzepte wurden für die Darstellung zur Laufzeit realisiert und erreichten in Tests bei angemessener Auflösung mindestens eine Bildfrequenz von 8 Hz.

Als Anwendungsszenarien für die entwickelten Visualisierungen dienen unter anderem die schnelle Positionierung des Roboters für verschiedene Anwendungen, die Evaluation der translatorischen bzw. rotatorischen Toleranz um eine geplante Trajektorie und die Untersuchung der Arbeitsraumform bei Variation verschiedener Konstruktionsparameter.

In einer Nutzerstudie wurden Unterstützung, Verständlichkeit und Benutzbarkeit der verschiedenen Konzepte evaluiert, wobei die Konzepte zur Visualisierung der translatorischen Erreichbarkeit durchweg *gut* bis *exzellent* bewertet wurden. Die Visualisierung zur direkten Erreichbarkeit für Rotation wurde mit großer Streuung im Mittel durchschnittlich bewertet, erwies sich jedoch trotzdem als effizient für die gegebene Aufgabe.

Diskussion, Fazit und Ausblick

Einen Mittelweg zwischen einer großen Menge an erreichbaren Orientierungen und einem großen erreichbaren Raum zu finden, den Zusammenhang zwischen Werkzeugform und -orientierung und dem resultierenden Arbeitsraum zu verstehen und letztendlich ein Gefühl für die Erreichbarkeit eines Manipulators zu entwickeln sind die Hauptanwendungen für die vorliegende Arbeit. Die Positionierfähigkeiten eines Manipulators werden durch visuelle Exploration greifbar gemacht. Der Zweck der Visualisierung liegt hierbei in weniger spezifischen bzw. planbaren Szenarien, die vorgestellten Visualisierungswerkzeuge wurden nicht entwickelt, um Optimierungsaufgaben wie Roboter- und Werkzeugkonstruktion, Positionierung und Pfadplanung durch Visualisierung an den Nutzer auszulagern.

Mögliche zukünftige Schritte umfassen die Generalisierung des Ansatzes durch Einbindung weiterer kinematischer Strukturen, das Einblenden von Arbeitsraumgrenzen in Kamerabildern oder auf Virtual Reality Brillen sowie die Berücksichtigung von Kollisionen bei der Erreichbarkeitsanalyse.

Contents

1 Introduction

Humans possess a natural feeling for the capabilities of their arms. Based on experience, they have an intuitive understanding for whether a point in space can be reached and if so, from which angles it can be accessed. They can also predict if the posture that the arm needs to be in, in order to reach a particular target, is comfortable or not.

If a person does not control their own arm but a robotic manipulator, this feeling is lost. Robots often have a joint configuration different from that of a human arm or just different joint limits. For instance, most robot "elbows" (if a joint can be identified acting as an elbow) can deflect in both directions from a straight position. As a consequence, it is hard to assess the reachability of target poses, and to determine a trajectory that leads there can be cumbersome.

In order to provide a substitute for the lack of a natural understanding for the capabilities of a robotic arm, different workspace visualization techniques and measures for dexterity were developed over several decades. However, determining a robot's workspace is generally a very complex and computationally expensive problem, which is so far tackled via simplification and reduction of dimensionality.

1.1 Motivation

Often, demanding problems can be solved by providing enough computation time. In many areas however, it is only due to the performance increase of hardware that paradigm-shifting possibilities were introduced: World-wide live video conferences are common, computer tomography can be used intraoperatively, virtual reality headsets do no longer induce motion sickness. Being able to explore, interact and play with a virtual object or scene in real-time conveys a much more vivid and graspable experience than rendered still images.

Since the advent of General Purpose Computation on Graphics Processing Units (GPGPU), many parallelizable problems can be computed much faster so that a solution is obtained without mentionable latency, even on standard home computers.

The primary aim of this thesis is to investigate and explore the possibilities that this technology can bring to workspace visualization techniques.

This will be attempted through the concept of massively parallel virtual exploration and visualization. Potential applications are visual support for tele-operation, robot and tool design and analysis, robot placement and groundwork for further optimization algorithms. The higher goal will be to allow the user to develop a feeling for the reaching capabilities of a robotic manipulator.

The idea is that the development of an intuitive understanding leads to more dexterous and intelligent handling in difficult situations as well as the ability to foresee and avoid them. Being able to grasp a scenario increases confidence and security. Furthermore, the comprehension of complex relations is a key concern in research and development.

1.2 Outline

Chapter 2 will first provide the reader with background knowledge necessary for understanding the concepts presented within this thesis. The robotic structure that serves as example as well as kinematic considerations is then introduced. Afterwards, existing workspace analysis approaches as well as dexterity measures are presented. The contribution of this thesis to the state of the art is then pointed out.

In chapter 3, methods for solving the inverse kinematics problem are presented, which form the basis for this work. Afterwards, the main concepts that were developed within the scope of this thesis are put forward. The distinction between generally and directly accessible workspace as well as visualization techniques for translatory and rotatory motion barriers is made. In the end, an approach for dealing with the robot's redundancy is presented.

Chapter 4 discusses the implementation of the visualization concepts. The system architecture as well as the utilized software and hardware are introduced and implementation details of the developed algorithms for computing the different workspace limits are given. The user interface is then put forward.

Potential application scenarios of the devised concepts are demonstrated in chapter 5.

Within chapter 6, the results of this thesis are presented. Different factors that influence the system performance are introduced. In a user study, it is investigated how intuitively the visualization concepts can be understood.

Chapter 7 discusses the presented visualization tools in a broader perspective and the results of the user study are interpreted. A few learned lessons are addressed. The presented work is compared to the closest works of other research groups and the differences are pointed out.

Lastly, in chapter 8, a conclusion is drawn and further research directions based on the proposed system are given.

2 Fundamentals & State of the Art

The concept of workspace analysis for robotic manipulators is as old as robotic manipulators themselves. In this chapter, a fundamental background is laid out and previous and related works are analyzed and discussed. In the first section, the kinematic structure that this thesis focuses on is presented. Afterwards, different concepts of workspace and reachability analysis and measures of manipulability and dexterity are put forward. Finally, the aim of this work and its contrast to related works are pointed out.

2.1 Redundant Robotic Manipulators

In order to freely position and orient an object in three-dimensional space, a robotic manipulator needs to have at least six degrees of freedom (DoF) in the form of joints. One of the most common structures for serial robotic manipulators is the anthropomorphic arm with six rotatory joints. Among those robots, most are wrist-partitioned, i.e. their last three joint axes intersect in a single point. This property is very helpful from a control point of view: Position and orientation can be regarded as decoupled, which facilitates many simplifications both for the inverse kinematics problem as well as for many workspace analysis methods.

Many newer robotic arms feature a seventh joint. The inverse kinematics problem becomes under-determined and the manipulators are kinematically redundant, which enables them to perform a self-motion while the end-effector does not change its pose. This facilitates a better avoidance of joint limits, singularities and obstacles and enhances the dexterity in general [30, 40]. In 1985, Hollerbach investigated several joint configurations for robotic arms in terms of singularity elimination, workspace optimization, kinematic simplicity and mechanical constructability [40]. The conclusion was, that a manipulator with a kinematic structure as depicted in Figure 2.1 best satisfies the criteria on balance. Figure 2.1 also shows the joint labeling that is used throughout this thesis.

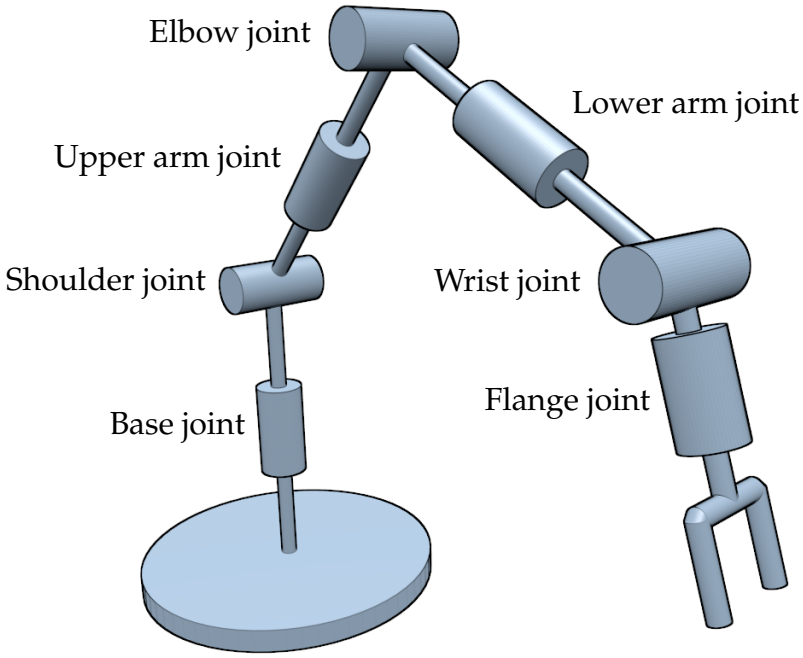

Figure 2.1: Optimal joint configuration of a 7-DoF robotic manipulator according to [40]. The cylinders represent rotatory joints. The joint nomenclature used in this thesis is shown.

Although most of the concepts presented in this thesis can be extended to arbitrary kinematic designs, the focus is laid on the mentioned structure, which is implemented in current robots such as the KUKA/DLR LBR IV [83], the Schunk LWA 4D [85] and the KUKA LBR iiwa [84] (see Figure 2.2). The KUKA/DLR LBR IV will serve as an example in this thesis.

Redundant robots are more flexible (as will be shown in section 5.4), but they are also more complicated to control. Some specific concepts will be covered in the following.

Having one more joint than necessary means that the desired end-effector or TCP (Tool Center Point) pose does not unambiguously define all joint angles. Two more parameters need to be specified: the null-space parameter and the configuration index.

[32] [44] [32]

Figure 2.2: From left to right: KUKA/DLR LBR IV, Schunk LWA 4D, KUKA LBR iiwa

2.1.1 Null-Space Parameter

When fixing the end-effector pose, a robot with the given kinematic structure can still rotate its elbow around the connecting line between shoulder and wrist, similar to the human arm.

In order to illustrate this self-motion, the robot structure can be simplified. Since both the first three and the last three joint axes intersect in one point, they can be substituted by spherical joints, as depicted in Figure 2.3. Note that this simplification does not allow to represent the original joint limits.

This additional null-space parameter will be specified by the angle η as shown in Figure 2.3. The reference elbow position (i.e. $\eta = 0$) is upwards.

2.1.2 Configuration Index

Even with the null-space parameter specified, the inverse kinematics problem still does not have a unique solution. The hinge joints (shoulder, elbow and wrist) can deflect in two directions and each of the 2^3 resulting configurations allows for a mathematical solution. This distinction will henceforth be referred to as configuration index. Figure 2.4 shows an LBR IV in eight different configurations for the same TCP pose and the same null-space parameter.

A change of configuration during operation is usually not performed since it requires driving through a singularity, which is an undesirable situation, as will be explained in section 2.1.3. Within this thesis, the configuration can be changed only in joint space by moving joints individually. Cartesian motion will not change the configuration index and all analysis is done for the current configuration only. An extension to check the other configurations as well would be trivial to add if required.

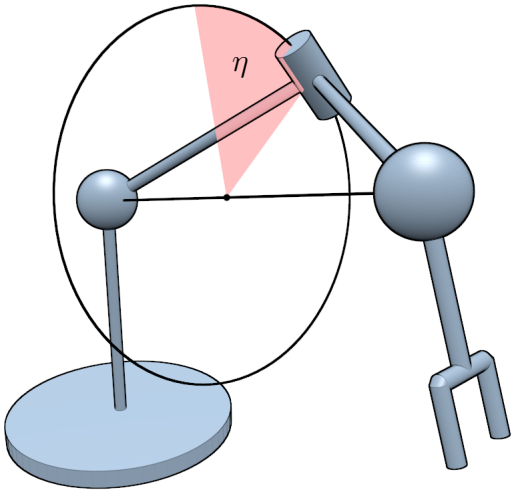

Figure 2.3: Simplification of the robot structure. The three first and the three last joints are replaced by spherical joints. The self-motion capability can be seen: The elbow (cylindrical joint) can rotate on a circle about an axis from shoulder to wrist. This motion is characterized by the null-space parameter η (red angle).

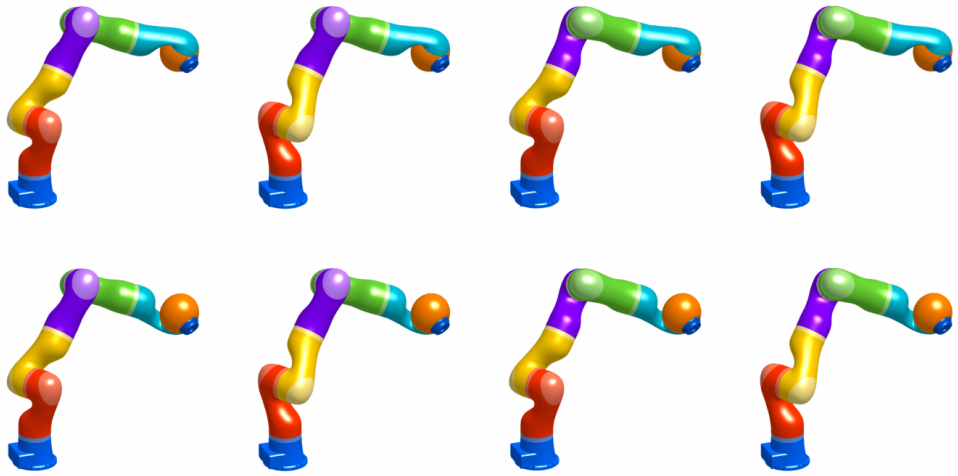

Figure 2.4: LBR IV in eight different configurations, colored for easier distinction. Flange pose and null-space parameter are identical for all postures.

2.1.3 The Jacobian Matrix and Singularities

The Cartesian pose of the TCP can be specified by a six-dimensional vector $\vec{p}_{\text{tcp}} = (x, y, z, \alpha, \beta, \gamma)^T$, where x, y and z are Cartesian coordinates defining the position and α, β and γ are angles (in an arbitrary convention) that describe the orientation of the TCP pose. The vector $\vec{p}_{\text{tcp}}$ can be written as a function of the joint angles $\vec{\theta} = (\theta_{1..j})^T$ (where j is the number of joints) and the Jacobian matrix describes the derivative of $\vec{p}_{\text{tcp}}$ with respect to $\vec{\theta}$:

$$\vec{p}_{\text{tcp}} = f(\vec{\theta}) \tag{2.1}$$

$$\mathrm{d}\vec{p}_{\text{tcp}} = J(\vec{\theta})\mathrm{d}\vec{\theta} \tag{2.2}$$

Singularities are situations where at least one direction of end-effector movement is blocked due to an unfavorable manipulator posture. This can occur when two joint axes align and the corresponding joint movements have the same effect on the TCP pose, so – very simplified – their flexibility is missing in another direction. In a singularity, the Jacobian becomes rank-deficient.

For the standard 6-DoF manipulator structure, which is similar to the one that is used in this thesis but with no upper arm joint, the pose is singular whenever two joint axes are coaxial. This happens when at least one of the hinge joints is stretched out or if the robot is in an overhead position so that the first and the last joint axes become coaxial.

For the considered 7-DoF robot structure, the situation is less trivial. A singularity in the mentioned sense is only reached, when the elbow joint is stretched or if more than two joint axes align.

However, if the null-space parameter is included as part of the Cartesian pose vector, which increases the Jacobian to a 7×7 matrix, the aforementioned cases can be considered singular, too. They render the null-space parameter fix or undefined. Figure 2.5 shows an overview of all possible singularities. The topmost posture constitutes a singularity in the classical sense (TCP movement restricted), the three lower postures only block the null-space movement.

When possible, the proximity to all types of singularities will be actively avoided within this thesis. Apart from that, they will not be taken account of since singularity treatment is not the focus.

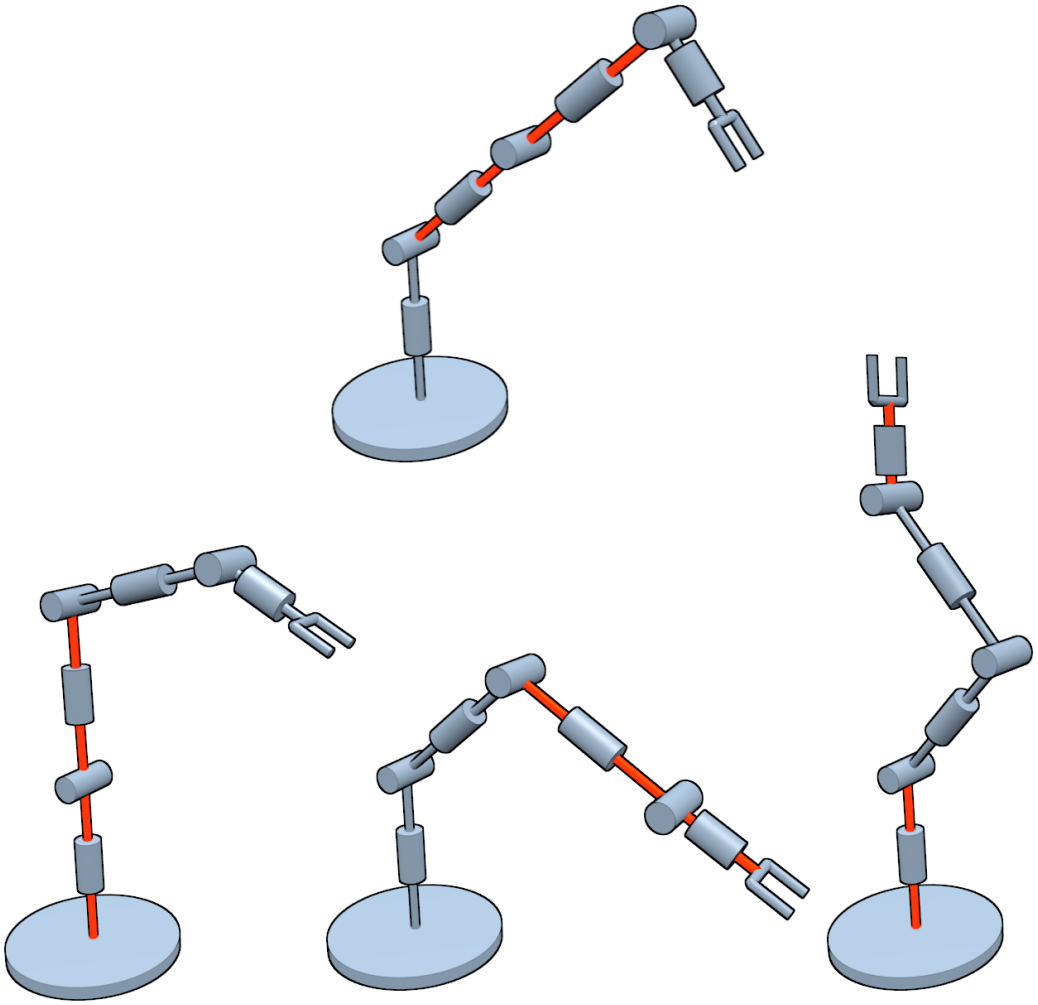

Figure 2.5: All possible singularities for the regarded structure, aligning joint axes colored in red. Only in the posture depicted at the top, the motion of the TCP is restricted.

2.1.4 Forward and Inverse Kinematics

Forward and inverse kinematics describe the mapping between joint angles and the end-effector pose. A certain joint configuration is also referred to as a position or vector in n-dimensional joint space, where n is the number of joints. The pose of the end-effector, described by position and orientation, is also called a pose in Cartesian space or configuration space.

Forward kinematics is the determination of the end-effector pose, given the joint angles. For serial manipulators of the given structure, this mapping is unique and straightforward to compute using consecutive matrix multiplications.

Inverse kinematics is the determination of joint angles for a given end-effector pose. In order to make this mapping unique, the configuration index and all null-space parameters (just one for the robots considered in this thesis) have to be specified. In a singular configuration, additional parameters are needed.

There are several approaches to solving the inverse kinematics problem. The most general one is a numerical gradient descent, invoking only forward kinematics computations and the determination of the Jacobian. An introduction is given by Buss [16]. This approach works with any type of serial robot. However, it comes with all the disadvantages of a numerical gradient descent, including the possibility of getting stuck in a local minimum, no guarantee that a solution will be found and possibly high computational cost.

For many robot structures, including the one used in this thesis, analytical solutions exist. Although devising such a solution is far more complex and restricted to a specific robot structure, it results in an algorithm that is exact, fast and it can be guaranteed that a solution will be found if one exists. In this thesis, this is the means of choice.

Although an additional joint increases the flexibility of the robot, it also complicates the inverse kinematics problem and creates a necessity for optimization. Finding a suitable value for the null-space parameter can be quite challenging. The elbow can not continuously rotate because its movement is restricted by the joint limits as can be seen in Figure 2.6. Figure 2.7 illustrates how the different joints can block different intervals on the elbow circle while the TCP moves on a straight line.

If the robot motion is determined by a predefined trajectory, the elbow position can be optimized globally. However, for the sake of generality, this cannot be assumed. Alternatively, it would be desirable to find a surjective mapping from TCP pose to null-space parameter, so that the whole inverse kinematics problem becomes unique. Shimizu et al. [68, 69] propose such

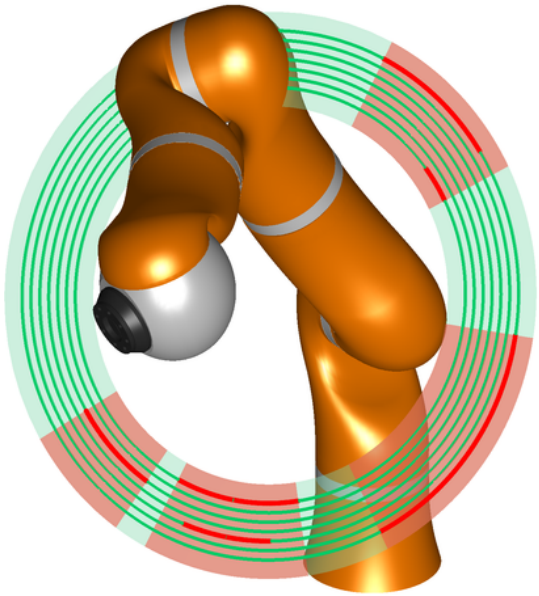

Figure 2.6: Joint limits of the LBR IV, projected onto the null-space parameter: The green regions are accessible, the red regions are blocked, since one or more joints would exceed their limits within these intervals. The innermost line in the ring corresponds to the base joint, the outermost line corresponds to the flange joint.

an approach. They present an optimization criterion, which can be solved analytically and which depends only on the current TCP pose. However, the optimal position can be within a blocked interval, in which case the closest possible elbow position satisfies the criterion best. In general, such an approach can lead to a jumping elbow behavior when the robot moves, which is not possible on an actual robot. Hence, this algorithm can only be applied for single, static poses.

Generally, a pre-determined elbow position for a given TCP pose would either introduce this jumping problem or restrict the flexibility too much. Also, a suitable elbow position for a specific end-effector pose depends on the trajectory to the pose. To give an example, Figure 2.8 illustrates how the LBR IV can place the TCP inside an area that would be unaccessible if there was no upper arm joint, since this slice lies outside the limits of the base joint. A suitable elbow position depends on the previous pose of the robot, whether the base joint was close to its positive or its negative limit.

Since both global null-space optimization as well as a surjective mapping from TCP pose to elbow position are unfeasible, the best option is a local optimization as described by Chan and Dubey [23], referred to as gradient projection method. A performance criterion or cost function is defined depending on the robot position and its gradient is projected on the null-space

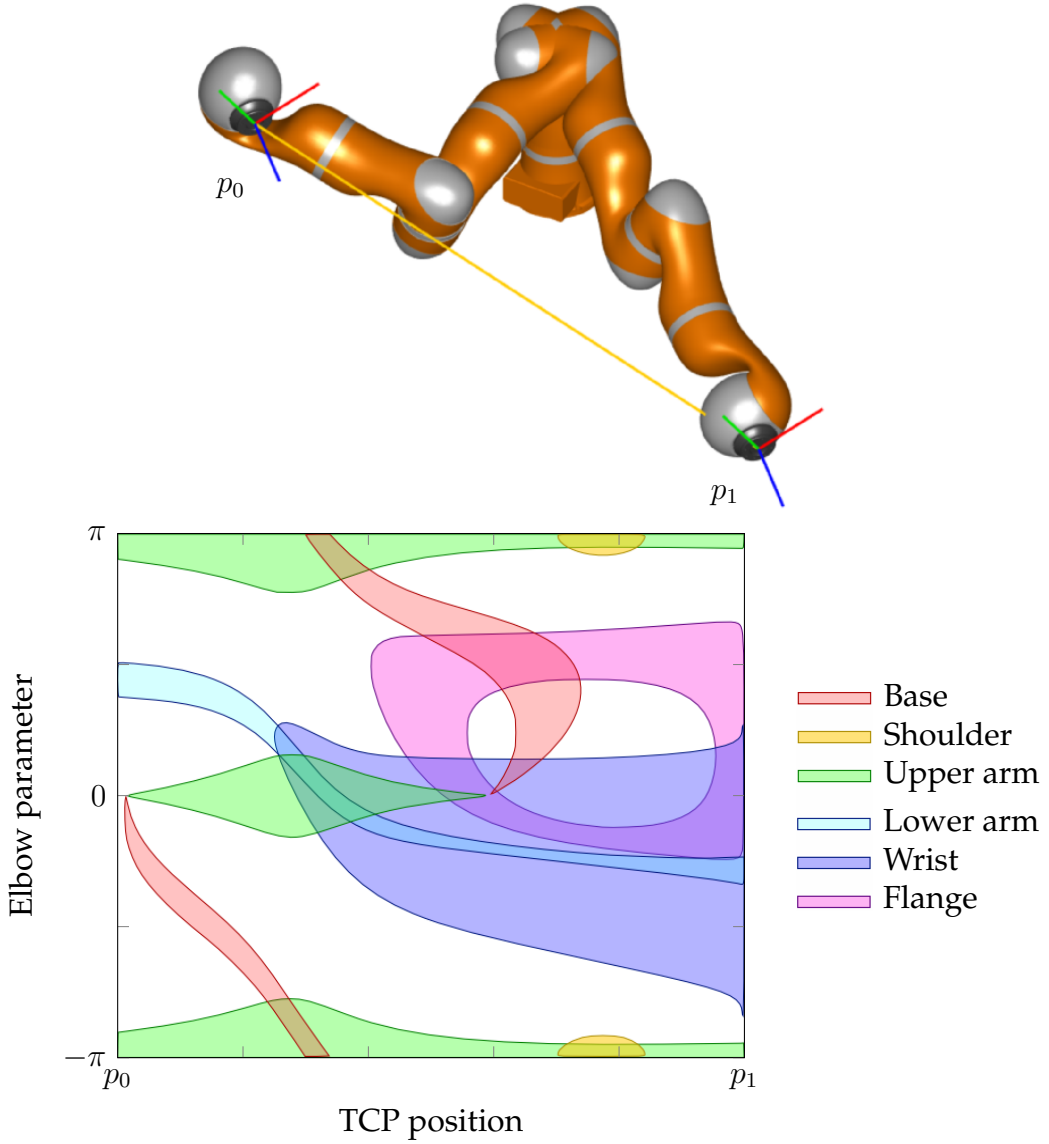

Figure 2.7: Blocked elbow intervals per joint for a straight movement: The colored areas show how the intervals change position, size, appear and disappear over the motion along the line. In the depicted case, no feasible trajectory for the elbow can be found.

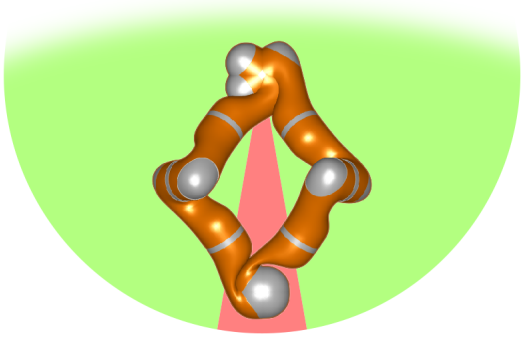

Figure 2.8: Demonstration of the flexibility that is introduced by a seventh joint: The red area is blocked for the upper arm by the base joint. Without the upper arm joint, the TCP could not be positioned there.

parameter. The parameter is then varied accordingly. Using such a strategy, it may occur that the elbow moves into a position which is unfavorable for a particular motion. However, if the motion trajectory is not known beforehand, this is unavoidable in general.

In this thesis, a local null-space optimization is the strategy of choice. The cost function that was developed within this work will be presented in section 3.4.

2.2 Workspace Analysis and Reachability

The first workspace considerations were published by Roth in 1976 [65]. Basic conventions and concepts (such as types of joints, decoupling of position and orientation, number of solutions) are discussed and sketches of workspaces (or "working spaces") are presented for different manipulator types, based on geometric considerations.

Within the following 40 years, numerous concepts and approaches have been devised, which will be introduced below. First, a basic workspace classification is presented. Afterwards, different concepts for workspace analysis are introduced and the most relevant works are discussed.

2.2.1 Basic Considerations

In [35, 36], Gupta summarizes and discusses basic concepts and insights about robot workspaces in general. The influence of the hand size on reachable and dexterous workspace (definition in the following section) is shown, the humanoid robot arm is characterized, approach angle and lengths are explained. Voids and holes are revised, a method for obtaining the number of inverse kinematics solutions is introduced and the influence of joint limits is shown.

Rastegar and Deravi [61] discuss the effect of joint limits on the workspace and on the sub-workspaces (the different configurations). Without joint limits, the workspace of all the different configurations would look identical.

2.2.2 Workspace Categories

In general, there are two different perspectives on workspace analysis. In terms of safety, it is relevant which parts of the surrounding space can be reached by any point on the robot or its tool. From the application point of view, it matters which areas can be reached with the tool. This thesis focuses on the latter.

Among the application-related workspaces, there are different sub categories [19]:

1. The reachable (or primary) workspace contains all points that the robot TCP can reach in at least one orientation.
2. The constant orientation (or functional) workspace contains all points that the robot TCP can reach in a specified orientation.
3. The total orientation workspace contains all points that the robot TCP can reach in all orientations from a predefined set.
4. The dexterous (or secondary) workspace contains all points that the robot TCP can reach in all orientations.
5. The orientation workspace contains all orientations that the robot TCP can reach at one specified location.

Figures 2.9, 2.10 and 2.11 illustrate the different types of application-related workspaces. A planar 3-DoF robot, where the shoulder and the elbow joint can move between 0° and 90° and the wrist joint can rotate freely, serves as an example.

There are different concepts to obtain the shape of the various types of workspace. They can be classified into the following five basic categories.

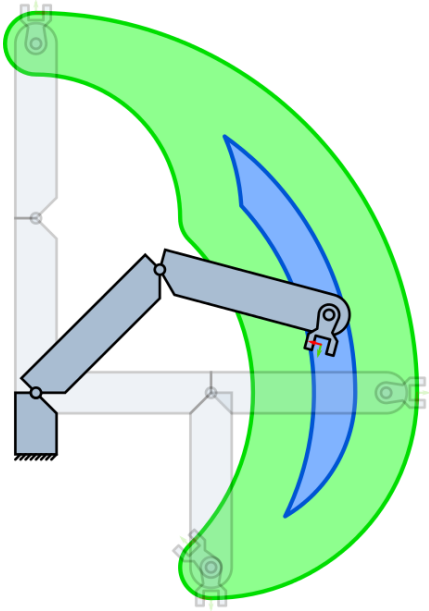

Figure 2.9: Reachable (green) and dexterous (blue) workspace for an exemplary planar manipulator with joint limits in its first and second joint.

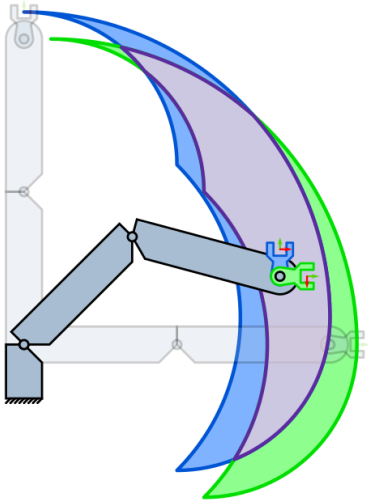

Figure 2.10: Constant orientation workspace for two different orientations (blue and green) and total orientation workspace (violet) for those two orientations.

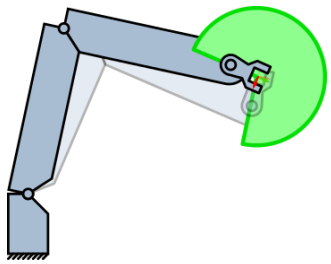

Figure 2.11: Orientation workspace for one particular end-effector position: The TCP can sweep along the depicted green angle without hitting joint limits.

2.2.2.1 Volume Sweeping or Convolution

The idea is to start at the last joint and analyze its movement. If it is a rotational joint, it is able to position the TCP on a circle. In case it is a prismatic joint, it can position the TCP on a line. Then the second to last joint can extrude this one-dimensional accessible region around (or along) its own axis, making it two-dimensional and so forth. Each joint can sweep the volume that all successive joints can cover around or along its own axis, as illustrated in Figure 2.12. Volume sweeping can only compute the accessible workspace. A discussion on the theory in general can be found in [2].

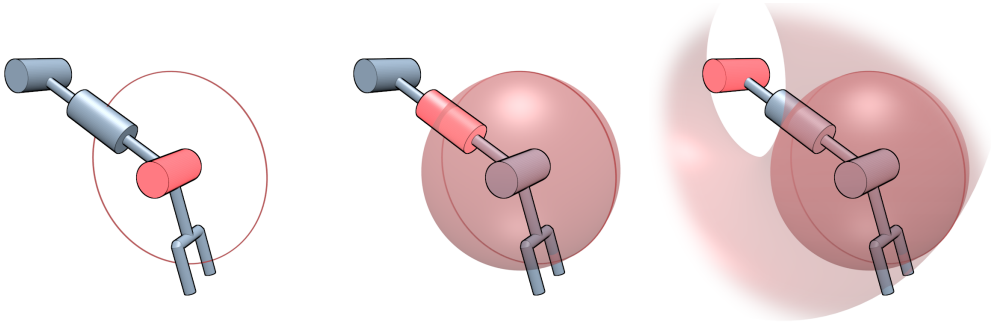

Figure 2.12: Illustration of volume sweeping for workspace determination: Each joint sweeps the workspace of all following joints about its own joint axis.

Hansen et al. [37] use this concept and reduce the sampling point count by using polar coordinate systems. Furthermore, the concept of approach angles and approach lengths is discussed: For a given point, the directions and lengths from which it can be approached are evaluated.

Ceccarelli [22] analytically describes the boundary surface of hyper-rings, which result from sweeping rings (or hyper-rings) along other rings. This

concept is used to describe the workspace of general n-DoF open-chain manipulators with revolute joints.

Chirikjian and Ebert-Uphoff [24, 25] compute the workspace based on convolution in Euclidean space. The workspace of each robot segment is approximated by a density function and the workspaces of all segments are then convoluted. This reduces the complexity of workspace computation for a robot with n joints and K discretized states per joint from $O(K^n)$ to $O(\log n)$.

Anderson-Sprecher and Simmons [6] use a convolution-based algorithm to compute a voxel grid, containing the minimum time it takes for any part of the robot to reach this grid cell from its current position. The approach can also be used for only the TCP.

2.2.2.2 Forward Kinematics

The basic idea is to sample all joint angles either equidistantly or randomly (Monte Carlo method) and compute the forward kinematics for each joint combination. The result is a three or six-dimensional point cloud (depending on orientation handling), where each sample lies within the reachable space of the robot. Now, either a hull around all points is created for visualization, or the samples are fitted into a voxel grid for further computations, e.g. much faster but less accurate reachability checks. Figure 2.13 illustrates the method.

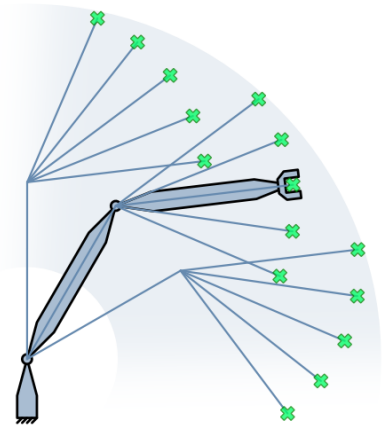

Figure 2.13: Illustration of the workspace determination by forward kinematics: The joint space is sampled, the forward kinematics is computed for each sample and a hull is generated around the resulting poses.

Yang and Lee [88] sample the joints and compute forward kinematic solutions for a slice of the workspace. The slice is then discretized and the outline pixels are found. From this, the volume of a revolving slice is computed. The ratio of the workspace volume over the total link lengths is introduced as a performance index. Cwiakala and Lee [28] improved this approach by significantly reducing the number of scanned samples.

Rastegar and Fardanesh [63] use Monte Carlo to determine the volume of a robot workspace, the accessibility and the effect of changes in the robot's parameters.

Alciatore and Ng [5] apply the approach to two-dimensional cases. Subsequently, line and arc segments are fitted around the found areas and form the boundaries of the workspace.

Guan and Yokoi [33] perform a reachable space analysis for the humanoid robot HRP-2 under kinematic constraints. They propose a Monte Carlo method by randomly sampling applicable joint angles, computing forward kinematics and storing the results in a voxel database.

Castelli et al. [19] also split the space into voxels. Then they sample each joint in regular intervals, compute the forward kinematics for each configuration and count how often each grid cell is reached. Also, the surrounding surface is computed by applying a filter.

Wang et al. [81] sample the workspace using the Monte Carlo method as well. They take inequality and equality constraints into consideration.

Cao et al. [17, 18] also take the Monte Carlo approach, the boundary of the point cloud is then rendered in a commercial CAD software. The main focus of this approach is simplicity.

Gudla [34] determines the constant orientation workspace of a 6R robot (six rotatory joints) by sampling the first three axes and determining the last three axes in order to keep the orientation constant.

The most closely related work to this thesis is done by Zacharias et al. at DLR [95, 96]. The workspace around the robot is sampled in a cubic grid. The set of orientations is also discretized. The forward kinematics is then computed for random joint positions and each result is associated with its closest match from the sampling. It is then counted for each position, from how many approaching directions it was reached. The result determines the color of a sphere that is drawn at that location (called reachability sphere). The resulting visualization is called reachability sphere map or capability map and is intended to depict the robot workspace in an intuitive manner. In order to encapsulate even more information, different shapes were used, e.g. a cone with the cap rotated into the direction from which the most solutions were accessible. In [97, 98, 100], the capability map is used for grasp planning

and positioning of mobile robots. In [99], it is used in order to evaluate how well a haptic interface covers the reachable workspace of human arms.

2.2.2.3 Inverse Kinematics

Here, it is not the configuration space but the Cartesian space that is sampled, as illustrated in Figure 2.14. In most works, it is divided into a voxel grid and each cell is tested for the existence of solutions to the inverse kinematics problem. An advantage compared to the forward kinematics method is that the result is a well-structured grid with uniform density. It can easily be rendered using the Marching Cubes algorithm for instance. Furthermore, the orientation can be considered for obtaining the functional workspace. However, it is considerably slower, since the inverse kinematics usually takes longer to compute.

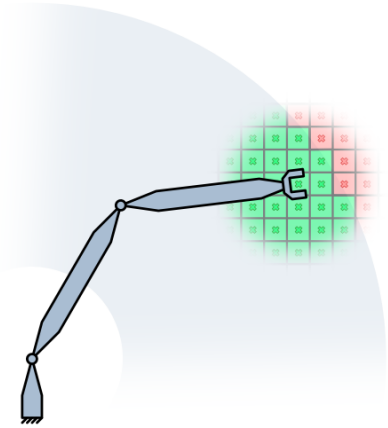

Figure 2.14: Illustration of the workspace determination by inverse kinematics: The configuration space is sampled and each pose is tested for solvable inverse kinematics.

Cavusoglu et al. [20] compare different robotic manipulators for the medical application of a teleoperated suturing task. Predefined trajectories are run through the inverse kinematics of all manipulators and it is checked if the desired motions can be achieved. They also account for the (often neglected) problem of continuity. If all poses on a trajectory are reachable, it is still possible that the robot needs to reconfigure between two consecutive poses.

Bonev and Ryu [11] investigate the orientation workspace of a parallel robot. The result is displayed as a boundary surface of Euler angles in a cylindrical coordinate system.

Petitt and Miller [58] map orientations into RGB color space via Euler angles in order to visualize the workspace including orientation information. The dexterity at a point is measured by the number of reachable orientations at that point. This idea was later also introduced by Zacharias.

Attamimi et al. [7] split the workspace into voxels and test if an inverse kinematics solution exists for each voxel. Afterwards, they compute a distance map (distance of each voxel to an edge voxel) and classify the reachability of a point based on this value. The map is used in connection with other maps, including object-specific maps in order to optimize manipulation.

Lohmann et al. [52] compute the workspace borders within a human body for minimally invasive robotic surgery. Here, the trocar reduces the dimensionality of the problem. The workspace is split into slices and each slice is sampled along its boundary. The algorithm is very fast and can visualize the workspace for a certain trocar within a fraction of a second. It is intended to assess the suitability of a trocar position.

This work was further pursued by Hutzl et al. [42]. Here, the trocar point position is autonomously optimized based on tracked and annotated operations. Grid points within a volume that has to be reachable are tested for reachability by checking if the inverse kinematics problem has a solution. The minimization criterion for the trocar point is the ratio of unreachable points to all points within the test volume.

Borchard et al. [12] also perform a workspace analysis for laparoscopic instruments. Tracked poses of surgical trajectories are tested for feasibility by computing the inverse kinematics solution for each recorded pose. The ratio of reachable to all poses is taken as a quality criterion. Collisions between the two instruments are also considered.

Porges et al. [59] speed up the generation of the capability map filling the voxel grid by sampling random forward kinematics until the probability to land in a voxel that was already covered becomes so high that it is faster to test the remaining voxels using inverse kinematics.

Vahrenkamp et al. [79] voxelize the workspace in six dimensions and assign a manipulability index. This reachability map is then inverted in order to generate an object-centered map that describes how well the robot can reach the object from different positions.

2.2.2.4 Analytical and Geometrical Boundary Determination

Here, slightly different approaches are summarized as one category, since they cannot be separated clearly in most cases. As explained in section 2.2, the workspace is bounded by surfaces at which either a joint limit is hit or the

robot is in a singular configuration. Geometrical, numerical and analytical methods have been devised in order to trace and visualize these surfaces. It is hard to make general statements, since the specific approaches vary largely from a conceptual point of view.

Kumar and Waldron [48] as well as Selfridge [67] determine the outer hull of the workspace of a robot with rotatory joints using different algorithms. Both use the fact that, in order to maximize the extension into a certain direction, all joint axes have to intersect one line in this direction.

Tsai and Soni [77] compute an arbitrary slice of the workspace of an n-DoF robot with revolute joints. A boundary point is searched from which the rim is traced using a linear optimization technique and incremental displacements.

Spanos and Kohli [45, 71] propose equations for describing the workspace of wrist-partitioned robots and present a method for workspace generation based on polynomial displacement equations and their discriminants. Analytical expressions for the boundary surfaces are obtained.

Borrel and Liegeois [13] show that the joint space can be decomposed into so-called aspects. The aspect is equivalent to what is called configuration index in this thesis. The inverse kinematics solution (for fixed null-space parameters) is unique within each aspect and the aspects are separated by hyper surfaces in which all m-order minors of the Jacobian are non-zero, where m is the dimensionality of the workspace. It is shown that the workspace boundaries are part of the aspect boundaries (where in this work the workspace boundary is understood as furthest reachable point in each direction). Aspects are visualized in a graph. The workspace is rendered by transforming the aspect boundaries from joint space into Cartesian space.

Wu and Valencia [86] assess the feasibility of trajectories within the workspace of a wrist-partitioned, 6R manipulator. First, the space for all possible wrist positions is obtained geometrically and it is tested whether the wrist trajectory lies within this space. It is then checked whether the last three joints can reach all desired orientations along the trajectory.

Tsai [76] geometrically traces the workspace borders using screw theory. The constant orientation workspace is also considered.

Cimino and Pennock [27] as well as Lia and Menq [51] compute analytical borders of the dexterous workspace of simple, wrist-partitioned robot manipulators.

Rastegar and Deravi [62] consider a robotic arm with a fixed end-effector pose as a closed kinematic chain and determine the number of possible configurations by finding the roots of a polynomial displacement equation.

Oblak and Kohli [56] investigate the surfaces in a workspace where the Jacobian matrix is singular or where joints reach their limits. They distinguish between inner and outer surfaces and check whether a surface is crossable or non-crossable. They apply the theory on simple three-jointed robots.

Yang et al. [89] mathematically separate a 6R wrist-separated robot arm at its wrist, analyze regional structure and wrist separately and reassemble the results via vector field analysis. Thus, the boundaries of the reachable workspace can be drawn analytically.

Kwon and Youm [49] devise an algorithm that allows drawing the outline for the workspace of a robot with an arbitrary number of links. However, it only works in a plane and assumes that the base can rotate that plane.

Lück and Lee [53, 54, 55] investigate the topology of self motion manifolds for redundant robots. Based on this, a discretization method for workspace analysis is proposed. They introduce the concept of a joint limit as a semi-singularity because it restricts motion along an axis in only one direction.

Abdel-Malek et al. [3] use the sweeping approach. Furthermore, joint limits are considered and a rank deficiency condition is imposed to determine singular sets. In [1], they compare a numerical continuation method and an analytical boundary parameterization, both based on rank deficiency, and use insights gained from one on the other.

Rauchfuss et al. [64] present methods for computing the accessible orientations around a point (service sphere or service angle) analytically for a 6-DoF arm.

Snyman et al. [70] search for a point inside the workspace and radiate outwards into multiple directions on a plane until the limit is hit (which is implemented as an optimization problem).

Wang and Chirikjian [82] describe workspace generation of hyper redundant robots as a diffusion process, which can be solved as a partial differential equation on the motion group SE(N). Furthermore, a workspace density approach to the inverse kinematic problem is proposed.

Cebula and Zsombor-Murray [21] formulate the workspace equation for arbitrary wrist-partitioned robots by using Study Kinematic Mapping.

Bohigas et al. [10] use a linear relaxation method in order to find all boundary points where a specially formulated Jacobian becomes rank-deficient. Joint limits can be handled and the type of singularity is classified (outer boundary barriers, interior barriers or traversable singularities). Depending on the formulation of the problem, constant orientation workspace as well as accessible and reachable workspaces can be computed.

Urbanic and Gulda [78] compute the constant orientation workspace of a 6R robot geometrically. However, they determine only one slice of the workspace and revolve the 2D boundary around the first joint. This entails that the checked orientation is not globally constant but it also rotates with the first joint. They use boolean operations to compute the workspaces for multiple orientations and for finding common workspaces of multiple robots.

2.2.2.5 Learning-based Approaches

Two publications have been found in which the authors pursue a learning-based approach. Advantageous are here the general applicability of the concepts. However, the price is a somewhat inaccurate solution.

Stulp et al. [72] combine analytic models, imitation and learning in order to obtain a model of the reachability for mobile manipulation tasks. This results in good heuristics with few data.

Jamone et al. [43] use a bio-inspired learning approach. The humanoid robot iCub serves as use case and its head and eyes are considered part of the kinematic chain to the hand. Motor babbling lets the robot learn a model of its forward kinematics. The model is inversed and a large set of inverse kinematics queries is computed and serves as Reachable Space Map. It can then be predicted whether a point within the robot's gaze is reachable and if so, how well.

2.3 Manipulability and Dexterity

So far, the primary concern was to determine the borders of the reachable space. However, it is furthermore of interest to assess the quality of a reachable pose, which is referred to as manipulability or dexterity measure. This measure evaluates how flexible a robot is at or around a given pose. Many works compute that index from the Jacobian which describes the connection between the configuration space and the Cartesian space. The idea is that if the robot needs to perform large joint motions in order to move the end-effector by a small distance, the pose is rather inflexible. Furthermore, the Jacobian becomes singular near the workspace borders which makes it well-suited as a basis for measuring manipulability. From a mechanical point of view however, large joint motion for small end-effector motion entails that a large force on the end-effector only requires small joint torque, which can be desired as well. Different scenarios require different measures. The most popular ones are introduced within this section.

Yoshikawa [92, 93] proposes $w = \sqrt{\det(J(\theta)J^T(\theta))}$ as a measure of manipulability, where $J(\theta)$ is the Jacobian matrix at a specific point θ in joint space. It corresponds to the volume of an ellipsoid that is formed by a sphere in joint space, projected into Cartesian space. If the robot is in a singular position, w becomes zero. It is called the manipulability ellipsoid. In [94], the measure is split into translational and rotational manipulability. In [91], a dynamic manipulability measure is introduced, which is computed by $w_d = \sqrt{\det(J(M^T M)J^T)}$, where M represents the mass matrix.

Yang and Lai [87] introduce the concept of service angle and service region. A point within the workspace of a robot is considered. Around that point, a sphere is constructed with the radius of the robot's hand size. For every angle from which the robot can place the TCP at the sphere center, a point is drawn where the wrist is located on the sphere. The sphere is called service sphere, such a point is called a service point and a connected set of points is called a service region. A similar concept is used within this thesis.

Togai [73] propose to use $M = \sigma_{\text{max}}/\sigma_{\text{min}}$, where σ_{max} and σ_{min} are maximum and minimum absolute singular values of the Jacobian.

Tsai [76] formulates the manipulability depending on the distance to the joint limits.

Park and Brockett [57] introduce a coordinate-invariant dexterity measure that is based on the mapping from joint space to SE(3).

Konietschke et al. [46] propose the reciprocal of the maximum joint velocity as a measure for manipulability. Furthermore, a positioning accuracy index is introduced, accounting for joint encoder steps.

Abdel-Malek et al. [4] place a trajectory within a robot's workspace so that every point can be reached with maximum dexterity, i.e. in a maximum number of orientations. The distance to the workspace boundary can also be specified. They furthermore extend the Jacobian matrix to account for joint limits.

Trivino and Martin [75] introduce a fuzzy measure in order to assess the manipulability. In contrast to the classical Jacobian approach, joint limits are taken into account.

Choi and Ryu [26] suggest a power manipulability index. It describes how much power can be generated along an arbitrary direction.

Vahrenkamp et al. [80] propose an extended manipulability measure which includes joint limits, obstacles and an augmented Jacobian measure.

Tondu and Bertrand [74] propose to use a zonotope-based manipulability index. It results from transforming a hyperrectangle in joint space, limited

by the joint speed limits, into Cartesian space using the Jacobian. From this, the maximum possible TCP velocity in each direction can be inferred. The volume of the zonotope can also be used as manipulability measure.

2.4 Contributions of this Thesis

Several weaknesses within state of the art workspace exploration techniques and dexterity measures have been identified. These will be pointed out in the following and the concepts of how they are addressed within this thesis will be introduced.

2.4.1 Intuitive Workspace Visualization

A major problem of workspaces per se is the fact that they are from a mathematical point of view boundaries of six-dimensional hypervolumes, which is impossible to visualize for a human eye in an intuitive manner. Many or even most works bypass this problem by only regarding the reachable or dexterous workspace of the robot (at least one or all orientations reachable) and disregard the influence of a specific target orientation. Within this thesis, a means will be presented by which all six dimensions are accounted for, while keeping the cognitive load on the operator within acceptable limits.

2.4.2 On-the-Fly Generation and Rendering of Workspace Boundaries

To the author's knowledge, all other works about workspace computation for robotic manipulators work offline, which means that the resulting visualization is a static map. The presented method is the first to be intended and suited for online computation. Today's hardware is powerful enough to tackle many problems on-the-fly, i.e. without precomputation and fast enough so that there is virtually no waiting time for a human user. Situation-specific workspace limits can be visualized while someone operates the robot. Tool changes can be accounted for without the need of having to precompute the whole reachability database for every new tool. As hardware becomes more and more powerful, the resolution, the frame rate and the number of considered influences can be increased.

2.4.3 Handling of Self-Overlaps

Self-intersections or self-overlaps of a robot's workspace pose a problem that is very rarely considered. The issue is depicted in Figure 2.15. A planar serial manipulator with two joints and joint limits can be seen. The blue-rimmed area around the robot contains all reachable points. The striped region can be accessed both from the red and from the green zone. However, it can only be left into the zone from where the robot came (here the green zone).

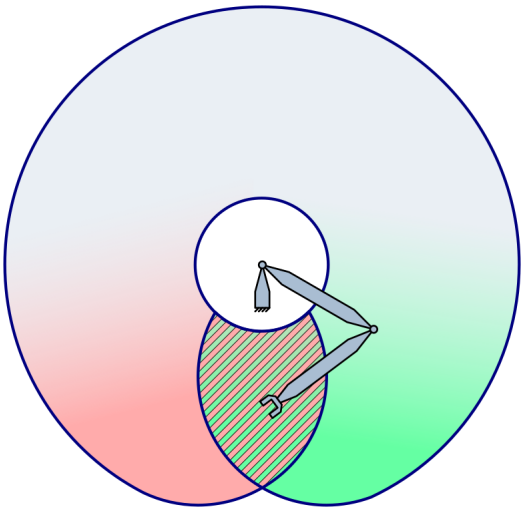

Figure 2.15: Illustration of workspace self-overlaps on an exemplary two-joint robot with joint limits: The striped region can be accessed from either the red or the green side but it cannot be traversed.

The problem is that the boundaries of a region lie within another section of the region itself. Whether a certain boundary limits the movement of the robot or not depends on the current robot position. Hence, it is not obvious whether a wall is currently relevant. For the depicted, two-dimensional case, the situation may be easily graspable. With a six (let alone seven) axis manipulator, it becomes much more complicated.

Handling this problem in a static map of the workspace seems impossible since the boundaries in question can either be shown or not (see Figure 2.16). It cannot be seen whether a wall is relevant for the current robot position.

This problem will be tackled within the scope of this thesis. The permanently updating live view allows to draw boundaries depending on the current pose.

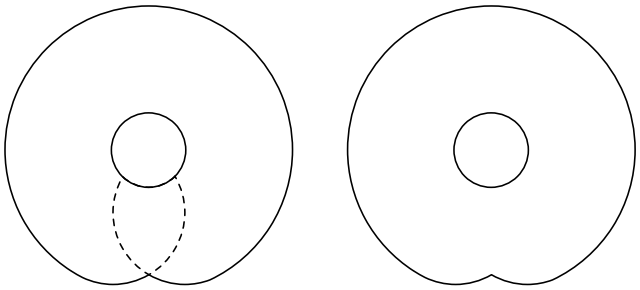

Figure 2.16: Possibilities of dealing with self-overlaps in a static workspace map: Inner boundaries can be drawn or not but whether they are relevant for the current pose cannot be determined.

2.4.4 Margin as Dexterity Measure

Several different dexterity measures have been proposed, most of them assign a scalar value to a robot pose, providing a quality criterion that can serve as a benchmark for comparing or optimizing positions within the workspace.

Within the thesis at hand, the actual margin of motion around the current pose is proposed to be interpreted as a measure of dexterity and manipulability. It can be much more meaningful than a metric that is purely based on the local Jacobian, since the Jacobian constitutes the differential at a single point and therefore linearizes the problem. Furthermore, if the robot has no unique solution to the inverse kinematics problem (as is the case for redundant manipulators), the Jacobian is not uniquely defined for a given pose, since it depends on the according joint angles. Many workspace determination techniques that rely on the Jacobian can also not be applied for this reason.

3 Proposed Workspace Visualization Concepts

In this chapter, the main ideas and concepts of this thesis are discussed in detail. First, closer insight into the computation of the inverse kinematics solution is given. Afterwards, the distinction between generally and directly accessible workspace is made. The different visualization techniques that form the core of this thesis are then presented. Finally, the handling of the null-space parameter is discussed.

3.1 Inverse Kinematics Computation

As stated in section 2.1.4, the inverse kinematics problem means that a pose in Cartesian space is given and it is required to find according joint angles. Within the scope of this thesis, we will distinguish between a particular and the general solution to this problem. A particular solution means that a pose, the configuration index and a feasible null-space parameter are given. The general solution means that only pose and configuration index are given and we are interested in finding the set of all null-space parameters that result in feasible joint angles for the robot.

In the following, a concise and graspable overview of the underlying algorithms for both particular and general solution of the inverse kinematics problem is presented, as all proposed visualization concepts base upon them. Particularly the connection between joint limits and blocked intervals of the null-space parameter is illustrated.

3.1.1 Particular Solution

A particular solution is comparatively simple to compute. The computation is done within the robot's base coordinate system. In the following, the term l_{xy} will denote the distance between two points $\vec{p}_x$ and $\vec{p}_y$. The indices stand for b – base, s – shoulder, e – elbow, w – wrist, f – flange, c – center of the elbow

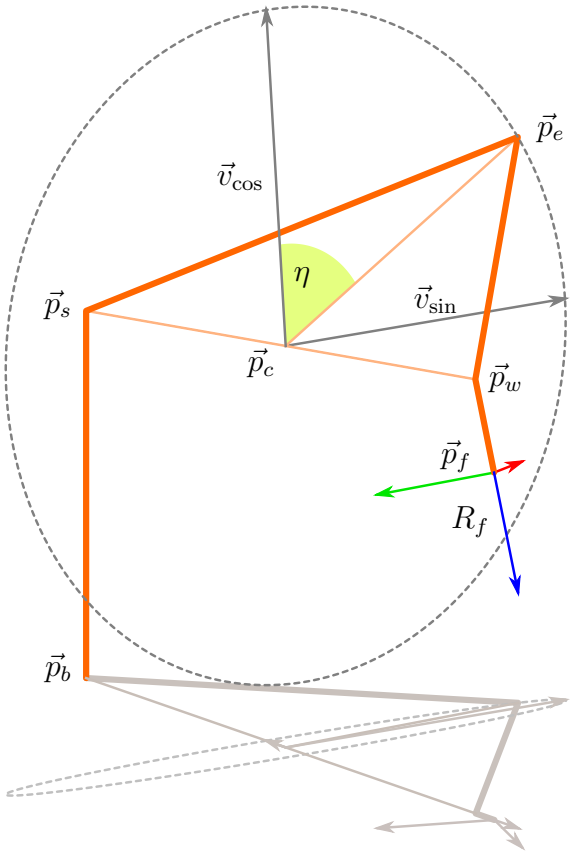

Figure 3.1: Sketch for computing a particular inverse kinematics solution: Shoulder, elbow and wrist form a triangle. As the triangle rotates on a connecting line l_{sw} from shoulder to wrist, the elbow travels on a circle (dashed line) with radius l_{ce}. This circle is centered at $\vec{p}_c$ and spanned by $\vec{v}_{\cos}$ and $\vec{v}_{sin}$, which can be obtained from shoulder and wrist position. The null-space parameter η describes the desired elbow position on the circle.

circle. Note that l_{bs}, l_{se}, l_{ew} and l_{wf} are determined by the robot construction. Figure 3.1 illustrates the geometric considerations.

At first, position $\vec{p}_f$ and orientation R_f of the flange have to be obtained from a given TCP position and orientation ($\vec{p}_{\text{tcp}}$ and R_{tcp}), based on the transformation of the tool ($\vec{v}_{\text{tool}}$ for translation and R_{tool} for orientation) that can be obtained from its CAD data for instance:

$$R_f = R_{\text{tool}}^T R_{\text{tcp}} \tag{3.1}$$
$$\vec{p}_f = \vec{p}_{\text{tcp}} - R_f \vec{p}_{\text{tool}} \,. \tag{3.2}$$

Now the wrist position can be obtained by

$$\vec{p}_w = \vec{p}_f + R_f \begin{pmatrix} 0 \\ 0 \\ -l_{wf} \end{pmatrix} \,, \tag{3.3}$$

while the shoulder position is simply given by

$$\vec{p}_s = \begin{pmatrix} 0 \\ 0 \\ l_{bs} \end{pmatrix} \,. \tag{3.4}$$

The distance between the shoulder $\vec{p}_s$ and the center $\vec{p}_c$ of the circle on which the elbow can rotate as well as the radius l_{ce} of this circle can be computed by

$$l_{sc} = \frac{l_{se}^2}{l_{sw}} \tag{3.5}$$
$$l_{ce} = \sqrt{l_{se}^2 - l_{sc}^2} \,, \tag{3.6}$$

so that

$$\vec{p}_c = \vec{p}_s + \frac{\vec{p}_w - \vec{p}_s}{l_{sw}} l_{sc} \,. \tag{3.7}$$

The vectors that span the elbow circle can be obtained by

$$\vec{v}_{\sin} = l_{ce} \frac{\vec{e}_z \times (\vec{p}_w - \vec{p}_s)}{\|\vec{e}_z \times (\vec{p}_w - \vec{p}_s)\|} \tag{3.8}$$
$$\vec{v}_{\cos} = l_{ce} \frac{(\vec{p}_w - \vec{p}_s) \times \vec{v}_{\sin}}{\|(\vec{p}_w - \vec{p}_s) \times \vec{v}_{\sin}\|} \,. \tag{3.9}$$

The actual position of the elbow is then determined by the null-space parameter:

$$\vec{p}_e = \vec{p}_c + \cos(\eta)\vec{v}_{\cos} + \sin(\eta)\vec{v}_{\sin} \,. \tag{3.10}$$

Once the positions of all joints are known, we can directly compute the hinge joint angles q_2, q_4 and q_6 (shoulder, elbow, wrist) as angles between the vectors along the neighboring links, as illustrated in Figure 3.2. This is exemplarily done for the elbow joint:

$$q_4 = \cos^{-1} \frac{(\vec{p}_e - \vec{p}_s) \cdot (\vec{p}_w - \vec{p}_e)}{l_{se} l_{ew}} \,. \tag{3.11}$$

For the axial joints q_1, q_3, q_5 and q_7 (base, upper arm, lower arm, flange), we need the rotation axes ($\vec{a}$) of the neighboring hinge joints and the axes which are perpendicular to that and to the connecting link ($\vec{a}'$). We will exemplarily compute the angle q_3 of the upper arm joint (see Figure 3.2) as follows:

$$\vec{a}_s = \frac{(\vec{p}_e - \vec{p}_s) \times (\vec{p}_b - \vec{p}_s)}{\|(\vec{p}_e - \vec{p}_s) \times (\vec{p}_b - \vec{p}_s)\|} \tag{3.12}$$

$$\vec{a}'_s = \vec{a}_s \times \frac{\vec{p}_e - \vec{p}_s}{l_{se}} \tag{3.13}$$

$$q_3 = \operatorname{atan2}(\vec{a}_e \cdot \vec{a}'_s, \vec{a}_e \cdot \vec{a}_s) \,. \tag{3.14}$$

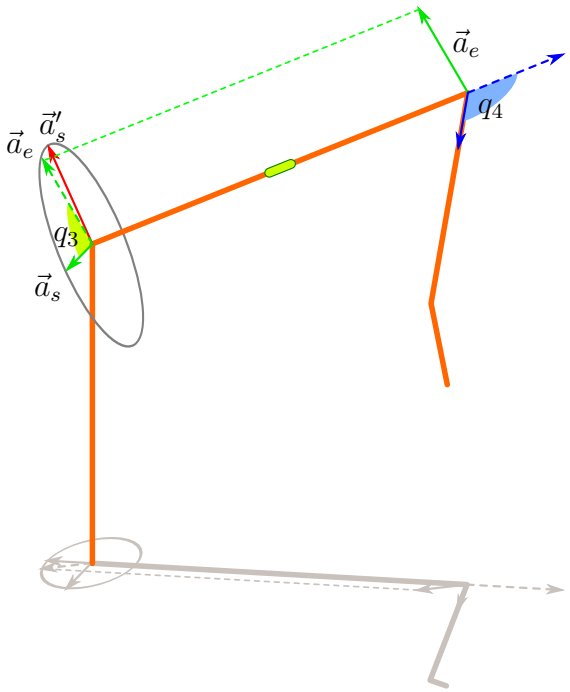

Figure 3.2: Sketch for the computation of q_3 and q_4

Note that with this algorithm, all hinge joints will have positive values. That corresponds to the first out of eight configuration indices. To obtain a different configuration, any hinge joint can be flipped to its negative value and the neighboring axial joints need to be rotated by 180°. The end-effector will then assume the identical pose.

3.1.2 General Solution

The general solution is much more complex and cannot be presented in full detail but the basic idea will be covered. In order to determine, which joint blocks which intervals on the null-space parameter, we first investigate the first three joints (base, shoulder and upper arm).

For the values of these joints, only the target position for the wrist $\vec{p}_w$ is relevant. This can easily be computed from the target TCP position and orientation (see equation (3.3)). In order to highlight the influence of each of the joints, they are colored in red, yellow and green throughout this section.

A sphere is constructed around the shoulder of the robot $\vec{p}_s$ as seen in Figure 3.3. Elbow $\vec{p}_e$ and wrist $\vec{p}_w$ are then projected onto this sphere ($\vec{p}_e^{\,\circ}$ and $\vec{p}_w^{\,\circ}$). If the joints had no limits, $\vec{p}_e$ and $\vec{p}_e^{\,\circ}$ could travel on a circle as the null-space parameter varies while $\vec{p}_w$ stays in position. The path of $\vec{p}_e^{\,\circ}$ on its circle on the surface of the sphere is shown as a dashed line.

The following restrictions are imposed by the respective joints (see Figure 3.4):

- The base joint renders a spherical lune Q_1 inaccessible for the projected elbow, as depicted in Figure 3.4. If $\vec{p}_e^{\,\circ}$ lies within the slice, the base joint is outside of its limits. For the LBR IV, it is within an azimuth above +170° or below -170°.
- Equally, the shoulder joint blocks a spherical sector around the base Q_2. The LBR IV cannot exceed a polar angle of 120° with its upper arm, hence the projected elbow cannot lie within the yellow region.
- The upper arm joint is a bit more intricate. It restricts the angle Q_3 between an arc from the north pole of the sphere $\vec{z}_+^{\,\circ}$ to the projected elbow $\vec{p}_e^{\,\circ}$ and an arc from the projected elbow to the projected wrist $\vec{p}_w^{\,\circ}$ (blue arcs in 3.4). For the LBR IV, this angle needs to be above 10°.

Depending on the configuration index, Q_1 can be located on the opposite side of the sphere and Q_3 may have to be constructed using the south instead of the north pole (so the adjacent angle of the one depicted in Figure 3.4 has to be above 10°).

Figure 3.5 shows how these restrictions form inaccessible sectors on the circle of projected elbow positions (and hence on the circle of elbow positions). For base and shoulder joint, these sectors lie within the intersections of the projected elbow and the inaccessible regions Q_1 and Q_2. Within the green sector, Q_3 is below 10°.

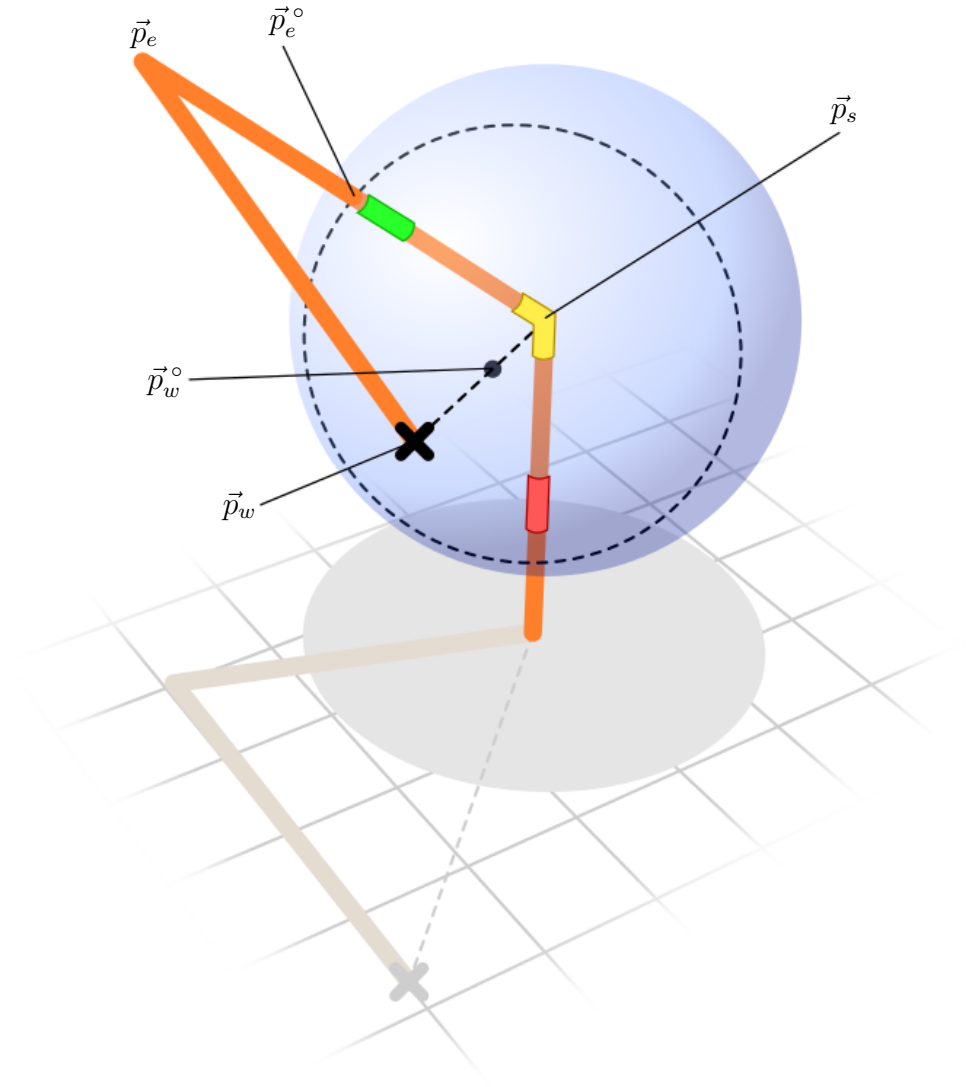

Figure 3.3: Sketch for computing the general inverse kinematics solution: Blocked null-space parameter intervals for the first three joints (red – base, yellow – shoulder and green – upper arm) are obtained with the help of spherical geometry on a sphere around the shoulder (blue).

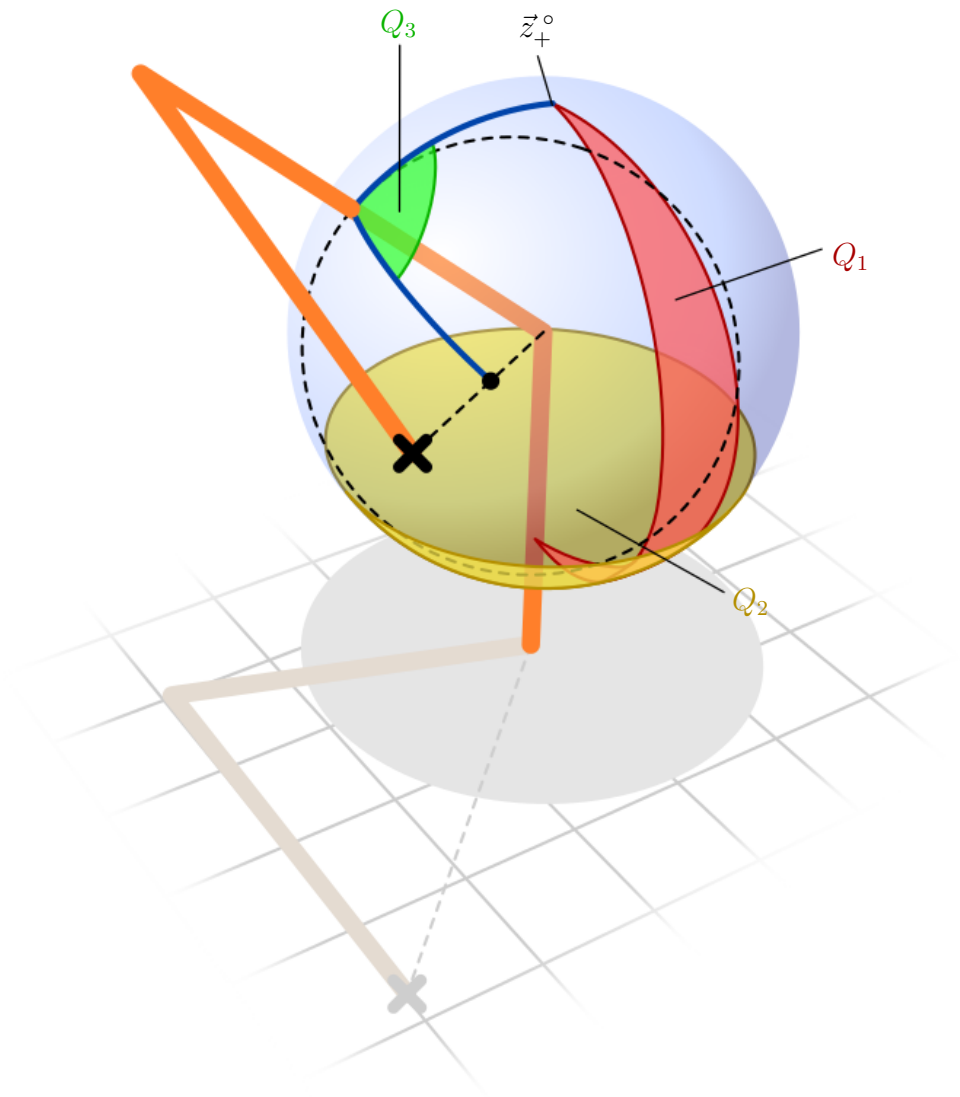

Figure 3.4: Sketch for computing the general inverse kinematics solution: Blocked areas due to different joints are depicted. The upper arm cannot move into the red spherical lune Q_1 because of the base joint or into the yellow cap Q_2 because of the shoulder joint. The green angle Q_3 has a lower limit due to the upper arm joint.

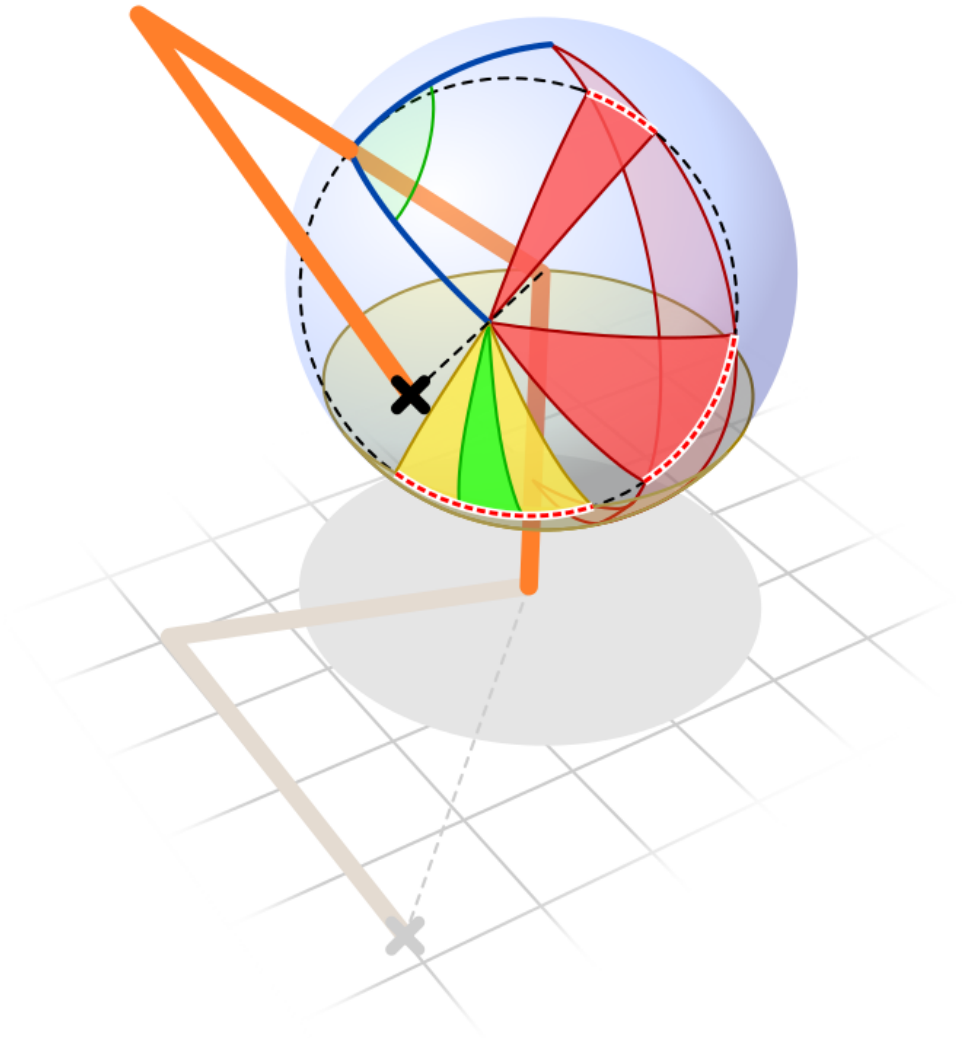

Figure 3.5: Sketch for computing the general inverse kinematics solution: The restrictions shown in Figure 3.4 block different intervals for the elbow movement on its circle. The blocked intervals are depicted by the colored circle sectors and red dotted lines on the circle.

As can be inferred, the base joint can block zero, one or two sectors on the elbow circle. If north or south pole lie within the circle, exactly one sector is blocked ("within" meaning the side of the circle that contains $\vec{p}_w^{\circ}$). The shoulder joint can block zero or one sector. The upper arm joint blocks zero, one or two sectors of the elbow circle. If the south pole lies within the circle, Q_3 is always above 90° and no sector is blocked by the upper arm joint. If neither south nor north pole are within the circle, one sector is blocked. If the north pole is within, two sectors are blocked. Depending on the configuration, it may be the other way around (north and south pole swap roles). Note that there is always at least one sector of the circle blocked and the elbow can never rotate freely.

So far, only the first three joints were taken into account. The considerations for the last three joints (lower arm, wrist and flange) are mathematically identical, since the robot's kinematic structure is symmetrical with respect to the elbow joint. The value of the elbow joint itself does not depend on the null-space parameter, only on the distance between shoulder and wrist, which is constant if the target pose is fix. Hence, the elbow joint can render the target pose completely unreachable, independent of the null-space parameter, if the target is too close. Otherwise, the elbow joint does not block any interval on the elbow circle.

3.2 Generally and Directly Accessible Workspace

In order to tackle the problem of self-overlaps that was introduced in section 2.4.3 and other potential internal barriers (e.g. if singularities are to be avoided), we will distinguish between two workspace types:

- The generally accessible workspace, which contains all poses that the TCP can assume in general
- The directly accessible workspace, which contains only poses that can be reached from the current pose in a straight motion (i.e. pure translation along one line or pure rotation about one axis).

Figure 3.6 illustrates the difference between the two workspaces. A two-joint parallel robot with joint limits serves as example.

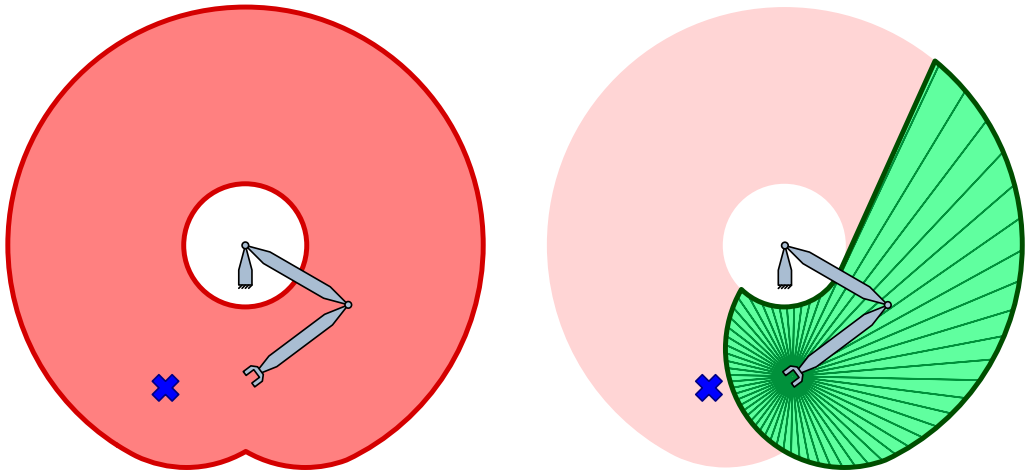

Figure 3.6: Comparison of the generally accessible workspace (left) and the directly accessible workspace (right) of an exemplary two-joint robot with joint limits: A target (blue X) can seem very close in the first visualization but it requires a large reconfiguration, which is only indicated by the second visualization.

For an operator, the directly accessible workspace can be much more relevant, since a large reconfiguration movement may be required in order to reach certain areas, which may not always be possible. For instance, the robot in Figure 3.6 would have to perform an overhead motion of the whole arm in order to reach the blue target.

However, sometimes it needs to be known whether a pose can be accessed at all, regardless what type of motion is required to get there. Since both pieces of information may be important, both concepts are pursued in this thesis.

3.3 Visualization

It is the aim to visualize the boundary surfaces around the reachable poses of a manipulator, which is a six-dimensional set. However, such a boundary is five-dimensional and (to the author's best knowledge) impossible to display in an intuitive way. This is why the problem is split into two parts: the display of translation boundaries and the display of rotation boundaries where the TCP acts as pivot point. For the visualization of the limits in one of the two domains, the current parameters in the other domain are kept constant.

3.3.1 Translation Boundaries

In general, the display of translatory motion limits in three-dimensional space is comparatively easy, it is simply a two-dimensional hull that can be rendered for instance as a triangle mesh.

For the visualization of the translation boundaries, it is reasonable to distinguish between generally and directly accessible workspace, since either concept can be applicable, depending on the situation.

The generally accessible translatory workspace is conceptually the same as the constant orientation workspace for the current orientation. A conceptual rendering can be seen in Figure 3.7. A rendering for the directly accessible translatory workspace is depicted in Figure 3.8.

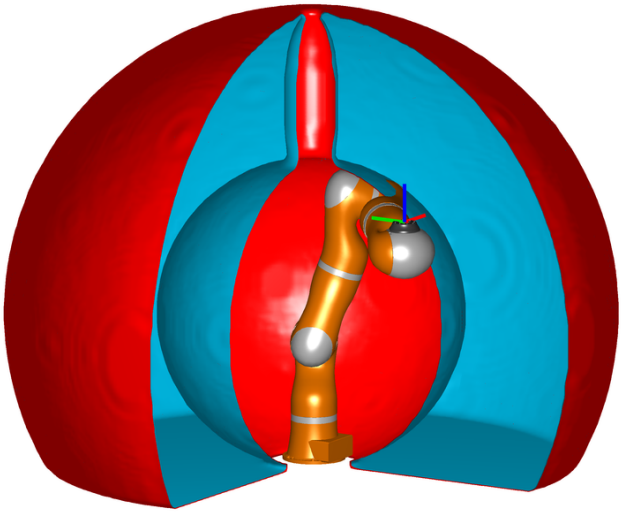

Figure 3.7: Translation boundaries of the generally accessible workspace of the LBR IV for the given orientation (cut open for visibility).

One concern that has to be taken care of is the fact that the view of the task at hand must not be obstructed by the rendered surface. This will be done by working with transparency and clipping. Further details are covered in chapter 4.

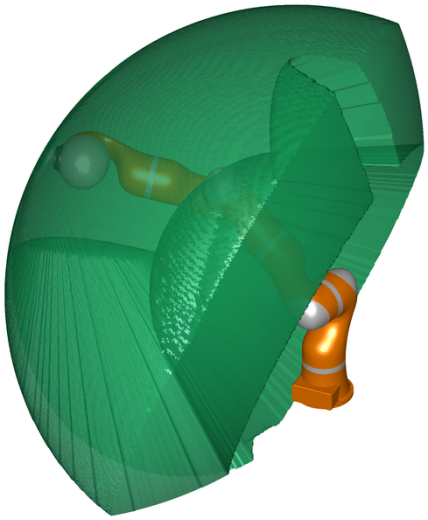

Figure 3.8: Translation boundaries of the directly accessible workspace of the LBR IV for the given orientation, up to a distance of 1 m around the flange center.

3.3.2 Rotation Boundaries

The limits of the directly accessible orientation workspace are more difficult to display. Several approaches were considered within the scope of this thesis, they will be presented in the following.

3.3.2.1 Boundary Surfaces in Orientation Domains

This concept displays the generally accessible orientation workspace for a fixed TCP position. The set of all orientations in three-dimensional space is itself three-dimensional, so it can be mapped to a three-dimensional chart. A boundary surface between all accessible and inaccessible orientations can be rendered within this chart. Four different orientation representations are considered. All use the LBR IV structure for the exemplary flange position (30 cm, 15 cm, 10 cm) in base coordinates.

The first representation is based on the classical yaw (Ψ)/pitch (Θ)/roll (Φ) convention, which is taken as one representative of the twelve Euler conventions. Figure 3.12 shows the resulting surface. Although the display contains all relevant information, it is very hard to interpret: Doing the mapping from yaw/pitch/roll to the corresponding orientation in the head for only one instance takes some seconds and often the help of three fingers. A complete chart is presumably ungraspable, especially if it is intended to be assisting instead of requiring full focus.

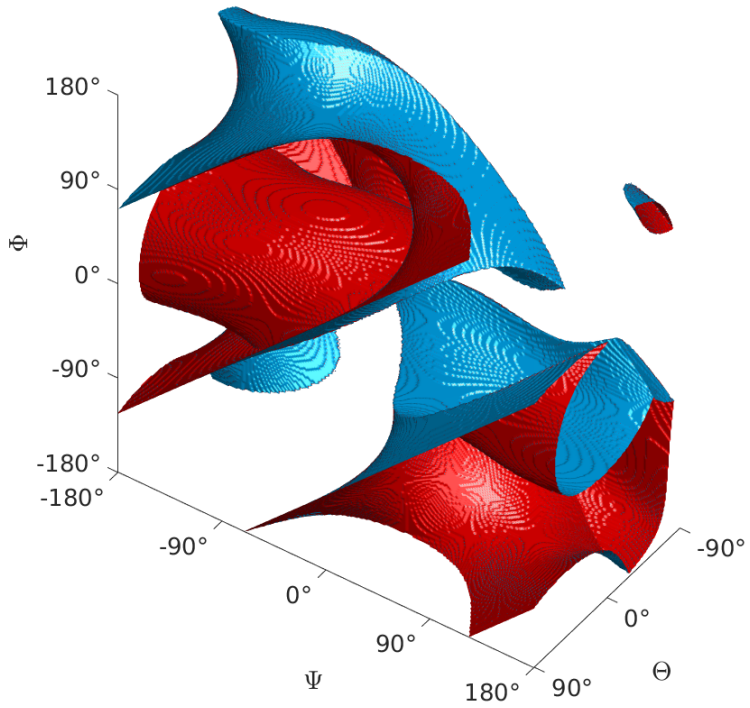

Figure 3.9: Exemplary orientation boundary surfaces in Euler angles. The red side is blocked, the blue side is reachable.

Using a spherical coordinate system instead of a Cartesian one and mapping azimuth, altitude and radius to yaw, pitch and roll (see Figure 3.10) could prove to be a little more intuitive since azimuth and yaw as well as altitude and pitch are similar. Only the mapping between axis length and roll angle has to be done in the head. This representation might be comprehensible with focus and practice but it is still not considered to be intuitively understandable by the author.

The next representation uses the rotation axis and angle convention, where the rotation angle is encoded linearly in the axis length. A length of 1 is mapped to a 180° rotation about the axis. The point (0, 0, 0) in the chart corresponds to no rotation, (0.5, 0, 0) corresponds to 90° rotation about the x-axis and so on. The result can be seen in Figure 3.11.

In the last representation, the three axes represent three components q_x, q_y, q_z of quaternions that correspond to tool orientations. The fourth component can be inferred by $q_w = \sqrt{1 - q_x^2 - q_y^2 - q_z^2}$ since orientation quaternions must have length 1. Figure 3.9 shows the resulting chart.

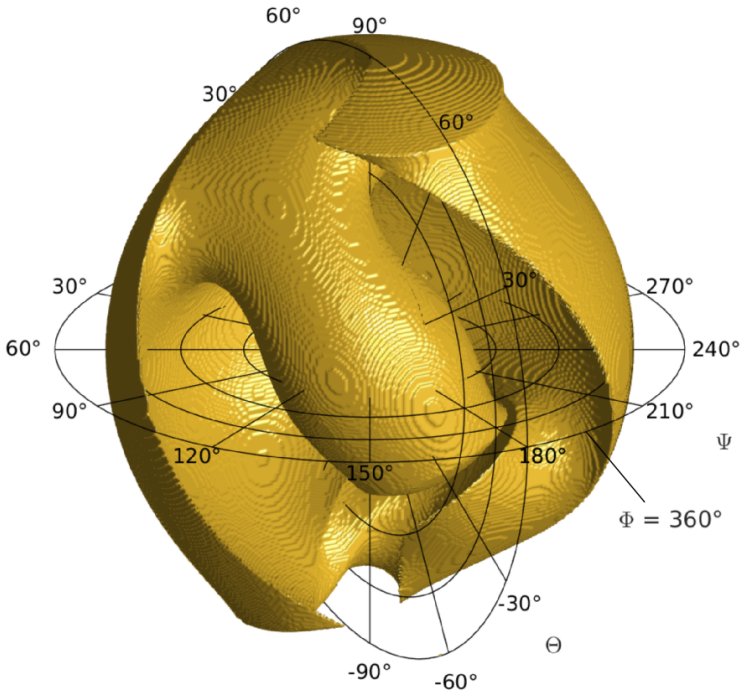

Figure 3.10: Exemplary orientation boundary surface in Euler angles, shown in a spherical coordinate system where azimuth corresponds to yaw, altitude to pitch and radius to roll.

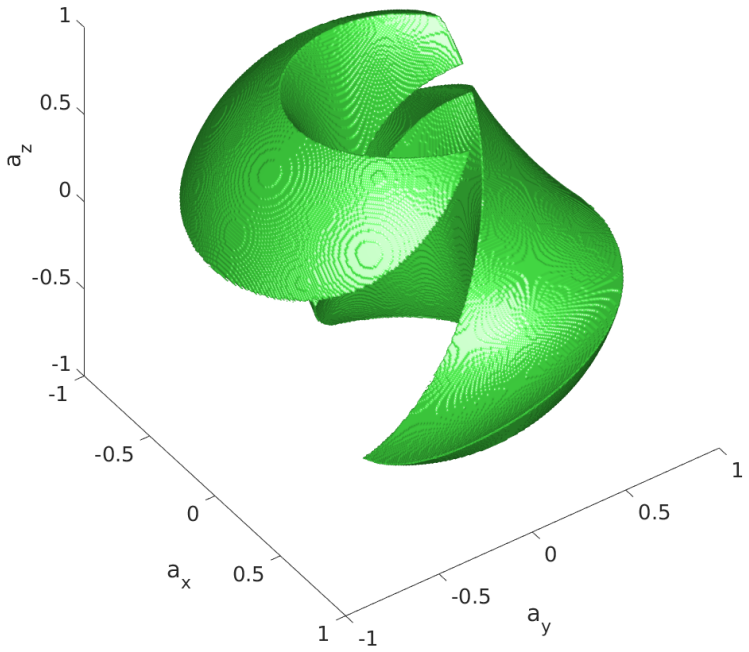

Figure 3.11: Exemplary orientation boundary surface in rotation axis angle convention.

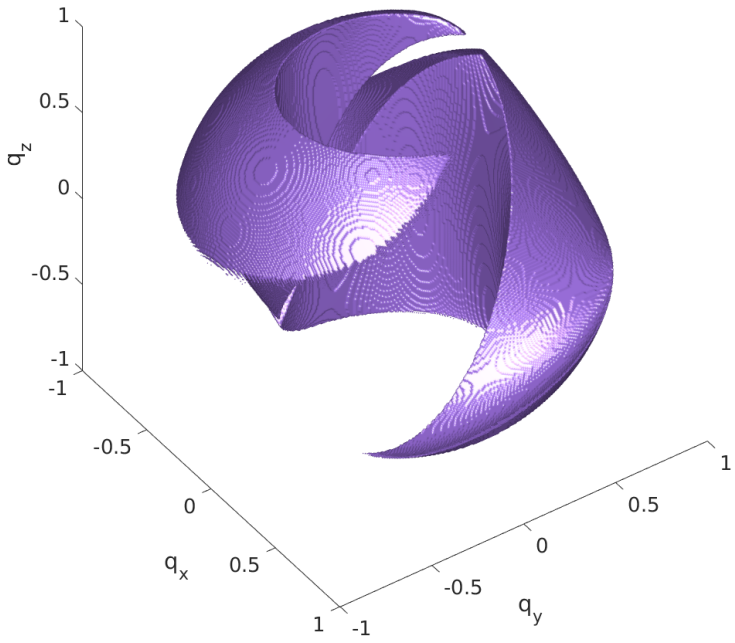

Figure 3.12: Exemplary orientation boundary surface where orientations are represented by three quaternion components. The fourth component can be inferred.

Axis angle representation and three quaternion components look very similar. This is due to the fact that the vector (q_x, q_y, q_z) of a quaternion already points along the rotation axis, only its length is not proportional to the rotation angle θ but to $\sin(\theta/2)$. Both concepts are considered to be equally hard graspable as the spherical Euler angles map above.

Although these kinds of chart may be the only way of visualizing the generally accessible orientation workspace, the idea of a bounding surface in an orientation domain was not pursued further due to its incomprehensiveness.

3.3.2.2 Sphere of Colored Rotation Axis Tips

Visualizing only the directly accessible orientation workspace requires less information to be shown, which can help to improve the perspicuity. The devised concept employs a sphere of colored rotation axis tips. Consider a sphere with radius r, centered at the TCP. Each point $\vec{s}$ on the sphere represents a rotation axis through the TCP and $\vec{s}$. The color of the point will be determined by the angle that the robot is capable of rotating the TCP about this axis. An exemplary result of this procedure can be seen in Figure

3.13. However, the resulting image was still hard to understand and it was simplified further.

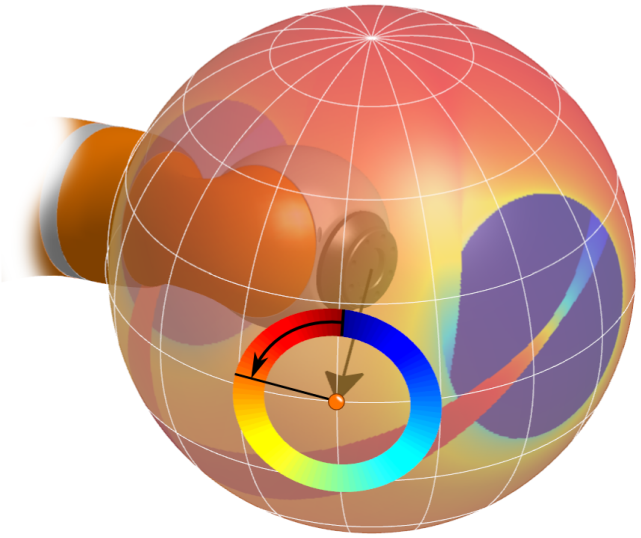

Figure 3.13: Exemplary sphere of colored rotation axis tips: In its current position, the robot flange could rotate $80°$ about the drawn axis (black arrow), which determines the tip color. The whole sphere comprises of colored rotation axis tips.

3.3.2.3 Tilt Sphere

For general tool shapes (or load shapes) with no dominant axes, the sphere of colored rotation axis tips might be the best option.

However, most tools do have a dominant axis. For the tilt sphere, the orientation is split into roll component (about the dominant axis) and tilt component (about all axes that are perpendicular to the roll axis). The first piece of information – how far can the tool be rolled clockwise and counterclockwise – simply consists of two scalars, which are easy to display, for instance by a little gauge. For the second part, we create another sphere around the TCP. Now we tilt the tool as far as kinematically possible about an axis through the TCP, which has to be perpendicular to the dominant tool axis. Where the dominant tool axis intersects the sphere after tilting, we draw a point. If we do this for all axes that are perpendicular to the dominant tool axis, we end up with a line on the sphere, that borders the orientational workspace in an understandable way, as illustrated in Figure 3.14.

The downside is that a lot of data is lost again. The information that is left, corresponds to the information that would be shown within one colored great circle and two dots on the sphere of colored orientation axis tips, since

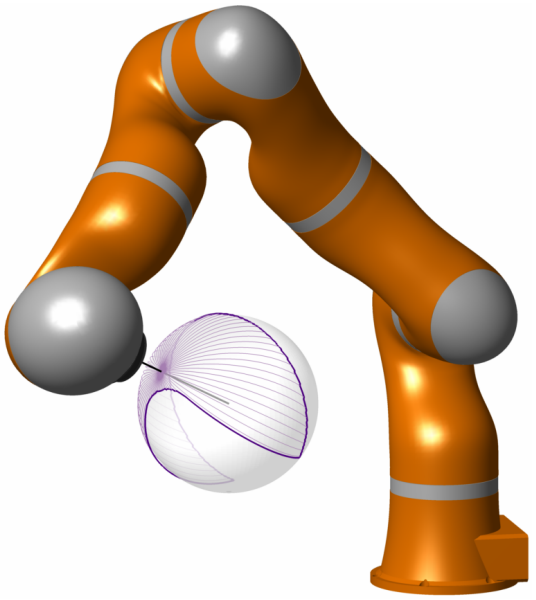

Figure 3.14: Tilt sphere for the depicted LBR IV posture with a 15 cm long tool (black line): The thick purple line shows the limit of tilting movements around the TCP in all directions, while both the TCP position and the tool roll angle are kept constant.

only rotation limits about axes that are perpendicular to the dominant tool axis and the two roll limits are considered. However, the brain does a great job at reconstructing the big picture from multiple slices of it and since the sphere updates instantly when the tool translates and rotates, the viewer can gradually form the complete image.

3.3.3 Total Orientation Workspace

The total orientation workspace can be seen as a generally accessible workspace that forms a hybrid between translation and orientation workspace. It is based on multiple constant orientation workspaces that have to be generated for several representative tool orientations for a specific task. Afterwards, the intersection of all results is computed and the outcome will be the boundary around the set of points that is reachable in all desired orientations, see Figure 3.15.

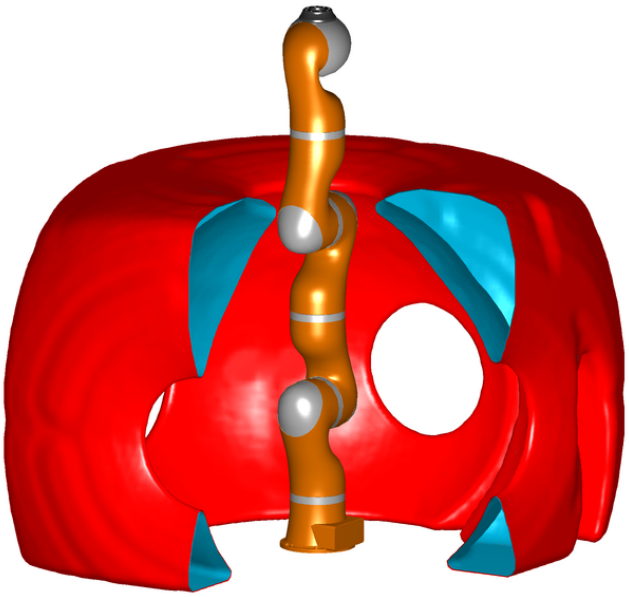

Figure 3.15: Generally accessible workspace of the LBR IV for 24 different orientations. Each point within the depicted volume is reachable in all orientations.

3.4 Redundancy Handling

As explained in section 2.2.2.3, the approach for optimizing the elbow position within the scope of this thesis is the gradient projection method. A local gradient descent on a cost function is chosen because the global path (i.e. the future trajectory) is assumed to be unknown to the system.

3.4.1 Cost Function Combining

A novel cost function that is based on the current pose and elbow position is designed to avoid joint limits and singularities. The gradient of this cost function is projected onto the null-space parameter and the parameter is varied accordingly until a local minimum is reached.

In most cases, the domain of a cost function includes admissible and inadmissible regions. In our case, the joint values within the limits are admissible and values beyond the limits are not admissible. A common approach to cost function design is to make its value reach infinity before the admissible region is left. The partial cost functions for all criteria are added and as a parameter reaches the limit of the admissible region, the combined cost function approaches infinity.

Based on these considerations, Zghal et al. [101] propose the following performance criterion

$$H_J = \sum_{i=1}^{7} \frac{(\theta_{\text{Max},i} - \theta_{\text{Min},i})}{(\theta_{\text{Max},i} - \theta_i)(\theta_i - \theta_{\text{Min},i})} \,, \tag{3.15}$$

where θ_i is the angle of the i^{th} joint which has to be between $\theta_{\text{Min},i}$ and $\theta_{\text{Max},i}$. An exemplary plot for $\theta_{\text{Min},i} = -120°$ and $\theta_{\text{Max},i} = 120°$ can be seen in Figure 3.16.

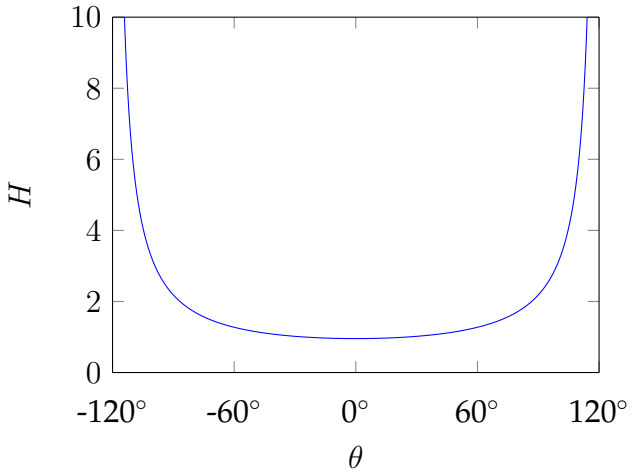

Figure 3.16: Exemplary cost function for one joint according to Zghal et al. [101]

However, this approach has a major drawback: Once the infinite barriers are crossed (which can happen numerically), the cost function becomes undefined or returns impractical values, which can entail an unstable behavior near the barriers.

Within this work, this limitation is overcome by modifying the cost function properties as follows: Each partial cost function c_i is designed to be negative in admissible regions and positive in inadmissible regions. The combined cost C is then not computed by the sum of the partial functions but by

$$C = \begin{cases} \dfrac{1}{\sum\limits_i \frac{1}{c_i}} \,, & \text{if } c_i < 0 \;\forall\, i \\ \sqrt{\sum\limits_{i:c_i>0} c_i^2} \,, & \text{otherwise.} \end{cases} \tag{3.16}$$

The function is plotted for $C(c_1, c_2)$ in Figure 3.17. If all partial costs c_i are negative, the merged cost C is the reciprocal of the sum over all c_i's reciprocals. Otherwise, it is the Pythagorean sum of all positive partial costs. This way,

the combined cost function is positive (inadmissible) if at least one partial cost function is positive and negative (admissible) if all partial costs are negative. It is also steady and monotonically increasing with respect to each partial value (apart from being discontinuously differentiable where all $c_i = 0$). The partial cost functions are designed in a way so that they cross zero when a joint reaches its limit or approaches a singularity.

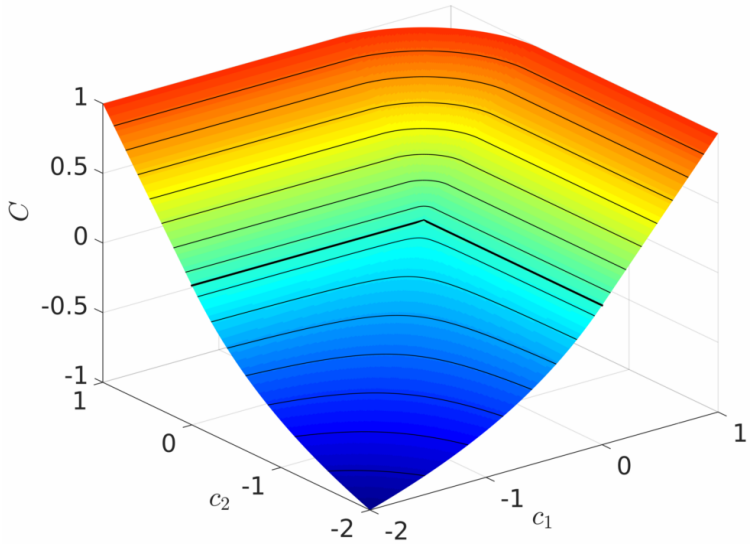

Figure 3.17: Merged cost function from two partial costs. If either c_1 or c_2 become inadmissible (> 0), C crosses 0 as well.

The benefit of this approach to cost function design is the fact that the transition from admissible to inadmissible regions is numerically much easier to find when it is a zero crossing, as opposed to a point where the cost reaches infinity and is undefined or meaningless outside of that limit. This is for instance used to obtain smooth surfaces using the marching cubes algorithm, as described in section 4.2.

3.4.2 Partial Cost Functions

As stated in section 2.1.3, a singularity is reached if two joint axes become coaxial, which is the case when the hinge joints are stretched. The alignment of the first and the last axis does not depend on the value of the null-space parameter, so it does not have to be included in the cost function.

The chosen formulae for the partial cost functions are of the form

$$c(q) = k_0 \cos\left(\frac{k_1}{|q| + k_2} + k_3\right) . \tag{3.17}$$

The parameters are chosen so that, for joint limit functions, $c(0) = -1$, $c'(0) = 0$, $c(\pm 180°) = 1$, $c'(\pm 180°) = 0$ and $c(\pm 170°) = 0$ for axial joints and $c(\pm 120°) = 0$ for hinge joints. For the singularity avoidance, the boundary conditions are $c(0) = 1$, $c'(0) = 0$, $c(\pm 180°) = -1$, $c'(\pm 5°) = 0$.

The resulting values are

$$
\begin{aligned}
c_{\text{axial, limit}}: &\quad k_0 = 1, &\quad k_1 = -0.6554, &\quad k_2 = -3.3379, &\quad k_3 = -0.1963 \\
c_{\text{hinge, limit}}: &\quad k_0 = 1, &\quad k_1 = -19.7392, &\quad k_2 = -6.2832, &\quad k_3 = -3.1416 \\
c_{\text{hinge, sing.}}: &\quad k_0 = -1, &\quad k_1 = -0.2988, &\quad k_2 = 0.0924, &\quad k_3 = 3.2340
\end{aligned}
$$

The cost functions are plotted in Figures 3.18, 3.19 and 3.20.

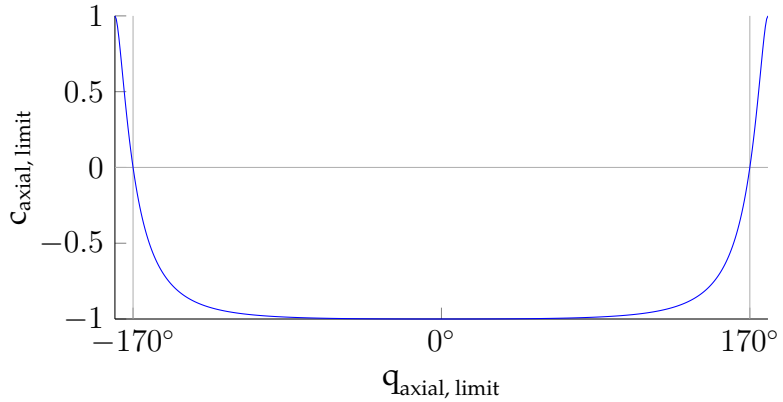

Figure 3.18: Partial cost function to avoid the limits of the axial joints ($\pm 170°$ for the LBR IV).

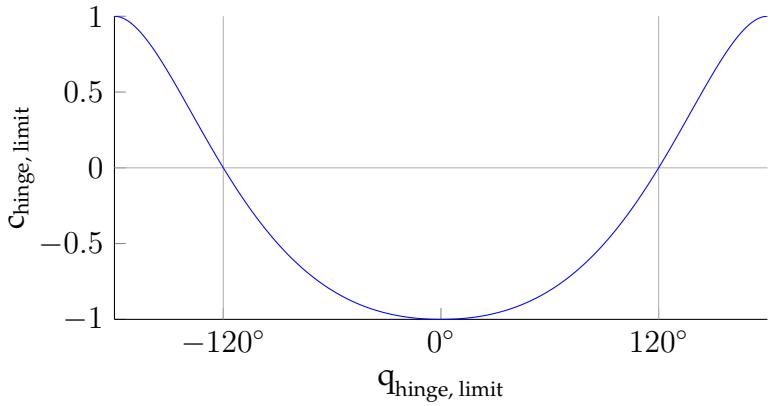

Figure 3.19: Partial cost function to avoid the limits of the hinge joints ($\pm 120°$ for the LBR IV)

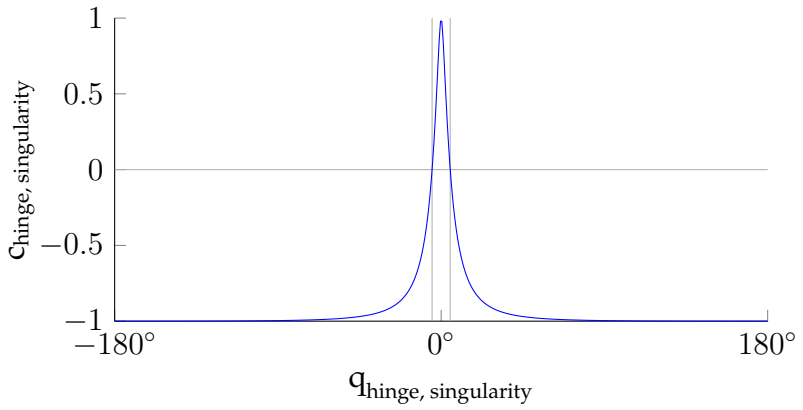

Figure 3.20: Partial cost function to avoid singularities, applied to hinge joints, set to $\pm 5°$

In order to prevent large reconfiguration motions, an additional cost was added for each joint:

$$c_{\text{norecfg}} = -\cos\left(\frac{q - q_{\text{old}}}{2}\right) , \tag{3.18}$$

where q_{old} is the previous joint position and q the current one. This partial cost makes a joint movement of over 180° in one step unfeasible.

More partial cost functions can easily be added. For instance, the elbow can be kept upwards by trend by including

$$c_{\text{elbow}} = \cos(\eta) - 2 \tag{3.19}$$

into the combined cost function. Keeping the elbow upwards during operation causes the robot to be less obstructive and both sides can easier be accessed by personnel, which can be desirable for instance in a surgical scenario.

Another possibility is to include the distance between the elbow and the closest obstacle to it. An external camera system and a computer system to find the distance to the closest object are required. This was implemented in the OP:Sense research system for robotic surgery [9].

$$c_{\text{obstacle}} = -\|\vec{p}_e - \vec{p}_{\text{obstacle}}\| . \tag{3.20}$$

4 Implementation

Within this chapter, the utilized hard- and software and their interplay are pointed out. Afterwards, implementation details for the devised workspace visualization techniques are presented. In the end, the user interface, consisting of visualization and input, is introduced.

4.1 System Setup

The software is implemented in C++ and runs on an Intel® Core™ i3-2100 Processor (Dual-Core, 3.10 GHz) with 4 GB RAM, running Ubuntu 14.04. The parts that are implemented for GPU are written in OpenCL (using simple-opencl [41] as framework) and are executed on an NVIDIA GeForce GT 440 (96 Cores, 1 GB VRAM) graphics card.

For online visualization, a dedicated PC communicates with a KUKA LBR IV[83] robot via Fast Research Interface (FRI) [66] and streams its joint angles into a ROS (Indigo) network [47, 60]. The visualization client (the PC described in the last paragraph) reads the angles, generates the workspace boundary surfaces and publishes them as visualization markers into the ROS network, from where they are finally visualized in RViz [38], running on the same PC.

For user input, two different modalities are used: a Microsoft Xbox 360 Controller as main input device and a 3Dconnexion SpaceNavigator was also integrated as an alternative.

The whole system architecture is depicted in Figure 4.1. The KRC box is the KUKA Robot Controller that provides an FRI via UDP, the RC is the dedicated PC that serves as interface between ROS and FRI, the WX computer runs the Workspace Exploration.

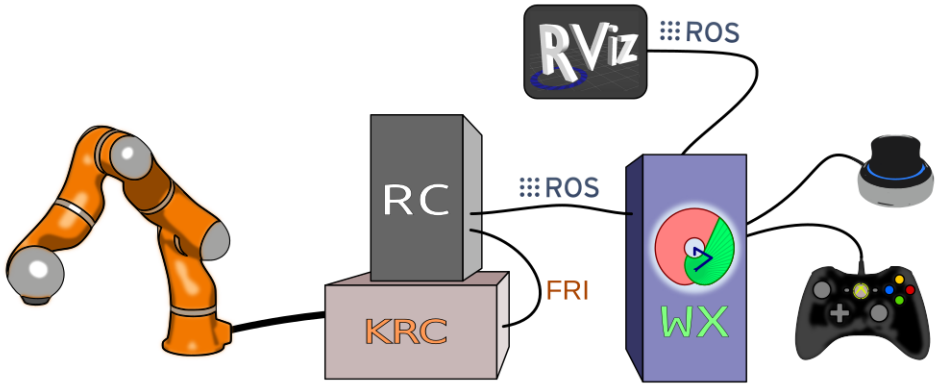

Figure 4.1: System setup: Robot controller, RViz visualization and input devices are connected to the workspace exploration computer.

4.2 Generally Accessible Workspace Computation

The generally accessible constant orientation workspace is computed using an inverse kinematics approach. First, the workspace is rasterized using a cubic grid. Then, the general inverse kinematics problem is solved for each grid node. This step can be parallelized and is executed on the GPU for speed up. If there is a solution for an interval of null-space parameters, the node is accessible, otherwise it is not. Afterwards, the hull around all accessible nodes is computed using the marching cubes algorithm (implemented by Borke [14]).

4.2.1 Inverse Kinematics

Both the general as well as the particular inverse kinematics solution have been implemented to run on graphics cards that run OpenCL, which includes Nvidia and ATI products. The algorithms are realized in accordance to section 3.1.

The general inverse kinematics computation is very complex and prone to implementation errors. It involves spherical geometry, many conditional branches and the resulting intervals are transformed, sorted and merged. In order to verify the implementation, millions of random poses have been checked. The general inverse kinematics solution was computed and asserted by computing the particular inverse kinematics solution for null-space parameters around the borders of intervals and checking whether joint limits were exceeded as predicted. In many cases, the borders were found to be

slightly but notably incorrect if single precision was used. Whether high accuracy is required or not has to be considered in further steps, since double precision computation runs significantly slower on many graphics cards.

4.2.2 Bounding Box Determination

In order to determine the limits for the space that needs to be sampled, the bounding box of the workspace has to be found. The constant orientation workspace of the TCP is equal to the constant orientation workspace of the wrist center, just shifted by an offset, as illustrated in Figure 4.2.

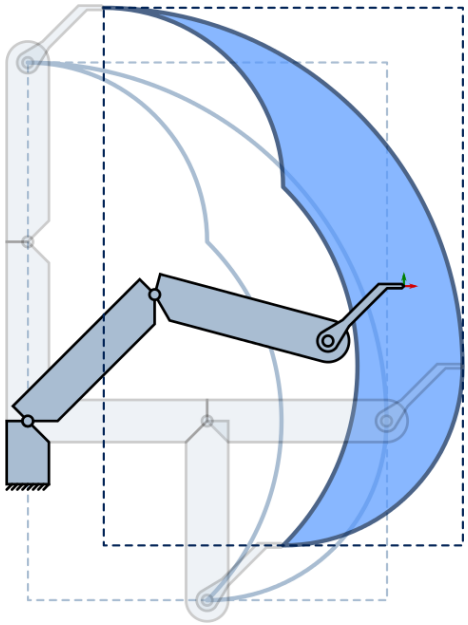

Figure 4.2: The constant orientation workspace and its bounding box are identical for wrist center and TCP, just shifted by an offset.

The offset $\vec{o}$ depends on the robot hand size, the tool matrix and the tool orientation and can be computed by

$$ {}^{b}T_{w} = {}^{b}T_{\text{tcp}} \cdot {}^{f}T_{\text{tcp}}^{-1} \cdot {}^{w}T_{f}^{-1} \tag{4.1} $$

$$ \vec{o} = {}^{b}T_{\text{tcp}\ 1..3,4} - {}^{b}T_{w\ 1..3,4}\ , \tag{4.2} $$

where ${}^{y}T_{x}$ is a homogeneous 4×4 matrix describing position and orientation of the coordinate system x within the reference coordinate system y. The upper left 3×3 matrix of ${}^{b}T_{\text{tcp}}$ describes the target orientation of the tool. The upper right 3×1 vector usually contains the position, but is canceled out by

(4.2) and can therefore be set to zero. ${}^{f}T_{\text{tcp}}$ contains the transformation of the tool and ${}^{w}T_{f}$ contains the robot's flange frame with respect to its wrist frame. The reference index b stands for the robot base.

Now all that is left is to determine the bounding box of the wrist center workspace of the robot, which – for the LBR IV – is independent of the flange orientation. It can be seen in Figure 4.3. Along $\pm$x and $\pm$y direction, the robot can extend its wrist by 79 cm. It can reach -28 cm in -z and 110 cm in +z direction.

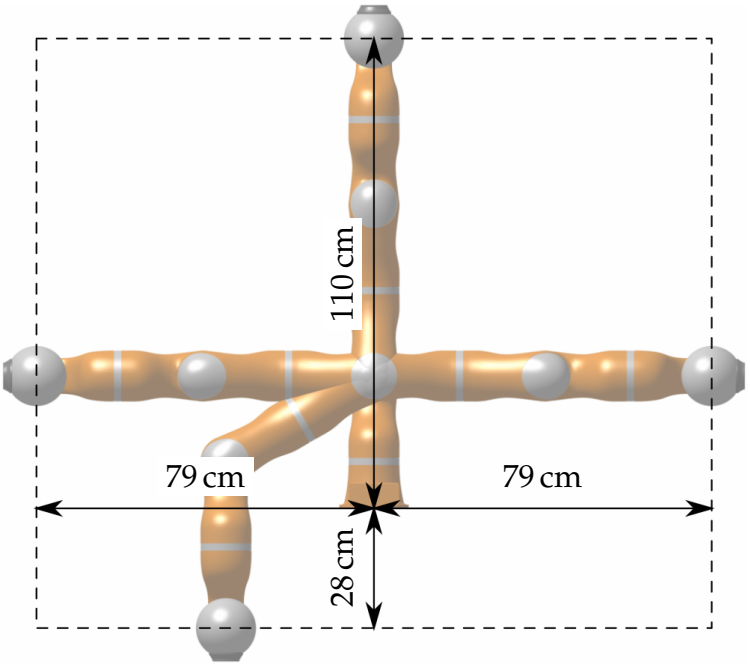

Figure 4.3: Bounding box of the workspace for the wrist center of the LBR IV.

4.2.3 Smoothing the Boundary Surface

Rendering a marching cubes surface on a grid with binary node values results in a discrete-looking, stepped surface, which can be difficult to interpret since the alias effect can conceal the underlying surface structure. To improve the mesh quality, the nodes can have a scalar value associated to them. The mesh vertices are then shifted along the grid edges in order to better approximate the zero crossing of the surface. The difference is illustrated in Figure 4.4. Depending on the gradient of the scalar values and the mesh resolution, the result can still contain ripples, however the overall quality is much better.

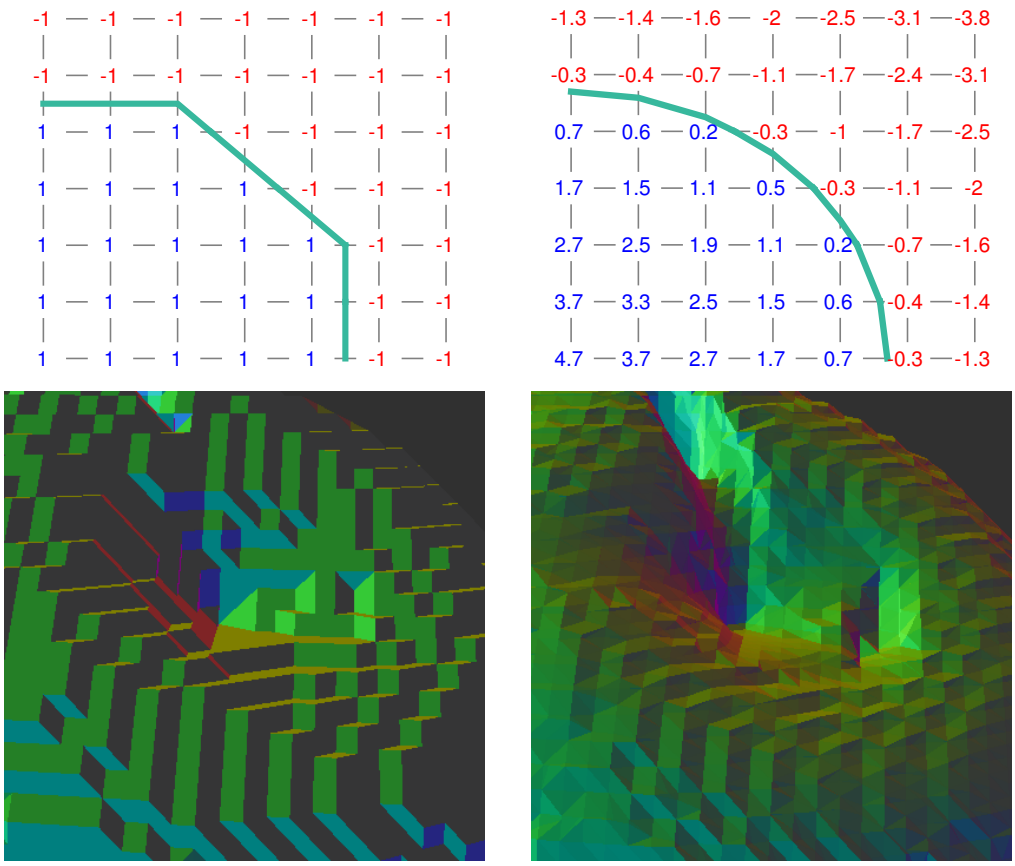

Figure 4.4: Comparison of marching cubes algorithm: With binary node values (left), the surface is stepped. Using scalar node values smoothens the surface.

Two different functions are used in order to determine the scalar field. The first function s_1 decides whether a target node is within arm range ($\leq r_{\text{arm, max}}$) but not too close for the elbow joint ($\geq r_{\text{arm, min}}$). It depends on the distance $r_{\text{arm, target}}$ between the shoulder position and the target wrist position. It is computed as follows:

$$r_{\text{arm, target}} = \|\vec{p}_{w,\text{ target}} - \vec{p}_s\| \tag{4.3}$$

$$r_{\text{arm, max}} = l_{se} + l_{ew} \tag{4.4}$$

$$r_{\text{arm, min}} = \sqrt{l_{se}^2 + l_{ew}^2 - 2 l_{se} l_{ew} \cos(\pi - q_{4,\text{ max}})} \tag{4.5}$$

$$s_1 = \begin{cases} r_{\text{arm, max}} - r_{\text{arm, target}} & \text{if } r_{\text{arm, target}} > \frac{r_{\text{arm, max}} + r_{\text{arm, min}}}{2} \\ r_{\text{arm, target}} - r_{\text{arm, max}} & \text{otherwise} \end{cases} . \tag{4.6}$$

The cost is zero at the boundary spheres, increases outside the outer sphere and inside the inner sphere and decreases in between down to a minimum halfway between the two shells.

For the second scalar function s_2, the computed free elbow circle intervals from a vertex' complete inverse kinematics solution are added up and a threshold is subtracted:

$$s_2 = \int_0^{2\pi} \eta_{\text{free}} \, \mathrm{d}\eta - \eta_{\text{threshold}} \, . \tag{4.7}$$

This way, only node points where the elbow has a certain leeway are included into the rendered boundary surface. This is necessary in order to obtain a smooth mesh since the scalar value has to cross 0 (it may not stay at 0). However the minimum value of the integral in equation (4.7) is 0 (if there is no valid elbow posture), so it has to be shifted slightly.

The larger $\eta_{\text{threshold}}$ is chosen, the more leeway the elbow has at each included pose and the smoother the surface becomes.

The final scalar value s at each node is the minimum of s_1 and s_2.

4.2.4 Total Orientation Workspace Determination

The total orientation workspace is the intersection of the constant orientation workspaces for all orientations in a predefined set. These will be called sub-workspaces in this section. The computation is similar to the one of a constant orientation workspace, only few additional steps are required. At first, the bounding box has to be determined. It is the intersection of the bounding boxes of all sub-workspaces. Therefore, the limit in +x direction is the minimum of the +x limits of all bounding boxes of the sub-workspaces and the limit in -x direction is the maximum of the -x limits. The same holds true for $\pm$y and $\pm$z.

The remaining bounding box (which may be an empty set) is sampled and each node has to be tested for reachability from all orientations that were defined. The resulting node value is the conjunction of all results at that point. If a smooth surface is generated, the maximum over the scalar cost values for all orientations at that point is used for the node value. Afterwards, the marching cubes algorithm can be applied.

4.3 Directly Accessible Workspace Computation

In order to generate the translatory directly accessible workspace, virtual motion has to be performed in all directions and the end points have to be visualized.

4.3.1 Probing Rays in All Directions

For sampling all directions in three-dimensional Euclidean space, an icosahedral sphere grid is generated (see Figure 4.5) using the sphere grid library by Burkardt [15].

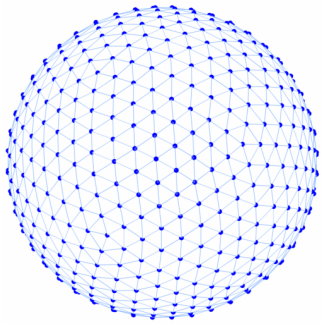

Figure 4.5: Icosahedral sphere grid for probing directions.

The number of probing directions is adjustable. Splitting the icosahedron edges into n segments and filling the faces with vertices accordingly yields $10n^2 + 2$ vertices, one vertex corresponding to one probing direction.

Each direction is sampled along its ray. The step width increases with the distance from the TCP in order to cover both a large range and a high accuracy close to the TCP. If a joint limit is hit or the position is too far to be reached by the robot, the workspace boundary will be drawn through the last feasible point. The process is illustrated in Figure 4.6.

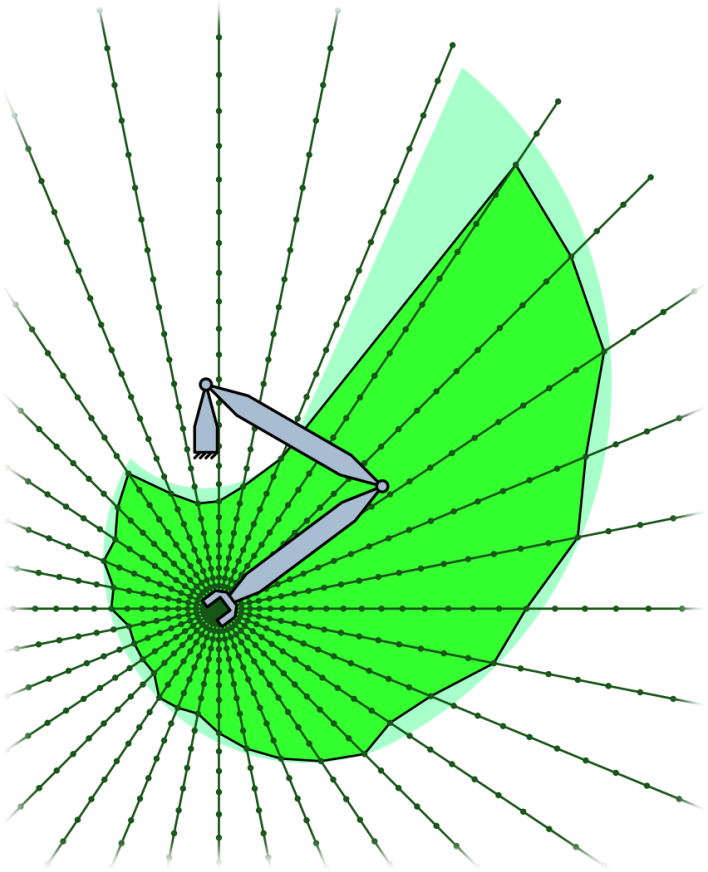

Figure 4.6: Two-dimensional visualization of the sampling process: The green lines represent probing rays along which the test points are drawn. The resulting area (bright green) will be contained within the directly reachable workspace (light green) and show alias effects around the edge.

4.3.2 Null-Space Optimization

At each step along the ray, the null-space parameter of the robot is locally optimized using a gradient descent method on the presented cost function: The parameter is incrementally increased or decreased, depending on the initial slope. At each optimization step, the local cost is computed and compared to the value from the last optimization step. If a cost minimum was passed or a certain limit is reached, the null-space parameter from the last optimization step is fixed and the propagation along the probing ray is continued.

The whole process is summarized in the following pseudo code.

```
vector probeRays[] = createIcoSphere(splitFactor) // unit vectors
vector currentTCP = getCurrentTCP() // current tool center point
vector currentNSP = getCurrentNSP() // current null-space parameter

// explore all probe rays:
parallel for i = 0 to numProbeRays-1 // running on GPU

  // explore along one probe ray:
  vector probeTCP = currentTCP

  for q = 0 to 1 step 1/numSteps // control variable
    d = q^2 * scanRange // step distance from TCP

    vector probeTCPBefore = probeTCP
    probeTCP = currentTCP + d * probeRays[i]

    // explore null-space at one point on probe ray:

    scalar cost0 = cost(probeTCP, currentNSP - epsilon)
    scalar cost1 = cost(probeTCP, currentNSP + epsilon)
    int gradFallDirection = -signum(cost1 - cost0)

    probeNSP = currentNSP
    costNSP = cost0

    do
      costNSPBefore = costNSP
      probeNSP = probeNSP + gradFallDirection * deltaNSP
      costNSP = cost(probeTCP, probeNSP)
    while probeNSP - currentNSP < deltaNSPmax // still within range
        && costProbe < costBefore // cost still falling

    if costNSPBefore > 0 // no solution, probe ray ends one step before
      displayHullVertex(probeTCPbefore)
    end if

  next
next
```

Figure 4.7 shows an exemplary progression of one virtual probing ray. The horizontal axis represents the progress along the ray while the vertical axis represents the null-space parameter. The background color indicates the

cost at this point. Green indicates negative cost (admissible), red indicates positive cost (inadmissible); the brighter the color, the higher the absolute value. The white line displays the optimized path. The white funnels display the maximum null-space variation for one step ($\pm 30°$), the maximum slope decreases as the step width increases.

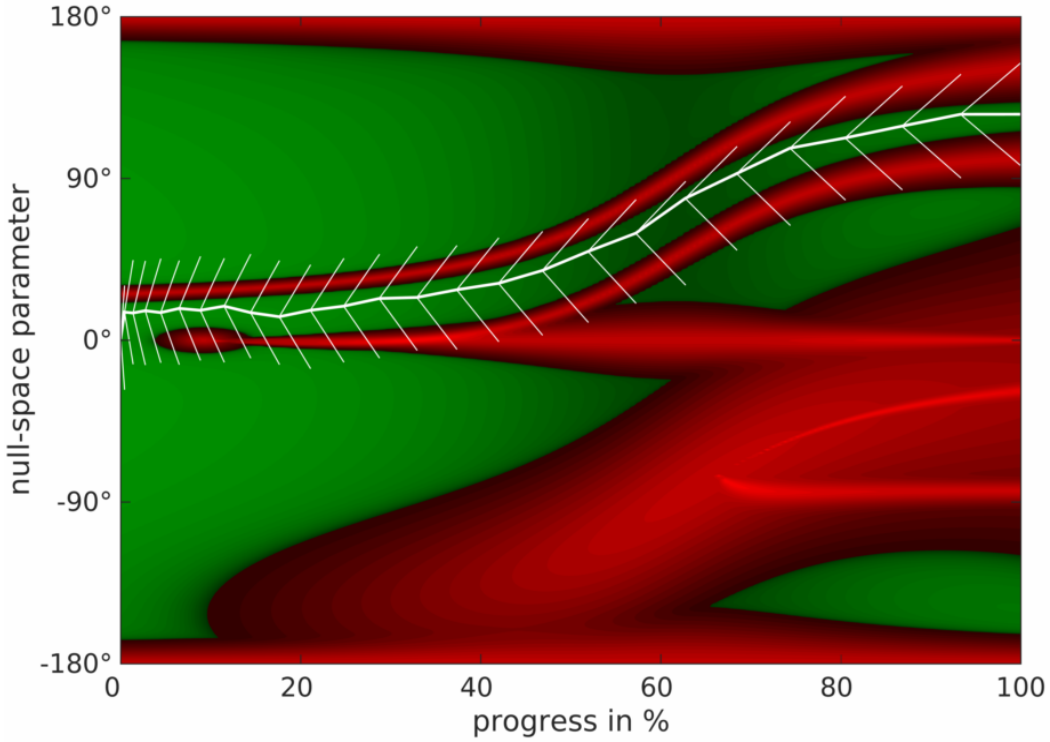

Figure 4.7: Progress of null-space optimization: The cost value is shown as a function of the progress along an exemplary linear motion and the null-space parameter. Red areas are blocked (the brighter, the higher the cost), green areas are accessible (the brighter, the lower the cost). The white path is found by the local optimization, the funnels indicate the scanning range per step.

4.3.3 Computation of the Tilt Sphere

The computation of the tilt sphere is similar to the computation of the directly accessible workspace for translation with the only difference that the TCP is not translated along rays in all spatial directions, but its pose is rotated around a set of axes that are perpendicular to the dominant tool axis, as described in section 3.3.2.3.

Note that the tilt sphere is computationally far less expensive, since only a one-dimensional manifold of tilt axes has to be discretized (plus the two

roll limits, which are computationally insignificant), while the translatory directly accessible workspace requires sampling a two-dimensional manifold of spatial directions.

4.4 User Interface

The user interface consists of the graphical output in order to display the computed workspace limits and an input device (gamepad or SpaceNavigator) for interacting with the system. Both elements are discussed below.

4.4.1 Workspace Visualization

Because of the broad acceptance of ROS as robotics framework, its visualization interface RViz has been chosen to display the workspace boundaries. Due to ROS' popularity and modularity, the workspace visualization can easily be integrated into the many existing virtual environments that use it. Figure 4.8 gives an overview of the basic RViz GUI layout.

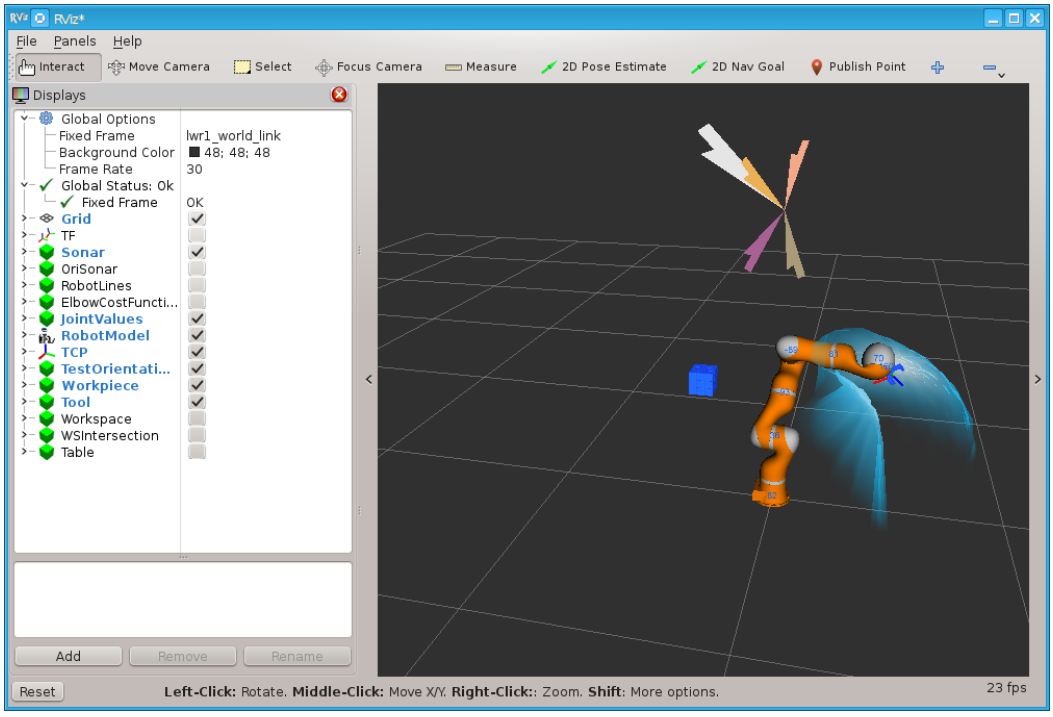

Figure 4.8: GUI layout containing the selection list for displayed elements (left), and the virtual environment (right).

The interface that RViz offers accepts different types of markers [39] which have to be sent via ROS messages. Within the scope of this thesis, the used markers are: predefined mesh files, triangle lists, line strips, spheres, text labels and coordinate frames.

The robot is constructed from predefined Collada meshes, the link relations are defined in RViz' URDF file format. A dedicated node accepts ROS joint messages, and sends the poses of the robot links to RViz. The node was developed within [8]. Tool and workpiece can also be visualized using predefined meshes. The TCP is shown as a coordinate system using the tf mechanism [31]. Half way between robot joint positions and camera position (so they stay in the foreground), the current joint values are displayed as text labels. This way, it is easily visible, which joint is approaching its limit. The described elements are depicted in Figure 4.9

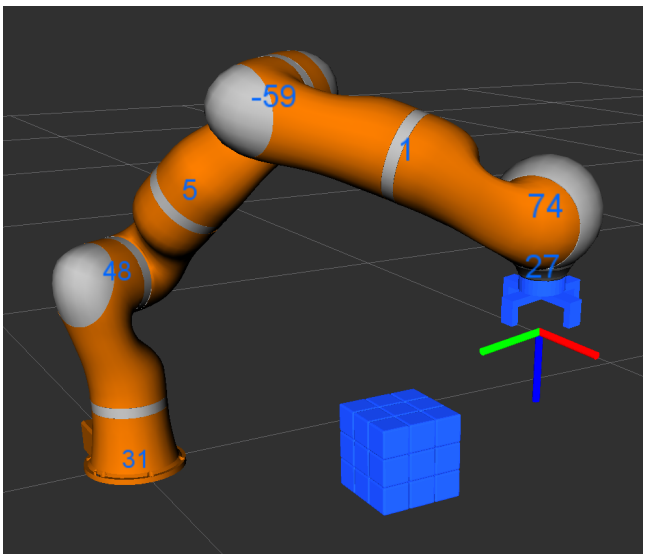

Figure 4.9: Visualization of the robot with current joint angles (in degrees), an exemplary workpiece, an according gripper and the TCP coordinate system.

The triangle mesh for visualizing the constant orientation workspace and also the total orientation workspace (see Figure 4.10) is transmitted as triangle list marker. The triangles are semi-transparent so all parts of the workspace are visible. Since triangle markers are currently not shaded in RViz, the triangles are colored depending on their surface normals in the common $(r, g, b) = \frac{1}{2}((n_x, n_y, n_z) + (1, 1, 1))$ scheme so the structure is visible.

The orientations that are considered within the total orientation workspace are visualized using half arrows (see Figure 4.11). They represent the dominant tool axis and are better understandable than coordinate axes when many of them are depicted around a single point. Half (in contrast to complete) arrows

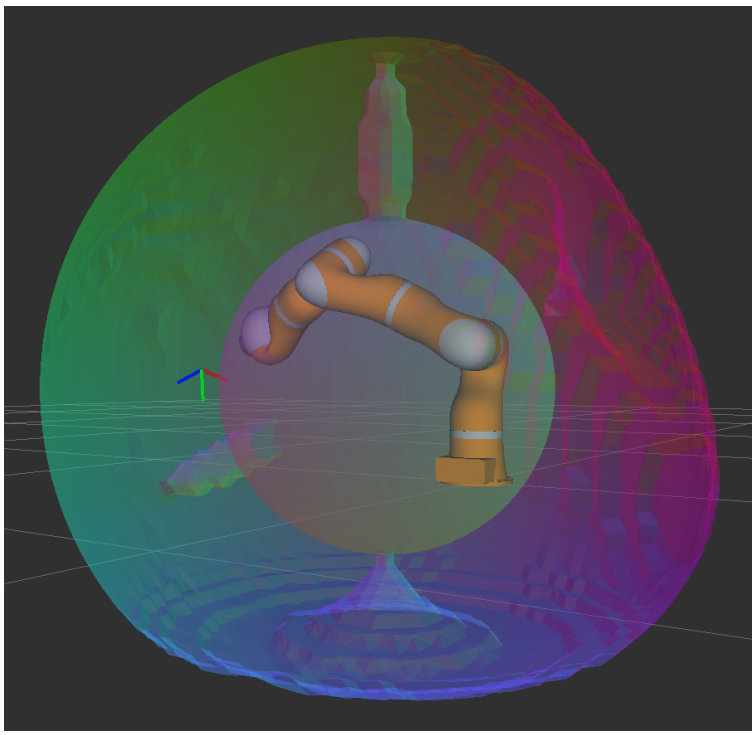

Figure 4.10: Visualization of the constant orientation workspace for the current tool offset and TCP orientation.

also make the roll angle of the tool unambiguous. Each half arrow is colored depending on its orientation, the whole cluster of arrows is transmitted as one triangle list marker.

The directly-accessible workspace for translation (see Figure 4.12) is also transmitted via triangle markers. The triangles connect the found end points of virtual motion in a way, so that if all of them were equally far away from the TCP, the triangle mesh would form the convex hull. The color of the triangles is determined via Gouraud shading (interpolation of neighboring vertex colors) and the vertices are shaded depending on their distance to the TCP. If the vertex is as far away as the maximum probing range, the vertex is completely transparent. Hence, distant borders do not obstruct the vision to the closer, more relevant motion limits. Very close to the TCP, the limit surface becomes slightly transparent white in order to emphasize the proximity. In between, the surface is colored in a stylish semi-transparent blue.

The directly-accessible workspace for orientation (the tilt sphere, see Figure 4.13) uses a sphere marker and line segments for visualization. A translucent white sphere is drawn around the TCP. As described in section 3.3.2.3, to obtain the tilt sphere, the tool is virtually tilted about (a representative subset of) all axes that are perpendicular to its dominant axis. Where the dominant

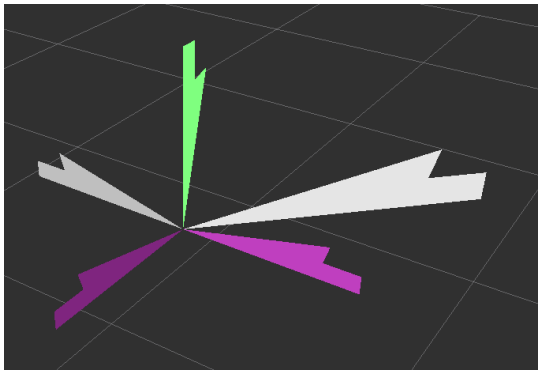

Figure 4.11: Visualization of the orientations that are considered in the total orientation workspace. The white big arrow represents the current workpiece orientation.

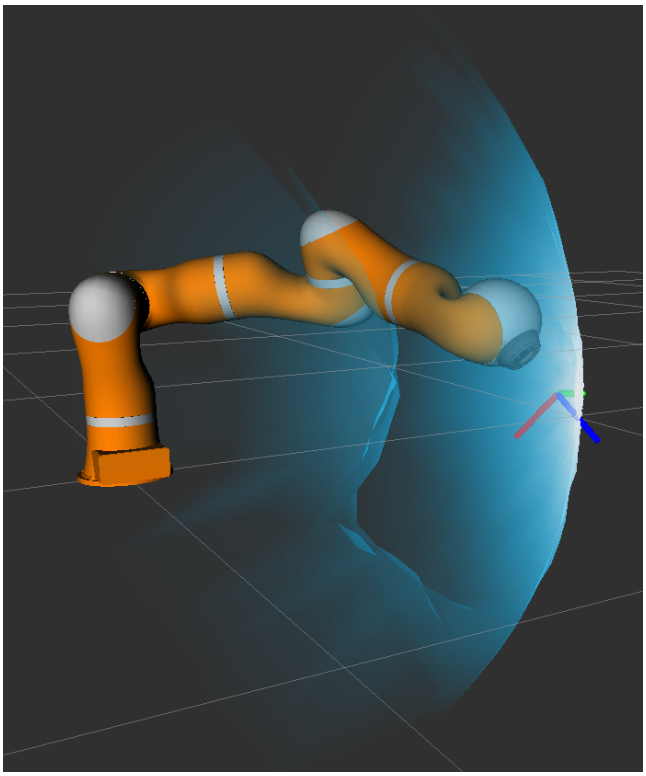

Figure 4.12: Visualization of directly accessible workspace for translation around the current TCP pose. As the TCP is very close to the boundary, it turns white. Far away from the TCP, the boundary is rendered transparent.

axis intersects the sphere at the end point of its probing motion, a point is drawn. All neighboring points are then connected via line strip and represent the motion limit. Again, the further a limit is away, the more transparent it becomes. In order to visualize the limits of the roll motion around the dominant tool axis, two lines are drawn that represent the range angle before the limit is reached.

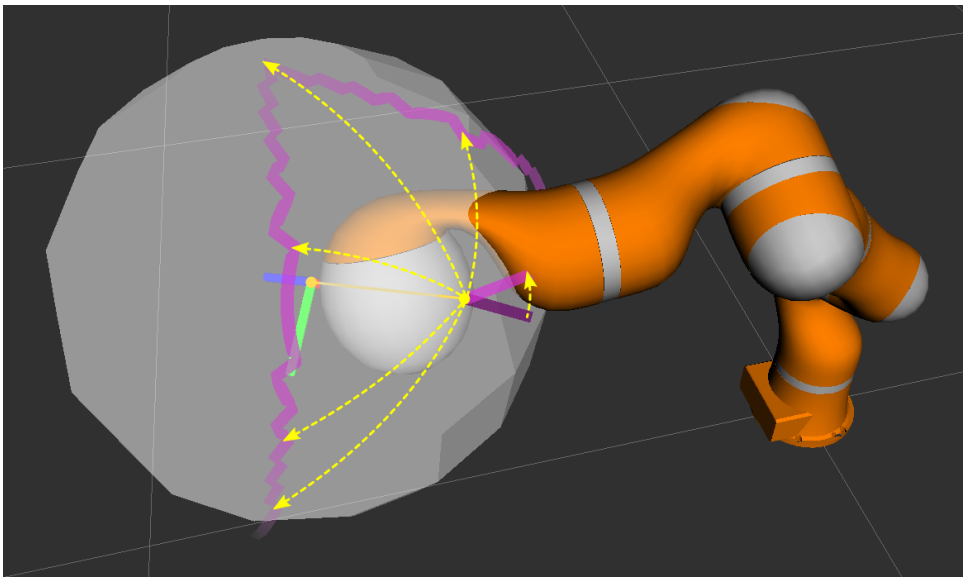

Figure 4.13: Visualization of the directly accessible workspace for orientation around the current TCP pose (tilt sphere). The yellow annotations show how the representation is to be interpreted, they are not actually displayed in the visualization. The long yellow arrows show possible tilt motions up to the boundary. The shortest yellow arrow indicates the roll limit: The purple line is the translated x-axis of the tool, which is about 30° away from the limit (pink line). The roll limit in the opposite direction is further than 90° and therefore not drawn.

4.4.2 Gamepad Control

Since the control method of the robot is not in the focus of this thesis, a simple gamepad control scheme was implemented. It is only representative and can easily be replaced by any other means of human-operated robot movement.

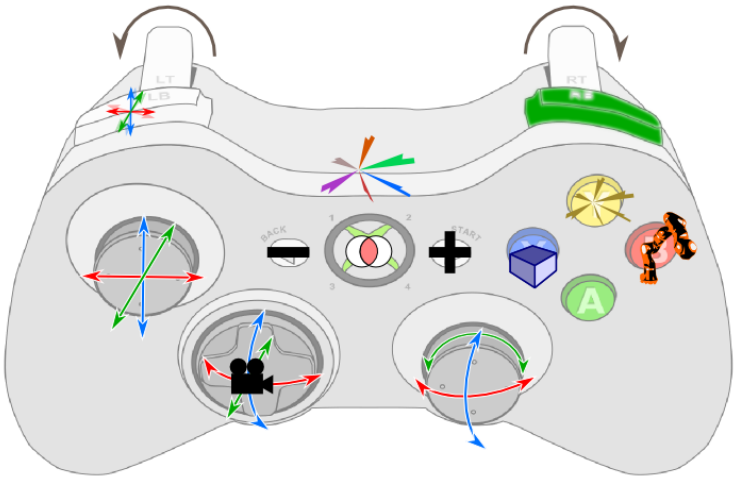

Figure 4.14: Sketch of the button mapping for robot control using the Microsoft Xbox 360 gamepad

Figure 4.14 illustrates the implemented control scheme.

- Left joystick: Translation of the controlled frame (initially within the screen plane), pressing the stick snaps the position to a predefined grid
- Right joystick: Rotation of the controlled frame (initially about arbitrary axes that lie within the screen plane), pressing the stick snaps the position to the closest in a set of predefined orientations
- Directional pad: Tilt and initially pan of the camera
- Shoulder triggers: Rotate about tool axis
- Right shoulder button: Tilt the control plane by 90° so it becomes horizontal → Left joystick up/down motion now moves away from and closer to the camera plane, right joystick left/right motion rotates about the line of sight, d-pad up/down press moves camera away from and closer to the object
- Left shoulder button: Align motion axes to world coordinate system (out of the 24 possibilities, the one that is closest to the current camera orientation is chosen)

- Start button: add current TCP orientation to the list of relevant orientations for the total orientation workspace
- Back button: remove last orientation from the list of relevant orientations for the total orientation workspace
- Right thumb buttons: Switch object that is moved → (X) workpiece, (B) robot TCP, (Y) rotate all orientations for total orientation workspace computation
- Central (X) button: compute and display total orientation workspace

As mentioned in section 4.1, a space mouse (3Dconnexion SpaceNavigator) was included as an alternative control modality. However, it only allows to move the currently controlled object, the gamepad is still required for input. On button 1 next to the control knob, three different control modes can be chosen:

1. Move all six dimensions according to space mouse input
2. Only perform translation or rotation, depending on which of the two has the higher absolute input value
3. Only move according to the one most dominant input out of all six values

5 Applications & Use Cases

The presented tools can be employed in a wide range of scenarios. This chapter presents a selection of exemplary applications which were implemented and evaluated within this thesis. The LBR IV serves as exemplary manipulator.

5.1 Robot Placement for Palletizing

Pick and place problems form a very basic subtask in many manipulation scenarios. From a reachability aspect, they can also be seen as being representative for tool changes or for clearing an area from obstructions. The first example is a palletizing task. The robot has to be positioned so it can place nine boxes on a pallet in a predefined orientation (see Figure 5.1).

To successfully fulfill the palletizing task, two conditions have to be met: (1) All target positions need to be reachable and (2) there should be no self-overlapping region of the workspace that enforces reconfiguration between any of the target positions.

To aid with the initial positioning, the constant orientation workspace for the current TCP orientation can be displayed. All TCP target positions for the palletizing task have to be placed within the visualized boundaries. The TCP orientation can also be varied. Although the box orientation for the palletizing task is fixed, the pose of the robot base is variable, which is mathematically identical. Hence, the pallet may also be turned upright or upside down.

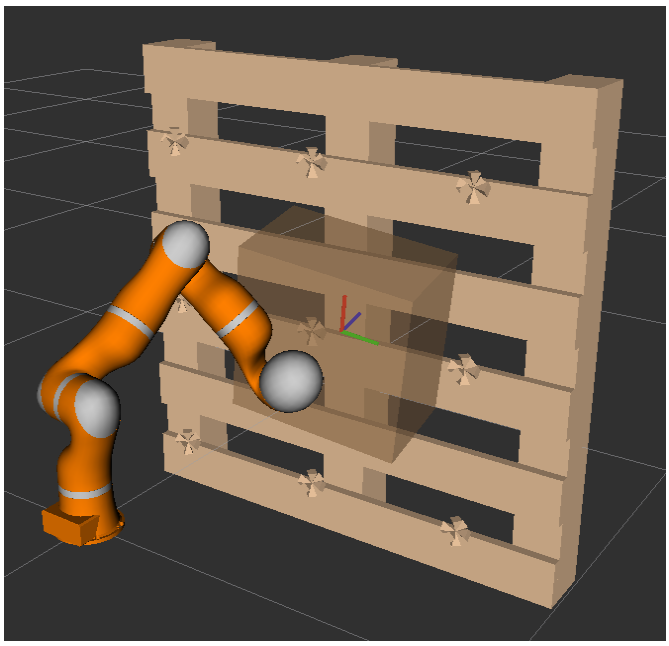

Figure 5.1: Palletizing task: The pallet has to be placed so that the robot can reach all nine box positions (TCP targets are marked on the pallet with crosses).

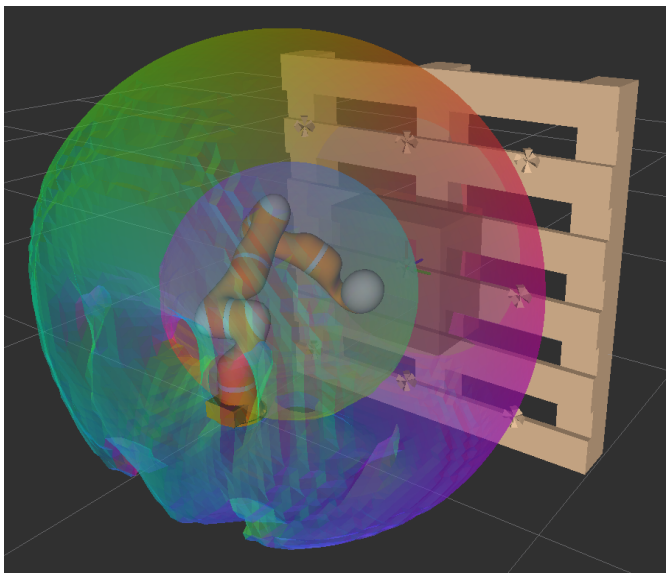

Figure 5.2: Visualization of the constant orientation workspace for the current tool orientation in order to assist during the initial pallet positioning: All nine target locations have to lie within the shown boundary. (In the virtual environment with movable camera, the situation is more graspable than in a static picture.)

Once a suitable pose is found so that all target positions are well-covered (and provided the sampling grid is fine enough), one can be sure that all places can be reached in principle. However, it is possible that the pallet is split by self-overlapping region boundaries of the workspace (as illustrated in Figure 5.3) and a large motion is required to move from one target location to another, which can be unnecessary or unfavorable. In order to test this, the whole pick and place procedure can be carried out in simulation while the directly accessible workspace is displayed and updated on-the-fly. For the palleting task, the radar-like translatory workspace representation is more suitable than the tilt sphere, since the orientation is identical for all target poses on the pallet. The described situation is visible in Figure 5.1. In case an overlap separates the pallet, it has to be repositioned or reoriented until the result of the simulation is satisfactory.

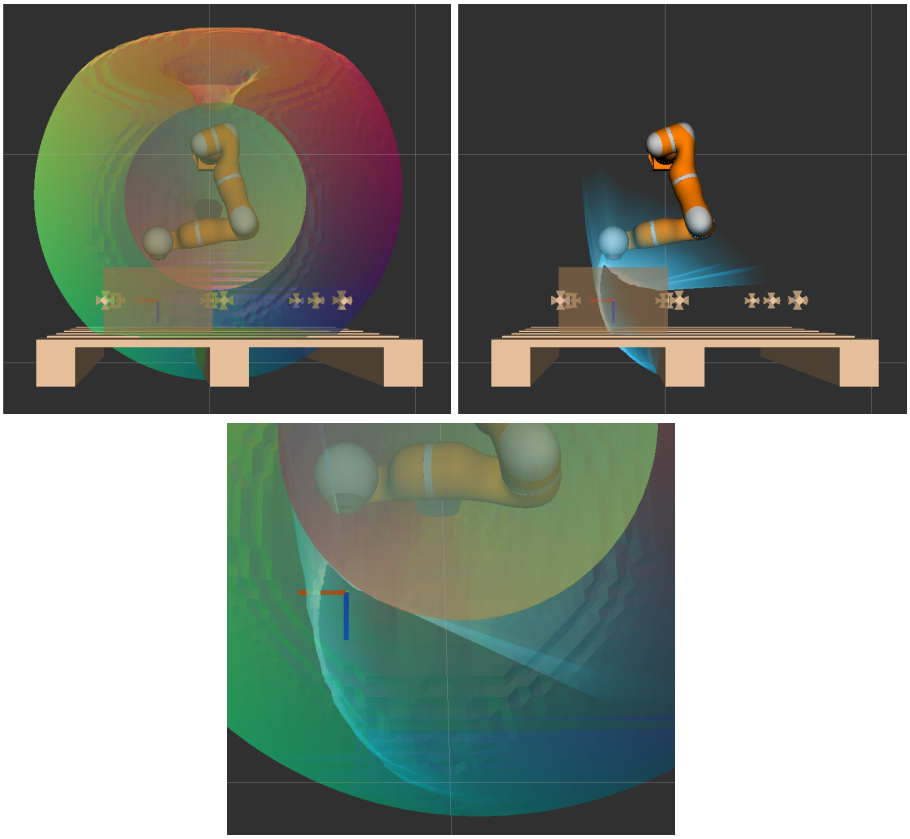

Figure 5.3: The self-overlapping problem, which can not be seen in the constant orientation workspace (top left), becomes visible when the directly-accessible workspace is rendered (top right). On the bottom, the overlay of both workspaces is shown. The straight lines can be understood as a shadow that is thrown by the inner inaccessible sphere.

5.2 Robot Placement for Machining a Cube

The second task can be seen as representative for drilling holes, milling the surface, welding, coating or basically any type of machining on an object: A cubic workpiece has to be accessed from all six sides and the tool has to be positioned along the respective surface normal (see Figure 5.4). Accessing all sides of a cubic shape is particularly difficult, since the dexterous workspace of the LBR IV is empty even when the attached tool has length zero and the six required orientations form a well-distributed subset of all orientations.

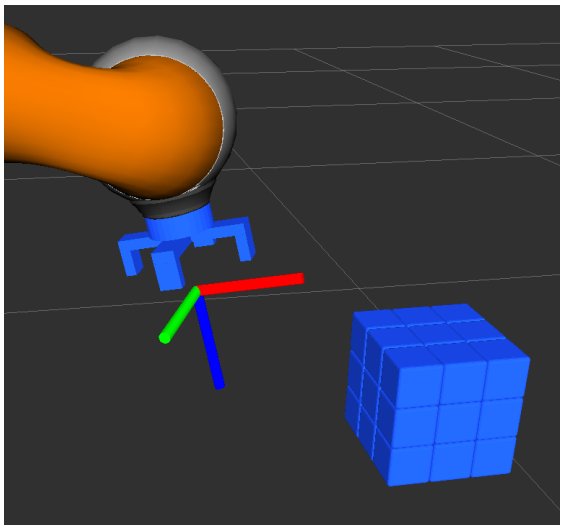

Figure 5.4: A cube has to be accessed from all six sides.

In order to make the task less abstract for the subsequent user study, the workpiece is visualized as a Rubik's Cube and the objective is to twist each side once using an appropriate-looking tool. Each side of the cube could be approached in four possible ways but since the roll axis of the tool aligns with the last joint axis of the robot and it can rotate between -170° and +170°, finding one suitable orientation for each cube face is sufficient. The TCP is positioned so that it lies in the center of the cube when the cube is held.

Similar to the palletizing task (section 5.1), it has to be ensured that (1) all TCP target poses for the cube are reachable (while this time they differ in orientation, not in position) and (2) the poses can be accessed consecutively without reconfiguration.

The first step is to define all desired orientations for displaying the total orientation workspace. It is done by rotating the workpiece (cube) and adding the selected orientations to a list. All added orientations are displayed as half arrows above the robot to visualize the current orientation set (see Figure 5.5).

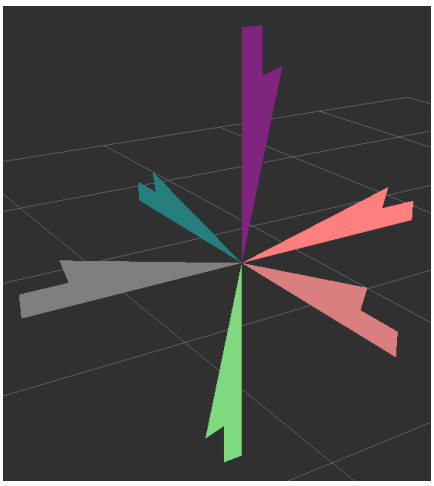

Figure 5.5: List of orientations to be considered for the total orientation workspace, visualized as half arrows.

While the TCP is rotated, the current constant orientation workspace can be displayed on-the-fly. When finished, the total orientation workspace can be displayed as intersection of the constant orientation workspaces of all selected orientations. The computation time is proportional to the number of selected orientations. The Rubik's Cube can then be placed with its center inside the displayed volume. Figure 5.6 shows that the cube can even be placed completely within the total orientation workspace. This means, that each point within the cube is reachable in all considered orientations for the TCP.

As with the first task, one can now be sure that the robot can reach all cube faces in principle. A simulation has to be carried out in order to be certain that no self-overlaps require large reconfiguration motions during operation. In the present scenario, the tilt sphere is more helpful than the radar-like translation limit display.

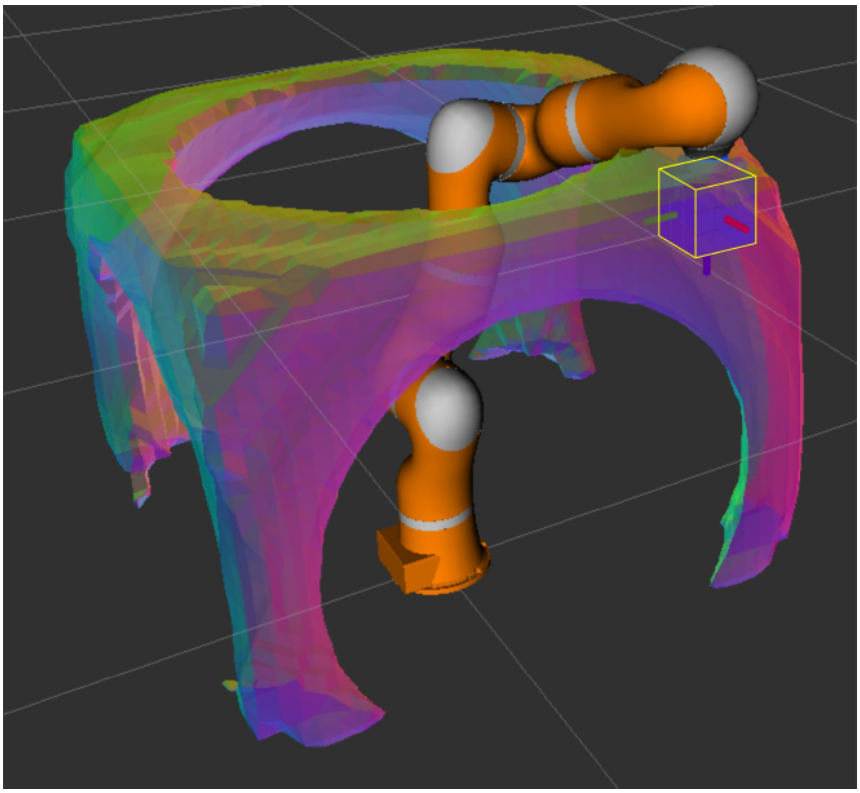

Figure 5.6: Cube inside the total orientation workspace. The cube edges are highlighted in yellow for better visibility.

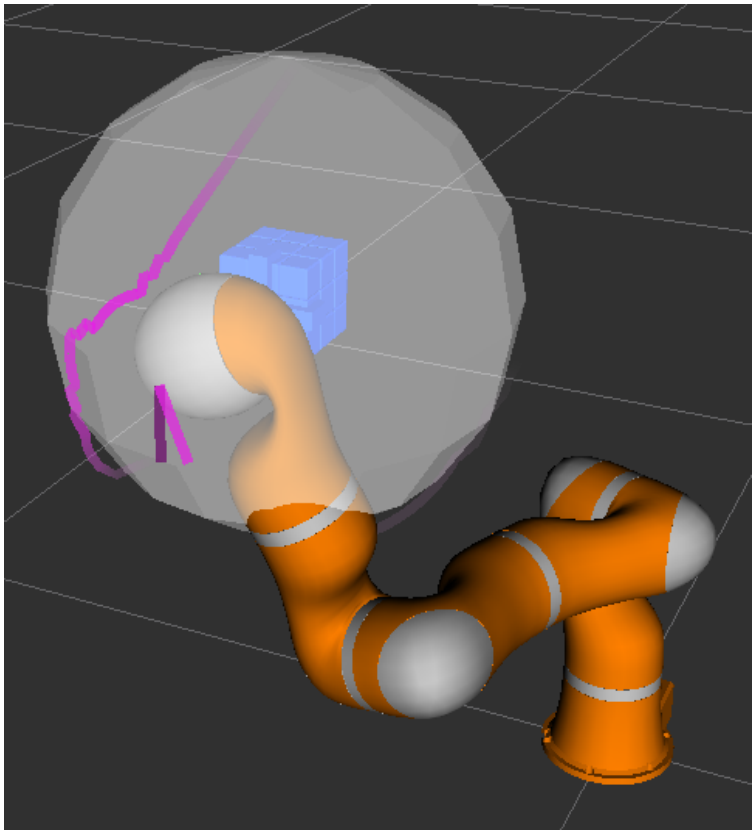

Figure 5.7: Tilt sphere for a difficult position: The robot can tilt the tool about 20° to the upper left before the jagged pink line is hit by the dominant axis of the tool (which passes through the sphere where the purple and the pink pointers meet) and a limit is reached. The tool can also be rolled about 15° clockwise about its dominant axis before hitting a joint limit.

5.3 Range of Motion for a Tool

The previously presented scenarios of robot placement for palletizing and cube machining are supposed to demonstrate the basic usage in an understandable manner. However, they could also be solved purely algorithmically without visual support. The actual strength of the visualization lies in less specific and less predictable tasks. Finding a trade-off between a large amount of coverable orientations and a large accessible region, understanding and internalizing the connection between tool shape, tool orientation and workspace shape and ultimately getting a feeling for the reachability around the robot are the main objectives of this work. It is a new means of illustrating the positioning capabilities of a robot by visual exploration. This scenario is intended to illustrate this abstract purpose by demonstration. A long tool is connected to the robot and a region with large motion range is to be found. A direct application is robot-assisted laparoscopic surgery.

Three sets of target orientations are defined, containing a reference position and eight different orientations around it where the tool is tilted by an angle α. At each of the nine orientations, the tool is further rolled by $-\alpha$, 0° and $+\alpha$, so that 27 different orientations are checked per set. For the three different sets, α is now set to 20°, 40° and 60°. Figure 5.8 shows the different orientation sets as displayed within RViz (with added black contour for visibility).

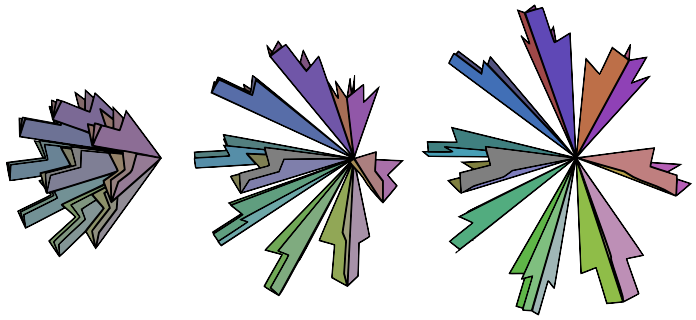

Figure 5.8: Tested sets of orientations with 20°, 40° and 60° tilt/roll angle between orientations

Furthermore, three different tool lengths were tested: 20 cm, 35 cm and 50 cm. Figure 5.9 displays the resulting total orientation workspaces. The depicted tool pose is the central orientation within the respective set.

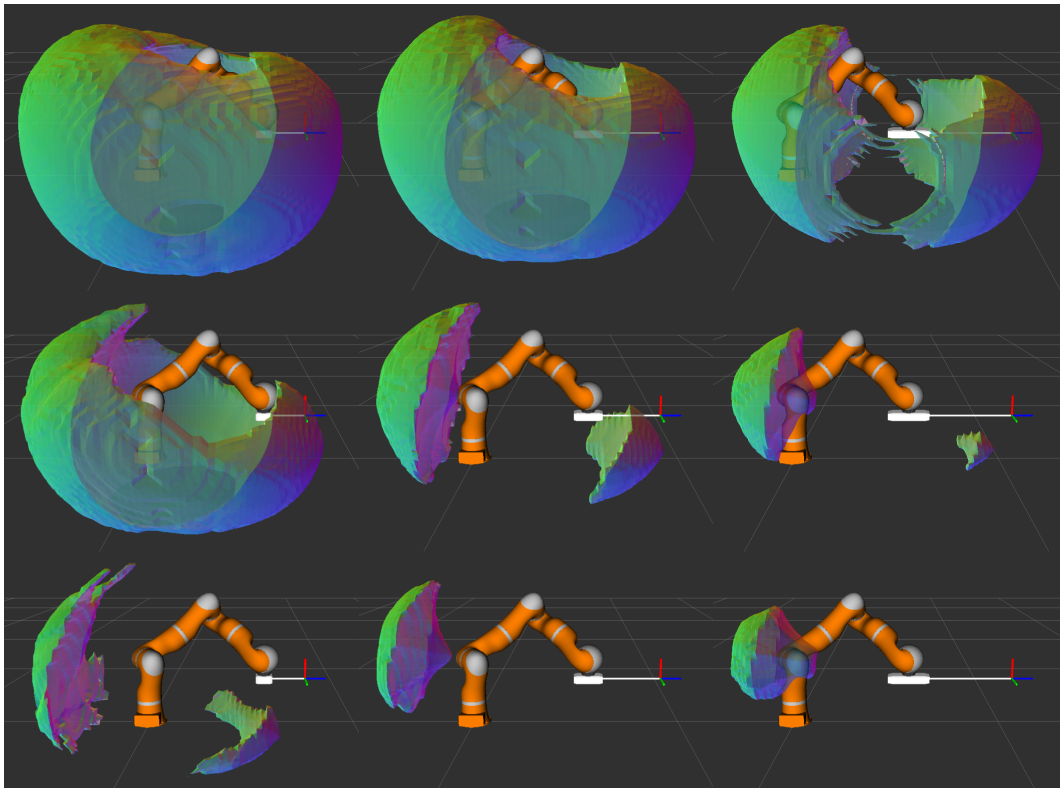

Figure 5.9: Total orientation workspaces for different approach angle sets (20°, 40° and 60° tilt from top to bottom) and tool lengths (20 cm, 35 cm and 50 cm from left to right).

As expected, the longer the tool and the larger the spread of covered orientations, the smaller the set of points that can be reached from all angles. For the three tool length/orientation spread combinations in the upper left of Figure 5.9 (20 cm/20°, 20 cm/40° and 35 cm/20°), the workspace is one connected set. As tool length or orientation spread increases, it begins to split into two sets, one in front of the robot and one around its shoulder. For the two longer tools with $\alpha = 60°$, the region in front of the robot has disappeared and only the area around the shoulder remains. Whether this region is convenient or not is an entirely different subject since the robot would block the access for other potential operators.

In addition to the length of the tool and the orientation spread, there are multiple other parameters that influence the total orientation workspace, which can be investigated as well:

- the start orientation of the tool (3 parameters),
- the transformation matrix between flange and TCP (6 parameters),
- in case of the chosen robotic structure the configuration of the robot (1 discrete parameter with 8 possible values),
- the parameters of the robot such as link lengths (only 2 additional parameters, base length can be compensated with the base position and hand length can be compensated by the tool matrix), the joint limits (14 additional parameters)
- and ultimately the set of tested orientations ($3n$ parameters where n is the number orientations in addition to the start orientation).

It is virtually impossible to compile a comprehensive in-depth analysis that discusses all parameter combinations and influences properly. The presented tools provide a tangible means of gaining insight into the positioning capabilities of a robotic system by displaying variations of all parameters on-the-fly.

5.4 The Benefit of Seven Joints

The visualizations presented in this thesis can be employed to investigate to what degree a seventh joint enhances the reachability of a robotic manipulator. If the upper arm joint of the LBR IV is set fix to 0°, it has the same kinematic structure as many 6-DoF anthropomorphic arms. This corresponds to a null-space parameter of either 0° or 180°, depending on the configuration index. In order compare the workspaces with or without redundant structure, the constant and the total orientation workspace are depicted in Figures 5.10 and 5.11 for exemplary tool orientations. For the redundant structure, the

workspaces are obtained as described in section 4.2. In order to obtain the workspaces for the non-redundant structure, the computation of the general inverse kinematics solution for each grid node is replaced by solving the particular inverse kinematics problem with the null-space parameter set to 0°.

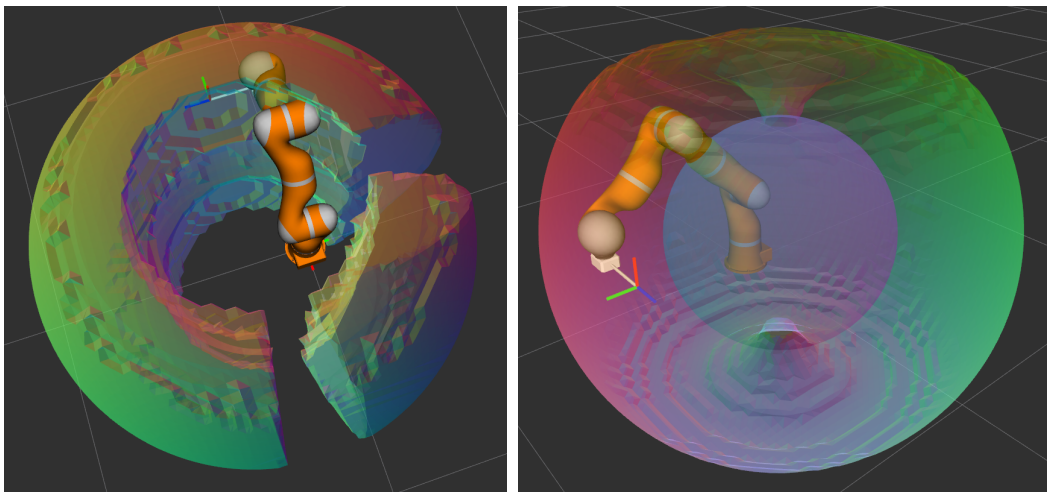

Figure 5.10: Comparison of the constant orientation workspace for the depicted tool orientation: On the left side, the null-space parameter (and therefore the upper arm joint) is set fix to 0°. On the right side, elbow movement is permitted and the upper arm joint can rotate within a range of ±170°.

Figure 5.10 clearly shows the effect that was explained in Figure 2.8. Two sections of the workspace are cut out due to the limits of the base joint and the flange joint for the depicted tool orientation. Furthermore, an area around the "north pole" of the workspace is not as accessible if the elbow cannot rotate downwards so that the lower arm can reach upwards. The situation around the "south pole" is similar.

The difference becomes much more considerable if the total orientation workspace is regarded, as visible in Figure 5.11. Here, the workspace for the depicted tool orientation and 26 more orientations around it with a spread angle of $\alpha = 30°$ (as explained in section 5.3) is compared. A tool with 20 cm length is attached to the robot. With only six movable joints, the workspace has the basic shape of a comparatively small spherical cap. With enabled redundancy, the workspace encloses more than half of the robot.

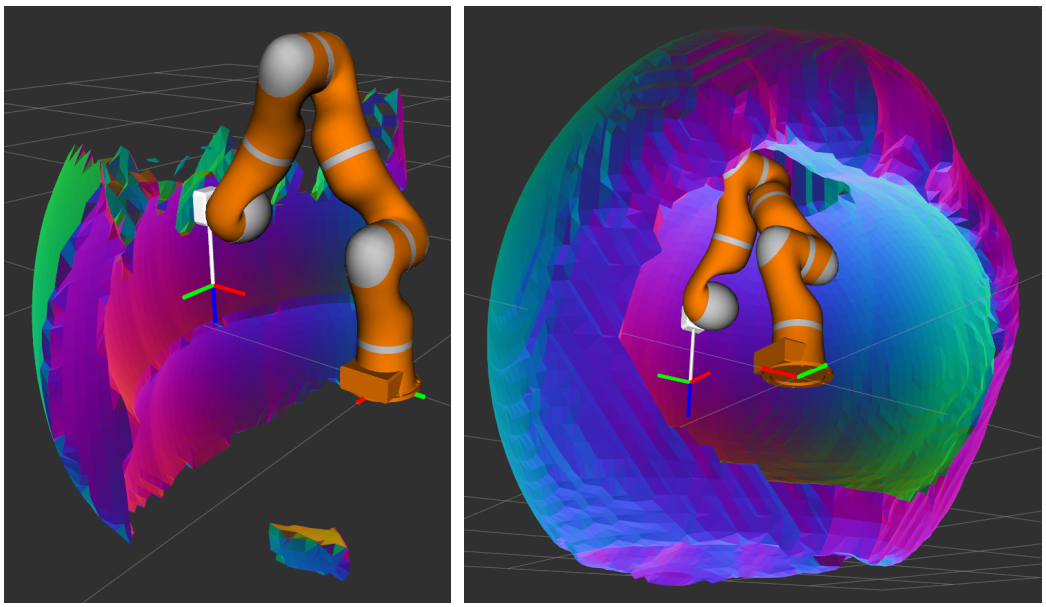

Figure 5.11: Comparison of the total orientation workspace for the depicted tool orientation and an orientation spread (as described in section 5.3) of $\alpha = 30°$: On the left side, the null-space parameter (and therefore the upper arm joint) is set fix to $0°$. On the right side, elbow movement is permitted and the upper arm joint can rotate within a range of $\pm 170°$.

As was the case in section 5.3, this example is not supposed to constitute a complete analysis of the usefulness of including a seventh joint into the robot structure, since there are equally many influencing parameters as before. The purpose is to illustrate the function of the devised visualization tools for kinematic analysis.

6 Results

Within this chapter, objective criteria of the overall system performance, such as resolution, precision and frame rate, are evaluated and the influencing factors are discussed. Subjective attributes, such as support capabilities, the usability and the perspicuity of the different visualizations, are then assessed in a user study.

6.1 System Performance & Influencing Factors

The system performance depends on several factors that influence each other and a trade-off between them has to be found. The parameters that were chosen for the scope of this thesis will be presented in the following. Ways of increasing the performance will also be discussed.

6.1.1 Frame Rate

The frame rate of the different visualizations depends on the situation that the robot is currently in, since longer simulated motions (in a situation where boundaries are further away) require more computation. The average timing can be seen in Table 6.1.

Table 6.1: Computation time for different workspace visualizations on the used hardware; memory transfer times are included into the total times

	const. ori. workspace	dir. acc. workspace
single prec.	70 .. 80 ms (4 .. 8 ms mem.)	12 .. 20 ms (2 ms mem.)
double prec.	110 .. 120 ms (6 .. 12 ms mem.)	60 .. 90 ms (3 ms mem.)

The constant orientation workspace requires the general inverse kinematics solution to be computed at each point. The interval handling requires many loops and conditional statements for which GPU processors are not well-suited. This explains the less significant difference between single and double precision.

The computation of the directly accessible workspace contains both the translatory as well as the rotatory component. In the current resolution, the former takes up 88% of the workload, the latter 12%.

In the present parameter configuration, a frame rate of 8 Hz could always be achieved. This does not produce a fluent impression but it was considered to be sufficient to provide an intuitive visualization without perceived lag. If a higher frame rate is needed, more powerful hardware is required or a different trade-off with other factors, which will be discussed in the following, has to be found.

6.1.2 Hardware

As most of the calculation is running on the GPU, the computational power of the graphics card forms the performance bottle neck. The NVIDIA GeForce GT 440, which was utilized for this work, was released in February 2011. Its specifications are listed in Table 6.2. The specifications of the currently fastest available graphics card, the NVIDIA GeForce GTX Titan Z, are listed for comparison.

Table 6.2: Comparison between the utilized NVIDIA GeForce GT 440 graphics adapter and the NVIDIA GeForce GTX Titan Z, the currently fastest available graphics adapter.

	GeForce GT 440	GeForce GTX Titan Z
Cores	96	5760
GPU clock	810 MHz	705 MHz 876 MHz with boost
Shader clock	1 620 MHz	1 620 MHz
Computation speed single precision double precision	 311 GFlops 25.9 GFlops	 8 121.6 GFlops 2 707.2 GFlops
Memory	1 GB (DDR3)	12 GB (DDR3)
Memory bandwidth	28.8 GB/s	672 GB/s

It is difficult to estimate the actual performance boost purely from these specifications, since the code running does not consist out of floating point operations only, many conditional statements, loops, comparisons etc. are involved. Judging from the number of cores (at the same clock speed), the GTX Titan Z should achieve 60 times higher performance. However, a comparison of the single precision computation speed indicates a factor of only about

26 while the factor for double precision is about 104. The memory transfer rate can be sped up by a factor of about 23. Based on these estimates, we can vaguely assume that a performance increase of a factor 20 is feasible in theory.

6.1.3 Resolution

In general, increasing the resolution results in higher accuracy and a more detailed boundary surface, sharper edges and less artifacts.

Constant Orientation Workspace

For computing the constant orientation workspace, a cubic grid of $54 \times 54 \times 47$ vertices with an edge length of 3 cm was chosen. If shifted properly (as described in section 4.2.2), that volume can always enclose all reachable points for the TCP of the LBR IV. This results in 137 052 vertices. Note that the number of grid points and therefore the influence on the computational cost is proportional to the third power of the resolution.

Depending on the structure of the robot, a certain speed-up could be achieved by using a more suitable grid design. For the LBR IV, the maximum arm length restricts the reachable volume to a sphere with radius 79 cm. Furthermore, the distance between wrist and shoulder has a lower limit (see equation (4.5)). This blocks a sphere of radius 39.5 cm for the LBR IV. The remaining spherical shell only occupies 46% of its bounding cube, so sampling only this shell can decrease both GPU load and memory transfer. However, points outside this shell are excluded by the GPU via simple range check much faster than the time it takes to compute the general inverse kinematics solution for points inside the shell, so the speed-up would be much less significant than a factor two.

Another possibility would be to check the grid in multiple passes, refining it only in the proximity of boundary points found during previous passes.

However, for the sake of simplicity and generality, only the basic cubic grid was implemented.

Total Orientation Workspace

As described in section 4.2.4, the bounding box of the total orientation workspace is the intersection of the constant orientation workspaces for all considered orientations and each orientation is tested at every grid knot. Within this thesis, the grid of $54 \times 54 \times 47$ is scaled to fit the size of the resulting intersection. Hence, the computation time is not influenced by the intersection volume but the resolution can be higher. However, the computational

load is proportional to the number of checked orientations at each vertex, which depends on the number of orientations that the user wants to test.

The considerations for speed-up that have been discussed for the constant orientation workspace computation (better matching or refined grid) apply here as well.

Directly Accessible Workspace for Translation

As mentioned in section 4.3, an icosahedral grid is used to sample the set of all possible directions (i.e. unit vectors in $\mathbb{R}^3$). The subdivision used in this thesis is 10, which results in 1 002 vertices, distributed as equally as possible over the sphere. The maximum angle between any given direction and the closest direction within the set is about 5.7°. Along each direction, a maximum of 25 steps up to a distance of 0.7 m is tested for a feasible inverse kinematics solution. This parameter influences the strength of aliasing artifacts (as shown in Figure 4.6). At each step, the null-space parameter is varied up to 30° in a maximum of 8 steps, visible in Figure 4.7.

While the subdivision count of the grid (and therefore the reciprocal of the maximum distance to the closest direction) has quadratic influence on the overall computation cost, the two other parameters (maximum number of steps along each direction and maximum number of null-space parameter tests at each step) have linear influence.

Directly Accessible Workspace for Rotation

The set of tilt axes that are perpendicular to the dominant tool axis is one-dimensional. Hence, sampling it is fairly straightforward and the number of samples only has linear influence on the computational load. Currently, 128 tilt directions are checked (+2 for $\pm$ roll about the dominant tool axis). Choice and influence of further parameters (number of steps for each direction and number of tests for the elbow) are equal to the translational computation.

6.1.4 Precision

While double precision requires significantly more computation, the accuracy loss when using single precision was visible since artifacts occurred, especially near singular poses. For specific applications, it has to be decided whether the possible gain in accuracy is important or not.

6.2 User Study

In order to evaluate how intuitive and graspable the visualization concepts are, a user study with 15 participants (10 of which have controlled a robot before) was conducted. For each participant, three virtual reality trials were carried out and after each trial, a User Experience Questionnaire (UEQ, [50]) had to be filled in.

The UEQ is a standardized test for evaluating the user experience of an interactive product, which also allows the comparison against a benchmark. It contains 26 items (e.g. "boring" vs. "exciting" or "obstructive" vs. "supportive") which have to be rated on a scale between -3 and +3. The items are then aggregated into six scales (taken from the UEQ handbook):

- Attractiveness: Overall impression of the product. Do users like or dislike the product?
- Perspicuity: Is it easy to get familiar with the product? Is it easy to learn how to use the product?
- Efficiency: Can users solve their tasks without unnecessary effort?
- Dependability: Does the user feel in control of the interaction?
- Stimulation: Is it exciting and motivating to use the product?
- Novelty: Is the product innovative and creative? Does the product catch the interest of users?

The trials are designed based on the scenarios presented in chapter 5. To allow for repeatability and to keep the duration acceptable for the participants, predefined positions were evaluated instead of asking the users to position (and potentially reposition) objects themselves. The aim of this user study is to assess its usability and perspicuity. A discussion about the value and the significance of these trials follows in chapter 7.

6.2.1 Introduction to the Study

All participants were given a short standardized introduction. It was stated that the aim of this research is to visualize the workspace boundaries of a robotic arm on-the-fly in an intuitive manner in order to give the user a sense of reachability of the arm.

All participants were allowed and encouraged to ask questions in general or if anything is unclear.

The users were then introduced to the concept and application of the UEQ: It is a standardized test in order to allow comparison against other products, so some aspects of the evaluation might not fit very well to the concept that is assessed. The UEQ has to be filled in shortly after the respective trial in a spontaneous manner.

The limits of the individual joints were stated and the current position of each joint was displayed throughout the trials.

They were then introduced to the gamepad control, the SpaceNavigator was offered as an alternative input device (it was used by three participants). For practice, a little box had to be put on a pallet in a defined orientation. As it is not the purpose of this study to evaluate the robot control modality, the users were offered to leave the control to the conductor of the study (i.e. the author), which was the case in one instance.

6.2.2 Reachability Evaluation

In the first trial, users were asked to assess the placement of objects. Three pallets were placed around the robot with nine target positions each, as can be seen in Figure 6.1. For each pallet, the user was asked to decide which of the nine target positions were reachable with a box attached to the robot. First, only camera control was permitted and the assessment had to be done visually. Afterwards, robot control was permitted and each position could be checked for reachability by trying to move the TCP to the target.

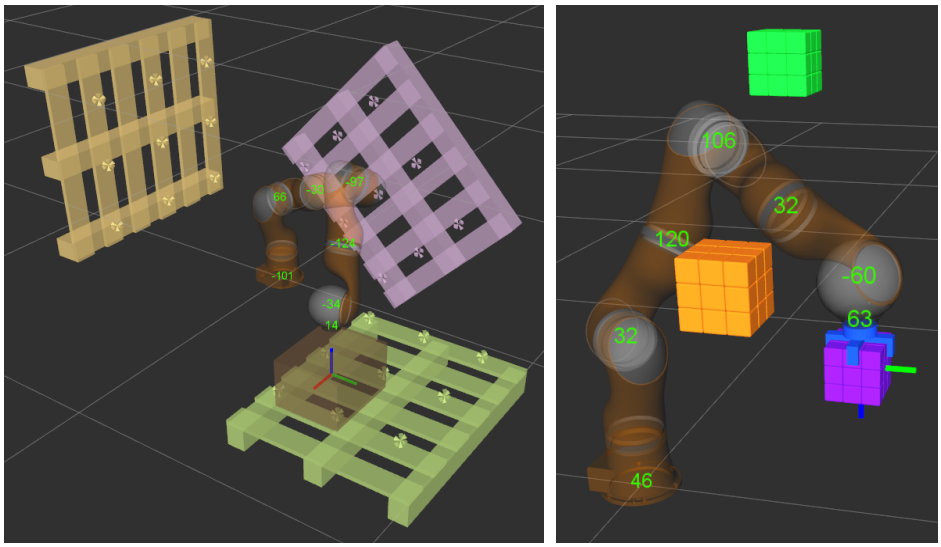

Figure 6.1: Placement evaluation for pallet positions (left) and cube positions (right).

A similar experiment was conducted with cubes instead of pallets. This time, all six sides of the cubes had to be accessed with a tool (see Figure 6.1). The users were asked for each cube whether all sides are reachable or not. Again, they were only allowed camera control at first and then robot control.

Afterwards, the users were introduced to the visualizations of the constant orientation workspace and the total orientation workspace (see Figure 6.2). For checking the reachability of pallet positions, the TCP with the box had to be orientated correctly and it had to be checked whether the nine target TCP positions were within the displayed workspace hull. For the cube positions, the total orientation workspace for six target orientations (one for each cube side) was displayed. The users had to check whether the cube centers are within the rendered surface or not.

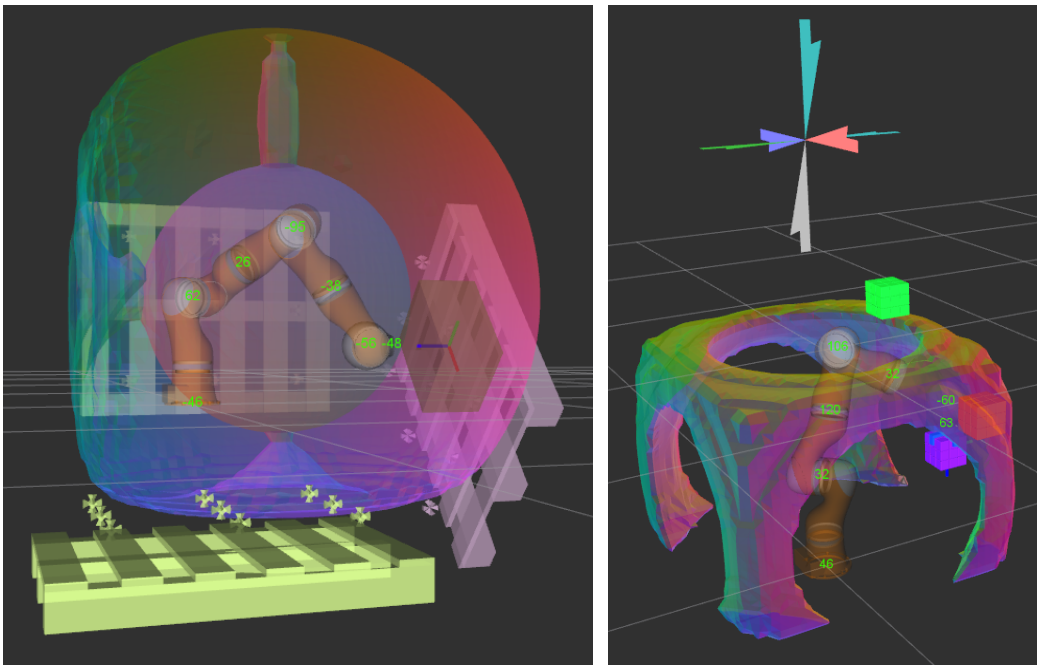

Figure 6.2: Placement evaluation for pallet positions (left) and cube positions (right) with workspace visualization support.

Subsequently, the users were asked to fill in the first UEQ. Specifically, the visualization concepts of the constant orientation workspace and the total orientation workspace (and not the control modality) had to be assessed.

The results of the palletizing trial are given in Figure 6.3. Each target position is represented by a pie chart, showing whether the participant's estimate was incorrect (red) or correct (green if the pose is reachable or blue if the pose is not reachable). As can be expected, pure visual and instinctive assessment is not very accurate. However, a majority vote for each pose assesses the

reachability incorrectly in only 4 out of 27 cases. Trial and error via robot control resulted in much higher accuracy but it still did not yield perfect results. In some cases, the users did not test a position accurately enough. One user correctly tested two poses but confused them before marking them. The large error for the top right pose on the yellow pallet was due to the fact that the last joint reached its limit of 170° shortly before reaching the pose (which was not intended during trial design). For some users, the elbow reconfigured advantageously while testing the other points of the pallet so that the pose was reachable in the end. One user rotated the box by 360° around the flange axis and was then able to reach the pose. The results of the boundary visualization were overall better. The problem with the top center position of the green pallet was that it was inside the inner unreachable sphere and it was hard to assess whether it was behind the boundary or not, due to a transparency problem that will be discussed in section 7.3.

The results of the cube experiment are depicted in Figure 6.4. The visual instinctive assessment was correct in about two out of three cases. Impressively, for the orange cube, it was better than trying to reach each face with the robot. The situation was labeled a "seven-dimensional labyrinth" by one participant due to the fact that the movement often ran into joint limits. Reaching a cube from all six sides can be done in numerous ways and the posture of the robot depends on the chosen path, as was discussed in section 2.1.4, so the limits were different for every user. Some users tried to access cube faces from other directions if they failed the first time. The green cube was unreachable on its top but only by a few degrees, so many users falsely identified it as being reachable. With the visualization of the total orientation workspace, all cubes were correctly assessed by all 15 participants.

Figure 6.5 shows the outcome of the User Experience Questionnaire. The constant and total orientation workspace visualization concepts are perceived very well in the overall impression, which is indicated by the "Attractiveness". Whether the concepts are intuitively understandable or not (which is the prime concern of the assessment) is probably best reflected by "Perspicuity", which ranges from "Above Average" to "Excellent". "Efficiency" and "Stimulation" were rated "Excellent" by the majority of users, which indicates that the visualization is fast enough to be pleasant to use and not tedious. The high "Dependability" shows that there was no unexpected or strange behavior of the visualization. Although the "Novelty" is rated high, it cannot be expected that the users are familiar with other concepts of the field, so this value cannot be taken literally.

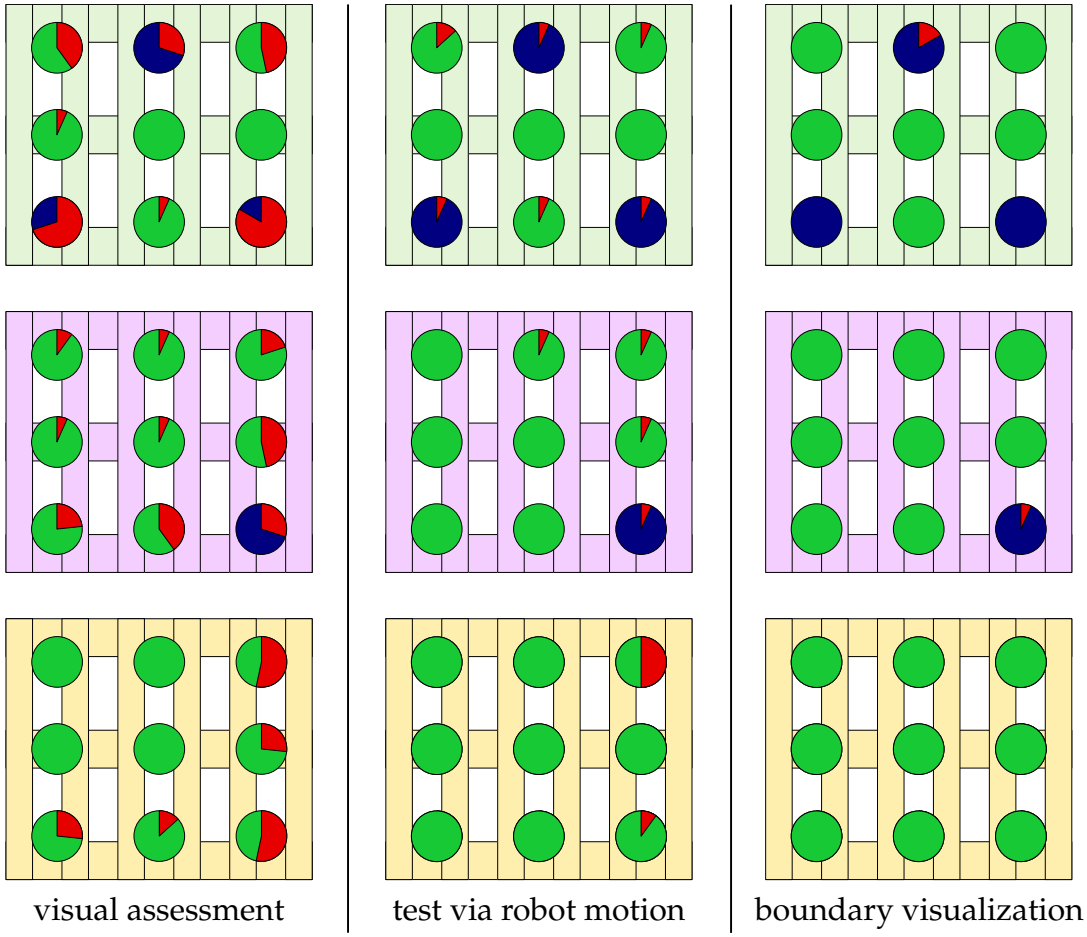

Figure 6.3: Results of the palletizing trial. Green: correctly identified as reachable, blue: correctly identified as unreachable, red: incorrectly identified.

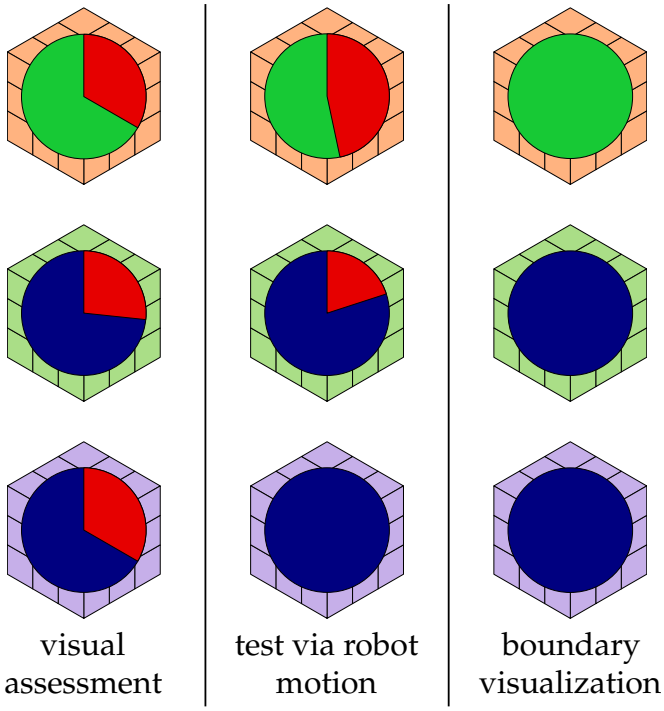

Figure 6.4: Results of the cube machining trial. Green: correctly identified as reachable, blue: correctly identified as unreachable, red: incorrectly identified.

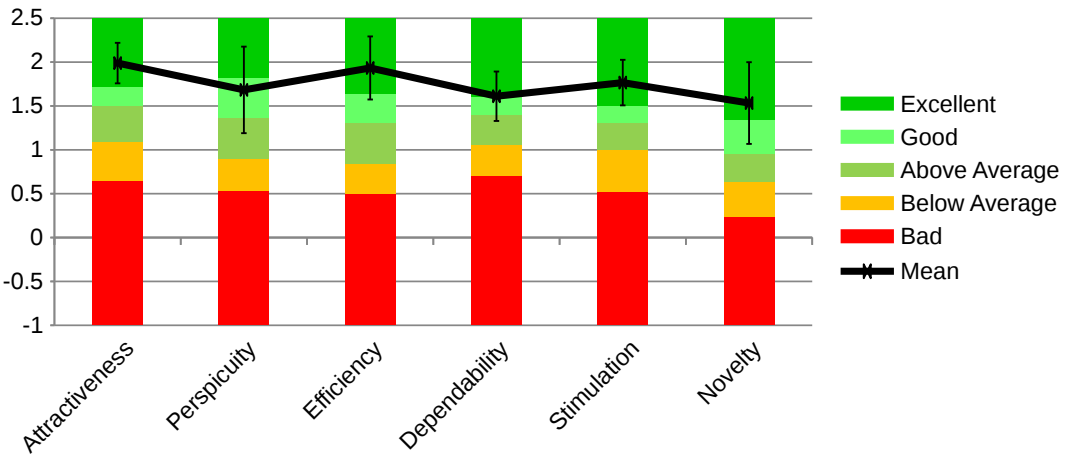

Figure 6.5: Results of the User Experience Questionnaire and comparison against benchmark for the generally accessible workspace visualizations

6.2.3 Position Tolerance Assessment

In the second trial, users had to evaluate the positioning tolerance of predefined trajectories. Two identical trajectories that resemble the outline of a car door at different poses were displayed (see Figure 6.6) along which the user was able to move the robot. The TCP stayed on the trajectory and could only be moved back and forth, the elbow angle was preplanned. First, the positioning tolerance had to be estimated based on displayed joint positions and overall impression. Users had to mark critical regions along the trajectories on a piece of paper. They were asked, which trajectory they would favor and how certain they are about their decision (in percent).

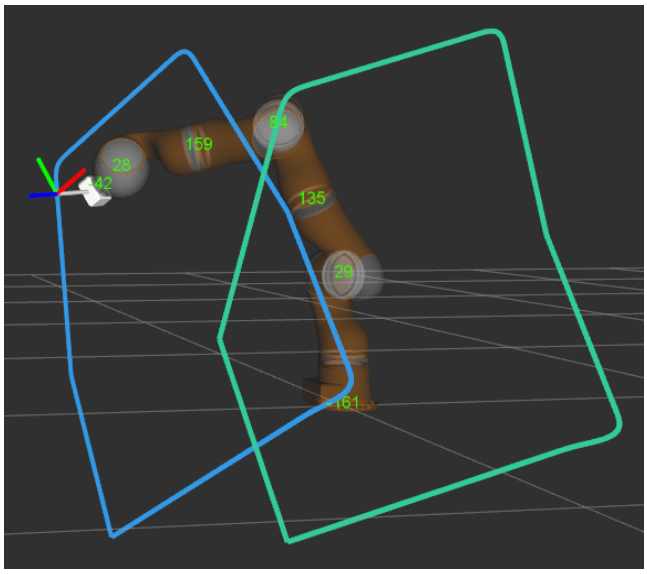

Figure 6.6: Translatory tolerance estimation for predefined trajectories without visualization support. The trajectories resemble the contour of car doors.

Afterwards, the concept of the directly accessible workspace for translation was introduced and the boundary was displayed. With the help of this visual support (as seen in Figure 6.7), users were asked to reevaluate the positioning tolerance, decide again between the two poses and state their certainty again. In the end of the second trial, a second UEQ had to be filled in.

The results are shown in Figure 6.8. An overlay of all user sketches shows where critical points were seen. For the green trajectory, the purely visual tolerance assessment based on joint angle values and overall impression was quite accurate and critical points were identified well. On the blue trajectory, users were much more accurate using the visual assistance of the translatory motion boundary. The decision about which trajectory has the higher tolerance was much clearer when the visual assistance was used, as

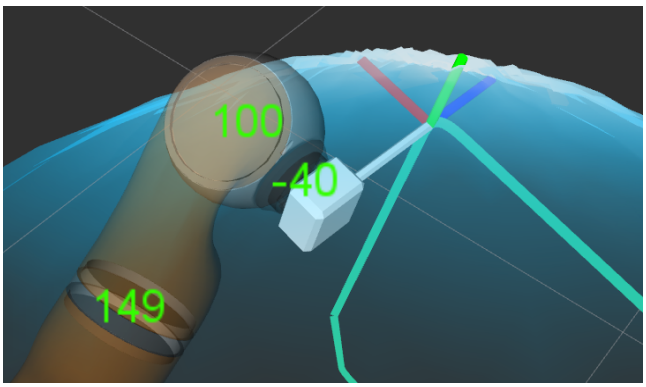

Figure 6.7: Translatory tolerance estimation for predefined trajectories with visual support by the directly accessible workspace rendering.

can be seen in the histograms. However, it has to be mentioned that some users completely ignored the most critical point on the blue trajectory (only 2 cm away from the boundary). It might be due to the fact that this point was close to the inner unreachable sphere, not to the outer one. If the scene is viewed from an unfavorable angle, the blue color of the uncritical outer hull tints the critical white color of the inner hull.

The UEQ's results can be seen in Figure 6.9. They resemble the results of the UEQ for the generally accessible workspace visualization very closely, only "Perspicuity" and perceived "Novelty" are slightly higher.

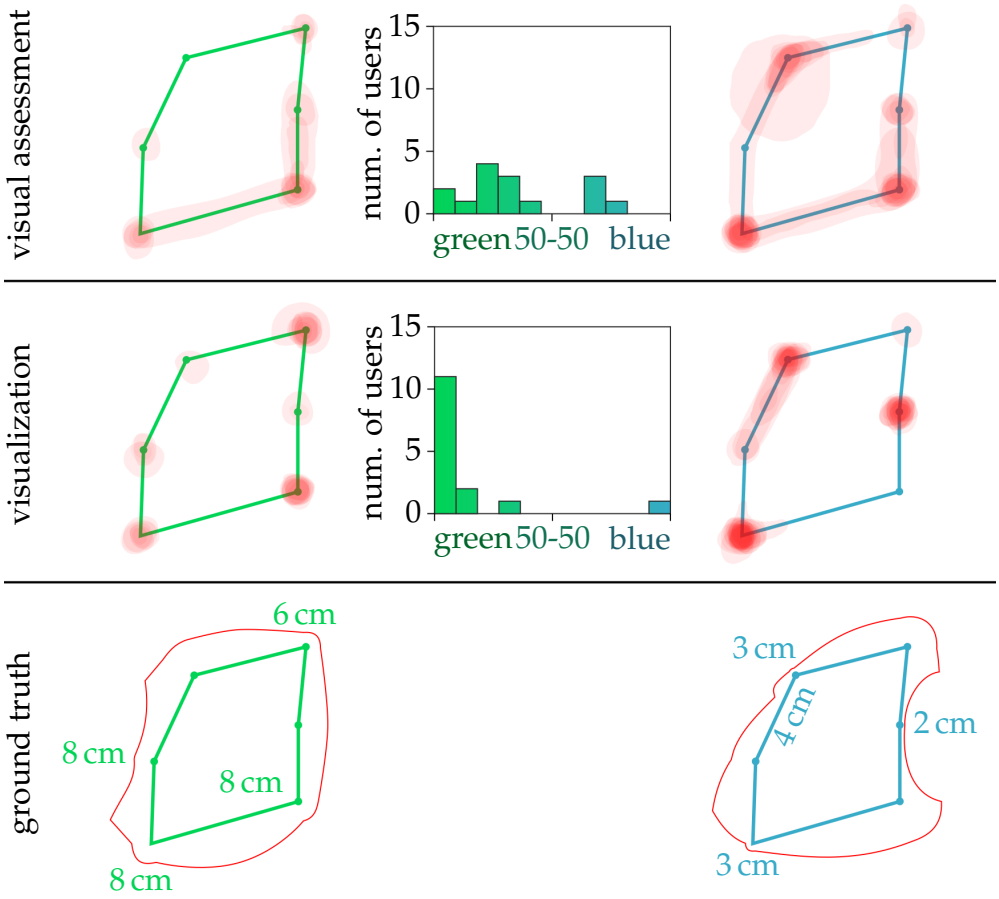

Figure 6.8: Results of the translatory tolerance experiment: Reddish areas are overlays of all user marks, the histograms show how certain the users picked a trajectory.

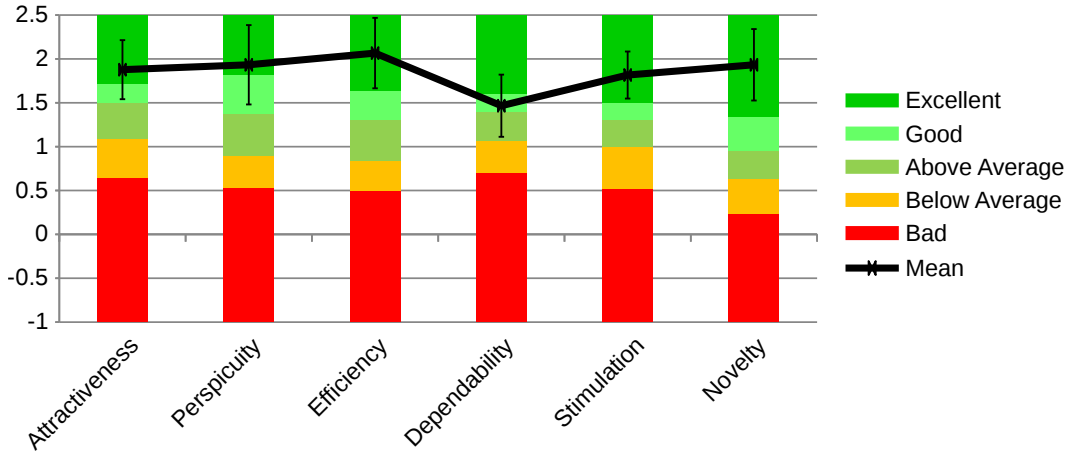

Figure 6.9: Results of the User Experience Questionnaire for the directly accessible workspace visualization for translation.

6.2.4 Orientation Tolerance Assessment

The third trial is again an evaluation of tolerance around predefined trajectories. This time, the angular clearance has to be assessed. The trajectories can be seen as green and blue meander-shaped lines in Figure 6.10, the white arrows around the green trajectory indicate the five different tool orientations that the tool sweeps through. The scenario was a milling motion on a femur for knee arthroplasty. Similar to the trial before, users had to do a visual estimation, mark critical regions, decide for a trajectory and state their certainty.

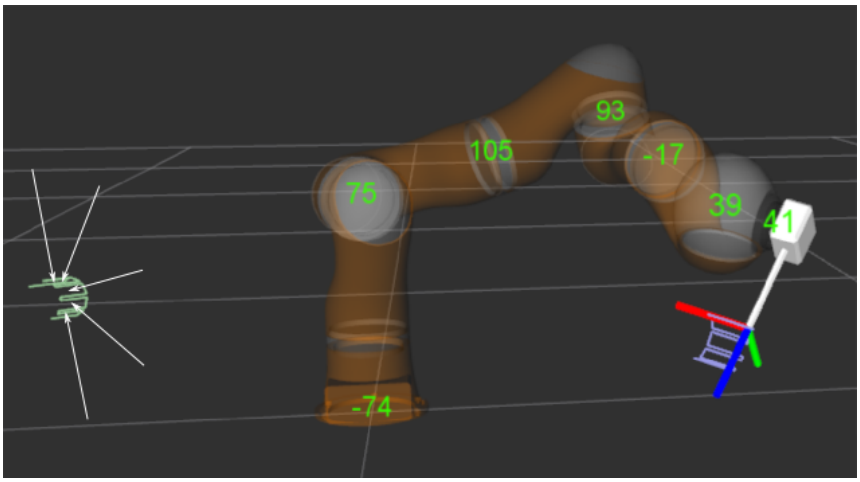

Figure 6.10: Rotatory tolerance estimation for predefined trajectories. The white arrows indicate the five different tool directions.

The tilt sphere was then introduced as support for this task (see Figure 6.11) and the users were asked to do the assessment again and fill in a third UEQ afterwards.

Figure 6.12 shows the outcome of the third trial. The trajectories are unfolded for better visualization. The red areas indicate how many users marked this milling plane as critical. At the bottom, the actual tolerance is displayed as a circle segment for each of the five planes. A 40° angle at the segment tip means that the tool can rotate by a minimum of 20° in all directions. Note that the purely visual assessment of orientational tolerance around a point is extremely difficult at best. The difference in rotational clearance between both trajectories was quite large. It was between 7° and 23° for the green trajectory and between 40° and 80° for the blue trajectory. The majority of users correctly chose the blue trajectory without visual support, but the certainty was very low, as visible in the upper histogram. Marking the critical areas only based on visual estimate did not resemble the ground truth very well for the green trajectory, which is due to the fact that one joint approached its limit at the end

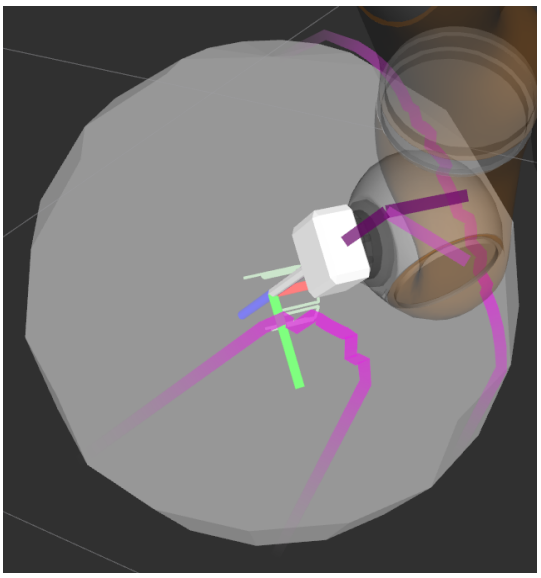

Figure 6.11: Rotatory tolerance estimation for predefined trajectories with visual support by the tilt sphere. The current position allows about 15° of tilt motion to the right, 20° of tilt motion to the lower left and 40° of roll motion clockwise.

of the trajectory, which did not influence the rotational clearance much. Using the tilt sphere, the critical areas were identified much more accurately and the decision was much clearer. However, many users ignored the depicted roll limits (indicated by pink pointers around a violet pointer, visible in Figure 6.11) and responded that they forgot to pay attention to this when asked about that afterwards. These limits have to be emphasized more or displayed in a different way.

Compared to the visualization of the directly accessible workspace for translation, the UEQ yielded lower results in all criteria. “Perspicuity” – the most important aspect – received an average rating with a large variance from “Bad” to “Good”, the same holds true for “Dependability”. The participants reported that the visualization is not as easily understandable. Furthermore, a pivoting motion of the tool always entails a certain roll movement, which causes the visualized boundary lines to shift more than expected. However, despite the lower scores on the user experience, the experimental results that were achieved with the help of the tilt sphere were quite accurate. The fact that the objective results are sound but the subjective impression could be better means that the display of the tilt sphere should be improved but the underlying concept is valid and understandable.

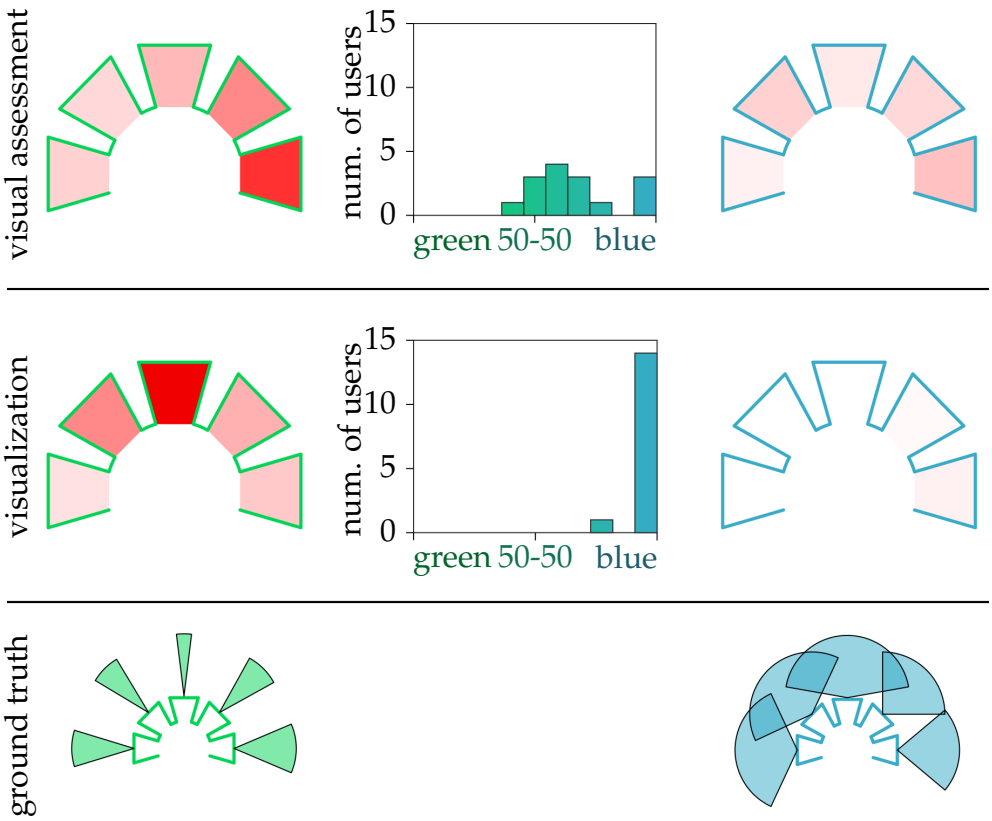

Figure 6.12: Results of the orientational tolerance experiment: The reddish areas are overlays of all user marks, the histograms show how certain the users picked a trajectory.

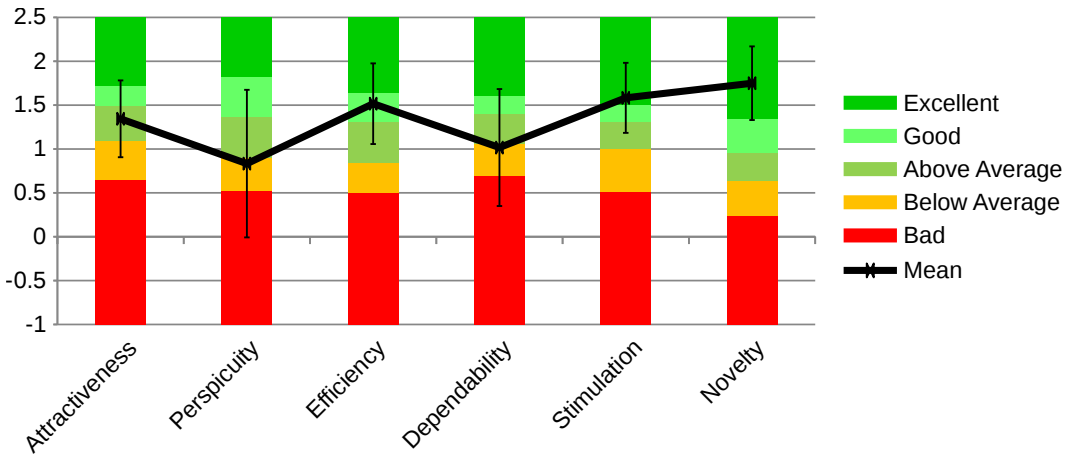

Figure 6.13: Results of the User Experience Questionnaire for the directly accessible workspace visualization for orientation.

7 Discussion

Within this chapter, the presented visualization tools will be discussed in a broader perspective. The results of the user study will be interpreted and a few "Lessons Learned" are addressed. A comparison with other works is made.

7.1 Reception

The overall reception of the visualization for the generally accessible workspaces (both constant orientation workspace and total orientation workspace) was very good. The concept was understood immediately, the visualization was deemed meaningful and supportive.

Compared to trial and error by robot control, the reachability was assessed as being much faster and more reliable. Simply solving the complete inverse kinematics for all target poses on the pallets (or all cube faces, respectively) and displaying a red or green label would most likely yield even better results. However, in a positioning scenario, the concept at hand assists in finding a solution if not all poses are reachable. The user can easily see and understand why this is the case and how to resolve the situation, if the workspace boundaries are shown.

The directly accessible workspace for translation was also received very well. The difference from the generally accessible workspace was easily understood. The users were able to solve the given task of evaluating the translatory tolerance around the trajectories. One user pointed out that this tolerance could simply be encoded as a color along the trajectory, which would allow for faster assessment and remove the need of visualizing the workspace and moving along the trajectory. Although this is a valid remark, forcing the user onto a predefined trajectory served the purpose of obtaining repeatable and comparable results from the study. The wider purpose of this visualization is to convey a detailed representation of the dexterity around the current pose, which is intended to be used in manually operated scenarios.

However, the fact that many users ignored the most critical point on the blue trajectory necessitates countermeasures. For specific scenarios, a distance threshold between TCP and workspace boundary could be defined and when the TCP is closer to the boundary, a warning could be issued. Furthermore, the surface could be rendered in red in areas where the distance falls below the threshold.

One user further commented that, if one is "trapped" in a small volume surrounded by walls of joint limits, the visualizations do not show a Cartesian way out of the situation. This reflects on the fact that path finding is not the purpose of this work. Numerous algorithms are available for path finding, which can work hand in hand with the visualization. Figuratively, the visualization is not intended to work as a navigation system but as a radar for this multi-dimensional maze which can be combined with the navigation system, if desired.

The directly accessible workspace for rotation (tilt sphere) received less positive reviews. This may be due to the fact that three-dimensional rotation and orientation is generally less intuitively graspable compared to translation and position. Of all the concepts that were investigated in section 3.3.2, the tilt sphere was already chosen to be the simplest. It can be a problem that the users had very little time to become familiar with the visualization. The root locus plot is also far from being intuitively graspable, yet it is a well-established visualization technique in control theory. As stated before, although the users were subjectively not as comfortable using the tilt sphere, objectively it served its purpose and the rotational clearance was assessed accurately.

7.2 Remarks on the User Study

Some aspects concerning the circumstances under which the user study was conducted are addressed and discussed in this section. Meaning and value of the study are pointed out.

Time Constraint

To ensure that the results are comparable and that users have to be briefed only once about the control and the idea of the UEQ, the evaluations of all concepts were conducted in a single session for every user.

Thoroughly evaluating three different concepts within an acceptable duration for each user and for the complete study is ambitious at best, especially since the visualizations are supposed to convey a natural feeling for the workspace of a manipulator over time. The study required 60 to 90 minutes per user.

Explanation and discussion, getting familiar with the gamepad control and completion of three UEQ forms already filled a large chunk of that time frame.

In order to unfold their full potential, the tools would have to be tested over a longer period of time. In a realistic scenario, a user has more patience both with and without visualization and an intrinsic motivation to solve the problem.

Control Conditions

Displaying the current joint angles directly at each joint within the scene can be considered a large support during the control experiments without boundary visualization. However, in a realistic scenario, a user can display the current joint positions one way or another, so they were included.

User Experience

The concept of the null-space and the robot configuration index was not introduced to the users since the visualization is supposed to be helpful also for unexperienced users or new robotic structures. Explaining these concepts and providing some further basic insights about the situation (e.g. turning the TCP by 360° or approaching the target from another angle might help when stuck) might have resulted in a better performance in the control experiments without visualization. However, some users already had experience in controlling the LBR IV and did not perform significantly better.

Value of the Study

The study showed that new users were quickly able to understand and use the developed tools the way they were intended to be used. All visualizations were assisting rather than obstructive and all concepts apart from the tilt sphere (which received average rating on the UEQ benchmark) were perceived as being helpful and pleasant to work with.

7.3 Lessons Learned

This section discusses additional points that surfaced during the work for this thesis and have to be addressed.

Three Dimensions on a Screen

Overall, it was noticeable that a lot of camera movement is required in order to fully grasp the depicted scene. Some users were complaining about that fact. This is an inherent problem of visualizing three-dimensional data on a two-dimensional screen. A solution may be to use more sophisticated display

hardware like head mounted displays, where three-dimensional perception and camera movement are more intuitive.

Influence of the Redundancy

Although the robot redundancy increases the workspace, the dexterity and the flexibility of the manipulator, it causes unwanted effects in the visualization of the directly accessible workspace, both for translation and rotation. During motion, workspace boundaries can suddenly appear or disappear. This effect can be explained using Figure 4.7. A blocked null-space parameter interval can occur during motion. The red inaccessible peninsula at 0.1 progress and 0° null-space parameter shows such a phenomenon. It splits the optimization path into two possibilities and the chosen path depends on the initial elbow position. While the elbow is optimized during motion, the path can suddenly swap. The path can also swap for only some of the probing rays, which results in corona-like spikes in the visualization, as can be seen in Figure 7.1.

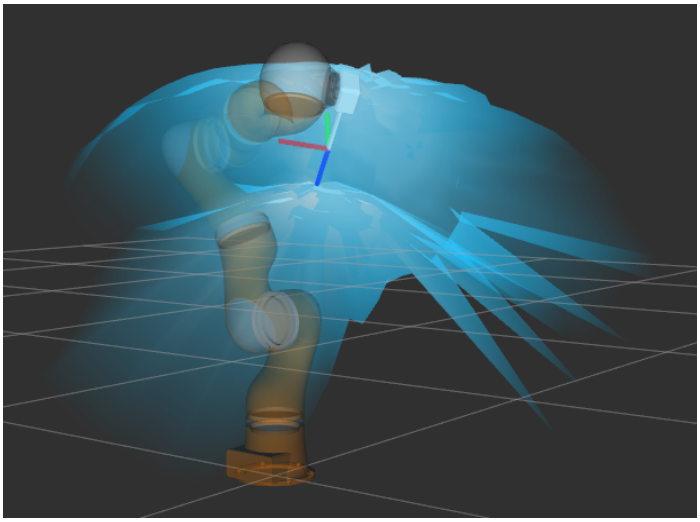

Figure 7.1: Some probe rays take a different optimization path for the elbow, which results in spike artifacts.

Although this issue does not occur very often, it can be confusing, especially if a wall appears where the robot just came from. This is an inherent problem of using a gradient descent method for elbow position optimization and can be solved in two ways: Either the elbow trajectory along each probing ray is explored using a much more sophisticated path finding algorithm (which would result in massive performance loss) or the initial elbow position is varied manually.

Problems with Layered Transparency

Most triangle-based rendering frameworks (such as RViz) cannot display intersecting transparent objects correctly. This is due to the fact that complete meshes are rendered consecutively, usually sorted by distance between their origin and the camera. This problem can be seen in Figure 7.2: The green pallet is either drawn completely before or completely behind the workspace. One of its target positions is inside the inner inaccessible sphere, which is very hard to see even when the camera is rotated.

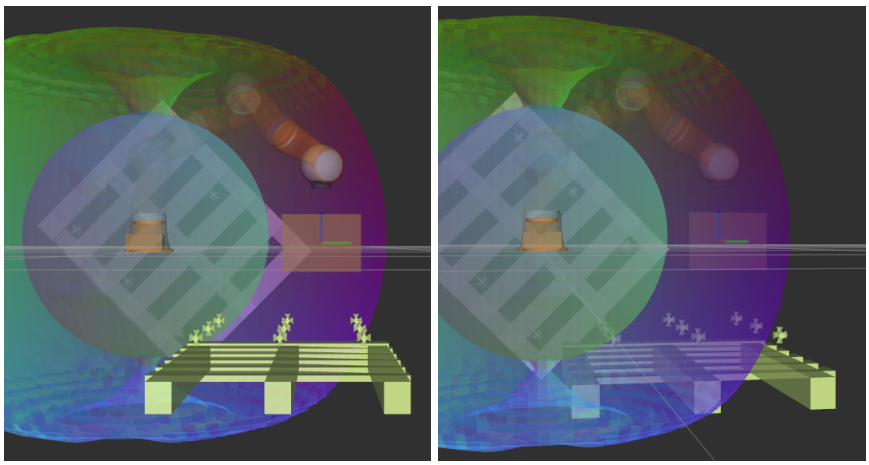

Figure 7.2: Effects of layered transparency: The pallet is either shown as being completely before or completely behind the workspace boundaries, depending on the camera position. No intersection is visible.

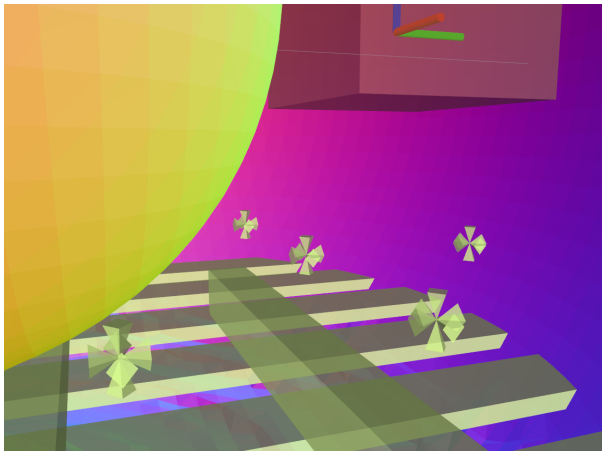

Figure 7.3: View from inside the opaque workspace: Intersections are displayed correctly and the reachability of the target positions can be assessed.

Using more advanced rendering software can solve this problem. If depth peeling is supported, objects are cut into successive depth layers, depending on the current camera view point. They can then be rendered in the correct order and intersections can be resolved. Ray tracing is another option, each pixel on the screen emits an individual viewing ray into the scene and each surface that is hit or crossed influences the resulting pixel color in the correct order. However, ray tracing comes with massive computation cost and is rarely used in interactive visualizations. Furthermore, using a visualization framework other than RViz – which is established as a standard in many robotic development projects – would have drastically reduced the compatibility with other works. RViz is under constant development and may support depth peeling in the future. The workaround that was used for the user study within this thesis was the option to turn the transparency off and move the camera into the region of interest (see Figure 7.3).

7.4 Comparison to Other Works

In this section, the differentiation from the closest work of other research groups will be made.

As pointed out in chapter 2, most other approaches generate or work with static maps. In some instances, such as in the work presented by Petitt and Miller [58] or Zacharias et al. [95, 96], the workspaces are precomputed in a database which can be accessed during runtime.

An advantage of their work with respect to the work at hand is, that only forward kinematics computations are used, which can easily be generated for arbitrary serial kinematic structures.

The disadvantages of precomputation are however, that the computation has to be restarted for parameter changes of the robot or the tool while the visualizations presented in this thesis update instantly. Furthermore, all offline sampling-based approaches can only generate a map of the generally accessible workspace. Internal boundaries due to workspace overlaps are neglected. The achievable resolution is also depending on the available memory.

Zacharias further presents a way of visualizing the complete workspace of a manipulator using colored spheres (capability map) or shapes representing from how many directions a point can be reached. This conveys an intuitively graspable overview of a robot's workspace in general within one single map.

The visualizations presented in this thesis are situation-specific and intended to be interacted with. In contrast to the capability map, it can be seen, which positions are reachable from a certain orientation.

The learning-based approaches presented by Stulp et al. [72] and by Jamone et al. [43] are also suitable for runtime employment. However, the obtained workspace representations are abstract, inaccurate and mainly usable by the robot itself. Aim of the thesis at hand is to make the reachability information visible and understandable to a human operator.

8 Conclusion & Outlook

This chapter provides a concluding summary over major scientific contributions to the state of the art and possible applications and an outlook with suggestions for further improvement.

8.1 Conclusion

As pointed out in the introduction, the major scientific questions that are addressed within the scope of this thesis are:

- How can workspace visualization benefit from on-the-fly computation on the GPU?
- Can it be used to substitute a natural sense of reachability for a robotic manipulator?

Different concepts were explored and devised within this context. A distinction was made between generally and directly accessible workspace and for both domains, visualization techniques for displaying translational and rotational boundaries have been investigated.

Four tools emerged, that were implemented and tested:

- An on-line visualization of the constant orientation workspace for the current TCP orientation allows to display the generally accessible workspace for translation.
- A fast visualization of the total orientation workspace for a given set of orientations serves as a hybrid for depicting the generally accessible workspace for position and orientation.
- An on-line visualization of the directly accessible workspace for translation from the current position in the current orientation offers a radar-like view of the motion limits.
- An on-line visualization of the directly accessible workspace for rotation from the current orientation at the current TCP position displays tilting and rolling limits for a tool.

Use cases for those tools are presented in chapter 5 and have been extended for evaluation in a user study in chapter 6. Some of the scenarios could also be solved without the need of any visualization or user interaction: Trajectories or objects can be positioned using optimization techniques. This applies to the parameters of a robotic structure and the shape of a tool as well. However, the strength of the presented approaches lies in their flexibility and in providing a visual understanding.

Human-operated robot control and human-robot interaction are becoming more and more popular. Robotic manipulators are used in a broader spectrum for many tasks, especially in research scenarios or in the context of the maker culture. Devising a dedicated optimization algorithm for every single task can require a lot of time and effort and having a useful multipurpose tool at hand is often sufficient. Furthermore, optimization might not provide a solution.

If the shape of the workspace is shown, one can often easily see if the shape of the trajectory will fit into it or not and how the trajectory has to be modified in order to fit. Also, optimization and visualization can be combined as was done in the trajectory trials in the user study.

Whether the presented concepts can serve as such a multipurpose tools is difficult to evaluate since it remains to be seen if they find acceptance in realistic scenarios. Two requests for making the visualization tools available to the ROS community and for further projects have already been made.

8.2 Contributions

Within the scope of the thesis at hand, the potential of on-the-fly workspace analyses has been explored. The following scientific contributions were made:

On-the-fly Visualization of Motion Boundaries

The live visualization of motion boundaries allows to display situation-relevant data only, which is especially beneficial for breaking down the five-dimensional workspace barrier into translation and orientation information for the current TCP pose.

Distinction Between Generally and Directly Accessible Workspace

By distinguishing between poses that are accessible in general and poses that are accessible in a straight motion, inner barriers that are the result of workspace self overlaps can be visualized.

Workspace Exploration by Parallel Simulated Motion

Probing the workspace by virtually moving in all directions constitutes a brute force strategy that can run in parallel on GPUs and can easily be extended to cope with other robotic manipulator structures.

Actual Freedom of Motion as Dexterity Measure

Displaying the effectively reachable volume around the current TCP position serves as a dexterity measure that can be more meaningful than measures based on the Jacobian matrix, which can only provide an estimate about the proximity to singularities and about direction-dependent speed or force/torque limits.

Novel Cost Function Merging Concept

A concept for aggregating joint-specific and general cost values into one combined cost scalar has been presented. It correctly propagates limits between inadmissible and admissible intervals for each partial cost into the final value without the need of infinite potential walls. This facilitates numeric algorithms to converge from inadmissible intervals into admissible ones and accurately detect the transition point.

8.3 Applications

All concepts have been implemented within the ROS environment and are ready to use. The only additional dependency is OpenCL.

Possible applications of the presented concepts include:

Visual Support for Manual Operation of Manipulators

For scenarios with tele-operated or hands-on control situations like surgery, installation or repair in human-unaccessible areas like space or under water or exploration of hazardous areas, visualization can help to navigate within the workspace.

Fast Workspace Analysis for Time-Critical Scenarios

If a robotic operation has to be performed quickly in an unknown environment, as can be the case in surgery or emergency management, being able to quickly assess the current or general reachability of the robot can be a crucial assistance for quicker and/or better performance.

Assistance for Robot or Target Placement and Repositioning

Displaying the scenario-specific workspace for a given tool and target orientation scope helps to easily determine a suitable robot pose in relation to the

target area (or vice versa), which can be particularly useful for mobile robots that have to be repositioned frequently.

Interactive Workspace Exploration for Design and Comparison of Robots and Tools

Being able to interactively determine the influence of parameters concerning the structure of the robot or the shape of a tool can assist during the design phase of new manipulators or tools.

Determination of Tolerance Around a Trajectory

Not only can critical areas be identified, the cause of the problem and a possible solution can be determined much more quickly compared to pure placement optimization results.

Flexible Tool for Robotics in Research Environments

Being able to see current motion boundaries can facilitate a quick setup of experiments and avoid time-consuming optimization or planning.

Fundament for Further On-the-Fly Analyses or Optimizations

The determined workspace boundaries can be used as input for further algorithms. The actual workspace volume around a position can serve as a quality criterion, trajectories can be planned and optimized using collision avoidance algorithms within the hull.

Creating a Feeling for a Robot‘s Positioning Capabilities

The results of the conducted user study are a strong indication that the visualization helps to convey an understanding of the reachability. Time will tell whether the concepts find broad acceptance in the robotics community.

8.4 Outlook

Some further research directions based on the proposed visualization concepts will be suggested in the following.

Generalization for Other Kinematic Structures

So far, the presented approach has been implemented for one robot architecture only. In order to extend it so it captures further structures, the inverse kinematics solution has to be implemented to run in OpenCL. Dealing with the very common 6-DoF anthropomorphic arm would actually simplify the situation a lot, it can be treated as a the regarded 7-DoF structure with fixed null-space parameter.

The visualization of the directly accessible workspace is applicable to robots with higher degrees of redundancy as well. The virtual motion for the probing rays can be implemented using numerical inverse kinematics solutions based on the pseudo inverse of the Jacobian, which works with any number of joints. For the generally accessible workspace however, being able to obtain a guaranteed solution for a given position is desirable, which favors a robotic structure that facilitates an analytical solution to the inverse kinematics problem.

Extended Workpiece Specification

With the methods presented in this thesis, it is possible to render the generally accessible workspace for one target TCP orientation as well as the intersection of the workspaces for different target TCP orientations. It may be desirable to access one point of the workpiece in one orientation and another point in one out of several orientations if the tool is symmetrical. In principle, this requires shifting of workspaces and further boolean operations which bears little implementation effort and no further computational load (one workspace computation per orientation).

Using Upcoming Display Technology

When the operator does not see the actual robot but only a camera image (as it is done in surgical scenarios or when the robot is operated from a distance), the workspace limits can be displayed within the image.

Head-mounted displays allow to comfortably navigate in three-dimensional space without the need to deliberately control the camera, from which the presented concepts could benefit a great deal, since the necessary camera motion was perceived as cumbersome by some test users.

If a form of augmented reality display is used, the workspace limits can even be displayed around the actual robot. This would facilitate using the system in hands-on operation.

Collision Avoidance

Basically, collision avoidance and its effect on the workspace shape can also be considered. However, this requires capturing the three-dimensional scene around the robot and collision checking can be computationally very expensive, depending on the detail of the models. Whether today's hardware is powerful enough to sample enough probe rays in sufficient resolution has to be evaluated.

List of Figures

List of Tables

Bibliography

[1] Karim Abdel-Malek, Frederick Adkins, Harn-Jou Yeh, and Edward Haug. On the determination of boundaries to manipulator workspaces. *Robotics and Computer-Integrated Manufacturing*, 13(1):63–72, 1997.

[2] Karim Abdel-Malek, Jingzhou Yang, Denis Blackmore, and KEN JOY. Swept volumes: fundation, perspectives, and applications. *International Journal of Shape Modeling*, 12(01):87–127, 2006.

[3] Karim Abdel-Malek and Harn-Jou Yeh. Geometric representation of the swept volume using jacobian rank-deficiency conditions. *Computer-Aided Design*, 29(6):457–468, 1997.

[4] Karim Abdel-Malek, Wei Yu, and Jingzhou Yang. Placement of robot manipulators to maximize dexterity. *International Journal of Robotics and Automation*, 19(1):6–14, 2004.

[5] D Alciatore and C Ng. Determining manipulator workspace boundaries using the monte carlo method and least squares segmentation. *ASME Robotics: Kinematics, Dynamics and Controls*, 72:141–146, 1994.

[6] Peter Anderson-Sprecher and Reid Simmons. Voxel-based motion bounding and workspace estimation for robotic manipulators. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2141–2146. IEEE, 2012.

[7] Muhammad Attamimi, Keisuke Ito, Tomoaki Nakamura, and Takayuki Nagai. A planning method for efficient mobile manipulation considering ambiguity. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 965–972. IEEE, 2012.

[8] Andreas Bihlmaier, Tim Beyl, Philip Nicolai, Mirko Kunze, Julien Mintenbeck, Luzie Schreiter, Thorsten Brennecke, Jessica Hutzl, Jörg Raczkowsky, and Heinz Wörn. Ros-based cognitive surgical robotics. In *Robot Operating System (ROS)*, pages 317–342. Springer, 2016.

[9] Andreas Bihlmaier and Heinz Wörn. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter Hands-on Learning of ROS Using Common Hardware, pages 29–50. Springer International Publishing, Cham, 2016.

[10] O. Bohigas, M. Manubens, and L. Ros. A complete method for workspace boundary determination on general structure manipulators. *Robotics, IEEE Transactions on*, 28(5):993–1006, Oct 2012.

[11] Ilian A Bonev and Jeha Ryu. A new approach to orientation workspace analysis of 6-dof parallel manipulators. *Mechanism and machine theory*, 36(1):15–28, 2001.

[12] J-H Borchard, F Dierßen, J Kotlarski, LA Kahrs, and T Ortmaier. Workspace analysis for evaluating laparoscopic instruments. In *New Trends in Mechanism and Machine Science*, pages 81–89. Springer, 2015.

[13] Paul Borrel and A Liegeois. A study of multiple manipulator inverse kinematic solutions with applications to trajectory planning and workspace determination. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1180–1185. IEEE, 1986.

[14] Paul Bourke, 1994.
http://paulbourke.net/geometry/polygonise/ (visited 10.12.2015).

[15] John Burkardt, 2013.
http://people.sc.fsu.edu/~jburkardt/cpp_src/sphere_grid/sphere_grid.html (visited 10.12.2015).

[16] Samuel R Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. 2004.

[17] Yi Cao, Suiping Qi, Ke Lu, Yi Zang, and Guanying Yang. An integrated method for workspace computation of robot manipulator. In *Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on*, volume 1, pages 309–312. IEEE, 2009.

[18] Yi Cao, Haihe Zang, Lan Wu, and Tao Lu. An engineeringoriented method for the three dimensional workspace generation of robot manipulator. *Journal of Information and Computational Science*, 8(1):51–61, 2011.

[19] Gianni Castelli, Erika Ottaviano, and Marco Ceccarelli. A fairly general algorithm to evaluate workspace characteristics of serial and parallel manipulators. *Mechanics based design of structures and machines*, 36(1):14–33, 2008.

[20] M Cenk Cavusoglu, Isela Villanueva, and Frank Tendick. Workspace analysis of robotic manipulators for a teleoperated suturing task. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 4, pages 2234–2239. IEEE, 2001.

[21] Andrzej J Cebula and Paul J Zsombor-Murray. Formulation of the workspace equation for wrist-partitioned spatial manipulators. *Mechanism and machine theory*, 41(7):778–789, 2006.

[22] Marco Ceccarelli. A formulation for the workspace boundary of general n-revolute manipulators. *Mechanism and Machine Theory*, 31(5):637 – 646, 1996.

[23] Tan Fung Chan and Rajiv V Dubey. A weighted least-norm solution based scheme for avoiding joint limits for redundant manipulators. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 395–402. IEEE, 1993.

[24] Gregory S Chirikjian and Imme Ebert-Uphoff. Discretely actuated manipulator workspace generation using numerical convolution on the euclidean group. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 1, pages 742–749. IEEE, 1998.

[25] Gregory S Chirikjian and Imme Ebert-Uphoff. Numerical convolution on the euclidean group with applications to workspace generation. *Robotics and Automation, IEEE Transactions on*, 14(1):123–136, 1998.

[26] Hee-Byoung Choi and Jeha Ryu. Convex hull-based power manipulability analysis of robot manipulators. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2972–2977. IEEE, 2012.

[27] WW Cimino and GR Pennock. Workspace of a six-revolute decoupled robot manipulator. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1848–1852. IEEE, 1986.

[28] M Cwiakala and TW Lee. Generation and evaluation of a manipulator workspace based on optimum path search. *Journal of Mechanical Design*, 107(2):245–255, 1985.

[29] M Cwiakala and TW Lee. Generation and evaluation of a manipulator workspace based on optimum path search. *Journal of Mechanical Design*, 107(2):245–255, 1985.

[30] Farbod Fahimi. *Autonomous robots: modeling, path planning, and control*, volume 107. Springer Science & Business Media, 2008.

[31] Tully Foote, Eitan Marder-Eppstein, and Wim Meeussen. http://wiki.ros.org/tf (visited 10.12.2015).

[32] KUKA Roboter GmbH.

[33] Y. Guan and K. Yokoi. Reachable space generation of a humanoid robot using the monte carlo method. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1984–1989, Oct 2006.

[34] Arun Gowtham Gudla. A methodology to determine the functional workspace of a 6r robot using forward kinematics and geometrical methods. 2012.

[35] K.C. Gupta. On the nature of robot workspace. *The International Journal of Robotics Research*, 5(2):112–121, 1986.

[36] KC Gupta and B Roth. Design considerations for manipulator workspace. *Journal of Mechanical Design*, 104(4):704–711, 1982.

[37] James A Hansen, KC Gupta, and SMK Kazerounian. Generation and evaluation of the workspace of a manipulator. *The International journal of robotics research*, 2(3):22–31, 1983.

[38] Dave Hershberger, David Gossow, and Josh Faust. http://wiki.ros.org/rviz (visited 10.12.2015).

[39] Dave Hershberger, David Gossow, and Josh Faust. http://wiki.ros.org/rviz/DisplayTypes/Marker (visited 10.12.2015).

[40] John M Hollerbach. Optimum kinematic design for a seven degree of freedom manipulator. In *Robotics research: The second international symposium*, pages 215–222. Cambridge, MIT Press, 1985.

[41] Oscar Amoros Huguet and Cristian Garcia Marin, 2011. https://code.google.com/archive/p/simple-opencl/ (visited 10.12.2015).

[42] Jessica Hutzl, Andreas Bihlmaier, Martin Wagner, Hannes Götz Kenngott, Beat Peter Müller, and Heinz Wörn. Knowledge-based workspace optimization of a redundant robot for minimally invasive robotic surgery (mirs). In *Conference on Robotics and Biomimetics*, 2015.

[43] Lorenzo Jamone, Lorenzo Natale, Kenji Hashimoto, Giulio Sandini, and Atsuo Takanishi. Learning the reachable space of a humanoid robot: A bio-inspired approach. In *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*, pages 1148–1154. IEEE, 2012.

[44] SCHUNK GmbH & Co. KG.

[45] D Kohli and J Spanos. Workspace analysis of mechanical manipulators using polynomial discriminants. *Journal of Mechanical Design*, 107(2):209–215, 1985.

[46] Rainer Konietschke, Tobias Ortmaier, Holger Weiss, Gerd Hirzinger, and Robert Engelke. Manipulability and accuracy measures for a medical robot in minimally invasive surgery. In *On Advances in Robot Kinematics*, pages 191–198. Springer, 2004.

[47] Anis Koubaa. *Robot Operating System (ROS): The Complete Reference*, volume 1. Springer, 2016.

[48] A Kumar and KJ Waldron. The workspaces of a mechanical manipulator. *Journal of Mechanical Design*, 103(3):665–672, 1981.

[49] Sang-Joo Kwon and Youngil Youm. General algorithm for automatic generation of the workspace for n-link redundant manipulators. In *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, pages 1722–1725 vol.2, June 1991.

[50] Bettina Laugwitz, Theo Held, and Martin Schrepp. *Construction and evaluation of a user experience questionnaire*. Springer, 2008.

[51] Z.-C. Lia and C.-H. Menq. The dexterous workspace of simple manipulators. *Robotics and Automation, IEEE Journal of*, 4(1):99–103, Feb 1988.

[52] M Lohmann, R Konietschke, A Hellings, C Borst, and G Hirzinger. A workspace analysis method to support intraoperative trocar placement in minimally invasive robotic surgery (mirs). In *Computer- und Roboterassistierte Chirurgie*, 2012.

[53] Carlos L Luck. Robot cartography: a topology-driven discretization for redundant manipulators. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 2, pages 1446–1451. IEEE, 1996.

[54] C.L. Lück and Sukhan Lee. Redundant manipulator self-motion topology under joint limits with an 8-dof case study. In *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, volume 2, pages 848–855 vol.2, Jul 1993.

[55] C.L. Lück and Sukhan Lee. Topology-based analysis for redundant manipulators under kinematic constraints. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 2, pages 1603–1608 vol.2, Dec 1995.

[56] D Oblak and D Kohli. Boundary surfaces, limit surfaces, crossable and noncrossable surfaces in workspace of mechanical manipulators. *Journal of Mechanical Design*, 110(4):389–396, 1988.

[57] Frank C Park and Roger W Brockett. Kinematic dexterity of robotic mechanisms. *The International Journal of Robotics Research*, 13(1):1–15, 1994.

[58] Joshua D Petitt and Karol Miller. Six-dimensional visualisation of end-effector pose using colour spaces. In *Proc. 2002 Australasian Conference on Robotics and Automation*, volume 27, page 29, 2002.

[59] Oliver Porges, Theodoros Stouraitis, Christoph Borst, and Maximo A Roa. Reachability and capability analysis for manipulation tasks. In *ROBOT2013: First Iberian Robotics Conference*, pages 703–718. Springer, 2014.

[60] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[61] J Rastegar and P Deravi. The effect of joint motion constraints on the workspace and number of configurations of manipulators. *Mechanism and Machine Theory*, 22(5):401–409, 1987.

[62] J Rastegar and P Deravi. Methods to determine workspace, its subspaces with different numbers of configurations and all the possible configurations of a manipulator. *Mechanism and Machine Theory*, 22(4):343–350, 1987.

[63] J Rastegar and B Fardanesh. Manipulation workspace analysis using the monte carlo method. *Mechanism and Machine Theory*, 25(2):233–239, 1990.

[64] Jason W Rauchfuss and Daniel CH Yang. A geometric approach for determining exact point accessibility of robotic manipulations. *Journal of Mechanical Design*, 122(3):287–293, 2000.

[65] Bernard Roth. Performance evaluation of manipulators from a kinematic viewpoint. *NBS Special Publication*, 459:39–62, 1976.

[66] Günter Schreiber, Andreas Stemmer, and Rainer Bischoff. The fast research interface for the kuka lightweight robot. In *IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications How to Modify and Enhance Commercial Controllers (ICRA 2010)*, pages 15–21. Citeseer, 2010.

[67] RG Selfridge. The reachable workarea of a manipulator. *Mechanism and Machine Theory*, 18(2):131–137, 1983.

[68] Masayuki Shimizu, Hiromu Kakuya, Woo-Keun Yoon, Kosei Kitagaki, and Kazuhiro Kosuge. Analytical inverse kinematic computation for 7-dof redundant manipulators with joint limits and its application to redundancy resolution. *Robotics, IEEE Transactions on*, 24(5):1131–1142, 2008.

[69] Masayuki Shimizu, Woo-Keun Yoon, and Kosei Kitagaki. A practical redundancy resolution for 7 dof redundant manipulators with joint limits. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4510–4516. IEEE, 2007.

[70] JA Snyman, LJ Du Plessis, and Joseph Duffy. An optimization approach to the determination of the boundaries of manipulator workspaces. *Journal of Mechanical Design*, 122(4):447–456, 2000.

[71] J Spanos and D Kohli. Workspace analysis of regional structures of manipulators. *Journal of Mechanisms, Transmissions, and Automation in Design*, 107(2):216–222, 1985.

[72] Freek Stulp, Andreas Fedrizzi, Franziska Zacharias, Moritz Tenorth, Jan Bandouch, and Michael Beetz. Combining analysis, imitation, and experience-based learning to acquire a concept of reachability in robot mobile manipulation. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 161–167. IEEE, 2009.

[73] Masaki Togai. An application of the singular value decomposition to manipulability and sensitivity of industrial robots. *SIAM Journal on Algebraic Discrete Methods*, 7(2):315–320, 1986.

[74] Bertrand Tondu. A zonotope-based approach for manipulability study of redundant robot limbs. *International Journal of Humanoid Robotics*, 10(03):1350023, 2013.

[75] Gracian Trivino and Jose San Martin. A fuzzy logic approach to the concept of manipulability in mechanics. In *Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International*, pages 1–5. IEEE, 2007.

[76] Ming-June Tsai. *Workspace geometric characterization and manipulability of industrial robots*. PhD thesis, The Ohio State University, 1986.

[77] YC Tsai and AH Soni. An algorithm for the workspace of a general nr robot. *Journal of Mechanical Design*, 105(1):52–57, 1983.

[78] RJ Urbanic and A Gudla. Functional work space estimation of a robot using forward kinematics, dh parameters, and shape analyses. In *ASME 2012 11th Biennial Conference on Engineering Systems Design and Analysis*, pages 381–391. American Society of Mechanical Engineers, 2012.

[79] Nikolaus Vahrenkamp, Tamim Asfour, and Rudiger Dillmann. Robot placement based on reachability inversion. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1970–1975. IEEE, 2013.

[80] Nikolaus Vahrenkamp, Tamim Asfour, Giorgio Metta, Giulio Sandini, and Rudiger Dillmann. Manipulability analysis. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 568–573. IEEE, 2012.

[81] Liquan Wang, Jianrong Wu, and Dedong Tang. Research on workspace of manipulator with complicated constraints. In *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pages 995–999. IEEE, 2008.

[82] Yunfeng Wang and Gregory S Chirikjian. Workspace generation of hyper-redundant manipulators as a diffusion process on se (n). *Robotics and Automation, IEEE Transactions on*, 20(3):399–408, 2004.

[83] KUKA Website. http://www.kuka-robotics.com/en/pressevents/news/NN_060515_Automatica_02.htm (visited 10.12.2015).

[84] KUKA Website. http://www.kuka-robotics.com/germany/en/products/industrial_robots/sensitiv/ (visited 10.12.2015).

[85] Schunk Website. http://mobile.schunk-microsite.com/en/produkte/products/dextrous-lightweight-arm-lwa-4d.html (visited 10.12.2015).

[86] Chi-Haur Wu and Hernando Valencia. Trajectory feasibility based on cartesian workspace for robot manipulators. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1865–1870. IEEE, 1986.

[87] DCH Yang and ZC Lai. On the dexterity of robotic manipulators—service angle. *Journal of Mechanisms, Transmissions, and Automation in Design*, 107(2):262–270, 1985.

[88] DCH Yang and TW Lee. On the evaluation of manipulator workspace. *Journal of Mechanism, Transmission, Automation in Design*, 105:70–77, 1983.

[89] DCH Yang, EY Lin, and SY Cheng. Primary workspace of industrial robots with roll-pitch-yaw wrists. *Journal of Mechanical Design*, 112(3):347–353, 1990.

[90] Anna Yershova, Swati Jain, Steven M Lavalle, and Julie C Mitchell. Generating uniform incremental grids on so (3) using the hopf fibration. *The International journal of robotics research*, 2009.

[91] T. Yoshikawa. Dynamic manipulability of robot manipulators. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 1033–1038, Mar 1985.

[92] Tsuneo Yoshikawa. Analysis and control of robot manipulators with redundancy. In *Robotics research: the first international symposium*, pages 735–747. Mit Press Cambridge, MA, 1984.

[93] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985.

[94] Tsuneo Yoshikawa. Translational and rotational manipulability of robotic manipulators. In *Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON'91., 1991 International Conference on*, pages 1170–1175. IEEE, 1991.

[95] Franziska Zacharias. *Knowledge representations for planning manipulation tasks*, volume 16. Springer Science & Business Media, 2012.

[96] Franziska Zacharias, Christoph Borst, and Gerd Hirzinger. Capturing robot workspace structure: representing robot capabilities. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3229–3236. Ieee, 2007.

[97] Franziska Zacharias, Christoph Borst, and Gerd Hirzinger. Object-specific grasp maps for use in planning manipulation actions. In *Advances in Robotics Research*, pages 203–213. Springer, 2009.

[98] Franziska Zacharias, Christoph Borst, and Gerd Hirzinger. Online generation of reachable grasps for dexterous manipulation using a representation of the reachable workspace. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–8. IEEE, 2009.

[99] Franziska Zacharias, Ian S Howard, Thomas Hulin, and Gerd Hirzinger. Workspace comparisons of setup configurations for human-robot interaction. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3117–3122. IEEE, 2010.

[100] Franziska Zacharias, Wolfgang Sepp, Christoph Borst, and Gerd Hirzinger. Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 55–61. IEEE, 2009.

[101] H. Zghal, R. V. Dubey, and J. A. Euler. Efficient gradient projection optimization for manipulators with multiple degrees of redundancy. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 1006–1011 vol.2, May 1990.