Chair of Applied Mechanics
Department of Mechanical Engineering
Technical University of Munich

Kinematic Structure Optimization for Humanoid Robots

Kinematische Strukturoptimierung für Humanoide Roboter

Scientific work for obtaining the academic degree
Master of Science (M.Sc.)
at the Department of Mechanical Engineering of the Technical University of Munich

Supervisor Prof. Dr.ir. Daniel J. Rixen
Chair of Applied Mechanics

Advisor Philipp Seiwald, M.Sc.,
Chair of Applied Mechanics

Dr.-Ing. Daniel Wahrmann (Co-Supervision, FRANKA EMIKA GmbH)

Submitted by Nicolas Neuburger

Submitted on July 7, 2019 in Garching bei München

Abstract

Humanoid robots represent complex technical systems with a high number of degrees of freedom which must provide human-like capabilities to perform everyday tasks. Mechanical designers must therefore carefully determine the robot's kinematic structure by determining a joint arrangement which facilitates a suitable workspace for these task applications. Research has provided several approaches to optimize a kinematic structure numerically. However, the proposed objective functions are mostly simple and do not provide the incorporation of task properties except for the reachability of predefined task poses. This thesis derives a methodology for a task-oriented structure optimization for humanoid robots. It allows to formulate an objective function which statically evaluates the robot's kinematic dexterities within predefined task areas regarding position and orientation properties of the task descriptions. Additionally it is considered, that the robot can execute the task with a single end effector or with both hands in a coupled state. The thesis finally provides a software framework which takes an initial structure topology and finds a good or even optimal set of dimensional parameters based on the formulated objective funtion, using meta-heuristic numerical optimization methods. The applicability of the optimizers and the methodology in general is shown with three application examples.

Zusammenfassung

Humanoide Roboter stellen technisch komplexe Systeme mit einer hohen Anzahl an Freiheitsgraden dar, die menschenähnliche Fähigkeiten zur Aufgabenbewältigung besitzen müssen. Während der Entwicklung eines Roboters muss eine Gelenkstruktur entwickelt werden, die aufgabenbezogene kinematische Fähigkeiten in relevanten Bereichen des robotischen Arbeitsraums bereitstellt. Dies stellt ein Optimierungsproblem dar, das mit Hilfe von numerischen Optimierungsmethoden gelöst werden kann. Bisherige Forschungsarbeiten konzentrieren sich auf die Formulierung einfacher Zielfunktionen, die einen Aufgabenbezug nur durch die Erreichbarkeit aufgabenrelevanter Roboter Posen zulassen. In dieser Masterarbeit wird eine Methodik zur statischen aufgabenorientierten Optimierung der kinematischen Struktur humanoider Roboter erarbeitet. Die Formulierung der Zielfunktion erfolgt durch die Definition aufgabenspezifischer Eigenschaften. Neben der Auswahl aufgabenbezogener kinematischer Gütekriterien, wird auch die Definition eines Aufgabenbereichs innerhalb des robotischen Arbeitsraums miteinbezogen. Außerdem wird berücksichtigt, ob der Roboter die jeweilige Aufgabe mit einem oder mehreren gekoppelten Endeffektoren ausführt. Die vorgestellte Methodik wird durch die Implementierung eines Software Frameworks evaluiert, das Parameter einer initialen Gelenkstruktur mit Hilfe meta-heuristischer Verfahren optimiert. Die Anwendbarkeit der Verfahren und die Validierung der Methodik wird durch drei Anwendungszenarien nachgewiesen.

Abbreviations

2D two-dimensional.

3D three-dimensional.

AM Chair of Applied Mechanics.

BB Bounding Box.

CI Condition Index.

CMA-ES Covariance Matrix Adaption - Evolution Strategy.

CPU Central Processing Unit.

CSV Comma-Seperated-Values.

DH Denavit-Hartenberg.

DLR German Aerospace Center.

DMI Dual Arm Manipulability Index.

DoF Degrees of Freedom.

EM Endeffecting Mode.

ES Evolution Strategy.

FK Forward Kinematics.

GA Genetic Algorithm.

GCI Global Condition Index.

GII Global Isotropy Index.

HYB Hybrid Kinematics.

I/O Input/Output.

IK Inverse Kinematics.

JRA Joint Range Availability.

JSON JavaScript Object Notation.

MC Metropolis Criterion.

MM Manipulability Measure.

NR Newton-Raphson.

PLY Polygon File Format.

PSO Particle Swarm Optimization.

RI Reachability Index.

SA Simulated Annealing.

SE(3) Special Euclidean Group.

SO(3) Special Orthogonal Group.

STL Standard Tessellation Language.

SVD Singular Value Decomposition.

SWA Spiral Walking Algorithm.

TOC Task-centric Optimization of robot Configurations.

TUM Technical University of Munich.

UML Unified Modeling Language.

URDF Unified Robot Description Format.

VFS Voxel-Filling Sphere.

VTR Velocity Transmission Ratio.

XML Extensible Markup Language.

Notation

Pattern	Meaning	Example
lower case	scalar	n
bold, lower case	column vector	$\boldsymbol{x}$
bold, upper case	matrix	$\boldsymbol{J}$
left subscript	coordinate frame of representation	${}_0\boldsymbol{x}$

Diacritical mark	Meaning	Example
$\widetilde{}$	skew-symmetric representation of a vector	${}_{k_D}\widetilde{\boldsymbol{a}}_k$
$\dot{}$	derivative with respect to time	$\dot{\boldsymbol{x}}_k$
$\hat{}$	projected representation	$\hat{\boldsymbol{a}}_{c,x}$

Important Symbol	Meaning	Example
$\mathbb{R}$,$\mathbb{N}$/$\mathbb{R}$,$\mathbb{N}$	real/natural number range including 0	-
$\boldsymbol{a}$	frame or joint related axis	$\boldsymbol{a}_{c,x}$
$\boldsymbol{R}$	3×3 rotation matrix	${}_{k_D}\boldsymbol{R}_k$
$\boldsymbol{t}$	3×1 translation vector	${}_0\boldsymbol{t}_k$
$\boldsymbol{T}$	homogeneous transformation	${}_0\boldsymbol{T}_k$
$\boldsymbol{I}$	identity Matrix	-
$\boldsymbol{J}$	end effector Jacobian Matrix	$\boldsymbol{J}_k$
n	number of joint configuration space dimensions	-
m	number of workspace dimensions	-
$\boldsymbol{x}$	workspace pose vector	-
$\boldsymbol{q}$	joint configuration	-
M	local kinematic metric	M_{RI}
$\mathcal{M}$	global kinematic metric	$\mathcal{M}_{VTR}$
$\mathcal{R}(J)$	range of all velocity direction components for the given joint configuration	-
$\mathcal{R}(J)^{\perp}$	range of all degenerated velocity direction components for the given joint configuration	-
σ	singular value of a matrix	$\sigma_{J,k}$

Operator	Meaning
∂	partial derivative
$\otimes$	outer vector product
$\preceq$ / $\succeq$	condition holds if for all elements $\leq$ / $\geq$ is true.
$\|\|...\|\|$	euclidean norm
$\|...\|$	absolute representation
atan2(...)	range-displaced $atan2 \in [0, 2\pi]$
diag(...)	block diagonal matrix
$\arg\min(...)$	argument that minimizes (...)
$\det\|...\|$	determinant of a squared matrix
$rank(...)$	rank of the (...) matrix argument

Script	Meaning	Example
$\dagger$	pseudo inverted matrix	$\boldsymbol{J}^{\dagger}$
0	relative to inertial frame	${}_{0}\boldsymbol{T}_{k}$
D	related to default joint status	${}_{p(k)}\boldsymbol{T}^{D}_{k_d}$
J	related to current joint status	${}_{p(k)}\boldsymbol{T}^{J}_{k_d}$
T	transposed representation	$\boldsymbol{a}^{T}_{c,x}$
k	related to frame of k-th segment	$\boldsymbol{x}_{k}$
k_D	related to frame of k-th segment in default joint position	${}_{k_D}\boldsymbol{R}_{k}$
$p(k)$	related to parent frame of k-th segment	${}_{k}\boldsymbol{T}_{p(k)}$
$x/y/z$	$x/y/z$ axis or component	$\boldsymbol{a}_{c,x}$, $\boldsymbol{a}^{x}_{c,x}$
ta	related to a task area	$W_{ta,table}$
e	related to a specific end effector	$\boldsymbol{x}_{e}$
l	related to the l-th task	$EM(l)$
$dext$	related to a dexterity	$f_{dext,l}$
d	discrete representation	$\boldsymbol{x}_{d}$
c	continuous representation (for emphasis)	$\boldsymbol{x}_{c}$
EM	related to a specific Endeffecting Mode (EM)	$M_{EM}(\boldsymbol{x}_{d})$
$EM(l)$	related to a the Endeffecting Mode (EM) of task l	$M_{RI,EM(l)}(\boldsymbol{x}_{t,d})$

Contents

Chapter 1

Introduction

Advances in force-based control, perception and mechatronics, among others, have facilitated a safe application of robots in environments that are shared with humans. This caused a great diversification in the range of tasks robots are suitable to fulfill. Now a robot can fulfill human collaborative tasks like supporting the assembly of a differential gear housing [1]. Tasks of such kind partly require human-like skills and the ability to interact with humans. BROOKS [2, p.3f] states that humans naturally switch to a human-human interaction pattern whenever they are interacting with robots that possess human-like traits. A humanoid appearance is therefore an increasingly important component of robotics research.

Several implementations of humanoid robots have been investigated with various application focuses. *Armar-IV* for instance is the consequent advancement of the omnidirectional wheel-based humanoid platform *Armar-III* and adds bipedal motion among other new features to the platform's abilities. A special focus was put on the design of arms and hands to allow bimanual grasping and manipulation for collaborative task execution in service applications [3]. Research of two-armed manipulation was also conducted at the German Aerospace Center (DLR) using the robot *Rollin' Justin* [4]. At the Chair of Applied Mechanics (AM) of the Technical University of Munich (TUM) the humanoid robot *LOLA* has been developed to investigate bipedal walking. 24 Degrees of Freedom (DoF) enable human-like walking motion. While force-sensing, actuated toes and seven DoF legs allow a rolling foot motion when walking, an inertia sensing torso and movable arms allow control of the full-body center of gravity [5].

Figure 1.1: Humanoid robots in the field of research: LOLA (left), Rollin' Justin (middle) and Armar-IV (right).

The limbs of most of these humanoid implementations have been explicitly designed to fulfill desired tasks. *LOLA's* legs allow versatile foot poses for contact controlled stabilization on uneven ground while simultaneously maintaining a controllable motion state of the full-body center of gravity. The arm structure possesses a less complex structure with three DoF, which is enough to guarantee the required compensation motion counteracting the walking rhythm [5]. *Armar-IV* on the other hand, is equipped with eight DoF and eleven DoF fluidic actuated hands. The shoulder positioning and topology of the arm was chosen to provide high dual-arm dexterity in areas in front of the robot [6].

From these examples it becomes obvious that the mechanical design of a robot determines if and how well they are kinematically and dynamically capable of performing a task. In order to ensure the necessary kinematic capabilities, the robot's kinematic structure must provide a workspace that serves the necessary operability within the task regions. Suitable joint arrangements for low complex structures and basic tasks are well achievable from engineering experience and with the use of analytical methods. For the design of highly complex structures, like those for humanoid robots, computational assistance is indispensable, since task descriptions of humanoids are more intricate and such structures can have redundant joint space dimensions. This assistance requires solving an optimization problem that determines a kinematic structure offering the best balance regarding its structural complexity and its fitness to perform preknown and future tasks.

The task-related kinematic fitness of a robot can be interpreted as the provision of required kinematic dexterities within task spaces. The following definition for kinematic dexterity in general is given [7, p.3], [8].

> *Kinematic dexterity encompasses the reachability of poses and the ease of arbitrarily changing the position and orientation from these poses*.

In order to provide a corresponding fitness formulation, the robot's dexterity must be represented quantitatively. This can be achieved by using kinematic metrics, which are capable of representing specific aspects of kinematic dexterity for a given joint configuration in the local scope and their distribution within the robot's workspace, thus global scope. Using the global scope representation, a robot's kinematic dexterity can be evaluated and compared to its alterations. An alteration of a structure can be imposed topologically and dimensionally [9]. The topology of a robot's structure relates to the number, type and axis orientation of joints. Link length and the deduced joint poses belong to dimensional modifications. According to KIVELÄ ET AL. [9] an optimization process of kinematic structures starts with the optimization of the robot's topology and subjects the results to dimensional optimization.

Most approaches regarding kinematic structure optimization for robots, e.g. [10], [7] or [11], tend to increase dexterity of the full workspace which can be classified as task-unrelated optimization. Task-related optimizations have been investigated e.g. in [9] and [12]. The two approaches choose the maximal reachability for task-related poses as the optimization objective. According to the definition given above, this only refers to a part of kinematic dexterity. Valuable information which can be derived from several existing kinematic metrics is not incorporated. To the author's knowledge, the application of kinematic structure optimization has not been investigated very intensively. Studies can be found that evaluate, but not optimize, the common workspace of a dual-armed humanoid (see [13]). Further research on humanoid structures, like the work of VAHRENKAMP ET AL. [3] or KAPUSTA AND KEMP [14] have concentrated on determining optimal joint space configurations for the execution of tasks.

1.1 Outline

Based on the presented state-of-the-art, further investigation on design optimization of humanoid robots is required. It is additionally desirable to develop a methodology which allows the incorporation of task properties to an objective function in an optimization problem. Here, the significance of several developed kinematic metrics should be regarded. Ideally, such methodology should also provide a general applicability to various forms of robots which requires a generic software framework. This thesis tries to approach these aspects and fulfill the following key objectives.

- Efficient computation of a humanoid robot's workspace through forward and inverse kinematic relations.
- Formulation of cost functions for user-defined tasks.
- Implementation of suitable numeric optimization algorithms.
- Usability of the implementation through an intuitive configuration of the software and an appropriate visualization of results.

In order to derive a suitable concept for these objectives, the thesis first analyzes the previous research work and defines fundamental background information in Chapter 2. A generically applicable kinematic workspace analysis of (non-) redundant robots and humanoids is examined. Then, the numerical representation of kinematic dexterities in form of kinematic metrics is investigated. With the dexterity objective in mind, numerical optimization methods are introduced which allow the efficient determination of an optimally parameterized structure.

In the conceptual Chapter 3 a methodology for the numerical representation of a task-related fitness of a structure is presented which is applicable to humanoid structures and other forms of robots. Finally, the usability of these methods is evaluated through the implementation of a software framework which is explained in Chapter 4. Among others, it is applied to the design optimization of a new arm for the humanoid *LOLA* for in-motion stabilization tasks and the optimization of two 7 DoF robots for dual-arm manipulation at a conveyor belt.

Chapter 2

Theoretical Background

This chapter provides the theoretical basis for the concept and methodology of this thesis. It starts with a general overview of state-of-the-art methodologies in Section 2.1. From the presented related work the necessity for the presentation of fundamental background information is derived. In Section 2.2 methods for kinematic modeling of humanoids are introduced and kinematic relations for robotic systems are derived. Subsequently, kinematic metrics are introduced which evaluate the kinematic dexterity of a structure (see Section 2.3). The analysis of these metrics within the robot's workspace must be performed numerically, because analytical expressions with respect to poses within the workspace cannot be derived in general. This requires a suitable discrete representation of the workspace. Therefore, discretization schemes for a robot's workspace are discussed in Section 2.4. Finally, optimization techniques, applicable to the expected form of optimization problems, are examined in Section 2.5.

2.1 Related Work

Kinematic structure optimization for robotic systems has been studied intensively in the past 35 years. First investigations, like [15] in 1985 or [16] in 1991, were focused on inventing general metric formulations for the representation of kinematic dexterities. This is analyzed in Section 2.3. One of the first kinematic design studies was conducted in 1991 by MA AND ANGELES [17]. The ideal set of geometrical parameters of a platform manipulator was determined by an analytical investigation of the conditioning of the end effector Jacobian's using Condition Index (CI). The restriction to analytically derivable objectives has been disregarded, when it was shown that numerical optimization approaches allow to successfully solve non-analytical structure optimization problems. In 1998, STOCCO ET AL. [18] utilized a modified version of the *branch and bound* solver to find an optimally parameterized structure regarding kinematic isotropy. They successfully tested the methodology on a five-bar linked parallel manipulator and a Stewart platform. As explained in Section 2.5, the *Branch and bound* solver belongs to the family of *complete methods* and thus obtains the global optimum through an intensive exploration of the parameter space. However, in 2002, KHATAMI AND SASSANI [19] showed that *complete methods* are problematic for finer discretized or multi-dimensional parameter spaces. They therefore proposed a more efficient exploration of the parameter space by applying *approximate methods* to the optimization problem. Through this, they were able to show that the Genetic Algorithm (GA) can derive a suitable pair of segment lengths of a two-dimensional (2D) robot with two revolute joints.

Based on these discoveries, several design studies of robots have been conducted on parallel and serial robots. An exemplary work on parallel robots can be found in [11] from 2009. The

link-length ratios of a parallel ankle rehabilitation robot were optimized regarding the CI and minimum actuator force. This was achieved by the use of a modified version of the GA. A design study of a serial manipulator can for instance be found in [10]. Here, several industrial redundant manipulators were tested regarding their workspace size when mounted on top of a *Mars- and Moon Rover Unit (LRU)*. Secondarily, the best mounting angle was determined. Another design study of serial manipulators is found in [20]. While the design studies above focus on optimizing the robot regarding single kinematic metrics, the incorporation of multiple kinematic metrics for a single dexterity objective was presented by ABDEL-MALEK ET AL. [21] in 2004. This methodology was for instance applied by PUGLISI ET AL. [22] who optimized the leg-lengths, the radius of the pod's base and top plates for a spherical four-legged parallel robot. A different approach was taken by KHAN ET AL. [23] in 2014 who interpreted each metric as an objective of its own and formulated a multi-objective problem using a Multi-Objective GA. However, research has concentrated on single-objective formulations of kinematic structure optimization. The studies above can be considered task-unrelated, however, also the incorporation of task properties to the optimization problems has been studied. PATEL AND SOBH [12], for instance, have optimized Denavit-Hartenberg (DH) parameters of several serial manipulator topologies for a predefined set of task poses. Simulated Annealing (SA) was hereby classified as suitable to kinematic structure optimization. The same was achieved in [9] where an eight DoF tunnel-drilling robot was dimensionally optimized. Optimal DH parameters were determined regarding the collision-free reachability of drilling poses.

In the examples above the workspace computation determined a big part of the computational effort of the structure evaluation. Because of this, several different computation strategies have been proposed among the cited work. Approaches which purely depend on the kinematic relations of Forward Kinematics (FK) use general joint configuration space sampling (see ZACHARIAS ET AL. [7]) or strategic sampling via the Monte-Carlo method (see [24]). Latter allows the incorporation of kinematic constraints which are present e.g. for parallel robots. Although these approaches possess low computational requirements and are able to compute a large number of workspace mappings, they do not guarantee that the workspace is covered in finite time, e.g. when the robot is redundant [10]. On the other hand, a workspace computation purely based on Inverse Kinematics (IK) is computationally very expensive (see Section 2.2.3). Thus, two approaches can be found which combine the advantages of both kinematic relations by computing a part of the workspace with FK and fill unmapped workspace poses via IK. This was for instance investigated by PORGES ET AL. [10]. They introduced the term Hybrid Kinematics (HYB) for this combined approach. HYB was also used in 2013 by ZACHARIAS ET AL. [7]. They evaluated joint configuration samples with FK to obtain feasible joint space configurations for workspace positions. Then, these configurations are fed into IK to generate feasible joint space configurations within the orientation space of the given position.

Since the workspace of arbitrary robots can be computed in an efficient manner, the more involved applications of humanoid robotics have been studied. In [13], the evaluation of a reachable workspace of bimanual humanoid robots was investigated without closer consideration of the bimanual coupling of the end effectors. Further studies with a more task-oriented approach relate to the determination of an optimal task-relative positioning and orientation of the robot or humanoid, respectively. An important work was published in 2013 in [7] where the well-known *Capability Map* was proposed. It represents a workspace database where kinematic metrics can be queried in order to derive suitable poses and generate task-conform trajectories. The same form of database was subsequently exploited by VAHRENKAMP ET AL. [3] to find suitable base placements of a humanoid for grasping tasks. A different methodology was invented by KAPUSTA AND KEMP [14] with the Task-centric Optimization

of robot Configurations (TOC) methodology. It determines optimal initial joint space configurations to start human-assistive tasks. The TOC methodology was successfully tested by applying the Covariance Matrix Adaption - Evolution Strategy (CMA-ES).

Finally, a very different approach from the presented methodologies is mentioned which optimizes a serial structure based on the simulation of task execution [25], [26]. This work is used to emphasize the restriction of this thesis to the static consideration of kinematic dexterities, because it imposes several dependencies. First, it incorporates dynamic properties which can usually only be determined after defining the kinematic structure. The formulation of joint actuator requirements and the system's inertia typically depend on the kinematics. Second, the optimization results are only optimal for the given set of simulator, task planner and controller. These dependencies are, thus, considered undesirable for this thesis which aims for an isolated optimization of the kinematic structure.

2.2 Kinematics

The field of kinematics refers to the geometric description of a system in space with respect to time, whereas the subject of dynamics analyzes the cause of movements. This means that kinematics can describe the pose, the velocity and acceleration of any component of the system at a given point in time through the time-variant states of its DoF. This mapping is called FK whereas its inverse formulation is titled as the IK of a system.

Robotic systems can interact with the world through their end effectors. Therefore, the robot's FK normally relates to the mapping from joint states in the *joint space* to the end effector's pose, velocity or acceleration in the *workspace*. The kinematic mapping has different properties depending on the structure. In this thesis only hierarchical structures are considered for reasons of simplification and because this topology allows to model a humanoid by interpreting the torso as a floating root component with chains attached to it which represent the extremities [27, p.46]. This strategy is also applied in many of the introduced humanoid projects including *LOLA* which represents the context of this thesis [28, p.15]. Based on this assumption it is restricted that branching within the hierarchical structure only occurs in rigid connection to the system's root segment and that the structure does not contain kinematic loops.

The description of hierarchical kinematic models is investigated in the following Section 2.2.1. The FK of such tree structures is always surjective. A joint configuration $\boldsymbol{q}$ maps to one workspace pose $\boldsymbol{x}$ and can thus be calculated straightforwardly as described in Section 2.2.2. The IK of such topology is more complex. Most of the time, an analytical description is not easily derivable and the number of feasible joint configurations varies dependent on structural properties. The functional relations of FK and IK are presented in the following Subsections 2.2.2 and 2.2.3. Only relations necessary for the understanding of the thesis' concepts are derived. The presented information is based on [29], [27], [30]. Interested readers can consult this literature for further kinematic relations and fundamentals in the field of robotics.

2.2.1 Hierarchical Kinematic Modeling

A hierarchical model can be described by a sequence of segments and joints with their topological and geometrical relation. A segment in this context refers the rigid body of a system component without its connectors (joints) to other components. Within this thesis three assumptions are made for the hierarchical kinematic modeling which are conform to the literature:

- Joints are considered to be ideal without backlash, elasticity or intolerances.
- Segments are assumed to be rigid bodies.
- Complex joint types, like helical, cylindrical, planar, spherical or floating joints can usually be modeled by a chain of prismatic and revolute joints.

To define the hierarchical dependencies between joints and segments, as well as their relative poses, the kinematic model must withhold the following information:

1. Identification of the parent and child segments of a joint.
2. Relative pose of the child segment to its parent segment for when the joint is in its default position.
3. Joint type (prismatic or revolute) and the axis of translation and rotation.
4. Kinematic joint constraints, such as joint position, velocity and acceleration limits (optionally).

The description of these relations can be accomplished in different ways. A well-known modeling scheme for instance is the DH convention. It reduces the information to a sequence of four parameters, the so called DH parameters which simplify the calculation of the FK. This is accomplished by constraining the way coordinate systems are defined for each joint. A more general scheme without this constraint is used for the popular Unified Robot Description Format (URDF) [31]. It allows an arbitrary definition of coordinate frames for each joint and segment. A valuable asset provides the definition of *fixed* joints that provide no DoF to a connection of segments. Thus, it is possible to describe a rigid body by multiple rigid bodies. URDF is well accepted in robotics today and is used for the hierarchical modeling of robots in this thesis. The following subsection introduces the kinematically relevant description possibilities in URDF.

Unified Robot Description Format (URDF)

The URDF is based on the Extensible Markup Language (XML) and is intended to represent a rich description format for kinematic and dynamic modeling. The following kinematically relevant definitions are visualized in Figure 2.1 where the URDF snippet defining the kinematic relationship between *Link 1* and *Link 2* for an exemplary tree structure is given. Each segment of a system is represented by a *link* element and joints with an identically named element both containing a *name* attribute for identification purposes. *Link* elements represent a frame for the given segment and can contain descriptions like collision volumes or the inertia. However, kinematically relevant are the definitions inside of the *joint* element which follow the structure of the enumeration in the predecessing Section 2.2.1. First, the topological parent-child relation of the segments connected to the joint is formulated by defining the particular link's *name* attribute inside of the *parent* and *child* elements. Next, the relative

transformation between the parent's link frame and the child's link frame in the zero displacement state is defined in the *origin* element node. It is described by a translation vector in the *xyz* attribute, followed by a rotation vector in the *rpy* attribute, defining the roll, pitch and yaw angles of a subsequent fixed axis rotation around the X,Y and Z axis. The DoF of the joint are defined by an attribute *type* and a three-dimensional axis vector stored in the *axis* element. Lastly, kinematic joint limits can be defined with a minimal and maximal joint displacement in the corresponding *limit* node. For more information on the URDF format see [31].

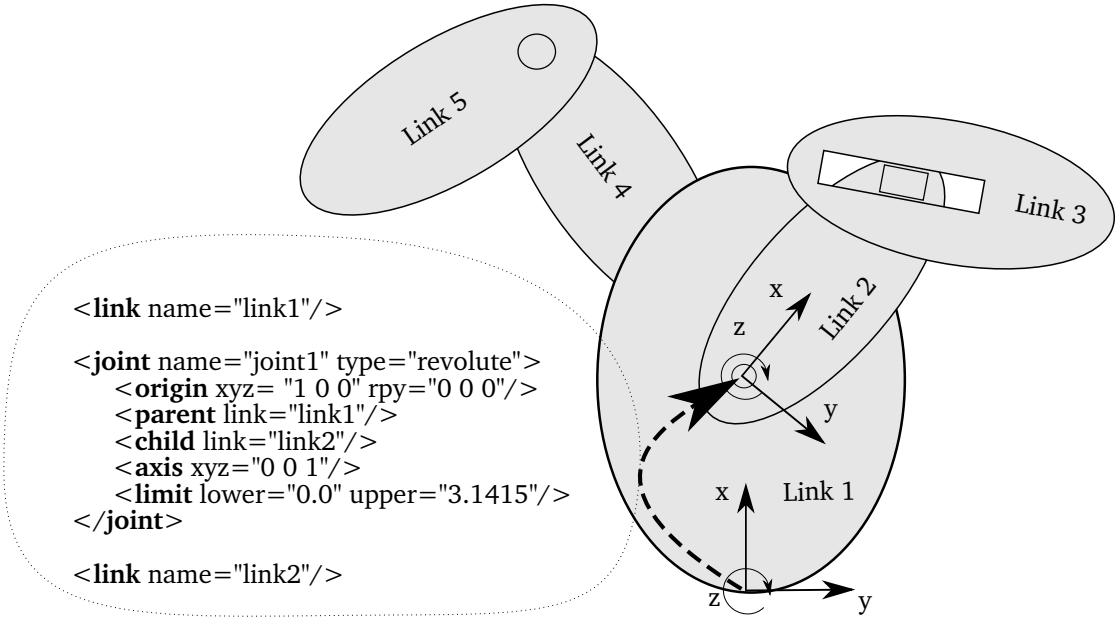

Figure 2.1: URDF-based definition of the relationship between *Link 1* and *Link 2* of an exemplary hierarchical kinematic structure.

2.2.2 Forward Kinematics

The FK mapping of hierarchical structures can be computed recursively, because the pose of the k-th segment frame in the tree structure is dependent on the pose of its $p(k)$ parent frame. The recursive calculation of the homogeneous transformation ${}_0\boldsymbol{T}_k \in \mathbb{R}^{4\times4}$ from the inertial frame to each segment's frame is described in the following algorithm relating to the conventions of URDF.

Algorithm 1: Recursive Computation of Homogeneous Segment Frame Transformations.

1 ${}_0\mathbf{T}_0 = \mathbf{I}$
2 **for** k *in range* 1 *to* N **do**
3 $\quad {}_0\mathbf{T}_k = {}_0\mathbf{T}_{p(k)} \left({}_{p(k)}\mathbf{T}_{k_D}^{D} \; {}_{k_D}\mathbf{T}_{k}^{J} \right)$
4 **end**
5 **return** ${}_0\mathbf{T}_k$

Here, ${}_{p(k)}\mathbf{T}_{k_D}^{D}$ denotes the modeled relative transformation between the frames of child segment k and parent segment $p(k)$ when the joint is not displaced with $q_k = 0$. This default state is referred to with the superscript D and the frame with k_D. For revolute or prismatic joints,

the rotation around the unit joint axis ${}_{k_D}\mathbf{a}_k$ or translation along this axis, respectively, has to be considered in form of ${}_{k_D}\mathbf{T}_k^{\,J}$. J denotes the the joint state dependent part of the transformation. The rotation matrix ${}_{k_D}\mathbf{R}_k^J$ of revolute joints is derived using the Euler-Rodrigues formula (compare [32]).

$$
{}_{k_D}\mathbf{T}_k^{\,J} = \begin{bmatrix} {}_{k_D}\mathbf{R}_k^J & {}_{k_D}\mathbf{t}_k^J \\ \mathbf{0} & 1 \end{bmatrix}
\begin{cases}
\text{if } \textit{revolute} \text{ joint:} & \\
\quad {}_{k_D}\mathbf{R}_k^J = \cos(q_k)\,\mathbf{I} + \sin(q_k)\,{}_{k_D}\widetilde{\mathbf{a}}_k & \\
\qquad + (1-\cos(q_k))\left({}_{k_D}\mathbf{a}_k \otimes {}_{k_D}\mathbf{a}_k\right) & {}_{k_D}\mathbf{t}_k^J = \mathbf{0} \\
\text{if } \textit{prismatic} \text{ joint:} & \\
\quad {}_{k_D}\mathbf{R}_k^J = \mathbf{I}, & {}_{k_D}\mathbf{t}_k^J = {}_{k_D}\mathbf{a}_k\, q_k \\
\text{if } \textit{fixed} \text{ joint:} & \\
\quad {}_{k_D}\mathbf{R}_k^J = \mathbf{I}, & {}_{k_D}\mathbf{t}_k^J = \mathbf{0}
\end{cases}
\tag{2.1}
$$

Based on the knowledge of the end effector pose, it is possible to calculate the Jacobian matrix $\mathbf{J}_k$ of a segment. It is the multidimensional form of the derivative and describes the transformation of n-dimensional joint velocities $\dot{\mathbf{q}}$ to the resulting m-dimensional workspace velocities $\dot{\mathbf{x}}_k$ of the segment k.

$$
\mathbf{J}_k = \frac{\partial \boldsymbol{x}_k}{\partial \mathbf{q}} \quad \leftrightarrow \quad \mathbf{J}_k = \frac{\partial \boldsymbol{x}_k}{\partial t}\,\frac{\partial t}{\partial \mathbf{q}} \quad \leftrightarrow \quad \dot{\boldsymbol{x}}_k = \boldsymbol{J}_k\,\dot{\boldsymbol{q}} \tag{2.2}
$$

Using the homogeneous transformation relations of Algorithm 1 the Jacobian matrix for a body k can be constructed by computing each column i depending on the joint *type*. Hereby it must be considered, that only predecessing joints within the same tree branch of k contribute to the moveability of segment k. For better traceability the Jacobian matrix is seperated into a translational part $\boldsymbol{J}_k^P$ and a rotational part $\boldsymbol{J}_k^R$ [33].

$$
{}_0\boldsymbol{J}_k = \begin{bmatrix} {}_0\boldsymbol{J}_{k,i=0} & \cdots & {}_0\boldsymbol{J}_{k,i=n} \end{bmatrix} \qquad {}_0\boldsymbol{J}_k \in \mathbb{R}^{mxn} \tag{2.3}
$$

$$
{}_0\boldsymbol{J}_{k,i} = \begin{bmatrix} {}_0\boldsymbol{J}_{k,i}^P \\ \\ {}_0\boldsymbol{J}_{k,i}^R \end{bmatrix} = \begin{cases}
\begin{bmatrix} {}_0\boldsymbol{a}_i \times ({}_0\boldsymbol{t}_k - {}_0\boldsymbol{t}_i) \\ {}_0\boldsymbol{a}_i \end{bmatrix}, & \text{if } i < k,\ \text{branch(i)=branch(k)},\ \textit{type} \mathrel{\hat{=}} \text{revolute} \\
\begin{bmatrix} {}_0\boldsymbol{a}_i \\ \mathbf{0} \end{bmatrix}, & \text{if } i < k,\ \text{branch(i)=branch(k)},\ \textit{type} \mathrel{\hat{=}} \text{prismatic} \\
\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, & \text{else}
\end{cases}
\tag{2.4}
$$

In the presence of revolute joints which contribute to translational workspace velocities the Jacobian matrix becomes dimensionally inhomogeneous. In order to overcome this problem the popular approach of translational scaling was introduced by Ma and Angeles [17]. It scales the translational components corresponding to revolute joints of $\boldsymbol{J}_k$ by a characteristic length L_k. Thus, the i-th column of the homogeneous Jacobian matrix $\mathbf{J}_{k,h}$ is formed by

$$
{}_0\mathbf{J}_{k,h,i} = \operatorname{diag}\begin{pmatrix} \frac{1}{L_k} & \frac{1}{L_k} & \frac{1}{L_k} & 1 & 1 & 1 \end{pmatrix} {}_0\mathbf{J}_{k,i} \tag{2.5}
$$

referring to (2.3). For simplicity, the homogeneous Jacobian matrix ${}_0\mathbf{J}_{k,h}$ at the end effector is referred to as the end effector Jacobian with $\boldsymbol{J}$. For the determination of L_k some structure dependent techniques have for instance been introduced by [23] where the characteristic length of a six DoF haptic robot device is determined as the distance between the base and the top place. However, definitions like these lack generality. A different approach is used by LOU ET AL. [34]. Here, the characteristic length is chosen to be the average lever of all joint axes towards the origin of the segment frame for the given joint configuration [35]. A disadvantage of this method is that a dynamic scaling between the rotational and translational elements is generated which produces an incomparability between Jacobians of different configurations.

As a consequence, KIM AND KHOSLA [16] provide a more general representation of L_k for a serial manipulator using the overall link length of the robot which represents the maximum lever that could create translational movement through a revolute joint. This creates an upper bound of 1 of the corresponding components of $\boldsymbol{J}_k$. Applied to humanoid tree structures this means, that the length l_i of the i-th link contributes to the characteristic length for segment k, if it belongs to the same tree branch and one of the predecessing joints j is a revolute joint.

$$L_k = \sum_{i=0}^{n} l_i, \qquad \text{with } l_i = \begin{cases} l_k, & \text{if branch(i)=branch(k)} \\ & \text{and any(type}(j)_{j\in[1,\ldots,n],\ \text{branch(j)=branch(i)}} \hat{=} \text{ revolute)} \\ 0, & \text{else} \end{cases} \tag{2.6}$$

2.2.3 Inverse Kinematics

An analytic solution to the inverse mapping of the non-linear FK relations is often not available. Thus, it is necessary to apply numerical methods in order to determine feasible joint configurations $\boldsymbol{q}$ to a corresponding workspace pose $\boldsymbol{x}$.

$$IK(\boldsymbol{q}) = \mathbf{x}\,, \qquad \text{with } \boldsymbol{q} \in \mathbb{R}^n \text{ and } \boldsymbol{x} \in \mathbb{R}^m \tag{2.7}$$

For the derivation of a general method some important properties of the IK mapping have to be considered. Depending on $\boldsymbol{x}$, it can yield

1. no solutions
2. one solution
3. finite solutions
4. infinite solutions ($n > m$)

For 3. and 4. the set of solutions among a given solution is called *null space* of the IK mapping. It represents all configurations with the same end effector pose, but with internal configuration variations. In the case of 4 the robot is titled to be redundant, whereas for $n \leq m$ it is non-redundant. An important task for numerical IK methods is therefore to find one solution out of multiple possibilities.

For the static kinematic workspace analysis of this thesis, the IK is used to determine feasible, particular joint configurations for workspace poses. In this case it is not desirable to impose

any trajectories to the solver which is necessary for velocity-based IK approaches. Instead, particular joint configurations are encountered by solving IK on position-level which can be realized using the established numerical Newton-Raphson (NR) method. This method builds upon a linear approximation from an initial joint configuration $\boldsymbol{q}_0$ such that equation (2.7) is rewritten on position-level as

$$\mathbf{x}_k(\mathbf{q}_0 + \Delta\mathbf{q}) \approx \mathbf{x}_k(\mathbf{q}_0) + \underbrace{\frac{\partial \mathbf{x}_k}{\partial \mathbf{q}}}_{\mathbf{J}_k} \Delta\mathbf{q}. \tag{2.8}$$

Using this relation it is possible to find an approximate $\Delta\boldsymbol{q}$ for the workspace pose $\mathbf{x}_k$ with the inverse representation of $\boldsymbol{J}_k$. However, the two-sided inverse $\boldsymbol{J}_k^{-1}$ is not always a suitable representation, because redundant robots have rectangular end effector Jacobian matrices and the matrix can in general become ill-conditioned. This happens when the robot is in a singular configuration. If the row rank of $\boldsymbol{J}_k$ is decreased, the robot will be in a *workspace singularity* which represents the loss of moveability in one or more workspace direction(s). If the column rank of $\boldsymbol{J}_k$ is decreased, then the robot will have reached a configuration where joints align in a way such that the movement of one or multiple specific joints can be expressed by a linear combination of others. In this case the robot loses one or multiple DoF which is called a *joint space singularity*. While for non-redundant robots this also automatically implies the presence of a *workspace singularity*, for redundant robots it can also mean a degeneracy of its *null space*.

In order to overcome these limitations, the more suitable generalized *Moore-Penrose pseudoinverse* of the Jacobian matrix $\boldsymbol{J}^\dagger$ is applied. Its properties and benefits in context of IK are discussed in Section 2.2.3 below. The pseudoinverse is applied in the presented form of the NR method in Algorithm 2 based on (2.8). Within every iteration step of the method, the approximation error between the $\boldsymbol{x}_d$ and $\mathbf{x}(\mathbf{q}_i)$ is used to compute an additional change of the joint configuration $\Delta\boldsymbol{q}$ successively reducing the approximation error with every iteration step. The iteration is repeated until the maximal number of iterations n_{max} is reached or $\Delta\boldsymbol{q}$ falls below the threshold ϵ, which implies that a good approximating joint configuration was found for $\boldsymbol{x}_d$.The pseudo-algorithm of the iterative procedure is presented in the following:

Algorithm 2: Newton-Raphson Method for IK.

```
1 i = 0
2 while ||Δq|| ≥ ϵ and i < n_max do
3 |   Δq = J^† (x_d − x(q_i))
4 |   q_i = q_{i−1} + Δq
5 |   i++
6 end
7 return q_i
```

Moore-Penrose Pseudoinverse

The Moore-Penrose pseudoinverse, from now on referred to as *pseudoinverse*, is a generalization of the two-sided inverse of a matrix. The two-sided inverse exists for square matrices which are full row and full column rank [36, p.412]. The pseudoinverse generalizes this relation to non-square $m \times n$ matrices which are not full rank. In case of a full rank matrix it is equal to the two-sided inverse. If not, the pseudoinverse yields particular solutions based on the vector that is to be transformed. In context of IK on velocity-level with $\dot{\boldsymbol{q}}_{\boldsymbol{x}_d} = J^{\dagger}\dot{\boldsymbol{x}}_d$ its behavior is dependent on $\dot{\boldsymbol{x}}_d$. Let $\mathcal{R}(J)$ represent the range of all $\dot{\boldsymbol{x}}_d$ velocity direction components which can be generated in the given joint configuration. The set of degenerated direction components, in which the end effector cannot move, is denoted by $\mathcal{R}(J)^{\perp}$. Thus, the following three assumptions can be made regarding the pseudoinverse based on [37, p.122]:

- If all directional components of $\dot{\boldsymbol{x}}_d$ belong to $\mathcal{R}(J)$, the pseudoinverse will yield $\dot{\boldsymbol{q}}_{\boldsymbol{x}_d}$ which has the minimal euclidean norm of all feasible $\dot{\boldsymbol{q}}$.
- If all directional components of $\dot{\boldsymbol{x}}_d$ belong to $\mathcal{R}(J)^{\perp}$, the pseudoinverse will yield $\dot{\boldsymbol{q}}_{\boldsymbol{x}_d} = \boldsymbol{0}$.
- If $\dot{\boldsymbol{x}}_d$ has some directional components belonging to $\mathcal{R}(J)$ and some that belong to $\mathcal{R}(J)^{\perp}$, the pseudoinverse will yield $\dot{\boldsymbol{q}}_{\boldsymbol{x}_d}$ which has the minimal euclidean norm of all $\dot{\boldsymbol{q}}$ which are feasible to $\hat{\dot{\boldsymbol{x}}}_d$ where each degenerated direction is set to zero.

As a consequence of these properties the application of the *pseudoinverse* to the NR method in Algorithm 2 implies that in each iteration step $\Delta\boldsymbol{q}$ with the lowest euclidean norm is generated. To put it in more expressive terms, a human observer would state that the closest of all feasible joint configurations to the initial configuration is found. An efficient way of computing the *pseudoinverse* can be achieved with the Singular Value Decomposition (SVD) (see Appendix A).

2.3 Kinematic Metrics

The idea behind kinematic metrics is the numerical representation of the structure's kinematic dexterities as derived in the introductory Chapter 1. Similar to the presented scope of dexterities, local metrics evaluate a joint configuration or a workspace pose while global metrics in general refer to a subset of the robot's workspace [8].

In the literature several *local* scope metrics can be found that numerically represent a *local* dexterity. The Reachability Index (RI) determines whether a configuration exists to strike a given pose [7]. With the Joint Range Availability (JRA) it is possible to quantify the distance of joint positions of a given configuration from the structure's joint limits [38]. In [15] the Manipulability Measure (MM) was proposed which evaluates the ability of transforming feasible joint space velocities to arbitrary directional workspace velocities. With the CI proposed by KIM AND KHOSLA [16] the proximity to kinematic singularities can be determined. Each of the metrics are examined in the subsequent Sections 2.3.1 to 2.3.4.

For the *global* scope, several metrics such as the Global Condition Index (GCI) can be found (see [39]). However, these metrics are constructed by integrating one of the above mentioned local metrics over a subset of the workspace. Another example is given by ZACHARIAS ET AL. [7] where the *reachability index with z-orientation* is proposed which excludes the direction around the end effector's major axis in the integration process. PUGLISI ET AL. [22]

speak of the *Workspace Index* relating to the integration of reachability over the subset of all positions within the workspace invariant of the orientation. Another example can be given with the *Shape Fit Error* [40]. This metric could be converted into RI which is only evaluated for task-related subset of orientations defined by a descriptive shape. An exception to the list of composed global metrics has to be made for the Global Isotropy Index (GII) [18]. Here the CI is computed examining extreme joint configurations which does not represent an integration process. However, this metric is senseless to robots, like humanoid robots, that possess singular joint configurations within their workspace which yields a GII of zero.

Additionally, metrics have been presented in literature which integrate several metrics into one representation. In [21] the MM is augmented by a joint limit penalization comparable to the JRA which complicates the interpretability of results.

Due to the presented overlap in metric definitions, the thesis concentrates on the above mentioned basic local metrics, expressed by M, which provide a distinct kinematic meaning. Their global representation is not wrapped into an individual metric, but is based on a general integration formalism mentioned in [41]. $\mathcal{M}$ is thus calculated by the integration of M over the subset W_s of the workspace W which is normalized by the volume [41].

$$\mathcal{M} = \frac{\int_{W_s} M(\boldsymbol{x})\ dW_s}{\int_{W_s} dW_s}, \quad \text{with } W_s \subseteq W \tag{2.9}$$

The discrete formulation of the integration over the set of $n_{s,d}$ discrete poses $x_d \in W_{s,d}$ is

$$\mathcal{M} = \frac{1}{n_{s,d}} \sum_{x_d \in W_{s,d}} M(\boldsymbol{x}_d) \tag{2.10}$$

Due to the fact that robots can provide redundant configurations for a given workspace pose, VAHRENKAMP ET AL. [3] discuss the opportunities of how the metric $M(\boldsymbol{x})$ of a workspace pose can be represented by the number of metrics corresponding to the feasible joint configurations $M(\boldsymbol{q})$. Depending on the application VAHRENKAMP ET AL. [3] suggest to use extreme metrics (worst or best) or the average.

2.3.1 Reachability Index (RI)

A very important basic feature of a kinematic structure is the reachability of workspace poses. For the local scope this kinematic feature can be reduced to

$$M_{RI}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \exists\ \boldsymbol{q}\ (FK(\boldsymbol{q}) = \boldsymbol{x}) \\ 0, & \text{if } \nexists\ \boldsymbol{q}\ (FK(\boldsymbol{q}) = \boldsymbol{x}) \end{cases} \tag{2.11}$$

RI is a discrete metric which, in contrast to the subsequently presented metrics, explicitly evaluates a workspace pose.

2.3.2 Condition Index (CI)

The condition number of an $m \times n$ matrix A represents the distance to a state where it is not full column rank. This information is generated through the ratio of the highest to the smallest singular value of A. Both relate to the norm of the best major and the worst major components of the transformation A.

$$C = \frac{\sigma_{A,max}}{\sigma_{A,min}} \tag{2.12}$$

This relation can be extrapolated to the transformation J of joint space velocities from (2.2). For any robot it can be said that if C is one, then the robot is in an isotropic state where the same velocity transformation ratio is given for any arbitrary direction. If C approximates infinity the interpretation is different for non-redundant and redundant robot. For non-redundant robots an infinite C means that the robot lost at least one DoF due to a singular joint configuration which led to the ill-conditioning of J. A redundant robot lost at least more than its redundant $n-m$ DoF due to its singular state.
In order to use the condition number as a metric for fitness evaluations its reciprocal formulation is used which ensures that it is bounded, because $\sigma_{J,min} \leq \sigma_{J,max}$ and $\sigma_{J,i} \geq 0$ of any component direction i holds.

$$M_{CI}(\boldsymbol{q}) = \frac{\sigma_{J,min}}{\sigma_{J,max}}, \quad \text{with } M_{CI}(\boldsymbol{q}) \in [0,1] \tag{2.13}$$

The singular values of an $m \times n$ matrix A can be computed using the SVD.

2.3.3 Manipulability Measure (MM)

In 1985, YOSHIKAWA [15] studied the shape of the Jacobian velocity transformation in the infinitesimal neighbourhood of a joint configuration. He realized that the condition number only considers the best and worst major component of the velocity transformation. However, as stated in the introductory Chapter 1, kinematic dexterity of a robot relates to the ease of arbitrary changing the position and orientation of the end effector in the workspace [7, p.3]. YOSHIKAWA [15] calls this *manipulability*. With the MM, as presented below, he incorporates all major components of the transformation into one metric. It is derived in the following.

The set of joint space velocities

$$\Omega = \{\dot{\boldsymbol{q}} \mid \|\dot{\boldsymbol{q}}\| \leq 1\} \tag{2.14}$$

represents a hypersphere in the n-dimensional joint space. The end effector Jacobian matrix maps the hypersphere into an m-dimensional hyperellipsoid Ψ, also called *manipulability ellipsoid*, representing the corresponding workspace velocities that can be generated in the infinitesimal neighbourhood of the given joint configuration [7]. Consequently, due to (2.2),(2.14) and

$$\|\dot{\boldsymbol{q}}\| = \dot{\boldsymbol{q}}^T \dot{\boldsymbol{q}} \tag{2.15}$$

the hyperellipsoid can be written as

$$\Psi = (\dot{\boldsymbol{x}} \mid \dot{\boldsymbol{x}}^T (\boldsymbol{J}\boldsymbol{J}^T)\, \dot{\boldsymbol{x}} \leq 1) \tag{2.16}$$

It seems reasonable, that YOSHIKAWA [15] relates the *manipulability* to the volume V_e of Ψ. He therefore provided the MM [15]

$$w(\boldsymbol{q}) = \begin{cases} \sqrt{\det|\boldsymbol{J}(\boldsymbol{q})\,\boldsymbol{J}^{\mathsf{T}}(\boldsymbol{q})|}, & \text{for } n \geq m \\ 0, & \text{for } n < m \end{cases} \tag{2.17}$$

which is linear to the hyperellipsoid's volume V_e with Γ representing the Gamma function, because the singular values correspond to the length of the ellipsoid's semi axes [16].

$$V_e = w\,\frac{\pi^{\frac{m}{2}}}{\Gamma\left[\frac{m}{2}+1\right]} \tag{2.18}$$

At this point it shall be mentioned that the determinant of a squared matrix is equal to the product of its singular values. In this case the MM reduces to product of the all k-th singular values $\sigma_{J,k}$ of the end effector Jacobian matrix.

$$w(\boldsymbol{q}) = \begin{cases} \prod_{k=1}^{n} \sigma_{J,k}, & \text{for } n \geq m \\ 0, & \text{for } n < m \end{cases} \tag{2.19}$$

In conclusion, the MM considers all major components of the velocity transformation in contrast to the CI in Section 2.3.2. This proves to bring valuable dexterity information, especially for redundant robots. Another property in comparison to CI is that it is generally possible to express MM analytically as a function of $\boldsymbol{q}$ [16].
In order to provide comparability between structures with a different number of n joints, a normalized form of w is proposed by KIM AND KHOSLA [16]. It relates to the geometric mean of all major components such that a valuable form M_{MM} of w for the optimization tasks of this thesis can be written as

$$M_{MM}(\boldsymbol{q}) = \sqrt[n]{w} \tag{2.20}$$

2.3.4 Joint Range Availability (JRA)

Any robot with discontinuous joints loses the joint-specific DoF, because the joint displacements reach the lower or upper limits which cannot be exceeded. It is therefore necessary when evaluating the kinematic dexterity of a robot to consider the joint limits. A metric was proposed in [38], named JRA, which calculates the proximity of a given joint configuration to the robot's joint limits by evaluating the deviation of each i-th joint displacement q_i from its mid range position $q_{i,mid}$ [38].

$$jra(\boldsymbol{q}) = \sum_{i=1}^{m} \frac{\left(q_i - q_{i,mid}\right)^2}{q_{i,max}} \tag{2.21}$$

with m being the order of the joint space, q_i representing the joint displacement, $q_{i,mid}$ and $q_{i,max}$ denoting the middle and maximal displacement limited through joint limits. This formulation of the JRA presents several problems. First, it is unbounded and increases with the number of joints of the robot. Second, the displacement of the joint from its middle position is squared to eliminate the sign. This imposes a dimension to jra. In order to overcome these issues, a slightly different definition of the JRA is derived.

A helpful inspiration can be taken from [42] and [21, p.7ff] where the robot's loss of DoF due to joint limits is considered through an augmentation of the MM with a corresponding penalization term. This strategy was already critized in Section 2.3 where a distinct meaning of kinematic metrics is desirable. Supportively, ZACHARIAS ET AL. [7, p.4] question the interpretability of this augmented form of the MM. The penalization term is an orientation for the construction of M_{JRA} in Section 3.3.1.

2.3.5 Application to Coupled End Effectors

For the execution of bimanual tasks humanoid robots use both end effectors in a coupled state e.g. in order to manipulate objects. It is therefore of great interest for the kinematic structure optimization of humanoids to ensure that the bimanual workspace relating to this coupling state has the required dexterity. To the author's knowledge, only little effort has yet been done in literature using kinematic metrics in relation to dual-arm tasks. BAGHERI ET AL. [43] state that it is possible to compute metrics for single systems and derive a combined ability based on the intersection of kinematic properties. For metrics like RI, JRA and CI this means, that the minimal metric value of each end effector e is representative for the dual-arm system.

$$\left.\begin{array}{rcl} M_{RI,coupled} & = & min(M_{RI,e}) \\ M_{CI,coupled} & = & min(M_{CI,e}) \\ M_{JRA,coupled} & = & min(M_{JRA,e}) \end{array}\right\}, \qquad \text{with } e \in \{1,..,n_{eef}\} \tag{2.22}$$

The theory of minimal intersection was also applied to the MM by BAGHERI ET AL. [43] for dual-arm systems in two dimensions. The hereby announced robot-specific Dual Arm Manipulability Index (DMI) is the lower bound of the intersected manipulability measure of both end effectors. This relates to the worst case, where both manipulability ellipsoids are perpendicular to each other forming a circular area. Thus, DMI is written as

$$M_{DMI} \quad = \quad \pi \; min(\sigma_{J_e})^2 \;, \qquad \forall e \in \{1,2\} \tag{2.23}$$

This concept can be exploited for higher dimensional spaces which is further examined in Section 3.3.3. A more general approach to calculate the intersection of two manipulability ellipsoids has been accomplished by LEE [44]. It is, however, approximate and requires extra computational effort. It is therefore not further taken into consideration in this thesis.

2.4 Workspace Discretization

As mentioned in the introductory Chapter 1 a numerical workspace analysis requires a partition into discrete poses that can be evaluated. The pose of a robot's end effector is composed of a position and an orientation. Following the rules of group theory, the workspace of a robot in Special Euclidean Group (SE(3)) can be separated into a subset of positions in $\mathbb{R}^3$ and orientations in the Special Orthogonal Group (SO(3)). Due to different geometrical rules discretization schemes for positions and orientations have to be accomplished separately and are interpreted in a hierarchical manner where every position is partitioned concerning SO(3).

For the partitioning of the position space of the end effector, $\mathbb{R}^3$ can be divided into equally-divided voxels. The center of each voxel resembles the discrete position. A robot-related

form of the voxelisation is the Bounding Box (BB) scheme e.g. applied in [7]. It sets the boundaries of the voxelisation with respect to the robot's overall size.

While the partitioning of positions is fairly straightforward without any considerable drawbacks, the discretization of the orientations can be achieved in various ways. A group of schemes relates to the sampling of representative orientation vectors, such as Euler angles or unit quaternions. Quaternions can be sampled by recursively subdividing the surface of a tesseract of all unit quaternions [45]. A different approach is the *Equivolumetric Partition*. It builds on the fact that all of the resulting discrete vectors of the above mentioned sampling methods do not relate to the same volume in the SO(3). In order to avoid numerical integration errors, the volume of each element would have to be considered when integrating. YANG AND CHEN [46] therefore propose an equi-volumetric partition of SO(3). It first describes the orientations in SO(3) as a vector $\omega \in \mathbb{R}^3$. This can be viewed as Cartesian coordinates over the interior space of a solid sphere of radius π [13]. The *Equivolumetric Partition* is therefore realized by dividing the sphere into equi-volumetric shell elements [13]. The same error disposition is also valid for the Voxel-Filling Sphere (VFS) scheme which is presented in [7]. It represents a different sampling method. Applied to the context of robotics it separates the discretization of orientations into rotations around the end effector's major axis and the resulting rest. This approach provides a huge advantage over the previously presented schemes. The parent joint of the end effector of many robots provides a revolute DoF around the end effector's major axis. In this case the dexterity analysis of orientations around the major axis yields equal results and can therefore be omitted for some kinematic structure optimization tasks which saves essential computational effort.

As a conclusion for this reason the VFS scheme is applied in this thesis. However, due to the proneness regarding integration errors, a well-balanced partitioning of both separate discretizations must be realized. If rotations around the end effector's major axis are not discretized into one element, the scheme still provides a good distribution of the rest of orientations yielding a low integration error [7]. In the following sections 2.4.1 and 2.4.2 the BB and VFS schemes are further examined and the forward, as well as the inverse mappings of positions and orientations are presented. The following notations are used for the mappings

$$PD_f : \boldsymbol{p}_d \rightarrow {}_0\boldsymbol{t}_d \qquad OD_f : \boldsymbol{o}_d \rightarrow {}_0\boldsymbol{R}_d \tag{2.24}$$

$$PD_i : {}_0\boldsymbol{t}_c \rightarrow \boldsymbol{p}_d \qquad OD_i : {}_0\boldsymbol{R}_c \rightarrow \boldsymbol{o}_d \tag{2.25}$$

where $\boldsymbol{p}_d$ and $\boldsymbol{o}_d$ denote to the vector of indices which map to the discrete position and orientation ${}_0\boldsymbol{t}_d$ and ${}_0\boldsymbol{R}_d$. The inverse mapping describes the back transformation from an arbitrary position ${}_0\boldsymbol{t}_c$ and orientation ${}_0\boldsymbol{R}_c$ to the corresponding index vector.

Note that ZACHARIAS ET AL. [7, p.18f] investigated the discretization error of the combined application of BB and VFS. The computed discrete workspace of industrial manipulators was tested with 10^5 randomly sampled joint configurations. The BB was discretized with 50 mm, the VFS with $n_k = 200$, $n_i = 12$ (see Section 2.4.1 and 2.4.2 for further explanations). Over 93% of the kinematic mappings were accurate for every robot. The error rate seems acceptable and further consideration during the course of the thesis is disregarded.

2.4.1 $\mathbb{R}^3$: Bounding Box Discretization

The center of the bounding box is the robot's root link which represents the rigid connection to the world. The box's length is defined by twice the sum over all n_k link lengths

$$l_{ws} = 2\sum_{i=1}^{n_k} l_k \tag{2.26}$$

The box is then voxelized into n_c voxels by an arbitrary discretization length l_c.

$$n_c = \frac{l_{ws}}{l_c} \tag{2.27}$$

Finally a mapping from the discrete index vector

$$\boldsymbol{p}_d = \begin{pmatrix} p_{d,x} \\ p_{d,y} \\ p_{d,z} \end{pmatrix} \tag{2.28}$$

to the coordinates of the center of every voxel ${}_0\mathbf{t}_d$ in the workspace is created [7, p.7ff].

$$PD_f : \mathbb{N}^3 \to \mathbb{R}^3; \quad \boldsymbol{p}_d \mapsto {}_0\mathbf{t}_d \qquad PD_f(\boldsymbol{p}_d) = \left(\boldsymbol{p}_d + (1 - \tfrac{n_c}{2}) \cdot \mathbf{1}\right) l_c - \tfrac{l_c}{2} \cdot \mathbf{1} \tag{2.29}$$

The inverse mapping of an arbitrary position vector ${}_0\mathbf{t}_c$ to the corresponding index vector is expressed through

$$PD_i : \mathbb{R}^3 \to \mathbb{N}^3; \; {}_0\mathbf{t}_c \mapsto \boldsymbol{p}_d \qquad PD_i({}_0\boldsymbol{t}_c) = \tfrac{1}{l_c}\, {}_0\boldsymbol{t}_c + \left(\tfrac{n_c}{2} - 1\right) \cdot \mathbf{1} \tag{2.30}$$

The overall link length as a measure of the extreme bounds of the robot's workspace does not hold for robots with prismatic joints which has to be considered for the applications of this thesis.

2.4.2 SO(3): Voxel-Filling Sphere Discretization

The following explanations built on the version of VFS used by ZACHARIAS ET AL. [7]. Slight modifications regarding the coordinate system conventions were imposed, which are explained within the following course of the section. A sphere with unit diameter is positioned congruently to the voxel's center. Next, the spherical surface is discretized in n_k uniformly distributed points p_i. In order to find a discrete set of well-distributed points on a sphere's surface the spiral algorithm after [47] is chosen, recommended by ZACHARIAS ET AL. [7, p.8]. A spiral is created along the sphere's surface which starts at the pole $\theta = \pi$ and ends at $\theta = 0$. On the spiral the n_k equidistant points are generated creating the mapping from the k-th point to the spherical coordinates $[\phi, \theta]$. The z-coordinate $h_k \in [-1, 1]$ of the k-th point is required to compute the mapping, accordingly to [47].

$$h_k = -1 + \tfrac{2(k-1)}{n_k - 1}, \qquad 1 \le k \le n_k \tag{2.31}$$

$$\theta_k = \operatorname{acos}(h_k) \tag{2.32}$$

$$\varphi_k = \left(\varphi_{k-1} + \tfrac{3.6}{\sqrt{n_k}} \tfrac{1}{\sqrt{1-h_k^2}}\right) \mod 2\pi, \qquad 2 \le k \le n_k - 1, \varphi_1 = \varphi_{n_k} = 0 \tag{2.33}$$

The results are then processed to compute the rotation matrices ${}_o\boldsymbol{R}_k$ which rotate a frame at the center of the voxel with the orientation of the inertial frame, such that the z-axis points to the point with index k on the sphere's surface. At this point ZACHARIAS ET AL. [7, p.8] chose to let the z-axis point in the opposite direction, such that an end effector with the z-axis as its major axis "pierces" through the sphere's surface towards its center. In this chapter this convention is disregarded in order to provide more generality to this scheme. Later in this thesis a different convention is chosen which is explained in Section 3.2. ${}_o\boldsymbol{R}_k$ can be calculated by rotating the identity frame at the voxel's center first around the y-axis by φ_k and then around the new x-axis by θ_k. This process is visualized in Figure 2.2. The grey cube represents the voxel of the BB. The green sphere with the black spiral at its surface represents the VFS. Each blue circle on the spiral corresponds to the k-th distributed point, generated on the sphere's surface. The black frame corresponds to the inertial frame while the red frame is the rotated k^{th} frame.

$$ {}_0\boldsymbol{R}_k = \boldsymbol{R}_{x'}(\theta_k)\boldsymbol{R}_y(\varphi_k) \tag{2.34} $$

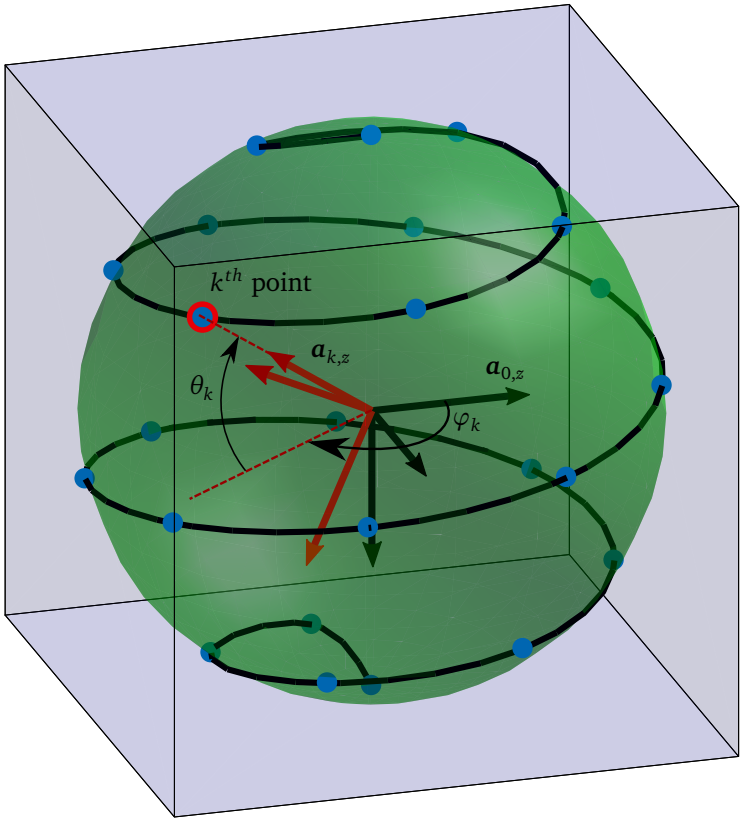

Figure 2.2: Computation of ${}_0\boldsymbol{R}_k$ through the subsequent rotation around the y-axis with φ_k and new $x^{'}$-axis with θ_k of the VFS discretization scheme.

Finally, the orientation around the z-axis of ${}_o\boldsymbol{R}_k$ is considered. The angle α_i is created based on the number n_i of discrete steps of α_i.

$$ \alpha_i = i\,\frac{2\pi}{n_i}, \qquad i = 0 \ldots n_i \tag{2.35} $$

The final mapping from the discrete orientation index vector

$$\boldsymbol{o}_d = \begin{pmatrix} k \\ i \end{pmatrix} \tag{2.36}$$

therefore yields

$$OD_f : \mathbb{N}^2 \to SO(3); \boldsymbol{o}_d \;\mapsto\; {}_0\boldsymbol{R}_d \qquad OD_f(\boldsymbol{o}_d) = \boldsymbol{R}_{z''}(\alpha_i)\ {}_0\boldsymbol{R}_k \tag{2.37}$$

For the inverse mapping OD_i, k must be determined first. In [7] the desired k is the one where the rotation matrix ${}_0\boldsymbol{R}_k$ has the lowest angle between its z-axis $\boldsymbol{a}_{k,z}$ and the z-axis of the arbitrary frame $\boldsymbol{a}_{c,z}$. This relation is visualized in Figure 2.3. The minimal angle of both axis is described by β_{min}.

$$\beta(\boldsymbol{a}_{k,z}, \boldsymbol{a}_{c,z}) = \operatorname{acos}\left(\frac{\boldsymbol{a}_{k,z}^T}{\|\boldsymbol{a}_{k,z}\|} \cdot \frac{\boldsymbol{a}_{c,z}}{\|\boldsymbol{a}_{c,z}\|} \right) \tag{2.38}$$

$$k = \underset{k \in [1,\ldots,n_k]}{\arg\min} \left(\beta(\boldsymbol{a}_{k,z}, \boldsymbol{a}_{c,z}) \right) \tag{2.39}$$

The second discrete orientation index i is derived from α_i. In this case the minimum angle in three dimensions between the x-axis of the k-th frame $\boldsymbol{a}_{k,x}$ and the arbitrary frame's x-axis $\boldsymbol{a}_{c,x}$ cannot be exploited in a direct manner, because the z-axes of the corresponding frames have different orientations. A solution to this is visualized in Figure 2.3. $\boldsymbol{a}_{c,x}$ is projected onto the x_k, y_k-plane of ${}_0\boldsymbol{R}_k$. In Figure 2.3 the xy-plane is visualized in form of the red surface. The projection of $\boldsymbol{a}_{c,x}$ is visualized by $\hat{\boldsymbol{a}}_{c,x}$ in green. This version can then be used to compute α_i in the x_k, y_k-plane

$$\hat{\boldsymbol{a}}_{c,x} = {}_0\boldsymbol{R}_k \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} {}_0\boldsymbol{R}_k^{-1}\ \boldsymbol{a}_{c,x}^T \tag{2.40}$$

Consequently, i is derived by computing the angle in the xy-plane via

$$\alpha(\boldsymbol{a}_{i,x}, \hat{\boldsymbol{a}}_{c,x}) = \left\| \left(\operatorname{atan2}\left(a_{i,x}^y, a_{i,x}^x\right) - \operatorname{atan2}\left(\hat{a}_{c,x}^y, \hat{a}_{c,x}^x\right) \right) \right\| \tag{2.41}$$

$$i = \underset{i \in [1,\ldots,n_i]}{\arg\min} \left(\alpha(\boldsymbol{a}_{i,x}, \hat{\boldsymbol{a}}_{c,x}) \right) \tag{2.42}$$

Finally the inverse mapping of OD writes

$$OD_i : SO(3) \to \mathbb{N}^2; {}_0\boldsymbol{R}_c \;\mapsto\; \boldsymbol{o}_d \qquad OD_i({}_0\boldsymbol{R}_c) = \begin{pmatrix} k \\ i \end{pmatrix} \tag{2.43}$$

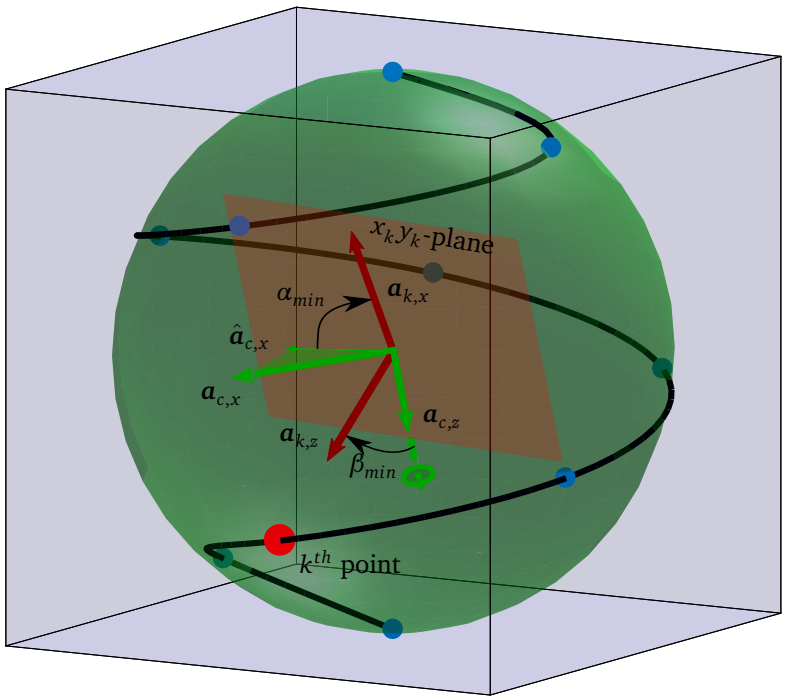

Figure 2.3: Inverse discretization mapping of the VFS scheme. First, β_{min} is determined to derive k, then i is computed by calculating α_{min} using the projected axis $\hat{a}_{c,x}$.

2.5 Numerical Optimization

The shape of the fitness function and the feasible parameter space of an optimization problem defines constraints to the set of solvers that are suitable to find the optimum. The following properties regarding the parameter space and the shape of the fitness function for the kinematic structure optimization in this thesis are assumed.

First, the feasible n-dimensional parameter space $\mathcal{F}$ is expected to be continuous in $\mathbb{R}^n$ and bounded between $\mathbf{b}_l$ and $\mathbf{b}_u$. This simplification is derived within the concept in Chapter 3. Another assumption is being made by classifying the optimization problem as single-objective, because the structural fitness is represented by one numeric value within the concept of this thesis (see Chapter 3).

$$\mathcal{F} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{b}_l \preceq \mathbf{x} \preceq \mathbf{b}_u\} \tag{2.44}$$

Second, the shape of the objective function f for all x in $\mathcal{F}$ is assumed to be non-linear and non convex. Further, $\mathcal{F}$ cannot be derived analytically and there is no gradient information available. These properties are given through the non-linear relations of FK, the inhomogenous influence of the proposed kinematic metrics and constraining joint limits (compare Chapter 3). Due to the non-convexity, local optimization techniques or gradient-based optimization solvers are not suitable, because in general they can only encounter local optima by architecture, dependent on where the methods are started from. SHAFIQUE [48] divides non-convex global optimizers into *Complete Methods* and *Approximate Methods*.

Complete Methods represent the approach of an exhaustive search in the parameter space to discover optimal regions. Solvers such as the *Branch and Bound* method divide the parameter

space into sub regions to each of which a local optimizer is applied. While this method ensures the discovery of a global optimum it requires a high amount of fitness evaluations which is computationally expensive for high dimensional parameter spaces and the elaborate fitness evaluation necessary for the kinematic structure evaluation [48].

Approximate Methods, on the other hand, use heuristics on a subset of points to efficiently estimate the probable location of the global optimum. The heuristics include techniques to transition out of local minima, but do not guarantee finding the global optimum [49, p.263]. However, *good enough* solutions can be detected which is sufficient for many applications [50, p.4]. Many meta-heuristic methods have been proposed and applied to a variety of problem domains, such as machining operations in manufacturing plants, optimal synthesis of water systems or the optimal design in chemical processes (see [48]). However, WEYLAND [51] criticizes that many methods, like the *Harmony Search* or the *Firefly Algorithm* wrap similar heuristics into different contexts disguising that only small adjustments to other existing methods have been made. To avoid this issue, solvers from differing solver families are chosen for the application in this thesis. The population-based algorithm Particle Swarm Optimization (PSO) algorithm, the evolutionary algorithm CMA-ES and SA as representative meta-heuristics inspired by a nonlinear physics process. They are introduced in the subsequent subsections 2.5.1 to 2.5.3.

2.5.1 Particle Swarm Optimization

KENNEDY AND EBERHART [52] proposed the population-based PSO algorithm in 1995 inspired by the social behavior of animal swarms, like birds or fish searching for food. An important property of swarms is that individuals move based on their own experience and their neighbour's actions. This allows the transport of a form of swarm knowledge via the neighbourhood relations maintaining a degree of freedom regarding the swarm's shape [53, p.16].

This behaviour is adopted in PSO. Initially, a set of randomly distributed particles form the swarm population P. Consequently, each particle moves to its new position x_{i+1} in the parameter space in each iteration. The position update results from summation of three attraction causes and is demonstrated in (2.45):

1. last movement $\mathbf{v}_i$ is considered as representation of the particle's inertia.
2. best position $\mathbf{x}_i^l$ in the particle's history.
3. best position $\mathbf{x}_i^g$ in the history of all neighbour particles.

$$\begin{aligned}\mathbf{v}_{i+1} &= \omega \left(\underbrace{\mathbf{v}_i}_{1.} + \underbrace{\eta_1 \mathbf{r}_1 \left(\mathbf{x}_i^l - \mathbf{x}_i\right)}_{2.} + \underbrace{\eta_2 \mathbf{r}_2 \left(\mathbf{x}_i^g - \mathbf{x}_i\right)}_{3.} \right), \text{ with } i \in \{1, ..., n_{iter}\} \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + \mathbf{v}_{i+1}\end{aligned} \tag{2.45}$$

Since its initial proposal several implementations for the position update have been suggested according to POLI ET AL. [54]. They differ in the determination of the particle's neighbourhood and the incorporation of parameters for convergence and movement control. A popular choice is the *lbest* neighbour topology and the *constriction coefficients* parametrization

scheme. The scheme is reflected by (2.45) where ω and the attraction-corresponding η_1, η_2 denote the control parameters and $\mathbf{r}_1, \mathbf{r}_2$ represent uniformly-distributed random weight values for each parameter space dimension d with $r_{d,1}, r_{d,2} \in [0,1]$. The neighbour topology *lbest* defines the n_{neighb} most adjacent population individuals as the particle's neighbourhood. If n_{neighb} equals the size of each distinct particle in the population the topology is called *gbest*. In comparison to *gbest*, *lbest* allows the forming of sub-swarms representing a "neighbourhood" that can behave isolated and even converge distant from other subswarms [54]. The parameter space is thus explored more intensively. *gbest*, however stands for faster convergence to one optimum. A promising approach is to implement a dynamic increase of n_{neighb} incorporating the advantages of both topologies with respect to iteration stage.

The following pseudo-code in Algorithm 3 introduces the overall procedure of PSO. In addition to the above described position update, the excess of dimension-corresponding velocity limits $\mathbf{v}_{max}$ and parameter space lower and upper box-bounds $\mathbf{b}_l, \mathbf{b}_u$ are considered. A good parametrization of η_1, η_2 and ω can be found in [54, p.37].

Algorithm 3: Particle Swarm Optimization

Input: P, n_{iter}, $\mathbf{b}_l$, $\mathbf{b}_u$, η_1, η_2, ω

```
scale v_max to dimension size
while i < n_iter do
    foreach individual in P do
        get neighbour mapping p_n(n_neighb)
        randomize r_1, r_2
        compute v_{i+1} (eq. (2.45))
        if norm(v_d) of v_d in v_{i+1} exceeds v_max then
            v_d = ± v_max
        end
        compute x_{i+1}
        if x_d in x_{i+1} exceeds b_l resp. b_u then
            x_d = b_l resp. b_u
        end
        update x_i^l
        update x_i^g
        update n_neighb
    end
    i++
end
return x_{i+1}
```

2.5.2 Simulated Annealing

The process of *annealing* describes the reformation of molecules or atoms (particles) of materials towards an energy-minimal crystalline state. This is achieved by the succession of heating and controlled cooling. Heating imparts energy to the system to promote the loosening of ion bindings within the crystalline structure. If the temperature is reduced in consequence of cooling, the moveability of atoms to each other is increasingly restricted and crystalline structures are reformed. However, fast cooling rates lead to poly-crystalline states which have a higher energy state than the crystalline state. This nonlinear physical coherence represents an optimization task in foundry engineering, where the temperature descent is controlled in order to generate low-distorted material structures [53], [50]. For slow cooling rates it can be assumed that the system is in a state of thermodynamic equilibrium. The probability p_i that a given particle has an energy state ϵ_i is then given by the Boltzmann distribution

$$P(\epsilon) = e^{-\frac{\epsilon}{k_B\, T}} \tag{2.46}$$

where T denotes the system's temperature and k_B is the Boltzmann's constant. It becomes obvious, that the probability of the system being at high energy state decreases with lower temperatures and converges [50].

In [55] the generalization of this annealing process and optimization task to numerical optimization problems was proposed in form of the SA method. Its core feature relies in the probability-based acceptance of worse parameter vectors based on the Metropolis Criterion (MC) of the *Metropolis Algorithm* [49]. As a result, SA is capable of avoiding local optima for high temperatures analog to the prevention of amorphous material states [50]. The procedure is presented in Algorithm 4. Each iteration step i represents the i'th cooldown step of the temperature $T \in \left[T_{init}, T_{fin}\right]$ in the annealing schedule. It can be modeled linearly like in [49] or exponentially by

$$T_i = T_{i-1}\, \frac{T_{fin}}{T_{init}}^{\frac{1}{n_T}} \tag{2.47}$$

which is later used in the implementation phase. A uniformly distributed parameter vector x_i is generated within the step size ν to the previous vector.

$$\boldsymbol{x}_i = rand(\boldsymbol{x}_{i-1} - \nu\, \boldsymbol{I}, \boldsymbol{x}_{i-1} + \nu\, \boldsymbol{I}) \tag{2.48}$$

In [49] a slight adjustment is proposed which alters only one parameter dimension at a time for every exploration step at a constant temperature. The new parameter vector is then tested against the MC and updated correspondingly [49, p.264]. A false positive acceptance of a worse parameter vector is produced, if the uniformly randomized parameter r is below the current Boltzmann probability.

$$MC : \boldsymbol{x}_i = \begin{cases} \boldsymbol{x}_i, & \text{if } f(\boldsymbol{x}_i) \leq f(\boldsymbol{x}_{i-1}) \\ \boldsymbol{x}_i, & \text{if } f(\boldsymbol{x}_i) > f(\boldsymbol{x}_{i-1}) \text{ and } r \leq e^{-\frac{\epsilon_i}{k_B\, T}} \\ \boldsymbol{x}_{i-1}, & \text{if } f(\boldsymbol{x}_i) > f(\boldsymbol{x}_{i-1}) \text{ and } r > e^{-\frac{\epsilon_i}{k_B\, T}} \end{cases} \tag{2.49}$$

The analogy of the atomic energy-state probability is wrapped into this update step. It is assumed that the probability of the next parameter vector being at $\boldsymbol{x}_i$ is Boltzmann distributed

(see (2.46)) and depends on the difference of the function evaluation $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_{i-1})$. In order to control the exploration space for each temperature step, CORANA ET AL. [49] introduced a step size control. It aims to maintain a 1:1 rate between accepted and rejected parameter vectors. v is adapted according to the number of acceptances n_u. It is assumed that the step size should increase if the acceptance rate is high, because the fitness gain is small compared to the temperature. This implies that the algorithm is evolving too slowly. On the other hand a decrease of the step size is initiated for low acceptance rates, because the computational effort is high without providing progress through accepted parameter vectors for the optimization problem. The derivation of (2.50) can be reviewed in CORANA ET AL. [49, p.268].

$$v = \begin{cases} v_{i-1}\left(1 + c_u \frac{\frac{n_u}{n_{alt}} - 0.6}{0.4}\right) & \text{if } n_u > 0.6\ n_{alt} \\ v_{i-1}\left(1 + c_u \frac{0.4 - \frac{n_u}{n_{alt}}}{0.4}\right)^{-1} & \text{if } n_u \leq 0.4\ n_{alt} \\ v_{i-1} & \text{else} \end{cases} \tag{2.50}$$

Algorithm 4: Simulated Annealing

Input: n_{step}, n_{alt}, T_{init}, T_{fin}, v_{init}

```
i ← 0
T ← T_init
v ← v_init
while T < T_fin do
    // Cooldown
    forall n_step search range adaptions do
        forall n_alt explorations do
            compute x_i (eq. (2.48))
            compute e^(-ε_i / (k_B T))
            compute r
            update x_i ← MC (eq. (2.49))
        end
        update v (eq. (2.50))
    end
    update T (eq. (2.47))
    i++
end
return x_i
```

2.5.3 Covariance Matrix Adaption - Evolution Strategy

Evolutionary algorithms, also called Evolution Strategy (ES), are meta-heuristic methods, that evolve an initial population by creation of new generations using genetic operations. A variety of variations of ES exist, that differ in the exact procedure and choice of the genetic operations. CMA-ES belongs to the $(\mu/\mu_I, \lambda)$-ES and realizes the genetic operations by creating new individuals based on a search space distribution $\mathcal{N}$ which is adaptively adjusted regarding its mean and shape. This process is visualized in the minimization of a two-dimensional sphere function using CMA-ES in Figure 2.4. The iso-contour of the 95% confidence interval is visualized for several generations such that the evolution of the search distribution can be seen.

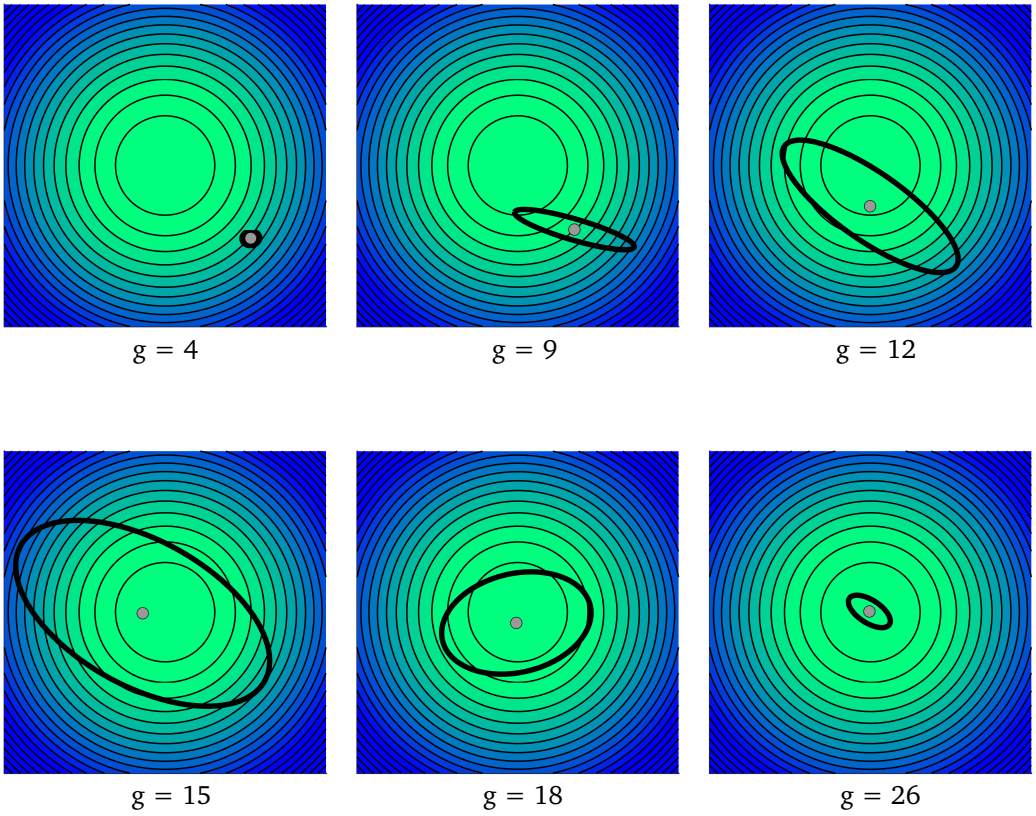

Figure 2.4: CMA-ES on two-dimensional sphere function ($g_{max} = 400, \sigma^{(0)} = 0.001, \boldsymbol{m}^{(0)} = rand([-1,1])$). Visualization of 95% confidence iso-contour at (1) $g = 4$, (2) $g = 9$, (3) $g = 12$, (4) $g = 15$, (5) $g = 18$, (6) $g = 26$.

In the procedure of $(\mu/\mu_I, \lambda)$-ES algorithms, a population P^g consists of μ individuals that are parents to λ offspring individuals of the next generation. The choice of the recreating parents is a genetic operation called *selection*, subsequentially interpreted as the operator s. The creation of an offspring individual from its parent is achieved through the application of the genetic operations *mutation* m and *recombination* r [56].

$$\begin{aligned} P_{\mu}^{(g)} &\leftarrow s(P^{(g)}) \\ P^{(g+1)} &\leftarrow m\left(r\left(P_{\mu}^{(g)}\right)\right) \end{aligned} \tag{2.51}$$

The **selection** operator sorts the offspring individuals in descending order regarding their fitness of the objective function f. Any future reference to $\boldsymbol{x}_i^g$ relates to the sorted vector

with the best vector at $i = 0$. The best μ individuals represent the parents for the recreation of the next generation.

With the genetic **recombination** operator, the mean $\boldsymbol{m}^{(g)}$ of $\mathcal{N}$ is determined. In CMA-ES the mean is the weighted average of the μ parent individual's position in the parameter space.

$$\mathbf{m}^{(g)} = \sum_{i=1}^{\mu} \omega_i \, \mathbf{x}_i^{(g)} \tag{2.52}$$

The **mutation** operation in CMA-ES creates an offspring individual $\boldsymbol{x}^{(g+1)}$ through sampling points of a multi-variate normal distribution $\mathcal{N}$.

$$\boldsymbol{x}_i^{g+1} \sim \mathcal{N}\left(\mathbf{m}^{(g)}, \left(\sigma^{(g)}\right)^2 \mathbf{C}^{(g)}\right), \qquad \forall i \epsilon \{1, ..., \lambda\} \tag{2.53}$$

This operation relates to "mutation", because $\mathcal{N}$ is generated regarding the parent's positions and their covariance.

The covariance matrix of $\mathcal{N}$ represents the strategy control of CMA-ES, because it determines the shape of $\mathcal{N}$ [57]. It can be computed via estimation or adaption. While a good estimate requires a high set of sampling points, adapting procedures allow the determination of $\mathbf{C}$ with a low number of individuals, because the covariance matrix of the current generation is adapted from the covariances of past generations [58]. This property is an essential benefit of the CMA-ES algorithm and allows a reduction of the computationally expensive workspace evaluations for the kinematic structure optimization in this thesis. The covariance matrix adaption consists of the following two step updating process. The steps are weighted by two factors. First, the exponential learning factor c_{cov} was proposed which determines the impact of the update and favors the influence of the covariances of chronologically close generations [58]. Second, μ_{cov} reciprocally weights the two adaption steps against each other.

$$\mathbf{C}^{(g)} = (1 - c_{cov}) \, \mathbf{C}^{(g-1)} + \underbrace{\frac{c_{cov}}{\mu_{cov}} \mathbf{p}_c^{(g)} \, \mathbf{p}_c^{(g)^T}}_{\text{Cumulation}} + c_{cov} \left(1 - \frac{1}{\mu_{cov}}\right) \underbrace{\frac{1}{\sigma^{(g)^2}} \mathbf{C}_\mu^{(g)}}_{\text{Rank-}\mu\text{-Update}} \tag{2.54}$$

The first update step is the *Rank-μ-Update* which adds the weighted mean of parent's covariances to update process. The distribution, thus, stretches into the better regions detected by the parents. The stretching can be seen in Figure 2.4 by e.g. comparing the iso-contour of the 9^{th} generation to the 12^{th} generation. In order to guarantee a comparability between the adapted $C^{(g+1)}$ and the parents' $C_\mu^{(g+1)}$, latter is divided by the step size parameter $\sigma^{(g)}$ [58, p.83].

$$\frac{1}{\sigma^{(i)^2}} \mathbf{C}_\mu^{(g)} = \sum_{i=1}^{\mu} \omega_i \left(\frac{\mathbf{x}_{i:\lambda}^{(g)} - \mathbf{m}^{(g-1)}}{\sigma^{(g-1)}}\right) \left(\frac{\mathbf{x}_{i:\lambda}^{(g)} - \mathbf{m}^{(g-1)}}{\sigma^{(g-1)}}\right)^T \tag{2.55}$$

With the *Cumulation* step the evolutional history of $\boldsymbol{m}^{(g)}$, called "evolution path" $\boldsymbol{p}_c^{(g+1)}$ by HANSEN [58], is incorporated into the CMA-ES. This adaption strategy can be roughly compared to the particle's inertia of the PSO, because the covariance of $\mathcal{N}$ is increased in the direction the mean has moved to, which prevents that a local optimum encountered by the parents of one generation leads to a drastic change in the shape of the search distribution due

to the *Rank-μ-Update*, hereby possibly avoiding the local optimum. This adaption step can be seen in Figure 2.4 comparing the iso-contours between the 12^{th} and 15^{th} generation. The distribution is stretched in the direction of the evolution path although the *Rank-μ-Update* pulls the covariances in direction of the global optimum. The path of the $(g+1)$ generation is derived by

$$\mathbf{p}_c^{(g)} = (1-c_c)\ \mathbf{p}_c^{(g-1)} + \sqrt{c_c\,(2-c_c)\,\mu_{eff}}\ \frac{\mathbf{m}^{(g)} - \mathbf{m}^{(g-1)}}{\sigma^{(g-1)}} \tag{2.56}$$

where c_c exponentially weights the history of the evolution path. While $\sqrt{c_c\,(2-c_c)\,\mu_{eff}}$ is incorporated for normalization purposes (see [58, p.87]), $\sigma^{(g)}$ represents a parameter for step size control.

Both update processes depend on the step size $\sigma^{(g)}$. HANSEN [58] proposed a *step size control* based on the length of the particular evolution path. It can be stated that if the evolution path is long and the steps point into the same direction, $\mathcal{N}$ can be updated more intensively, because the distribution is moving to areas with a higher fitness (compare Figure 2.4 between the 4^{th}, 9^{th} and 12^{th} generations). The opposite is present if the path is short and the steps annihilate each other regarding the direction. In the CMA-ES examples (see Figure 2.4), annihilation can be seen between the generations 15, 18 and 26. In this case it is desired to decrease the step size, because a local optimum is present. This classification of the evolution path is realized by comparing the conjugate evolution path [58, p. 86f]

$$\mathbf{p}_\sigma^{(g)} = (1-c_\sigma)\ \mathbf{p}_\sigma^{(g-1)} + \sqrt{c_\sigma\,(2-c_\sigma)\,\mu_{eff}}\ \left(\mathbf{C}^{(g-1)}\right)^{-\frac{1}{2}}\ \frac{\mathbf{m}^{(g)} - \mathbf{m}^{(g-1)}}{\sigma^{(g-1)}} \tag{2.57}$$

to the expected euclidean norm of a $\mathcal{N}$ distributed random vector $E\,\|\mathcal{N}(0,\mathbf{I})\|$ which yields the step size with a damping parameter d_σ and the control rate c_σ which influences the step size adaption analogously to c_{cov} [58, p. 88]

$$\sigma^{(g)} = \sigma^{(g-1)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma^{(g)}\|}{E\,\|\mathcal{N}(0,\mathbf{I})\|} - 1\right)\right). \tag{2.58}$$

For the application of CMA-ES a good parametrization of the presented constants can be found in [58, p.100] which is later used for the application of CMA-ES in this thesis. The number of individuals and generations of the evolution process, however, must be chosen carefully. While a higher number of individuals allow a denser coverage of the search space, it is of vital importance to define a sufficient number of generations so that the presented adaption strategies can reliably determine the covariance matrix. The overall procedure of CMA-ES is presented in the following pseudo-code Algorithm 5 inspired from [59] and [60] referring to the equations (2.51) to (2.58).

Algorithm 5: CMA-ES

Input: $\boldsymbol{m}^{(0)}$, $\sigma^{(0)}$, g_{max}, n

// Initialization of parameters (see [58, p.100])

$\lambda = 4 + \lfloor 3 \ln n \rfloor$

$\mu = \lfloor \frac{\lambda}{2} \rfloor$

$w_i = \frac{\ln(\mu+1)-\ln(i)}{\sum_{j=1}^{\mu}(\ln(\mu+1)-\ln(j))}, \qquad \text{for } i = 1, ..., \mu$

$\mu_{eef} = \frac{1}{\sum_{i=1}^{\mu} w_i^2}$

$c_\sigma = \frac{\mu_{eff}+2}{n+\mu_{eef}+3}$

$d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu_{eff}-1}{n+1}} - 1\right) + c_\sigma$

$c_c = \frac{4}{n+4}$

$\mu_{cov} = \mu_{eff}$

$c_{cov} = \frac{1}{\mu_{cov}} \frac{2}{(n+\sqrt{2})^2} + \left(1 - \frac{1}{\mu_{cov}}\right) \min\left(1, \frac{2\mu_{eff}-1}{(n+2)^2+\mu_{eef}}\right)$

$\mathbf{C} \leftarrow \mathbf{I}$

$\boldsymbol{p}_c, \boldsymbol{p}_\sigma \leftarrow \mathbf{0}$

$\sigma^0 = 0.5$

$g = 1$

// Algorithmic sequence

while g != g_{max} **do**

  foreach $i \in 1, ..., \lambda$ **do**

    $P^{(g)} \leftarrow m\left(P^{(g-1)}\right)$ (see eq. (2.53))

  end

  $P_\mu^{(g)} \leftarrow s(P^{(g)})$

  $\boldsymbol{m}^g \leftarrow r(P_\mu^{(g)})$ (see eq. (2.52))

  compute $\boldsymbol{p}_\sigma^{(g)}$ (see eq. (2.57))

  compute $\boldsymbol{p}_c^{(g)}$ (see eq. (2.56))

  adapt $\mathbf{C}^{(g)}$ from $\mathbf{C}^{(g-1)}$ (see eq. (2.54))

  update $\sigma^{(g)}$ (see eq. (2.58))

  g++

end

return $P^{(g)}$

Chapter 3

Concept and Methodology

The concept and underlining methodology of the solution to the optimization problem is presented in this chapter. As explained in Chapter 1 the topological and dimensional optimization requires the consequent evaluation of the structure's workspace in order to estimate the fitness of a modified structure.
The topological and dimensional kinematic structure optimization in this thesis is reduced to the comparison of dimensionally optimized topologies. This restriction is made, because the mechanical designer of the robot has a limited scope of topologies he/she can realize due to mechatronical limits and design constraints. These constraints are hard to express numerically, such that the pre-design of topologies is left to the mechanical designer beforehand. This restriction has also been discussed in [9] with the conclusion that the topology synthesis can be appropriately achieved by using existing knowledge. The structure optimization therefore reduces to the dimensional optimization which must yield a fitness value that is invariant to properties of specific topologies in order to provide comparability.

In contrast to related literature work, this thesis provides a more task-oriented approach to the kinematic structure optimization of humanoid robots and robots in general. While much effort was dedicated to optimizing structures regarding one kinematic metric within the workspace or a very exclusive selection of task-relevant poses, this thesis incorporates four important aspects into the objective function which relate to the key objectives which are formulated for this thesis in Section 1.1.

- Consideration of task areas
- Task-dependent composition of kinematic metrics for modeling of required dexterities
- Task-related involvement of one or multiple coupled end effectors
- Prioritization of tasks

The integration of these aspects into the objective function is derived in the following course of this chapter. First, the optimization workflow from an initial structure topology, via the evaluation of a parameterized structure model, to a set of optimal parameters is presented in Section 3.1. Based on the workflow and its structure, it is then explained in detail, how the workspace of a robot can be determined in a computationally efficient way, using the kinematic relations of Section 2.2. The computed workspace is then evaluated regarding its task-related dexterity, using the kinematic metrics presented in Section 2.3. Hereby, the consideration of the task-related involvement of the end effectors is incorporated when computing the corresponding metrics which is achieved in Section 3.3.

3.1 Workflow of Task-Oriented Kinematic Structure Optimization

It is stated, that a robotic structure can be considered *fit*, if it provides the required kinematic dexterities in task areas (see Chapter 1). Therefore, it seems reasonable to introduce a task fitness $f_{t,l}$ for the l-th of the n_{task} tasks, the robot is supposed to be optimized for. Their weighted sum produces an overall structure fitness representation which incorporates the aspect of prioritization from above through the choice of the weight $\omega_{t,l}$.

$$f_s = \frac{1}{\sum_{l=1}^{n_{tasks}} \omega_{t,l}} \sum_{l=1}^{n_{tasks}} \omega_{t,l}\, f_{t,l} \tag{3.1}$$

The task fitness $f_{t,l}$ is computed analogously to equation (2.9) from Section 2.3. In this way the aspect of task area integration from above is regarded. The task area W_{ta} contains each pose that must provide the required kinematic dexterities for task execution. Additionally, a normalized weighting factor $\omega_{ta,l}(\boldsymbol{x}_{t,d})$ allows to create a variance of how much a task pose within a task area contributes to the objective function. In this thesis the weighting is used to create a fluent transition between task areas and non-task areas. The schematics of the distribution of $\omega_{ta,l}$ with respect to the task area is shown in Figure 3.1. The grey squared grid represents a 2D version of the voxels of BB and the circles relate to the spheres of VFS where each slice denotes a discrete orientation. The task-related voxels within the BB are marked grey. While the bright blue slices denote the task relevant poses with the highest $\omega_{ta,l}$, the dark blue poses mark the non-task-area with a $\omega_{ta,l}$ above zero. Due to the linear modeling of the transition, $\omega_{ta,l}$ is lower by a factor of 2 for the dark blue slices compared to the bright blue slices for the given schematic scenario (see Figure 3.1).

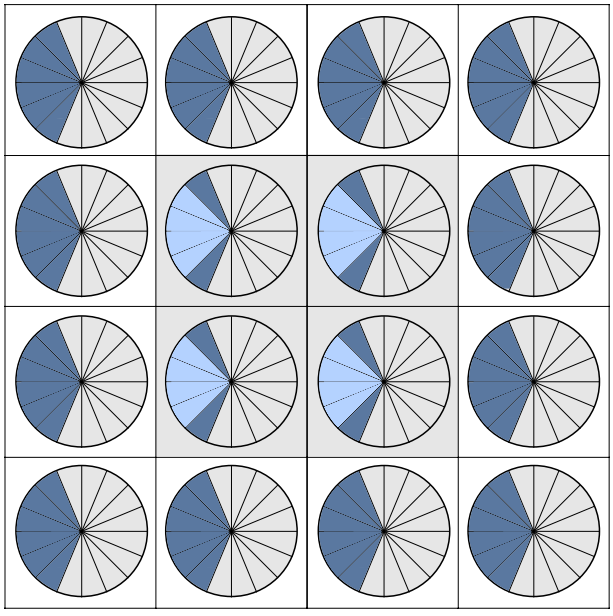

Figure 3.1: Schematics of the distribution of the weighting factor for fluent transition from task area to non-task area. The grey box denotes the positional task area. The bright blue slices of the VFS scheme denote a $\omega_{ta,l}$ higher by a factor of 2 compared to the dark blue slices.

Another of the listed features from above is included in the derivation of the task fitness $f_{t,l}$ in equation (3.3). It is supposed that humanoid robots can either interact with the environment with one end effector or using both in a coupled state. Each of these interaction types is from

now on called an Endeffecting Mode (EM). An EM hereby defines the relative poses of the specific end effectors e to the task pose $\boldsymbol{x}_{t,d}$. The consideration of the specific EM within the structure evaluation is realized through the task-related dexterity $f_{dext,l}$. Here the kinematic metrics are incorporated which are computed with respect to the robot's EM-specific state. If the reachability of task poses in W_{ta} should not be regarded inside of the $f_{dext,l}$, then the reachability must be excluded from the integration process. This is realized with the factor $c_{RI,normalize}$, using the task-related RI specific to the EM of task l.

$$c_{RI,normalize} = \begin{cases} 1 & \text{, if RI is task-relevant} \\ \left(\sum_{\boldsymbol{x}_{t,d} \in W_{ta}} M_{RI,EM}\left(\boldsymbol{x}_{t,d}\right)\right)^{-1} & \text{, if RI is task-irrelevant} \end{cases} \tag{3.2}$$

$$f_{t,l} = \frac{1}{\sum_{\boldsymbol{x}_{t,d} \in W_{ta}} \omega_{ta,l}\left(\boldsymbol{x}_{t,d}\right)} \; c_{RI,normalize} \sum_{\boldsymbol{x}_{t,d} \in W_{ta}} \omega_{ta,l}\left(\boldsymbol{x}_{t,d}\right) \; f_{dext,l}\left(\boldsymbol{x}_{t,d}\right) \tag{3.3}$$

The computation of the task-related dexterity $f_{dext,l}$ is realized through the composition of kinematic metrics for the specific EM. This step is accomplished by computing the product over each local metric M_{EM} that is part of the set of task-relevant metrics M_{tr}. Hereby, the mapping from the l-th task to the corresponding EM is considered by using the EM-specific local metrics $M_{EM,t}$

$$f_{dext,l}(\boldsymbol{x}_d) = \prod_{M_{EM} \in M_{tr}} M_{EM}(\boldsymbol{x}_d) \tag{3.4}$$

In [22] the sum over the global metric representation was used to create a single fitness value (see Section 2.1). However, computing the product over metrics has a significant advantage over the summation of the same. Although some of the metrics (RI, JRA and CI) are bound between 0 and 1, providing the sum could provide an imbalance regarding the impact of the metrics to M_{EM}. This can be shown with a simple example. If two robots were to compare for a task area containing only one task pose, the following results could occur. Robot A with $M_{JRA} = 0.95, M_{CI} = 0.0001 \rightarrow f_{dext,l} = 0.9501$ and robot B with $M_{JRA} = 0.93, M_{CI} = 0.01 \rightarrow f_{dext,l} = 0.94$. In this case robot A is favored although it provides a 100 times worse CI compared to the 2% improvement regarding JRA. The formulation of a product overcomes this issue.

From the contemplations of above it can be seen that equations (3.1) and (3.4) relate to a hierarchical workflow. The procedure is structured into four hierarchical layers and a sequence of actions which is visualized in Figure 3.2.
The highest layer of the procedure represents the *Optimization Layer*. First, the solver receives a model of the topology and decides on the parametrization of each iterative step during the optimization. Then, a model for the dimensionally modified structure is generated which is passed to the hierarchically lowest *Workspace Layer* (see Figure 3.2). Using the kinematic relations from Section 2.2 the workspace of every single end effector is subsequently computed. This means that a mapping $map(Q_{ws,e} \rightarrow W_{ws,e})$ between a set of end effector specific joint configurations $Q_{ws,e}$ and workspace poses $W_{ws,e}$ is derived. An efficient way of computing the workspace is proposed in Section 3.2. In a next step within the Workspace Layer, the EM-specific kinematic metrics are computed. Invariant to the type and number of EM, the computed end effector workspaces from the previous step can be used to obtain the metrics. This process is described in Section 3.3. For a humanoid robot, the objective function of

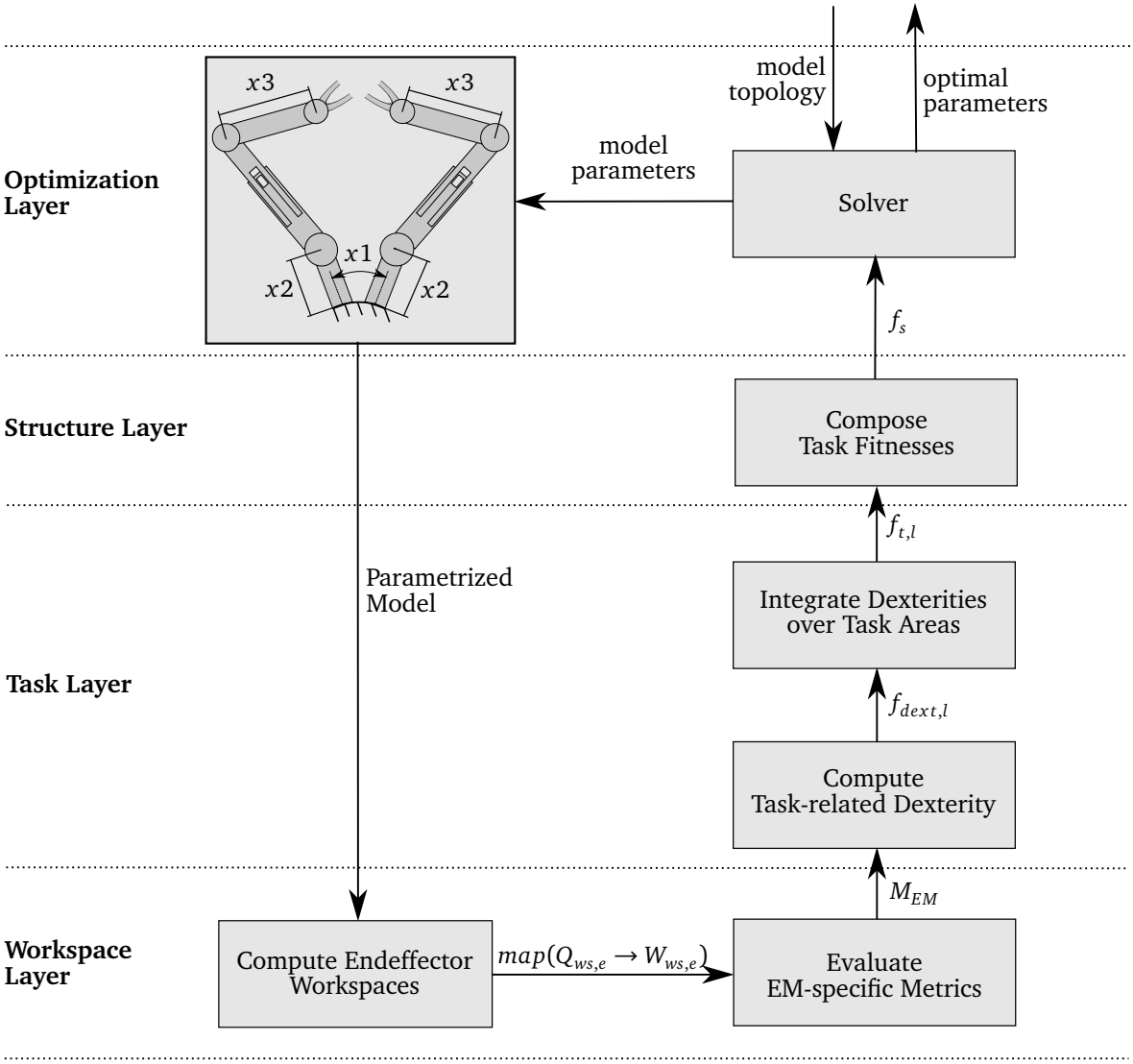

Figure 3.2: Conceptual workflow of task-oriented kinematic structure optimization of robots. The hierarchical dependencies of the worklow are classfied into the four layers *Optimization, Structure, Task* and *Workspace*.

this methodology could therefore describe a left-arm, a right-arm and multiple dual-arm EM where the arms are coupled in a different manner for various tasks. This can be achieved without recomputing the kinematics of the robot.

The local metrics are then composed within the *Task Layer* to the task-related dexterity $f_{dext,l}$ (see eq.(3.4)). Consequently, the fitness of every defined task is generated through integration of $f_{dext,l}$ over the task area (see eq. (3.3)). This procedural step is located in the *Task Layer*. From equation (3.1) results the computation of the structure fitness f_s in the *Structure Layer*. f_s represents the function value of the objective function and is subsequently used by the solver to either repeat the presented procedure in an iterative way or yield an optimal dimensional parameterization of the structure (see Figure 3.2).

3.2 End effector Workspace Computation in the Workspace Layer

The term *Workspace Computation* in this thesis refers to the computation of the mapping between joint configurations and workspace poses for each end effector of the robot. In Section 2.4 several ways of discretizing a robot's workspace in SE(3) are introduced. It is then derived why a combination of the BB and VFS scheme proposes essential benefits to the application of the workspace computation of humanoids. They are applied in this thesis with some adjustments which are presented in the subsequent Section 3.2.1.

Due to the fact, that many of the presented humanoid robots from Chapter 1 have redundant extremity topologies, the workspace computation purely based on FK through joint space sampling or IK through workspace sampling is computationally expensive. As derived in Section 2.1 both advantages can be combined through the HYB method where first random joint space samples are evaluated and then, remaining unmapped workspace poses are checked using IK. In this thesis the HYB approach is refined by exploiting the potential of good initial configurations more intensively for the IK. Further explanations follow in Section 3.2.2.

3.2.1 Adjustments to Discretization Schemes

In the classic BB, presented in Section 2.4.1, each of the boundary faces of the voxelized cube are located in a distance of the overall robot length. This definition is suitable for the given literature application (compare [7, p.6f]), because the serial robot only contained revolute joints. However, robotic structures can also have prismatic joints, so that this boundary condition does not hold. Further, the concept allows to define task areas which have to be discretized as well. For this thesis, it is therefore chosen to uncouple the choice of the box boundaries of BB from the robot's structure and allow the definition of arbitrary cuboids for the voxelization scheme.

The VFS also requires modification to suit the task of this thesis. As explained in Section 2.4.2, it is common to introduce a coordinate system convention to the way how endeffector rotation matrices are represented in the discretization scheme. For instance, for reasons of visualization it is suitable to adapt the mapping, so that for any orientation the endeffector "pierces" through the sphere's surface and points towards the voxel's center. The application of this convention has proven suitable in [7]. The same idea is pursued in this thesis. However, it is assumed that the endeffector's major axis is modeled with the x-axis of the end effector frame. The rotation matrix ${}_0\boldsymbol{R}_{eef}$ is therefore converted for the application to Section 2.4.2.

$$ {}_0\boldsymbol{R}_c = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} {}_0\boldsymbol{R}_{eef} \qquad {}_0\boldsymbol{R}_{eef} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} {}_0\boldsymbol{R}_d \tag{3.5} $$

Due to the requirement of efficient workspace computation, stated in the Outline (see Section 1.1) of this thesis, an improvement to the VFS scheme was created for the determination of index k from a given endeffector orientation ${}_0\boldsymbol{R}_{eef}$.

In equation (2.39) (see Section 2.4.2) k is derived by comparing the angle of the z-axis $\boldsymbol{a}_{k,z}$ of the k-th frame to the z-axis of ${}_0\boldsymbol{R}_c$. The number of calculations of the angles increases linearly to the number n_k of well-distributed points on the sphere. The following procedure

overcomes this dependency and behaves invariant to the chosen discretization granularity. It is named *Spiral Walking Algorithm (SWA)* from here on. Figure 3.3 visualizes the steps using the preknown visualization of the VFS from Section 2.4.2.

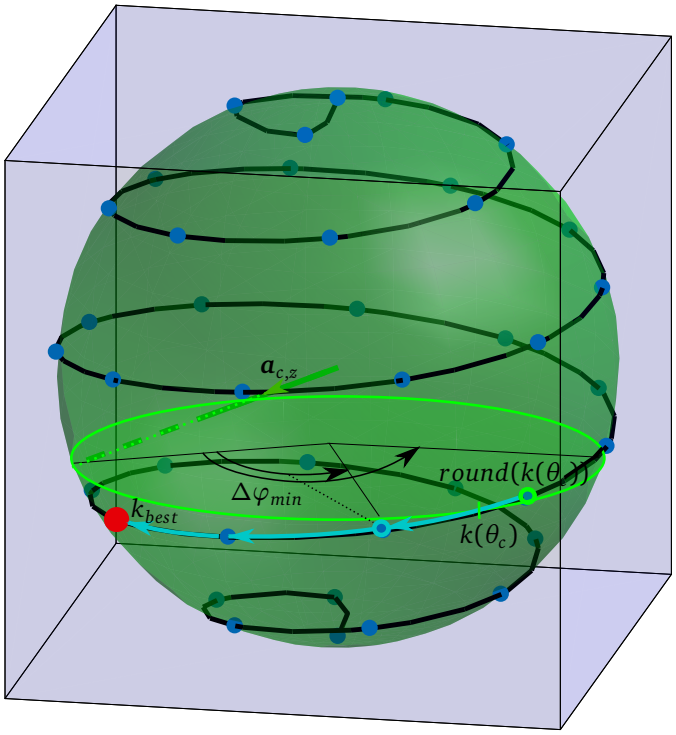

Figure 3.3: Schematic display of the procedure in the SWA. The starting point is marked green, derived from the intersection of the z-coordinate iso-contour with the spiral. $\Delta\varphi_{min}$ determines the walking direction.

In a first step, the spherical coordinates θ_c, φ_c of the z-axis ${}_0\boldsymbol{a}_{c,z}$ are determined. Solving equations (2.31) and (2.32) for k and rounding the result yields the index with the closest θ angle. This process is shown in Figure 3.3. The green circle at a given height of the sphere, represents all surface points with the same θ as the z-axis. The intersection with the spiral marks the point on the spiral with the same θ. Due to rounding the green marked spiral point is determined. From this discrete index, the algorithm "walks" the spiral up- or downwards. The spiral is walked upwards, if the neighbor index $k+1$ yields a reduction of the deviation of φ_{k+1} compared to φ_c. Otherwise, the algorithm will continue "walking" downwards which is the present case in Figure 3.3. In each "walking" step, the deviation is recomputed iteratively until the deviation increases again or the index exceeds one of the spiral's endpoints. The corresponding $k-1$ is assumed to yield the best map for ${}_0\boldsymbol{a}_{c,z}$ and is marked with red in Figure 3.3.

The mapping of orientations is therefore rewritten to

$$OD_i : \mathbb{R}^{3\times3} \rightarrow \mathbb{N}^2; {}_0\boldsymbol{R}_c \mapsto \boldsymbol{o}_d \quad OD_i({}_0\boldsymbol{R}_c) = \begin{pmatrix} k \leftarrow SWA\left({}_0\boldsymbol{a}_{c,z}\right) \\ i \leftarrow eq.(2.42) \end{pmatrix} \tag{3.6}$$

Algorithm 6: SWA: New approach for inverse mapping of k in VFS.

Input: ${}_0\boldsymbol{a}_{c,z}$

$\theta_c, \varphi_c \leftarrow sphericalCoordinates({}_0\boldsymbol{a}_{c,z})$
$k \leftarrow$ solve eq. (2.32) for k: $round\,(k(\theta_c))$
$\Delta\varphi_{best} \leftarrow |\varphi_c - \varphi_k(k)|$
if $|\varphi_{eef} - \varphi_k(k+1)| < \Delta\varphi_{best}$ **then**
  $d_{walking} \leftarrow +1$
else
  $d_{walking} \leftarrow -1$
end
while $k \geq 1$ *and* $k \leq n_k$ **do**
  $k = k + d_{walking}$
  $\Delta\varphi_k \leftarrow |\varphi_{eef} - \varphi_k(k)|$
  if $\Delta\varphi_k < \Delta\varphi_{best}$ **then**
    update $\Delta\varphi_{best}$
  else
    break
  end
end
return $k - d_{walking}$

3.2.2 Efficient Workspace Kinematics

The NR method, introduced in Section 2.2.3, successively reduces the approximation error of the given configuration to the desired workspace pose by iteratively computing an additional change to the initial configuration through linearization (compare Section 2.2). Consequently, due to properties of the *pseudoinverse*, an initial configuration with a close euclidean distance to a feasible configuration within the joint space converges faster (compare Section 2.2.3). The idea of an efficient workspace computation in this concept is therefore, to apply the NR method with the *pseudoinverse* and use initial configurations from neighboring workspace poses. These configurations have a high probability of being located closely in the joint space to feasible solutions. Additionally, the several neighbors of a workspace pose provide a high potential that joint configurations from different areas of the *null space* were mapped by the sampled FK. This facilitates to represent the robot's redundancy when solutions are generated from multiple neighbors.

A discrete workspace pose with the discrete index vectors $\boldsymbol{p}$ and $\boldsymbol{o}$ is considered neighbor of $\boldsymbol{p}_c$ and $\boldsymbol{o}_c$ within this concept, if the following conditions are met which use the relations from equations (2.29), (2.38) and (2.41) of the discretization schemes presented in Section 2.4

1. $\|PD_i(\boldsymbol{p}) - PD_i(\boldsymbol{p}_c)\| \leq r_v$
2. $|\beta(\boldsymbol{a}_{k,z}, \boldsymbol{a}_{k_c,z})| \leq \gamma_\beta$
3. $|\alpha(\boldsymbol{a}_{k,x}, \boldsymbol{a}_{k_c,x})| \leq \gamma_\alpha$

The conditions are explained with the support of Figure 3.4. Here, a three DoF robot operating in a 2D plane with three configurations is visualized. The middle configuration represents the initial configuration. The visualization of the discretizations are known from Section 3.1.

Thus, the discrete pose of the initial configuration is marked green. The 1. neighboring condition implies that a voxel contains neighbor poses, if its center is positioned inside of a sphere with radius r_v around the center at $\boldsymbol{p}_c$. This relation is visualized through the blue circle in the 2D example of Figure 3.4. Each voxel's center within this area meets this first condition for the exemplary 2D case. The 2. and 3. conditions state that a neighbor pose must have an orientation where α and β from equations (2.38) and (2.41) do not exceed the corresponding limits γ_β and γ_α. These orientations are marked as blue slices of the sphere in Figure 3.4. Although the discretization granularity of the example is very big compared to the robot's size, the three presented configurations of the robot show that relatively small deviations of individual joints were necessary to create the feasible neighbor configurations, marked with dotted shape-boundaries. The deviations are indicated roughly by the distance markers at every joint.

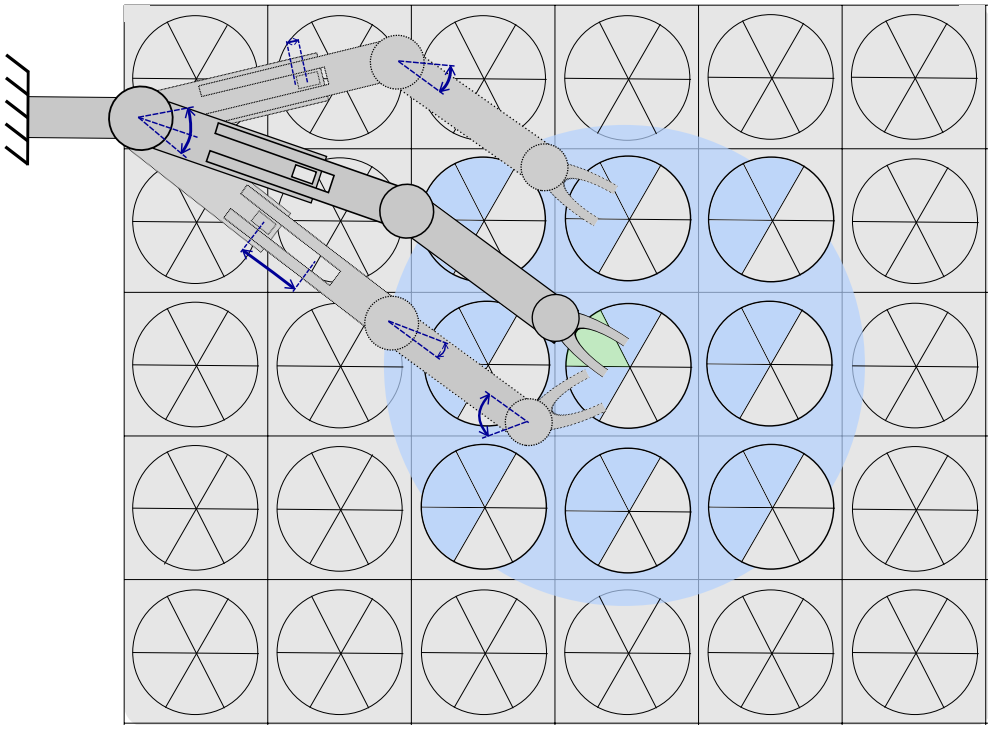

Figure 3.4: Schematic representation of the HYB scheme using neighboring initial configurations. The middle configuration is used to compute the upper and lower configuration with the NR method which yields solutions with small change in the joint deviations.

The procedure of HYB using these neighboring relations is explained in Algorithm 7. First, the joint configuration space is sampled which spans the set Q_{ws}. Using FK, the samples are mapped to W_{ws}. With the conditions for neighborhood from above, IK is applied to any neighbor $\boldsymbol{x}_n$ of each reachable pose $\boldsymbol{x} \in W_{ws}$ where the number of mapped configurations q_n is lower than the threshold n_{minMap} of each $\boldsymbol{x} \in W_{ws}$. This threshold restricts the number of times IK is applied to the same workspace pose. This is essential for redundant systems that can provide numerous configurations for one workspace pose which would be computationally expensive.

Algorithm 7: Workspace computation using HYB with neighborhood pose relations.

Input: n_{minMap}, r_v, γ_β, γ_α

```
Q_ws ← sample joint configuration space
map(Q_ws ↔ W_ws) ← compute Workspace mapping with FK(q ∈ Q_ws)
foreach x ∈ W_ws do
    foreach neighbor x_n of x do
        if q_n < n_minMap then
            map(Q_ws ↔ W_ws) ← append solution of IK(x, any(Q_ws(x_n)))
        end
    end
end
```

3.3 EM-specific Metric Evaluation in the Workspace Layer

n this section the computation of EM-specific kinematic metrics of robots is presented. Several metrics and their applicability to single or multiple end effectors have been discussed in Section 2.3. While metrics like the RI, CI and MM have proven to be useful in the course of this thesis, several critical aspects of the JRA, proposed in [38], are discussed. Thus, Section 3.3.1 introduces a more suitable formulation of the JRA. Additionally, a new kinematic metric is developed in Section 3.3.2 which is called Velocity Transmission Ratio (VTR). It employs the relations of velocity transmission which are presented in context of the MM. While MM is a directionless metric, VTR evaluates the transmission of velocity from joint space velocities to workspace velocities in a specific direction. This proves to be an important dexterity required for some tasks. Finally, this section concludes with Section 3.3.4 where the computation of a local metric considering the relative pose of a single or multiple coupled end effectors is presented. Thus, it is explained how the results of the workspace computation of each end effector can be used for this step.

3.3.1 Reformulation of the Joint Range Availability (JRA)

In order to overcome the unboundedness of the metric and its variance to the robot's DoF (see Section 2.3.4), the following redefinition of the JRA from equation (2.21) is proposed. As described in Section 2.3.4, it is inspired from the joint-limit based augmentation of the end effector Jacobian from [42] and [21, p.7ff].

$$M_{JRA} = \sqrt[n]{\prod_{i=1}^{n} \frac{\min\left(q_{max,i} - q_i, q_i - q_{min,i}\right)}{\frac{1}{2}\left(q_{max,i} - q_{min,i}\right)}} \tag{3.7}$$

This metric is bounded $M_{JRA} \in [0,1]$. Due to the application of the geometric mean to the normed joint displacements, M_{JRA} yields 0 if one of the robot's joints has reached its joint limits. If all joints are in their mid-position, M_{JRA} becomes 1. This behavior proves as a suitable dexterity criterion, because a joint space configuration in distance of joint limits ensures, that the full spectrum of the joint's agility space is available for disposal from the given joint space configuration.

3.3.2 Velocity Transmission Ratio (VTR)

YOSHIKAWA [15] has exploited the volumetric property of the Jacobian velocity transformation in order to numerically evaluate *manipulability* (see Section 2.3.3). While most of the tasks of humanoid robots require a high *manipulability* in all the existing workspace directions, a task can also require a high movability in one task-specific direction. In order to evaluate this dexterity the VTR metric is proposed in this section.

Based on equation (2.2) and the properties of the *pseudoinverse*, it can be rephrased that for

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}\dot{\boldsymbol{x}} \tag{3.8}$$

the *pseudoinverse* yields $\dot{\boldsymbol{q}}$ with the lowest euclidian norm of all feasible joint space velocities (see Section 2.2.3). This means, that if a unit workspace velocity $\dot{\boldsymbol{x}}_c$ is transformed via equation (3.8), the best possible velocity transmission ratio from the joint space to the workspace in the given direction can be computed as

$$vtr(\dot{\boldsymbol{x}}_c) = \begin{cases} 0 & \text{, if all directional components of } \dot{\boldsymbol{x}}_c \text{ belong to } \mathcal{R}(J)^{\perp} \\ \left(\left\| \boldsymbol{J}^{\dagger} \frac{\dot{\boldsymbol{x}}_c}{\|\dot{\boldsymbol{x}}_c\|} \right\| \right)^{-1} & \text{, else} \end{cases} \tag{3.9}$$

where $\mathcal{R}(J)^{\perp}$ denotes to the set of degenerated velocity directions (see Section 2.2.3). A robot with a high vtr can thus produce high velocities in the specific direction $\dot{\boldsymbol{x}}_c$. However, for most tasks it is not exactly one direction where the robot must be capable of producing velocities, because humanoid robots are deployed in changing environments (see Chapter 1). It is therefore desirable to consider the immediate surroundings of the task direction. This aspect is incorporated into the following metric formulation M_{VTR} which provides characteristics suitable for the application to the kinematic structure optimization tasks of this thesis.

It is assumed that a direction belongs to the task description if it lies within a hyper-spherical cap that is spanned across the task direction. This context is visualized in Figure 3.5 where the translational manipulability ellispoid of an exemplary configuration of the *Franka Emika Panda* collaborative robot is shown (see [61]). The ellipsoid is represented in blue and the hyper spherical cap in red.

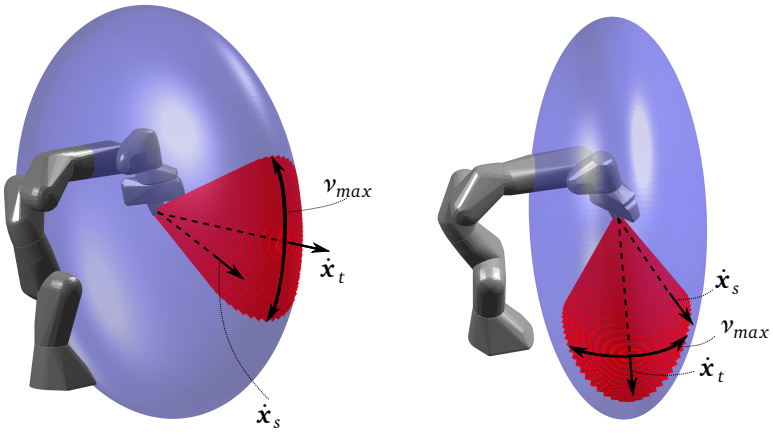

Figure 3.5: Hyper spherical cap of the translational manipulability ellipsoid of the *Franka Emika Panda* collaborative robot with the maximal opening angle ν_{max} [61].

The hyper spherical cap which is described through its maximal opening angle ν_{max}, surrounds the task direction, marked in red Figure 3.5. Using this relation, unit directions $\boldsymbol{x}_s \in X_s$ are evaluated with (3.9). In order to take into consideration that the vtr in task direction should be considered more intensively than directions close to the boundaries of the hyper-spherical cap, gaussian weighting is applied. The gaussian weighting function Φ is fitted with a mean $m = 0$ and a standard deviation of $\sigma = \nu_{max}/2$, so that approximately $95,5\%$ of the overall weight is distributed. A smaller σ would lead to evaluations of (3.9) at the boundary of the hyper-spherical cap which contribute too little to be reasonable regarding their computational effort. This provides a well balanced weighting progression from the task direction to directions at the boundary of the hyper-spherical cap. The final formulation of M_{vtr} is then normalized by the weights that were distributed within the calculations of vtr for each direction.

$$M_{VTR}(\boldsymbol{q}) = \frac{1}{\sum_{\dot{\boldsymbol{x}}_s \in \dot{X}_s} \Phi(\nu(\dot{\boldsymbol{x}}_s))} \sum_{\dot{\boldsymbol{x}}_s \in \dot{X}_s} \Phi(\nu(\dot{\boldsymbol{x}}_s))\ vtr(\dot{\boldsymbol{x}}_s) \tag{3.10}$$

This general formulation of the VTR can also be reduced to the isolated consideration of velocity transmission regarding translation or rotation. In this case $\dot{\boldsymbol{x}}_c$ is a three-dimensional (3D) translational or rotational velocity vector and the Jacobian is reduced to its translational or rotational components, $\boldsymbol{J}^T$ or $\boldsymbol{J}^R$, repectively. These representations can be useful when the translational respectively rotational components do not matter. The translational variation of vtr is applied in Chapter 5.

3.3.3 Application of MM and VTR to Coupled End Effectors

In Section 2.3.5 the capabilities of coupled end effectors can be represented by the composition of the dexterities of each individual end effector, where the coupled version of the metrics RI, CI and JRA have been proposed. This section starts with a more general formulation of the 2D version of the coupled MM (see Section 2.3.3). The intersection ellipsoid of all EM-specific end effector manipulability ellispoids with possibly heterogeneous dimensions must be formulated differently. Due to normalization techniques applied in (2.20), the MM represents the geometric mean over the length of all semi axes of the manipulability ellipsoid. Extrapolated to the coupled state of end effectors, it means that the geometric mean of the intersection ellipsoid must be lower bound to the minimum length of the semi axes of all end effector manipulability ellipsoids. The coupled version of the MM, thus, relates to the volume of the biggest hyper sphere which can be fit into all manipulability ellipsoids. Using (2.17) and (2.19), the coupled metric is written as

$$M_{MM,coupled}(\boldsymbol{x}) = \arg\min_{e \in EM}(\sigma_{J_e}) \tag{3.11}$$

However, it is mentioned that this conservative representation of the intersection ellipsoid does not add any value to the already derived coupled version of CI. Its applicability to coupled end effectors is thus disregarded and reveals an important field of improvement regarding this thesis which is discussed in Chapter 6.

The VTR of coupled end effectors is derived by the minimum $M_{VTR,e}$ of all end effectors analogously to other metrics in Section 2.3.5. It states, that the end effector with the lowest

velocity transmission in the task direction is representative for the maximum capabilities of the coupled state. Thus, $M_{VTR,coupled}$ writes

$$M_{VTR,coupled}(\boldsymbol{x}) = \underset{e \in EM}{\arg\min}(M_{VTR,e}) \tag{3.12}$$

3.3.4 EM-specific Local Metrics from seperated End Effector Workspaces

As already explained in Section 3.1 the kinematic mapping of each end effector workspace can be used to determine an EM-dependent local metric $M_t(\boldsymbol{x}_{t,d})$ for a given task pose $\boldsymbol{x}_d$. Each kinematic metric requires that the given pose can be reached by the robot. This is determined by backtransforming the task pose $\boldsymbol{x}_d$ to the pose, the end effector must be in, to fulfill the requirements of the corresponding EM. The backtransformation is visualized in Figure 3.6 where an EM with two displaced and rotated end effectors of a dual armed 2D robot is presented. The EM was defined in the described way, because the robot is supposed to accomplish a manipulation task for a food plate which is visualized in white. The figure contains the already known visualization of the discretization schemes (see Figure 3.4).

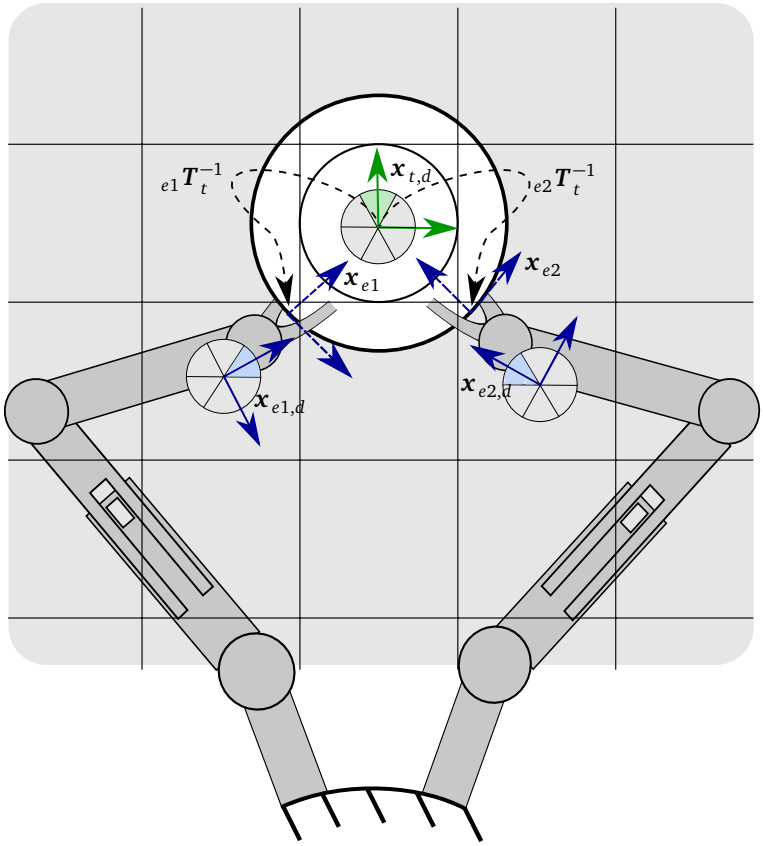

Figure 3.6: Schematic visualization of the transformation of the discrete task-related pose $\boldsymbol{x}_{t,d}$ to each discrete end effector pose $\boldsymbol{x}_{e,d}$ of the EM-specific coupling. An EM for a plate holding dual-arm robot is shown. The discrete poses within the schematic BB and VFS scheme are presented, as well as the intermediate non-discrete end effector poses $\boldsymbol{x}_e$ from the transformation process.

From the figure it can be seen that the EM-specific end effector pose $\boldsymbol{x}_e$ is computed by backtransforming the task pose $\boldsymbol{x}_d$ with the inverse of the homogeneous transformation $_e\boldsymbol{T}_t$.

This homogeneous transformation is available from the EM description, as it describes the relative pose of task pose frame within the end effector frame. This step is visualized in Figure 3.6 for both end effectors $e1$ and $e2$. However, the workspace computation of each end effector was accomplished in a discrete way. It is therefore necessary to find the discrete pose $\boldsymbol{x}_{e,d}$ for $\boldsymbol{x}_e$ based on the discretization scheme. This discretization step imposes an additional error that is visualized with the obvious offset of the two frames referring to $\boldsymbol{x}_e$ and $\boldsymbol{x}_{e,d}$. The error either annihilates the discretization error which was produced during the workspace computation or adds an additional discretization error to it. Latter case must be regarded when choosing the discretization steps.

Now, the EM-specific metrics $M_t(\boldsymbol{x}_{t,d})$ can be computed. This procedure is explained in Algorithm 8. The following steps are repeated for each end effector e that is part of the EM. Using the end effector pose $\boldsymbol{x}_e$, the kinematically mapped joint configurations can now be collected to a set Q_e. This set can subsequently be used to compute the Jacobian matrix $\boldsymbol{J}_{d,e}$ in $\boldsymbol{x}_{t,d}$ corresponding to the to the end effector e. It expresses how joint space velocities from the kinematic chain regarding the end effector e produce workspace velocities at the given task pose $\boldsymbol{x}_{t,d}$. This can be achieved by adding a fixed segment to the end effector within the tree model with the transformation of ${}_e\boldsymbol{T}_t$ and using the new segment to derive the Jacobian accordingly to equation (2.4).

The resulting configuration- and end effector-dependent, kinematic metric $M_{t,e}(\boldsymbol{q}_d)$ is then averaged to obtain the pose-dependent metric $M_e(\boldsymbol{x}_{t,d})$. The average was hereby favored over extreme representations (see Section 2.3). During task execution the robot control determines the configuration within the movements. Many control strategies e.g. for the exploitation of a redundant robot's *null space* exist. Due to the fact that any possible configuration might be picked dependent on the controller, it seems reasonable to average the kinematic dexterity for a given task pose. If more than one end effector takes part in the EM, the pose-dependent metrics $M_{t,e}(\boldsymbol{x}_{t,d})$ of each end effector e are composed by applying the coupled representation of metrics derived in Section 2.3.5 and Section 3.3.3. The final metric representation then yields $M_{t,EM}(\boldsymbol{x}_{t,d})$.

Algorithm 8: Computation of EM-specific Local Metrics

Input: $x_{t,d}$, $map(Q_{ws} \leftrightarrow W_{ws})$, $map(EM \rightarrow e)$, ${}_eT_t$

foreach *end effector e of EM* **do**
  add segment *em* to tree model(${}_e\boldsymbol{T}_t$, fixed joint)
  $\boldsymbol{x}_e = {}_e\boldsymbol{T}_t^{-1}\boldsymbol{x}_{t,d}$
  $\boldsymbol{x}_{e,d} \leftarrow$ discretize $\boldsymbol{x}_e$
  $Q_e \leftarrow map(Q_{ws} \leftrightarrow W_{ws})(\boldsymbol{x}_{e,d})$
  foreach $\boldsymbol{q}_e \in Q_e$ **do**
    ${}_{t,d}\boldsymbol{J}_e(\boldsymbol{q}_e) \leftarrow eq.(2.4)(em, \boldsymbol{q}_e)$
    compute $M_{t,e}(\boldsymbol{q}_e,\ {}_{t,d}\boldsymbol{J}_e)$ (s. *eq.* (2.11) (2.13), (2.20), (3.7), (3.10))
  end
  $M_{t,e}(\boldsymbol{x}_{t,d}) \leftarrow \text{avg}_{\boldsymbol{q}_e \in Q_e}(M_{t,e}(\boldsymbol{q}_e))$
end
if *more than one e in EM* **then**
  compute $M_{coupled}(\boldsymbol{x}_{t,d}) \leftarrow M_{e \in EM}(\boldsymbol{x}_{t,d})$ (s. eq. (2.22), (3.11), (3.12))
  $M_{t,EM}(\boldsymbol{x}_{t,d}) = M_{coupled}(\boldsymbol{x}_{t,d})$
else
  $M_{t,EM}(\boldsymbol{x}_{t,d}) = M_{t,e}(\boldsymbol{x}_{t,d})$
end
return $M_{t,EM}(\boldsymbol{x}_{t,d})$

Chapter 4

Implementation

In the outline of this thesis in Section 1.1 it is stated, that a software framework should be implemented which facilitates the evaluation of the concept from the previous Chapter 3. Due to the requirement of efficient computation during the structure evaluation, a compiled language is preferred over an interpreted language. The source code of the LOLA project is completely written in C++. It was therefore decided to implement the framework in C++, as well. Another advantage of this language is that several mature open-source libraries for some of the framework's functionalities exist. The framework, thus, depends on the following libraries and their field of application

- *Eigen*: Linear algebra [62].
- *Orocos KDL*: Hierachical kinematic modeling, FK and IK [63].
- *LibIGL* and *CGAL*: Triangular mesh processing [64], [65].
- *pagmo2*: Numerical optimization algorithms [66].
- *TinyExpr*: Parser and evaluator engine for string-based arithmetic evaluation [67].

The architecture of the framework was designed with close reference to the concept model imitating the layered dependencies presented in Section 3.1. An overview of the architecture is presented in the first Section 4.1. The user interfaces with the framework through data exchange which is explained in further detail in Section 4.2. Hereby, any software related settings, the discretization parameters, the structure model and the task descriptions must be defined. In return the user can obtain a visualization of the results. In order to assess how the framework processes and generates data, the software's process steps are explained in Section 4.3, disregarding low-level details of the implementation. However, the usage of the above mentioned external dependencies is included into the explanations.

The following sections only present information that is required to understand the procedural steps of the framework. For more information on the source code, a software documentation is available on the data medium which is attached to this thesis. Please consult the *README* file first, before using the software.

4.1 Framework Architecture

The framework builds on the conceptual workflow which is proposed in Section 3.1. Here, the hierarchical derivation of the structure evaluation is identified and the workflow was structured into four layers. This layered architecture is repeated in the software implementation which is visualized in Figure 4.1. The grey boxes represent a class in *C++*. A relationship between classes is either modeled as a composition relation with a diamond marked connection or a dependency with an arrow. This convention is known from the Unified Modeling Language (UML). The blue elements within the figure represent visualizations of class instances and are therefore contained within the class. The elliptic element *User Interface* represents a logic module which is represented by a conglomeration of data files and the program executables.

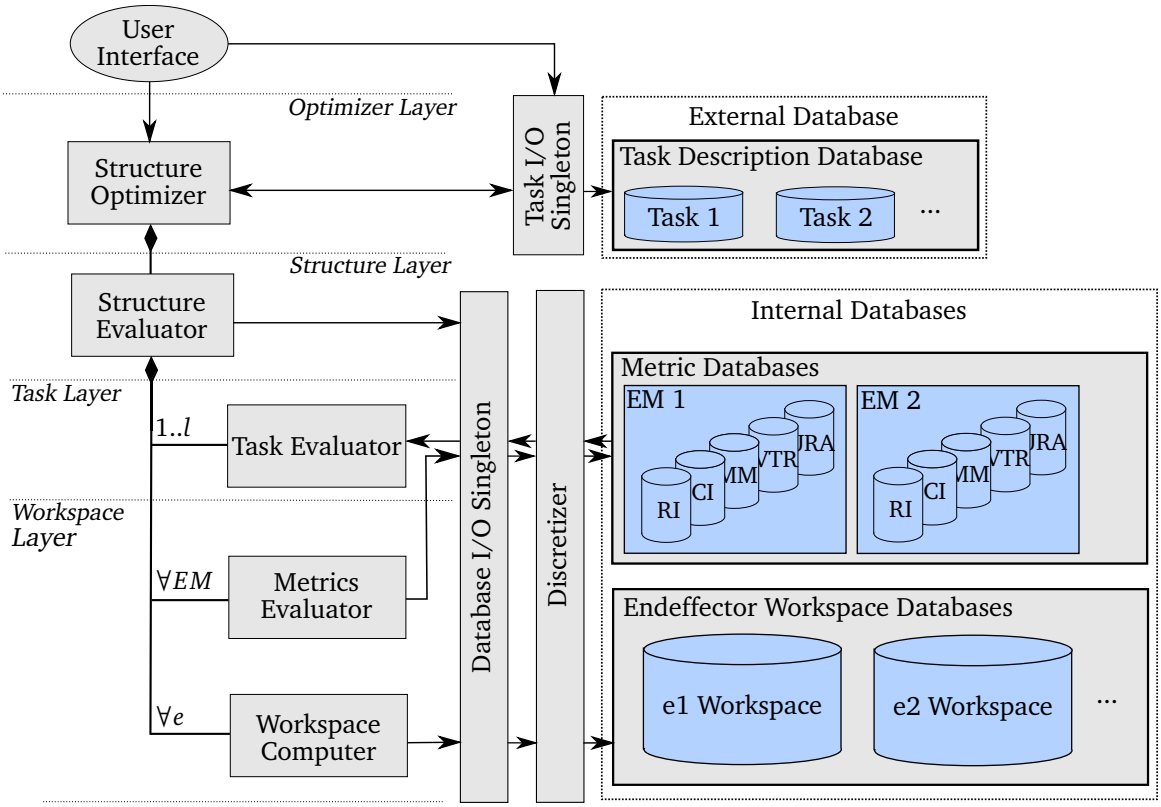

Figure 4.1: Schematic architecture of the framework. It shows the most important C++ classes identified by grey boxes and their relations, as well exemplary class instances marked in blue.

Each evaluation layer is represented by one C++ class which, among others, provides the mathematical process steps presented in Section 3.1. An exception must be made for the workspace layer. Here, two classes encapsulate the two main process steps of the workspace layer (compare Figure 4.1). First, the computation of one end effector workspace is processed by one instance of the *Workspace Computer* class and the computation of metrics corresponding to one EM is realized by an instance of the *Metrics Evaluator* class. Due to the hierarchical relations, each of the classes are instantiated by the higher level class. Again, the workspace layer is treated differently, because it is instantiated by the *Structure Evaluator* two hierarchy levels above (see Figure 4.1). This exception is desired, because it provides a decoupling from the *Task Evaluator* class instance to the *Workspace Computer* and *Metrics Evaluator* class instances of the workspace layer. Several tasks can relate to the same EM and integrate the same set of metrics.

The framework architecture provides additional classes for several functionalities. The storage of data is primarily organized outside of the layered architecture. This design was chosen, because several of the above mentioned classes access the same data which is why it must be accessible from any layer. The access to any database within the architecture is realized through an Input/Output (I/O) class which is implemented in the *Singleton Pattern* (further information on the *Singleton Pattern* can be found in [68]). This pattern assures that only one instance of the class can exist, so that each of the classes interacts with the same instance and the same data. This property is useful for the realization of multi threading which is required in order to efficiently use the existing processing power of computers with multiple Central Processing Unit (CPU) cores. It simplifies the establishment of a thread-safe environment during data access.

Two forms of data storage have been realized. One refers to external data which in the current state of the implementation is the task descriptions contained in the *Task Description Database* (compare Figure 4.1). The other corresponds to the internal data which is the *Endeffector Workspace Databases* and the *Metric Databases* (see Figure 4.1). Whenever a class saves or loads data described in the continuous workspace through the *Data I/O Singleton* class, the data is discretized by the *Discretizer* class which realizes the forward and inverse discretization mapping described in Section 2.4. With an identification variable of the endeffector and the EM, respectively, the required database is determined. The Task Description Database is fed with data from the *User Interface*. This database is seperated from the internal data. At one point it is imaginable that task descriptions are standardized so that a designer can have recourse on already created task descriptions and use them for new kinematic structure optimization tasks. The *User Interface* also triggers the structure optimization by calling the highest hierarchy level class *Structure Optimizer*.

Besides the visualized classes in Figure 4.1, the framework possesses several other classes which encapsulate small functionalities which are from now on referred to as *Helper Classes*. Three of the various *Helper Classes* are hereby important to understand the framework's procedure, presented in Section 4.3:

- *Structure Model Class*: encapsulates the kinematic tree model, holds the definition of all EM.
- *Forward Kinematics Class*: provides methods for FK using the model definitions of a Structure Model Class instance.
- *Inverse Kinematics Class*: encapsulates IK functionalities that are applied on a Structure Model Class instance.

4.2 User Interface

The framework provides the three following executables which can be called by the user. They provide different funtionalities of the system:

- *optimize*: Starts a kinematic structure optimization.
- *analyze_parameterspace*: Discretizes the parameter space with a defined step size and evaluates each parameter step.
- *visualize_structure*: Creates triangular mesh-files to visualize the workspace and kinematic metrics and task spaces for a structure with a defined parameter set.

As derived in the previous section, the user interface is currently based on a data exchange. The user must therefore provide a URDF file and two JavaScript Object Notation (JSON) files which is a popular file format for parameterization [69]. The URDF file describes the model's topology (compare Section 2.2.1). It also incorporates the optimization parameters. They are identified through a unique space-less string within the file. During the optimization process these strings are then replaced by the real value parameters chosen from the optimizer. The first JSON-file is the *General Configuration File*. It contains the following information wrapped into nested parameter structures containing dictionaries and lists of dicitonaries, respectively. Only important information for the comprehensibility of the implementation is presented here. Further and more detailed information can be found in the software's documentation, as explained in the introduction of this chapter.

- General parameters
  - Number of process threads
  - Parameters regarding FK (joint configuration space sampling/randomizing).
  - Parameters regarding IK (neighborhood definition, NR method).
  - Choice of the optimization method and corresponding configuration parameters.
- Structure model
  - URDF file path
  - Structure optimization parameters (identification string used in URDF file, lower and upper real value boundary).
  - Definition of the EM (name of the endeffector segment, transformation between endeffector and task pose ${}_e\boldsymbol{T}_{t,d}$).
- Discretization
  - BB parameters (see Section 2.4).
  - VFS parameters (see Section 2.4).

The second JSON file is called the *Task Configuration File*. Here, the following important aspects are stored which are then converted into a task description C++ struct within the *Task Description Database* (see Figure 4.1).

- List of Task Areas
  - Standard Tessellation Language (STL) file path that provides the boundaries within BB in form of a triangular mesh.
  - Translation vector for repositioning the STL mesh.
  - Minimum and maximum values for φ and θ within VFS (compare Section 2.4).
  - Discrete steps representing the length of the transition zone between task area and non-task area. Within the transition zone, $w_{ta,l}$ is modeled as linearly decreasing.
- Task Dexterity
  - Boolean parameter for every kinematic metric which decides, if it is part of the dexterity composition.
  - If VTR is part of the composition: Task direction vector, the opening angle ν of the hyper-spherical cap and a resolution parameter to determine the number of evaluations in $\dot{X}_s$ (see Section 3.3.2).

While the three above files represent the input of the User Interface, the framework also provides three forms of feedback. During software execution, information of each evaluation step is provided over the system's standard output. Additionally, each evaluation with the iterative structure optimization is documented in a Comma-Seperated-Values (CSV)[1] file which stores the optimization parameter vectors, the task and structure fitness.

Based on a specified parameter vector, several properties of the structure evaluation can be visualized in 3D, using the *visualize_structure* executable. The data is described in form of a triangular mesh containing a set of spheres. Each sphere corresponds to a voxel-filling sphere in the VFS scheme at the specific position. For the description of the mesh, the Polygon File Format (PLY) format[2] was chosen, because it has a simple syntax and allows to add a color property to each vertex and edge of the triangular mesh. This feature is necessary for a comprehensive visualization. On the one hand, this allows to give the sphere one color which can be used to represent a position averaged local metric. On the other hand the metrics of discrete orientations can be represented through the distinct coloring of vertices of a sphere. In correlation to the VFS, a vertex hereby relates to the orientation where the end effector "pierces" through the sphere, as described in Section 2.4. The color therefore represents the average metrics of all discrete orientations around the end effector's major axis of the given surface point of the VFS scheme. An exemplary visualization can be found in Figure 4.2.

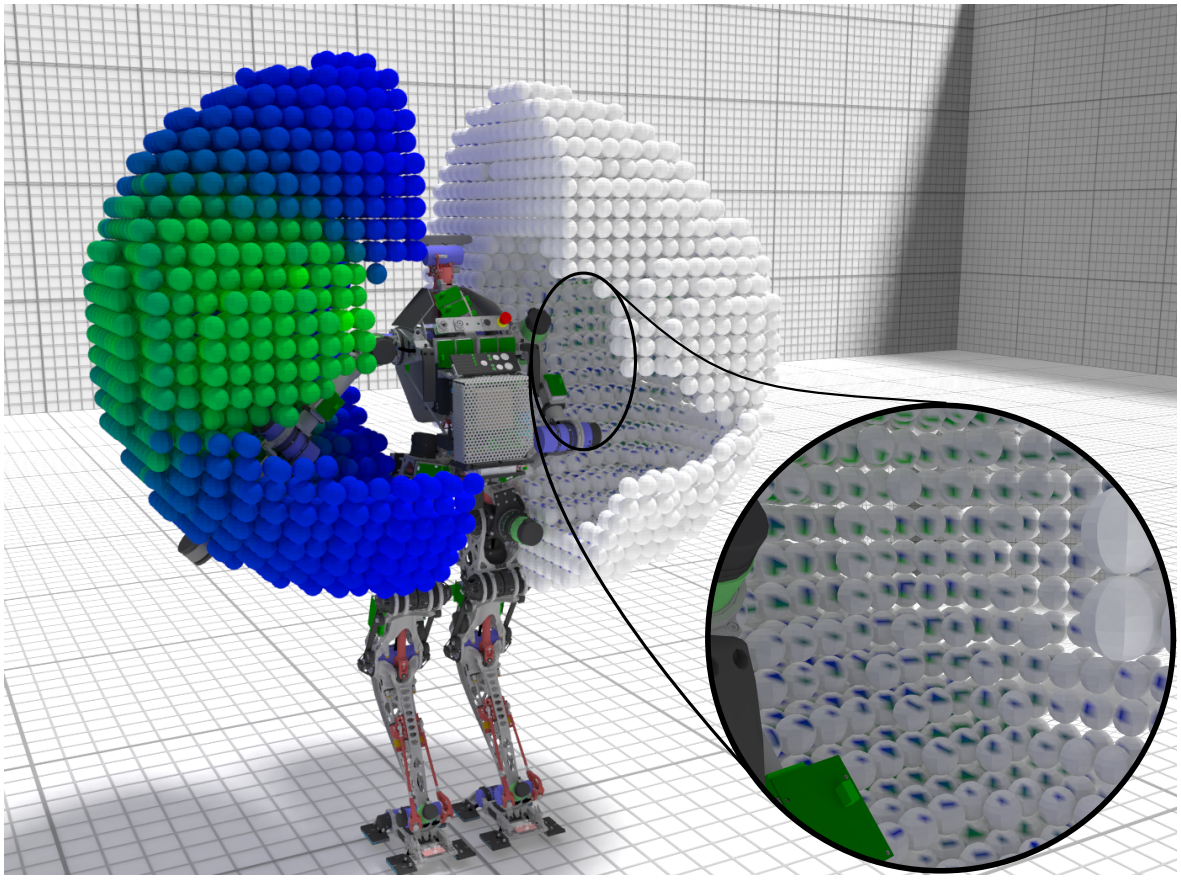

Figure 4.2: Visualization of LOLA's left arm workspace with position-averaged CI metric (green: CI = 0.88, blue: CI = 0.0) and right arm workspace with JRA metric (green: CI = 0.97, blue: CI = 0.0) .

[1]The CSV format is a lean storage format for tabular data. Seperator and newline operators structure the data.
[2]PLY is a file format for storing 3D graphical objects that are described as a collection of polygons. These are described with vertices and faces with additional property traits [70].

It presents the workspaces of the current arm structures of LOLA. The left arm workspace visualizes the position-averaged CI metric. The right workspace represents the JRA where the metric is visualized dependent on the resolution of the mesh.

These visualizations are only some of several others which are provided by the framework. The following list contains the meshes that can be generated for visualization:

- Task areas with the corresponding spatial weight $w_{ta,l}$.
- Local metrics (also averaged regarding position).
- Local metrics only within task areas where non reachable task poses are marked red.

4.3 Program Sequence

In the following, the most important procedural steps of the *optimize* executable are examined, which is representational for the framework's functionality. With the sequence diagrams in Figure 4.3 to Figure 4.6, the interaction of the framework's C++ classes are presented. A class is hereby marked blue, if all of its important interactions within the software is presented in the diagram. This should guide the focus of the reader.

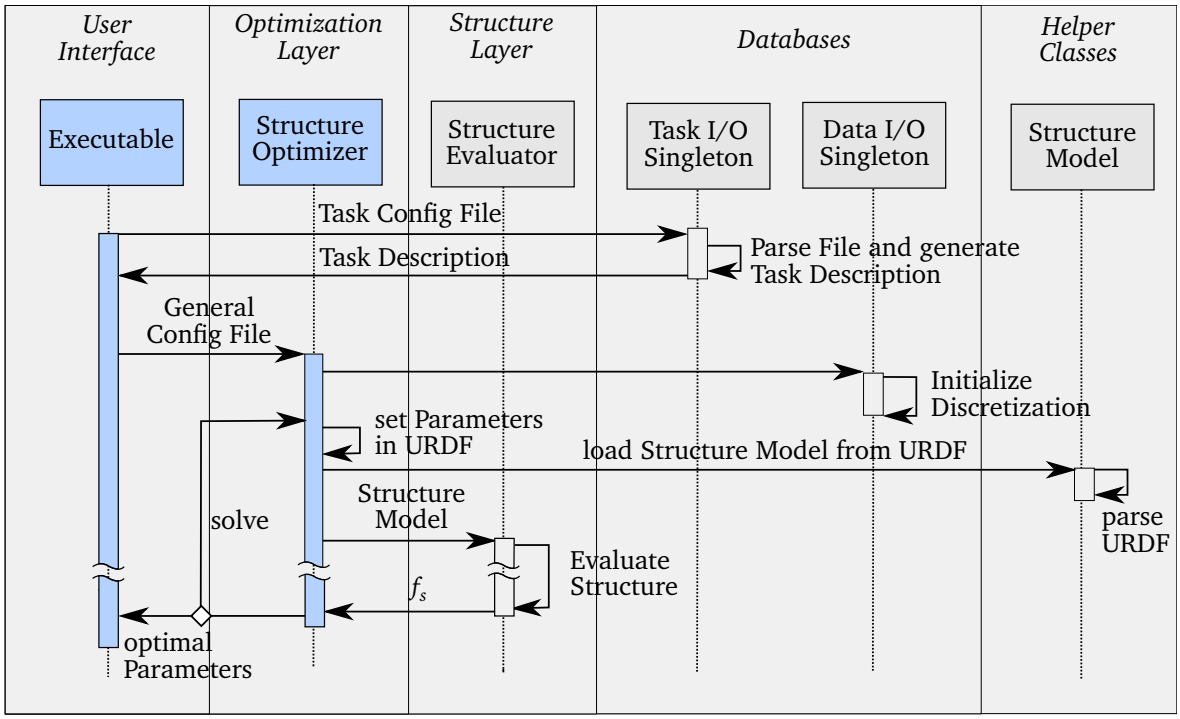

Figure 4.3: Sequence diagram with focus on the interactions of the user interface and the *Structure Optimizer*.

Through an argument defined by the user the executable passes the *Task File* to the *Task I/O Singleton*. Here, the task definitions are parsed and wrapped into a *Task Description* structure object. In a next step the *Structure Optimizer* is called which receives the user-defined *General Configuration File*. Based on the discretization parameters, the BB and VFS schemes are initialized through the *Data I/O Singleton* class which passes the information on to the *Discretizer* class (compare Section 4.1).

Subsequently, the optimizer starts the iterative optimization method. Within the *General Configuration File*, the user has defined and configured one of the three optimization methods PSO (see Section 2.5.1), SA (see Section 2.5.2) and CMA-ES (see Section 2.5.3), which are implemented through methods of the *pagmo2* library. In a next step, the identification string of the optimization parameter, defined in the *General Configuration File*, is replaced in the URDF file with a value chosen by the numerical solver. Additionally, any arithmetic operation encoded within the strings are evaluated using the *TinyExpr* library. With the updated file, the *Structure Model* class is instantiated and the kinematic tree is defined compatible with the *KDL* library. In a next step, the structure model is passed over to the *Structure Evaluator* which computes the structure fitness f_s (see Figure 4.3).

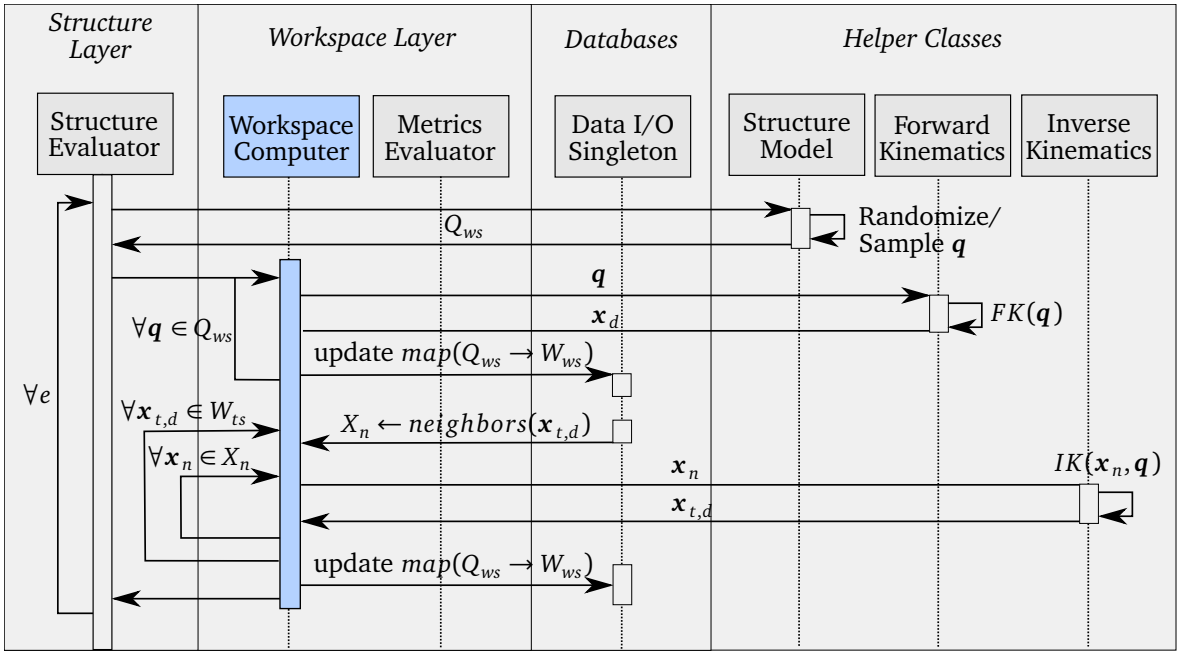

Figure 4.4: Sequence diagram with focus on the interactions of the *Workspace Computer*.

In Figure 4.4 the program sequences for the workspace computation of each endeffector are presented. First, the joint configuration samples Q_{ws} are produced with the help of the *Structure Model* class. Through the *General Configuration File* this can be obtained through sampling or randomization. While randomized samples are uniformly distributed within the joint limits, the discretization is realized as a linearly progressive discretization. Herefore, the user defines the minimal number of samples of the first joint and the amount discretization samples. The framework then calculates a suitable progressive factor and adds random samples at the end to reach the required amount of samples (see Figure 4.4).

Further, Q_{ws} is split into equally distributed sets in order to process each set in its own thread using the *Workspace Computer* class. The end effector poses $\boldsymbol{x}$ to the joint samples are then computed with FK-related methods, provided by the *Forward Kinematics* helper class. The software hereby relies on algorithms from the *KDL* library. Subsequently, the poses are discretized and saved in the *Endeffector Workspace Database* using the *Data I/O Singleton* interface class. If the user requested the usage of IK in the *General Configuration File*, the neighbors of pose $\boldsymbol{x}_{t,d}$, that is within one of the task areas W_{ta} are computed subsequently. This is again achieved by calling the singleton interface which retrieves the pose's neighbors from the *Discretizer* class (compare Section 4.1). The neighbor mapping is generated at the start of the program by computing the relative discrete steps for each neighbor. Finally, for

every neighbor pose which does not have a configuration mapping, the NR method is executed, using the *Inverse Kinematics* helper class which relies on the algorithms of the *KDL* library. If a solution was computed successfully, it is saved to the database and the steps are repeated for each neighbor and workspace pose.

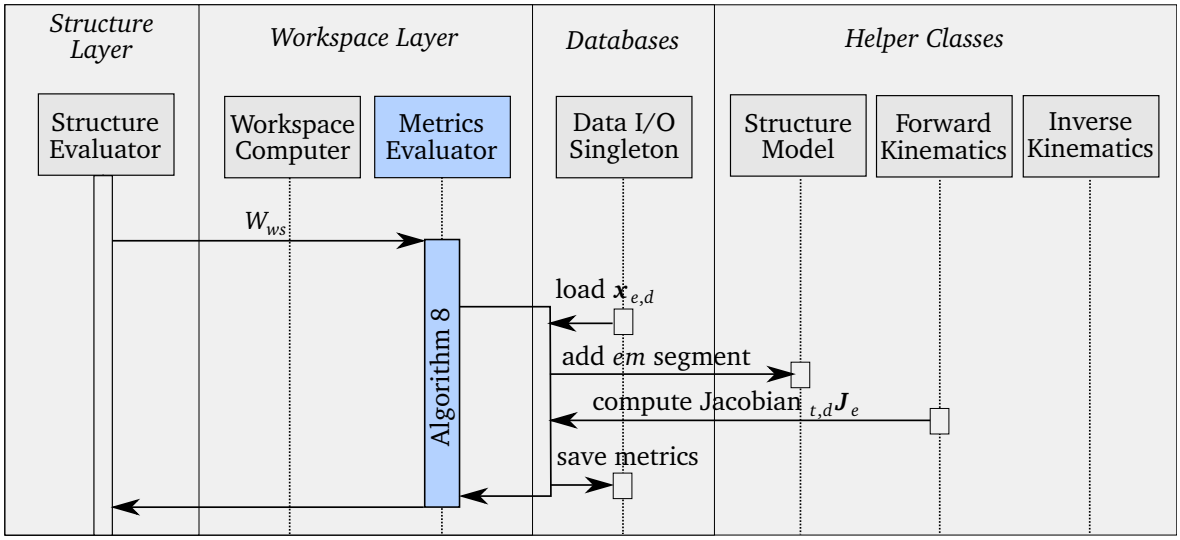

Figure 4.5: Sequence diagram with focus on the interactions of the *Metrics Evaluator*.

In Figure 4.5 the interactions of the *Metrics Evaluator* class are presented which computes the EM-specific metrics. The *Structure Evaluator* class splits the workspace poses which were generated above, into multiple sets. Again, this suits for the realization of multi threading, where one *Metrics Evaluator* instance is declared for each thread. The following sequence is based on Algorithm 8. Within the framework the algorithm induces several interactions between classes. First the particular discrete end effector specific pose $\boldsymbol{x}_{e,d}$ is loaded from the database via the *Data I/O Singleton* class. Then, the virtual *em* segment is added to the *Structure Model* class instance using the homogeneous transformation $_e\boldsymbol{T}_t$ which was defined in the *General Configuration File*. After computing the Jacobian $_{t,d}\boldsymbol{J}_e$ using the *Forward Kinematics* helper class, the metrics are computed and saved into the databases.

Based on the described computations which have determined the end effector workspaces and evaluated the EM-specific metrics, the *Structure Evaluator* instantiates a *Task Evaluator* class instance for each of the user-defined tasks. This is shown in Figure 4.6. With the definitions inside of the task description, the task area weights $w_{ta}(\boldsymbol{x}_d)$ are computed. Herefore the STL files defined in the *Task Configuration File* are processed, using the *LibIGL* and *CGAL* library in order to compute the BB-related boundaries of the specific task area. Hereby, the fluent transition from the task area to the non-task area is modeled by a linearly decreasing weight. The user hereby defines the discrete step size which represent the inflation of this transition area in all dimensions. Subsequently, the weights are used to integrate the task-related dexterity over the task area. For the computation of the task-related dexterity the *Task Evaluator* retrieves the EM-specific metrics from the *Data I/O Singleton* class instance (see Figure 4.6).

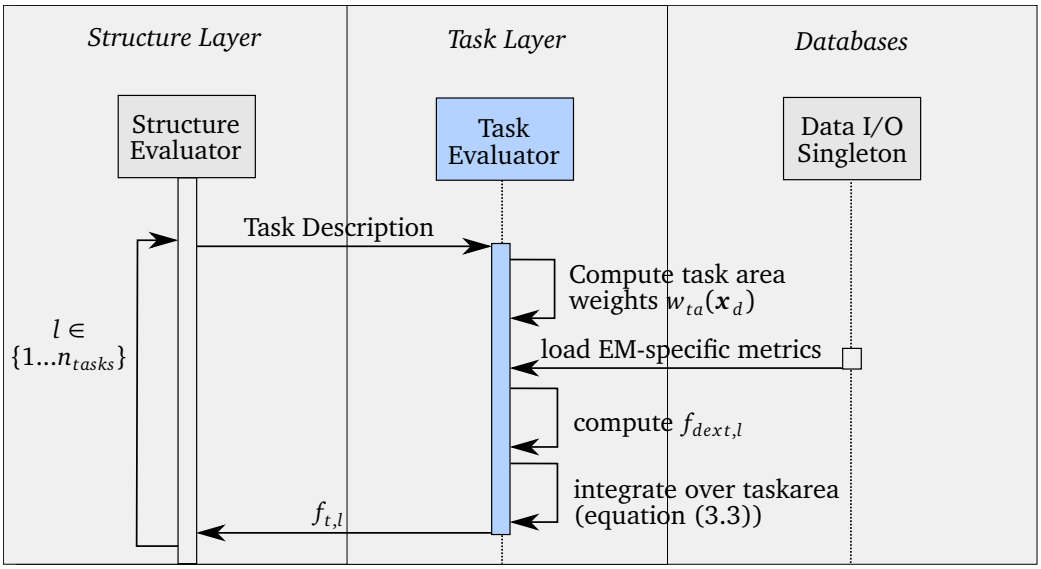

Figure 4.6: Sequence diagram with focus on the interactions of the *Task Evaluator*.

Chapter 5

Evaluation

Explications of the framework in Chapter 4 have shown that the software, due to its general configurability, facilitates a wide range of applications. In this chapter three application scenarios are therefore provided in order to evaluate the basic concept and show the applicability of the framework. Not every functionality is incorporated into the applications in order to keep the evaluation within a suitable scope. Any omitted definitions and results can be found on the storage medium which is attached to the thesis. The first application scenario is the determination of a task-optimal structure topology for a three DoF manipulator acting in the 2D plane. The robot is optimized for table top manipulation. The presentation of the results focuses on the reasoning of the dimensional optimization by discussing the development of kinematic metrics with respect to the chosen optimization parameters. Following this, the applicability of the three optimization methods PSO, SA and CMA-ES is assessed. The second application example was already introduced in Section 1.1. It analyzes a suitable new arm topology for LOLA which suits to perform in-motion stabilization contacts. In this section the focus is put on the interpretation and comparison of three suggested topologies in order to validate the concept from the application-oriented perspective. In the third application example the coupling of end effectors and the evaluation of the EM-specific shared workspace is evaluated. This is achieved by optimizing two 7 DoF robots for the execution of a dual-arm manipulation task at a conveyor belt in Section 5.4.

The chapter begins with an evaluation of the efficiency of the workspace computation and metric evaluation using the framework's implementation. It is intended to give an idea of the computation times which should help to estimate the implementation's capabilities and limitations.

5.1 Efficiency of Workspace Computation

The computational effort of a fitness evaluation is dependent on the constitution of the computer and the configuration of the framework. For a comparable analysis of the efficiency of the implementation, three structure evaluations with varying configuration parameters were performed on a commercially available laptop with two CPU cores with a basic clock frequency of 2.7 GHz. In the following Table 5.1 the computation times of one structure evaluation are demonstrated which were measured for the three application examples. The 3 DoF and 4 DoF examples relate to the application scenarios of the subsequent sections Section 5.2 and Section 5.3.

A low complex structure with 3 DoF with a low discretized workspace was evaluated in the first row of Table 5.1. It computes fast in under 2 s, using the full range of functionality,

including the HYB method. A first run of the LOLA application scenario from Section 5.3 is shown in the second row. Here, it is clearly shown that the increase of the discretization resolution and number of task-related datapoints leads to high computation times during the HYB method. The reason for this is that the HYB spends much time recurrently solving IK for workspace poses beyond the boundary of the workspace. The configuration of every neighbor is used as an initial configuration to find a feasible solution which does not exist. This leads to high computational costs. The application of the HYB method is therefore disregarded in the further course of this thesis and improvements are suggested for future work in Chapter 6.

From the fourth and fifth row of the table Table 5.1, the influence of the number of task-related discrete poses is presented for a dual arm application scenario. Two seven DoF manipulators were evaluated regarding one EM where both end effectors are coupled. In the fourth row it can be seen that the task area represents a small sub region of the workspace. The analysis of the coupled metrics takes only 6 s. However, when defining the full workspace as task-relevant, the computation time increases to 250 s. Considering the overall computational effort for the workspace computations and the overall time it is shown that even this last complex scenario can be evaluated in a reasonable amount of time. The optimization methods within this thesis have shown in test runs that the optimum is reliably identified after 100 to 300 evaluations for parameter spaces with $2-3$ parameters. This implies that this dual-arm example could be optimized in around 4 to 12 hours. This seems acceptable for a kinematic design analysis on the given ordinary computational hardware. However, complex application examples should be deployed on more capable, multi core workstations in order to make use of the software's parallelization properties.

Table 5.1: Computational execution times of one structure evaluation for various structures and distinct settings.

			Discrete poses		Workspace gen.			
Structure	**FK sampling**	**IK**	**No. in workspace**	**No. in task area**	**FK**	**IK**	**Metric eval.**	**Overall time**
3 DoF	$5 \cdot 10^5$	active	$3.2 \cdot 10^5$	$6.4 \cdot 10^3$	0.7 s	0.3 s	0.7 s	1.7 s
4 DoF	$2 \cdot 10^6$	active	$6.4 \cdot 10^6$	$8 \cdot 10^4$	8 s	1160 s	33 s	1201 s
4 DoF	$2 \cdot 10^6$	inactive	$6.4 \cdot 10^6$	$8 \cdot 10^4$	9 s	-	34 s	43 s
2×7 DoF	$2{\times}15{\cdot}10^6$	inactive	$2{\times}12.8{\cdot}10^6$	$1 \cdot 10^5$	$2{\times}70s$	-	6 s	146 s
2×7 DoF	$2{\times}15{\cdot}10^6$	inactive	$2{\times}12.8{\cdot}10^6$	$12.8 \cdot 10^6$	$2{\times}70s$	-	250 s	390 s

Efficiency of SWA

Another test was conducted to prove the efficiency of the SWA method. For a low discretization of $n_k = 100$ and $n_i = 1$ an improvement of factor 2, in comparison to the method provided in [7], was achieved. A higher discretization with $n_k = 400$ and $n_i = 6$ was approximately five times faster. Due to the fact that the orientation discretization must be conducted for each kinematic evaluation, this demonstrates that the thesis has provided a significant improvement within the implementation of the VFS scheme.

5.2 Example 1: 2D Table Top Manipulator

A first application example was created to test the functionalities of the system and evaluate parts of the concept. It deals with the optimization of a 2D manipulator that must perform table top manipulation tasks. The task scenario is presented in Figure 5.1 and is explained in the subsequenct sections. The following objectives are pursued when specifying the example and are therefore focus of this section's presentation:

- Comparability of topologies with revolute and prismatic joints.
- Plausibility of metrics for a low complex structure.
- Plausibility of the task-related dexterity formulation for a simple task description and structure.
- Applicability of the optimization methods PSO, SA and CMA-ES.

In Table 5.2 some of the configuration parameters of the *General Configuration File* for this application scenario are presented. Within each evaluation of the structure, FK is applied for 500.000 joint configuration samples. IK, using the HYB method, is not performed due to the performance issues presented in the previous Section 5.1. Each of the joint configurations is subsequently mapped to a pose within the discretized workspace. The discretization parameters are listed in Table 5.2 as well.

Table 5.2: Discretization scheme parameters (left) and general parameters (right) for the 2D table manipulator problem.

Scheme	Discrete index	Step size	Range
BB	x	0.05 m	$[0, 2.0]$
	y	0.05 m	$[0, 2.0]$
	z	0.0 m	$[0, 0]$
VFS	n_k	197	
	n_i	1	

Parameter type	Value
FK sampling	500.000 random samples
IK	inactive

Task Areas

The task area of the setting is presented in Figure 5.1. The manipulator is opted to perform manipulation tasks on top of a table which is located in a fixed distance of 1 m from the robot's base. The table possesses a width of 0.4 m and a depth of 0.2 m which represent the boundaries of the STL mesh of the task area $W_{ta,table}$ (see Figure 5.1). For the manipulation task it is assumed that the task area should represent the orientations that are required to grasp objects from the front of the table. The orientation task space is therefore defined within $-\frac{\pi}{3} <= \varphi <= \frac{\pi}{3}$. Due to the 2D properties of the example, θ must be chosen, such that all task-related orientations are located within one revolution of the spiral curve around its main axis. For the given discretization these conditions are met for θ between 1.5 rad and 1.65 rad.

Task-related Dexterity

For the given task, the robot requires a general dexterity within the task area, because the type of manipulation task is not further defined. Thus, it is striven to determine the dimensionally altered structure which provides the highest distance from singular configurations and ensures that the task poses are struck in configurations which are distant from joint limits. This is incorporated into the task dexterity by including RI, CI and JRA.

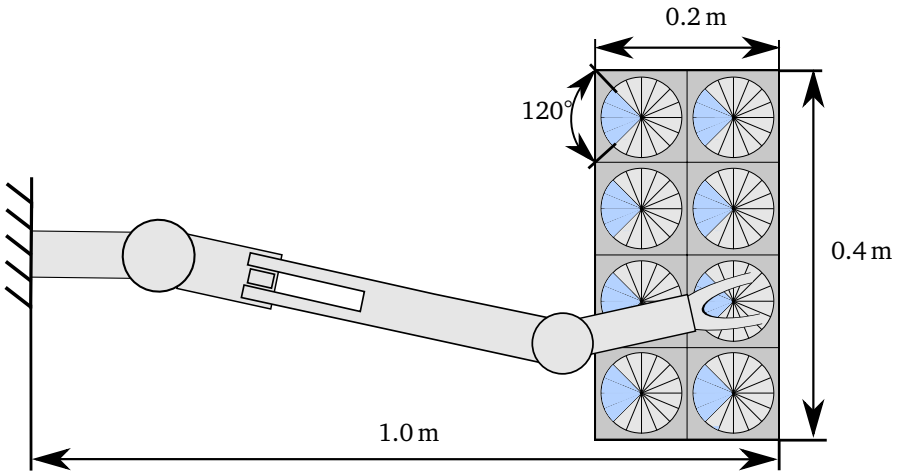

Figure 5.1: Schematics of the table task setting with a visualized exemplary topology.

Feasible Topologies

In order to evaluate the presented objectives of this example, as explained above, the three topologies are created which vary regarding the type of joint. All of them provide 3 DoF and are visualized in Figure 5.2.

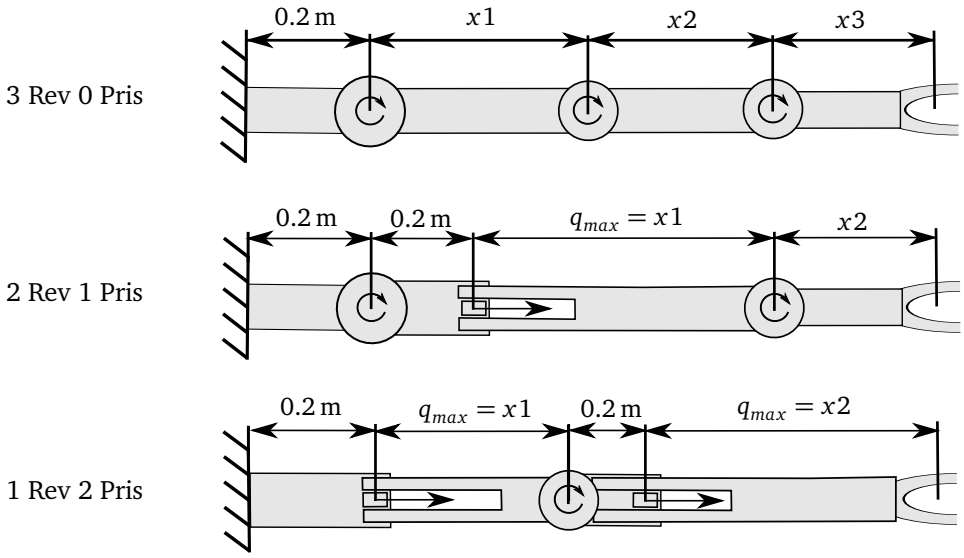

Figure 5.2: Arm topology suggestions for the 2D manipulator for manipulation tasks. The type of DoF of each joint is visualized through a descriptive arrow. Optimization parameters are identified by $x1$, $x2$, $x3$.

Each of the topologies is tagged with a name which is presented on the left side of Figure 5.2. It also represents the number of the respective joint type. From the illustration the definition of the optimization parameters $x1$, $x2$ and $x3$ can be derived. Whereas in the *3 Rev 0 Pris*

topology parameters are the link lengths and are bound to $[0.1\,\mathrm{m}, 0.6\,\mathrm{m}]$, in *2 Rev 1 Pris* and *1 Rev 2 Pris* only two parameters, bound to $[0.1\,\mathrm{m}, 0.8\,\mathrm{m}]$, are introduced. They are partly coupled to the upper joint limit of the prismatic joint(s). Further information on the structure and joint limits which are not presented in this section can be derived from the URDF model from the storage medium which is attached to this thesis.

Optimization Results

Each of the presented topologies was optimized regarding its dimensional parameters, as explained in the previous section. The results are presented in the following table.

Table 5.3: Task area related kinematic metrics and fitness results of the dimensionally optimized 2D manipulators.

		Table			Optimal parameters		
Topology	$\mathbf{f_s}$	$\mathcal{M}_{RI}$	$\mathcal{M}_{JRA}$	$\mathcal{M}_{CI}$	**x1[m]**	**x2[m]**	**x3[m]**
3 Rev 0 Pris	0.033	1	0.467	0.071	0.38	0.347	0.1
2 Rev 1 Pris	0.168	1	0.697	0.242	0.8	0.1	-
1 Rev 2 Pris	0.049	0.429	0.53	0.234	0.8	0.415	-

These results represent the best topologies which were found during several optimization runs of the three different optimization methods. In the following the applicability of the optimization methods to this sort of problem is presented. This is achieved by analyzing the optimization performance exemplarily for the *3 Rev 0 Pris* topology. However, the results are quite similar for any of the other optimizations executed for this thesis, because the solver have steadily performed in the presented way. This can be consulted from diagrams provided in Appendix B. CMA-ES, PSO and SA are capable of encountering solutions which are close to the global optimum of the problem. This can be seen from the box plot[1] diagram in Figure 5.3.

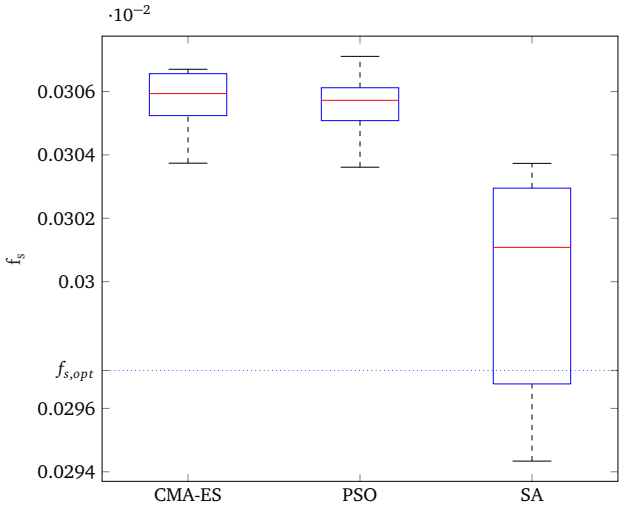

Figure 5.3: Distributions of the optimal *3 Rev 0 Pris* structure fitness determined during ten optimizations of CMA-ES, PSO and SA in a box plot[1]. The blue dotted line marks the optimal fitness structure which was determined during exploration with the *analyze_parameterspace* executable. The optimizer configuration parameters of each method are listed in Appendix B.

[1]box plot: The bottom and top edges of the box indicate the 25th and 75th percentiles. The central mark indicates the median

The diagram presents the distribution of the optimal fitness values from 30 runs of each optimization method. The horizontal line at $f_{s,opt}$ indicates the optimal fitness that was found by exploring the workspace with the *analyse_workspace* executable.

It becomes obvious that the median of all boxes is above this line within a range of 2 %. This shows that very good solutions have been found by each of the optimization methods. The box plots of the population-based CMA-ES and PSO provide slightly better and less deviated solutions than the SA. This might be caused by the random acceptance of worse parameters even in late stages of the SA which on one hand helps to avoid greater regions of local minima effectively, but also causes the acceptance of local minima close to the global optimum during the convergence phase. A different tuning of the temperature range and acceptance rate repetitions might lead to a decrease of this effect. The small deviations of the population-based methods are caused by the random joint sampling during the structure evaluation which imposes an uncertainty in the objective function of this optimization.
This effect can also be seen in Figure 5.3 where the structure fitness of each optimization method for all evaluations of one optimization run is shown.

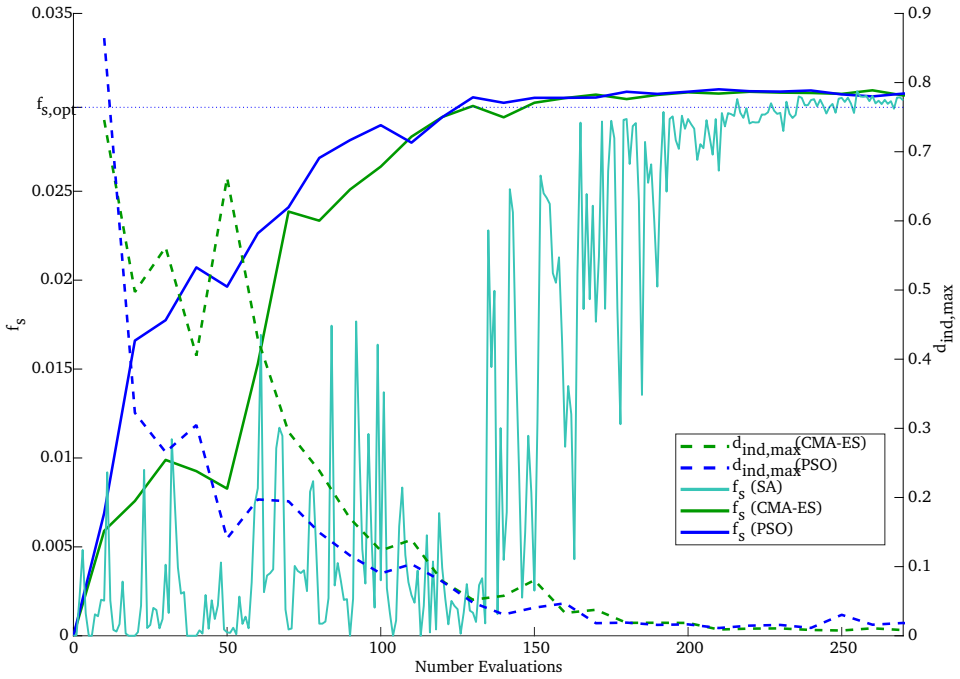

Figure 5.4: Structure evaluations of one optimization runs of each method related to the *3 Rev 0 Pris* topology problem. The fitness of SA for each evaluation is plotted. The structure fitness of (CMA-ES, PSO) represent the average fitness over all individuals within one generation. The right y-axis presents the maximal euclidean distance $d_{ind,max}$ of individual parameter vectors within the population of one generation. The optimizer configuration parameters of each method are listed in Appendix B.

The three methods obviously converge at a high structure fitness, but tumble slightly around this optimum. The convergence of the population-based approaches is also indicated by the line plots of $d_{ind,max}$ which represent the maximal euclidean distance of individual parameter vectors within the population of one generation. It is clearly shown that the distribution of the particle swarm of PSO and the shape of the normal distribution of CMA-ES shrink to almost zero which proves convergence. CMA-ES and PSO behave in a quite similar way and converge after around 170 iterations. However, SA, converges after more than 200 iterations. This is reasonable, because the convergence behavior is dependent on the temperature. During high temperatures many false acceptances are produced which are necessary to sufficiently

explore the space and evaluate the best region for further investigations. The SA related curve therefore tumbles strongly until approximately 150 iterations and starts to converge to higher fitness values.
In the following the presented results of Table 5.3 will be discussed by investigating the evolution of metrics over the parameter spaces of the *3 Rev 0 Pris* and *2 Rev 1 Pris* topologies which were generated through the *analyze_parameterspace* executable. The validity of the concept for serial manipulators with different joint types is provided. Results from the *1 Rev 2 Pris* are disregarded for the sake of brevity, because they do not provide new essential information. However, the corresponding results can be seen in Appendix B.

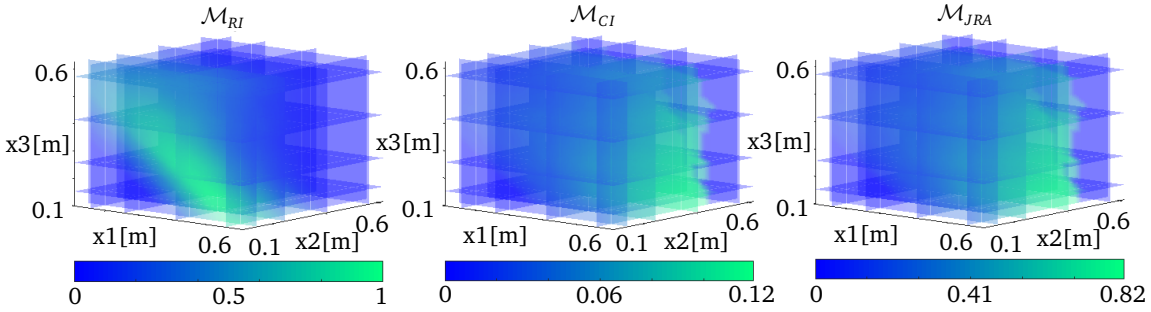

Figure 5.5: Dexterity-related metrics within the parameter space of the *3 Rev 0 Pris* topology. From left to right the RI, CI and JRA are plotted dependent on each optimization parameter $x1$, $x2$ and $x3$.

From the parameter space visualizations of the dexterity-related metrics in Figure 5.5, it can be seen that the *3 Rev 0 Pris* topology provides a high reachability in an area which cuts through the parameter space box in shape of a thick diagonal plane. This plane represents the areas where the overall length of the manipulator is closely above 1 m. The manipulator's topology only provides reachability for poses with the task-related orientations in regions close to the outer workspace boundary, because here the end effector faces away from the robot's base to the table. A very similar distribution of well-performing sets of parameters can be seen for the JRA (see Figure 5.5). If the robot can reach many of the poses in a well-stretched configuration which is provided close to the outer workspace boundaries, the joint positions of the revolute joints tend to small deviations which is reflected in the JRA. This property is also visualized in Figure 5.6 where the CI (left) and JRA (right) over the workspace of the optimal *3 Rev 0 Pris* topology is demonstrated. However, a well-strechted pose implies that the given three DoF structure is close to a singular configuration. The optimization objectives of JRA and CI can, thus, be considered as contradictive for the given topology which becomes obvious when comparing both metrics in Figure 5.6.

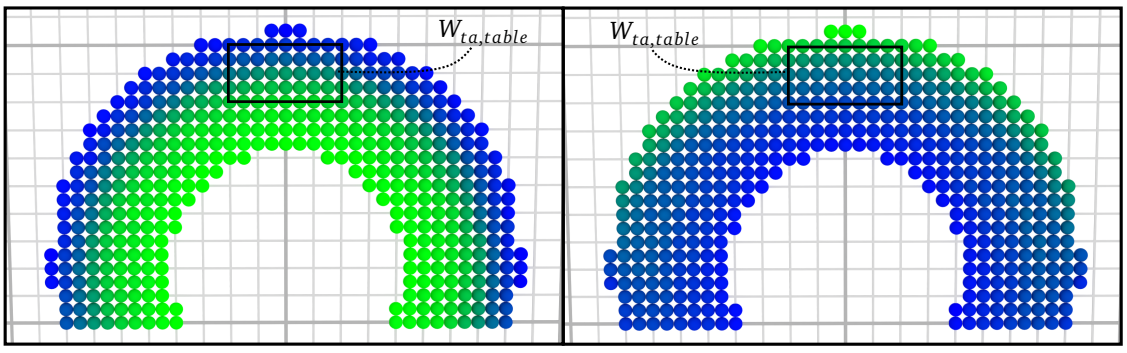

Figure 5.6: Averaged dexterity-relevant metrics of the end effector workspace of the *3 Rev 0 Pris* topology. Left: (green: $\mathcal{M}_{CI}$ = 0.11, blue: $\mathcal{M}_{CI}$ = 0.0), Right:(green: $\mathcal{M}_{JRA}$ = 0.97, blue: $\mathcal{M}_{JRA}$ = 0.0).

Due to the similar behavior of RI and JRA, the contradictive impact of the CI remains small for the structure fitness. In conclusion, the optimal solution provides full task-related reachability with a high distance from joint limits.

The second *2 Rev 1 Pris* topology is capable of fully reaching every task pose as well. Again, the variation of the three task dexterity related metrics over the parameter space is visualized in Figure 5.7.

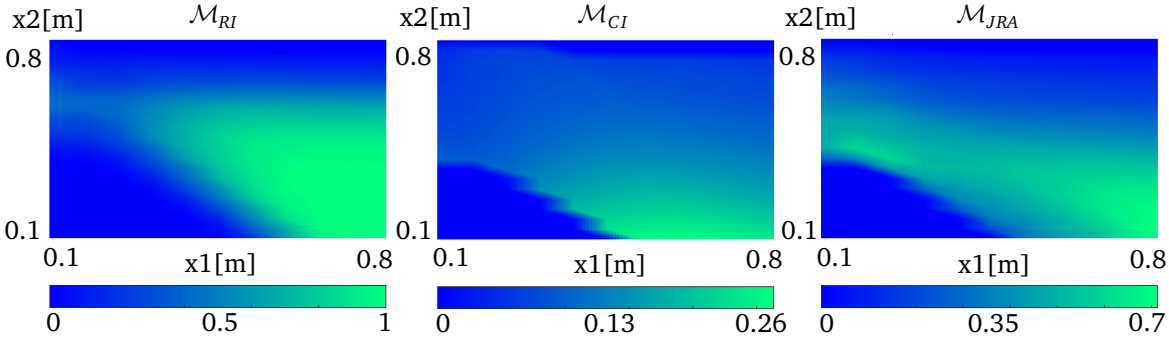

Figure 5.7: Dexterity-related metrics within the parameter space of the *2 Rev 1 Pris* topology. From left to right the RI, CI and JRA are plotted dependent on each optimization parameter $x1$ and $x2$.

In contrast to the results of the *3 Rev 0 Pris* topology, the metrics behave similarly. For instance they favor a long second link. This can be explained by considering that the increase of the second link implicitly extends the joint range of the prismatic joint and workspace, because the link length was coupled to the upper joint limit of it (see Section 5.2). Whereas the full task reachability for a long second link is provided for a length of 0.1 m to 0.5 m, a high JRA and CI is only ensured if the last link of the manipulator is chosen small.

These relations derive an optimal set of parameters which provides a well-balanced compromise between the CI and JRA for a maximal task dexterity. This compromise for the optimal structure can be derived from Figure 5.8. It can be seen that the task area is placed in the area with the highest distance to the joint limits. It is located in the middle of the workspace, because the workspace boundaries are mostly caused from the joint limits of the prismatic joint. In the same area CI provides a solid distance from singular configurations. For the given topology singular configurations are caused, if the prismatic joint comes close to its lower joint limit. In that case the length of the second link becomes zero, such that the two revolute joints are congruent and the manipulator loses one *DoF*. In conclusion, the presented optimal structure fulfills the requirements formulated for the task dexterity (see Section 5.2).

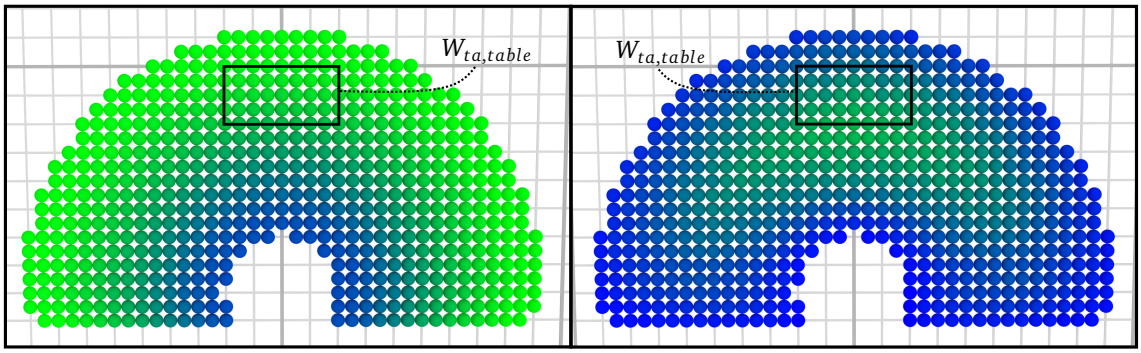

Figure 5.8: Averaged dexterity-relevant metrics of the end effector workspace of the *2 Rev 1 Pris* topology. Left: (green: $\mathcal{M}_{CI}$ = 0.31, blue: $\mathcal{M}_{CI}$ = 0.0), Right: (green: $\mathcal{M}_{JRA}$ = 0.95, blue: $\mathcal{M}_{JRA}$ = 0.0).

5.3 Example 2: LOLA Single-Arm Stabilization

As presented in the introductory Chapter 1, a modification to the arm topology of the humanoid LOLA is supposed to be designed which is the key objective of this example. Due to current research work, the new arm should not only produce counterbalancing movement, but, additionally, provide the ability of producing contact with the environment for stabilization (see Chapter 1). Stabilization in general means, that a force is produced from the environmental contact which helps to maintain or restore the positioning of the center of gravity.

Five restrictions are made in consultation with the research work:

- Stabilization contacts are only generated on the side of the robot, because LOLA should only stabilize in walking state.
- Lola uses the end effector to push itself away from the contact point.
- The current end effector does not possess any DoF to realize grasping or other complex environmental interactions (see [5]). Rotations around the major axis are therefore irrelevant.
- The type of objects which are used for the stabilization tasks are limited to railings, walls and tables. Their importance for the stabilization task are considered equal.
- Both arms are designed and built symmetrically. The optimization of one arm is, thus, representative for both arms.

Due to the differences regarding geometry and location of the objects, a subtask for each environment object is defined. Based on these assumptions three feasible arm topologies have been dimensionally optimized. They are further presented in the following subsections, as well as the task description and a discussion about the results. In Table 5.4 the discretization and kinematic parameters are listed in order to estimate the realization of the results. The number of joint samples is set to two million. More samples did not provide any change in the structure evaluation. Further settings can be found within the corresponding *General Configuration File* on the storage medium of this thesis.

Table 5.4: Discretization scheme parameters (left) and general parameters (right) for the LOLA problem.

Scheme	Discrete index	Step size	Range
BB	x	0.05 m	[0, 2.0]
	y	0.05 m	[0, 2.0]
	z	0.05 m	[0, 2.0]
VFS	n_k	197	
	n_i	1	

Parameter type	Value
FK sampling	2.000.000 random samples
IK	inactive

Task Areas

The positional space of the railing stabilization task is defined by a box shaped STL mesh. In Figure 5.9 the dimensions of the box can be assessed. Due to German standards, defined in *DIN 18065*, railings in German institutions must provide a minimum height of 0.9 m for a drop height of under 12 m. A maximum height of railings which serves stabilization purposes for humans does usually not exceed 1.20 m (compare Figure 5.9). An important objective for LOLA is to have human-like abilities. A human usually touches a railing which is located in a distance of 0.3 m and 0.6 m from the human hip. These measures therefore define the lateral box boundaries. The task-related orientation space is represented by upper hemisphere which is visualized through the green surface area on the VFS spheres based on the visualization capabilities of the framework. Any end effector orientation which pierces through the VFS sphere from above can create a stabilization force on the railing. Figure 5.9 shows a task-related feasible configuration of the current arm of LOLA.

The same orientation space is defined for the table's task area which is also demonstrated in Figure 5.9 on the right hand side. Tables can have various geometric properties. Again, a box is chosen to mark the task-related positions. The minimum table height for workspaces in Germany is used as the lower limit. It is assumed that LOLA could even operate in a scenario where standing tables are present. The maximum height is assumed regarding a research on the highest standing tables available on the market with approximately 1.50 m. The distances of a table towards the robot is considered with the same dimensions as of the railing task area.

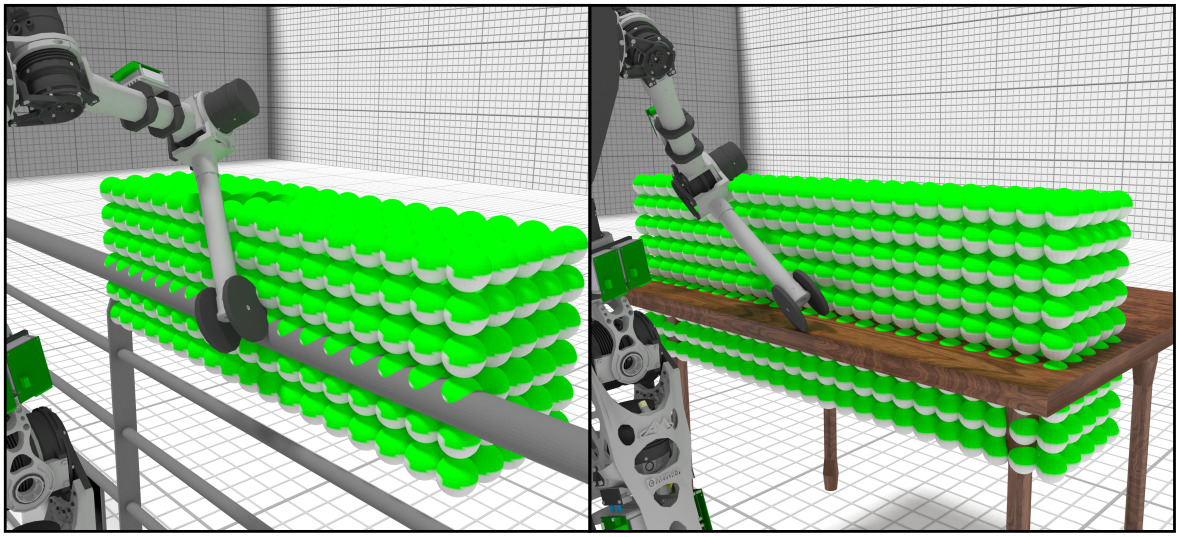

Figure 5.9: Distribution of the task area weighting factor $w_{ta,railing}$ of the railing task (left) and the table task (right). Both task area meshes were bisected prehand in order to show feasible configurations of the current LOLA arm.

For the consideration of wall stabilization, the task area must be divided into a higher area and a lower area. Both task areas are visualized in Figure 5.10. Again the meshes were bisected in order to descriptively visualize a configuration of the current LOLA arm which is considered task-relevant. Due to the suggested topologies from Section 5.3 and restrictions regarding the arm length, high areas of the wall can only be reached with the end effector from below. The orientation space of the upper part is thus characterized by the lower semi hemisphere which faces away from the wall towards the robot. The opposite is present when LOLA is supposed to use lower parts of the wall for stabilization. Here, the upper semi hemisphere models that the end effector can only create a contact with the lower part

of the wall, coming from above (see Figure 5.10). Neither of both restrictions holds for the medium high segments of the wall. In conclusion, the two defined task areas overlap regarding their position in the middle sector of the wall. Using the task area weights for a smooth transition to non-task areas, a transition from the lower semi hemisphere, via the full hemisphere to the upper semi hemisphere is realized through both task areas. This relation can be comprehended better by observing the two task areas from Figure 5.10.

Regarding the positioning of both task areas, again, two boxes are defined. The distance from LOLA's hip to a wall where stabilization can be performed, is defined between 0.3 m and 0.7 m. While the lowest reasonable contact points on the wall are located in height of LOLA's knees, the highest points are defined at a height of 1.8 m. The overlapping region is quantified by one third of the overall height of the combined areas.

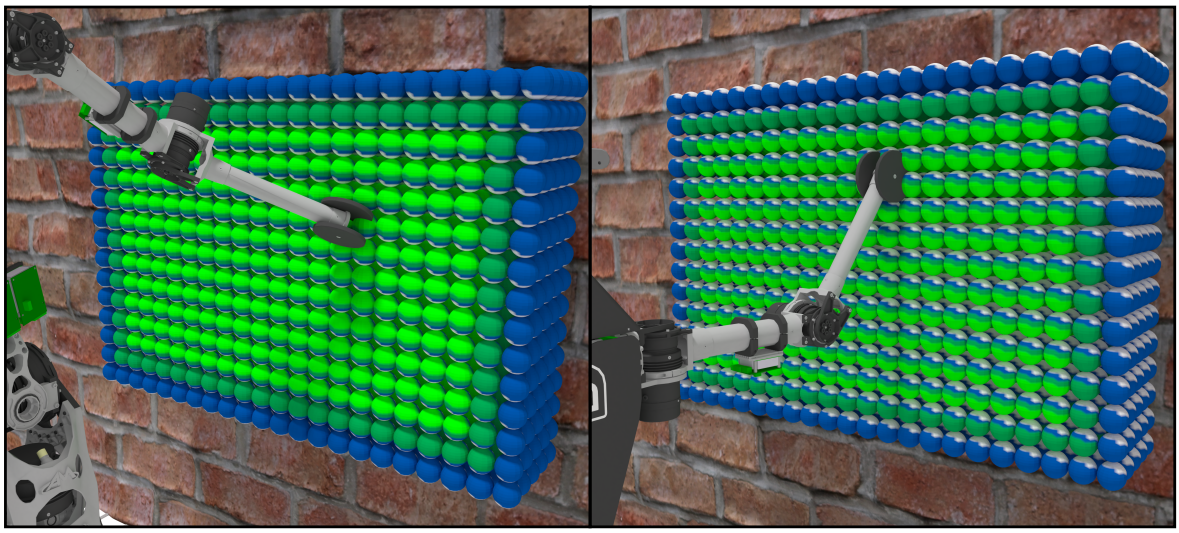

Figure 5.10: Distribution of the task area weighting factor $w_{ta,wallLow}$ for the lower part of the wall (left) and the higher part $w_{ta,wallHigh}$ (right). The task area meshes were bisected in half to descriptively demonstrate the weight distribution.

Task-related Dexterity

As in most task descriptions, a high reachability within the defined task areas is desirable. However, a reachable task pose is only considered dexterous, if the configurations provide a sufficient distance to joint limits and singular configurations which is quantified with the JRA and the CI. Additionally, a good configuration can generate a velocity in the negative walking direction with ease. This is an important criterion, because the robot should provide steady contact which can only be facilitated if the given joint space configuration can counteract the movement of the upper body. For this reason the VTR is incorporated into the dexterity formulation with the maximal opening angle $\nu_{max} = 0.3\pi$. Due to the symmetrical geometry of the end effector, the translational velocity transmission invariant to the rotational velocity is considered for the VTR (see Section 3.3.2).

Feasible Topologies

The current arm topology of LOLA is presented in Figure 5.11. Three new arm topologies were classified as mechatronically feasible in consultation with the research team of LOLA.

- *Arm-1*: A revolute joint is inserted into the upper arm with the joint axis aligned to the arm axis.
- *Arm-2*: A prismatic joint is inserted into the upper arm with the joint axis aligned to the arm axis.
- *Arm-3*: A prismatic joint is inserted into the lower arm with the joint axis aligned to the arm axis.

With the addition of one extra DoF it is expected that the workspace increases sufficiently to provide a suitable workspace for the stabilization task with a low effort regarding the redesign of the arm. A visualization of the topologies is presented in Figure 5.11. It also contains the definitions of the optimization parameters and their lower and upper bounds. They are expressed in arm-related measurements which are presented for the *Current Arm*. $d_{shoulders}$ hereby represents the distance between both shoulder joints, while l_{upper} and l_{lower} denote the upper arm length and the lower arm length, respectively.

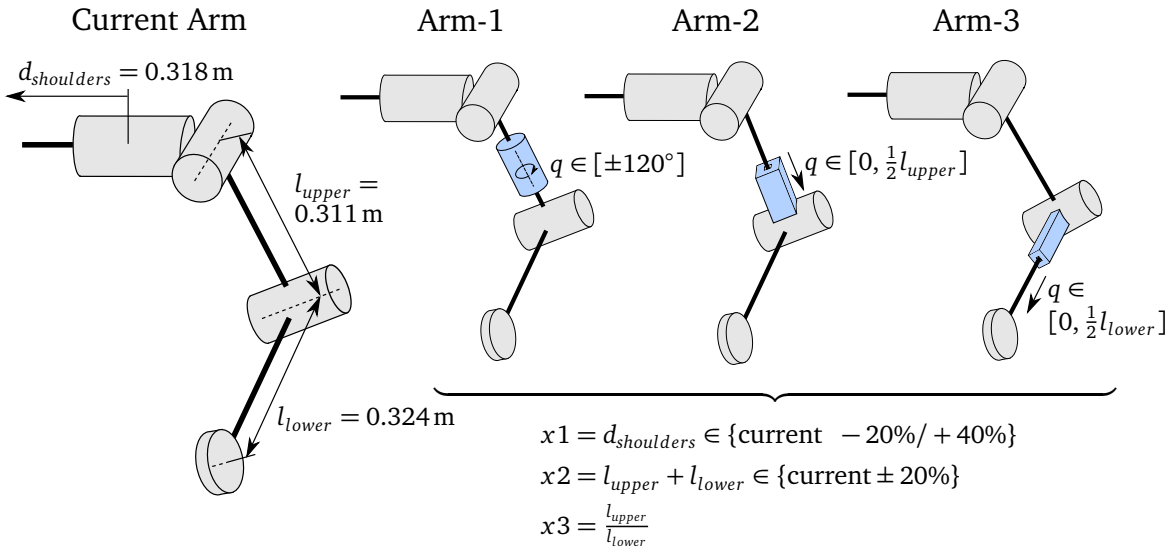

Figure 5.11: Schematic visualization of the current arm topology with significant measurements and arm topology suggestions for the new LOLA arm for stabilization. The DoF of each added blue-marked joint is visualized through an arrow and the optimization parameters $x1$, $x2$, $x3$ are defined relating to the current arm measurements.

Optimization Results

After dimensionally optimizing each of the suggested new topologies regarding the given problem definition, the following structure fitnesses and task area averaged kinematic metrics for each specific topology were obtained (see Table 5.5). f_{s-RCJV} denotes to the structure fitness based on the task dexterity from Section 5.3.

Each of the suggested topologies leads to an improvement of the structure fitness in comparison to the current arm of LOLA, as expected. The second topology with a prismatic joint

Table 5.5: Task area related kinematic metrics and fitness results of the dimensionally optimized arm topologies for LOLA.

			Railing				Wall low			
Topology	$\mathbf{f_{s-RCJV}}$	$\mathbf{f_{s-RCJ}}$	$\mathcal{M}_{RI}$	$\mathcal{M}_{CI}$	$\mathcal{M}_{JRA}$	$\mathcal{M}_{VTR}$	$\mathcal{M}_{RI}$	$\mathcal{M}_{CI}$	$\mathcal{M}_{JRA}$	$\mathcal{M}_{VTR}$
Current	0.001	0.008	0.028	0.467	0.41	0.167	0.047	0.47	0.408	0.139
Arm-1	0.004	0.018	0.131	0.223	0.411	0.175	0.334	0.213	0.453	0.152
Arm-2	0.016	0.039	0.115	0.45	0.41	0.426	0.309	0.478	0.408	0.447
Arm-3	0.014	0.027	0.087	0.504	0.374	0.513	0.197	0.49	0.369	0.546

	Wall high				Table				Optimal parameters		
Topology	$\mathcal{M}_{RI}$	$\mathcal{M}_{CI}$	$\mathcal{M}_{JRA}$	$\mathcal{M}_{VTR}$	$\mathcal{M}_{RI}$	$\mathcal{M}_{CI}$	$\mathcal{M}_{JRA}$	$\mathcal{M}_{VTR}$	**x1[m]**	**x2[m]**	**x3**
Current	0.072	0.609	0.476	0.138	0.017	0.397	0.402	0.159			
Arm 1	0.335	0.168	0.46	0.15	0.133	0.237	0.425	0.156	0.256	0.728	0.9
Arm 2	0.382	0.612	0.422	0.338	0.094	0.394	0.402	0.517	0.256	0.84	0.9
Arm 3	0.287	0.641	0.407	0.489	0.062	0.4	0.375	0.581	0.256	0.84	0.48

within the upper arm is favored over all of the others with a factor of 3.3 to 3.6. The result, however, clearly shows a limitation of the concept. *Arm-2* is classified more than five times better than *Arm-1*. This does not reflect the actual discrepancy regarding "dexterity" between both topologies, because *Arm-1* provides a much higher reachability in all of the task areas. For the table task for instance, more than twice as many task poses were reachable compared to *Arm-2* (see Table 5.5). The reachability is obviously not represented intensively enough in the task-related dexterity composition of this example. The optimizer should always favor a robot which provides a high reachability and a low dexterity, compared to a robot with low reachability and high dexterity.

Another limitation of the conceptual composition refers to the divergence of the VTR metric for differing topologies. From Table 5.5 it can be shown that the VTR of *Arm-3* for the table task is more than six times better than for *Arm-1* which has a correspondingly great impact on the structure fitness and leads to the disregard of RI within the structure fitness. One reason for this is that the VTR metric is not normalized by a suitable characteristic variable, yet. This complicates the comparability between topologies with a distinct number of DoF. Another reason can be found when comparing both prismatic topologies (*Arm-2* and *Arm-3*) with the revolute topology of *Arm-1* regarding the VTR (see Table 5.5). This discrepancy regarding the VTR might also be caused by the homogenization of the end effector Jacobian matrix, because the scaling of the translational velocity transmission of revolute joints affects the VTR strongly.

The two limitations propose an important task for future work, as explained in Chapter 6. Due to the given difficulty regarding the VTR, the task-related dexterity is reformulated to compose RI, CI and JRA which is denoted by f_{s-RCJ} (see Table 5.5). However, it is emphasized that the result must be interpreted carefully, because the disregard of the RI can occur for specific structures and an adverse distribution of the metrics over the workspace.

Using the new dexterity definition, the dimensionally optimized topologies were computed successfully through the three available optimization methods. The optimal parameters for each topology can be found in Table 5.5. From the results of the new structure fitness f_{s-RCJ}, it can be clearly seen that *Arm-2* is considered most dexterous. This result comes from two major influences. The first influence can be well shown by comparing *Arm-1* to *Arm-2*. Both

optimized topologies provide a high reachability within the task areas. While only slightly more than 10 % of the railing and table task poses can be reached, stabilization using the wall yields as decent RI of around 30 % to 38 % for the lower areas and higher areas of both topologies. A very similar performance is also provided with the JRA for both topologies in all task areas. The preference of *Arm-2* over *Arm-1* by a factor of 2 comes from the CI. *Arm-2* is much closer to singular configurations by a factor of 2 to 3.6 regarding the CI. The reason for this is descriptively visualized in Figure 5.12 where the CI of the optimal *Arm-1* within the lower task area is shown. The red marked areas represent the task-relevant poses which were not reached, as explained in Section 4.2.

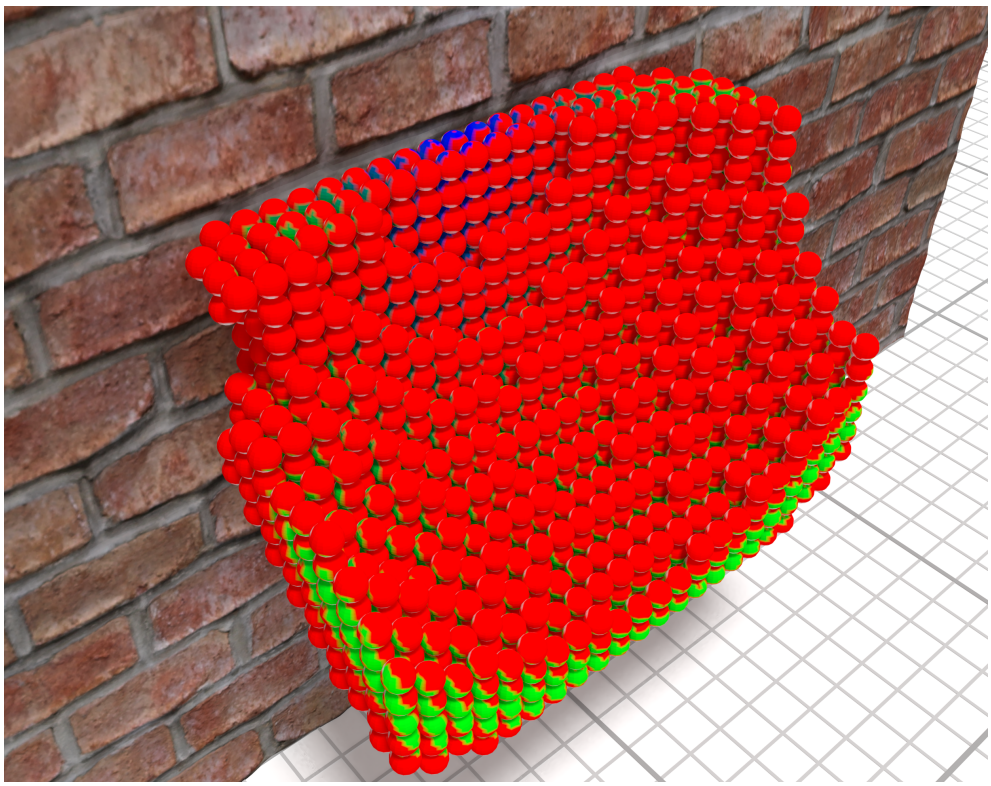

Figure 5.12: CI of the optimized topology *Arm-1* in the low wall task area (green: $CI = 0.27$, blue: $CI = 0.0$).

It becomes obvious that the outer workspace boundary cuts through the task area. Here, the *Arm-1* topology loses a *DoF* which can be clearly seen at the upper poses right in front of the visualized wall where the robot is in a fully stretched configuration and the CI drops to zero which is marked by the blue areas of the spheres (see Figure 5.12). There, the shoulder joint aligns with the newly imposed elbow joint. In conclusion, *Arm 2* is more dexterous than *Arm 1*, because it provides a higher distance from singular configurations.

The second influence which explains the preference of *Arm 2* over *Arm-3* is induced by the higher reachability of *Arm 2* within the task areas. The RI is higher by a factor of approximately 1.3 to 1.6 for all of the stabilization related task areas which is the reason why it possesses a higher structure fitness f_{s-RCJ}. *Arm 2* is, thus, favorable over *Arm-3*.

In conclusion, *Arm 2* outperforms the other two dimensionally optimized topologies, because it possesses a high dexterity regarding RI, CI and JRA. This is well represented by the structure fitness. The gradation in comparison to the other two topologies is reasonable, because *Arm-1*

provides a low CI and is, thus, half as dextereous as *Arm 2* while *Arm-3* provides an only slightly worse reachability which leads to a fitness value in between the other two topologies.

This example has shown that concept and implementation can be applied to a humanoid application scenario with multiple tasks areas. The capability of defining smooth transitions between task areas was successfully applied, as well as the presentation of the visualization techniques. If the composition of metrics is performed carefully e.g. respecting limitations of the VTR metric formulation, a dexterous structure could be found for human applications.

5.4 Example 3: Dual-Arm Manipulation

In the third application scenario of this thesis the optimal placements of two 7 DoF collaborative manipulators for a dual-arm manipulation task at a conveyor belt are determined. Both manipulators are *Panda* robots by the company *Franka Emika* [61]. The scenario is further explained in the subsequent subsection and in Figure 5.13. It is constructed and analyzed in order to evaluate the

- task-related coupling of end effectors
- applicability of the VTR metric

Consequently, the subsequent analysis focuses on the evaluation of these aspects and disregards aspects of the optimization process which are evaluated in the application examples of Section 5.2 and Section 5.3. In Table 5.6 the discretization and kinematic parameters of the optimization problem are listed. As in Section 5.2 and Section 5.3 the orientation discretization around the end effector's major axis is disregarded, because the *Panda* robots possess a revolute joint with its axis aligned with the end effector's major axis.

Table 5.6: Discretization scheme parameters (left) and general parameters (right) for the dual-arm manipulation problem.

Scheme	Discrete index	Step size	Range
BB	x	0.05 m	$[0, 2.0]$
	y	0.05 m	$[0, 2.0]$
	z	0.05 m	$[0, 2.0]$
VFS	n_k	197	
	n_i	1	

Parameter type	Value
FK sampling	2x7.500.000 random samples
IK	inactive

Task Definition

The task scenario is visualized in Figure 5.13. The two *Panda* robots are positioned on both sides of the conveyor belt rotated by 180° to each other. During the task execution they are supposed to lift an object while the conveyor belt is moving and stack the object on top of another. Lifting and stacking is performed by both end effectors in a coupled state. It is assumed that the grasping contacts at the object are in a distance of 10 cm and both end effector major axes are congruent in the grasping poses. This requirement implies the end effector coupling for the task-related EM which is shown in Figure 5.13. An exemplary

feasible dual-arm pose for the given task is shown with the visualization of the coupling transformation to the task-related pose $\boldsymbol{x}_{t,d}$.

Due to variances in the positioning of the task-related objects on the conveyor belt during operation, the object center is assumed to be located within a maximum distance of 0.125 m of the conveyor belt's transport axis (see the dashed line in Figure 5.13). The two robots must fulfill the dual-arm manipulation task within a segment of 0.5 m of the conveyor belt. Therefore, they are positioned in the middle of this segment. Another task-related requirement is defined by the maximum height of the lifting sequence with 0.4 m. The conveyor belt is 0.7 m high. Thus, the task area is modeled as a box with the dimensions 0.5 m x 0.25 m x 0.4 m. Task-relevant orientations form the half of the VFS sphere which faces the opposite transport direction (see Figure 5.13). The other half of the sphere is not part of the task area, because a crossover of the two robots is not desired for the manipulation task.

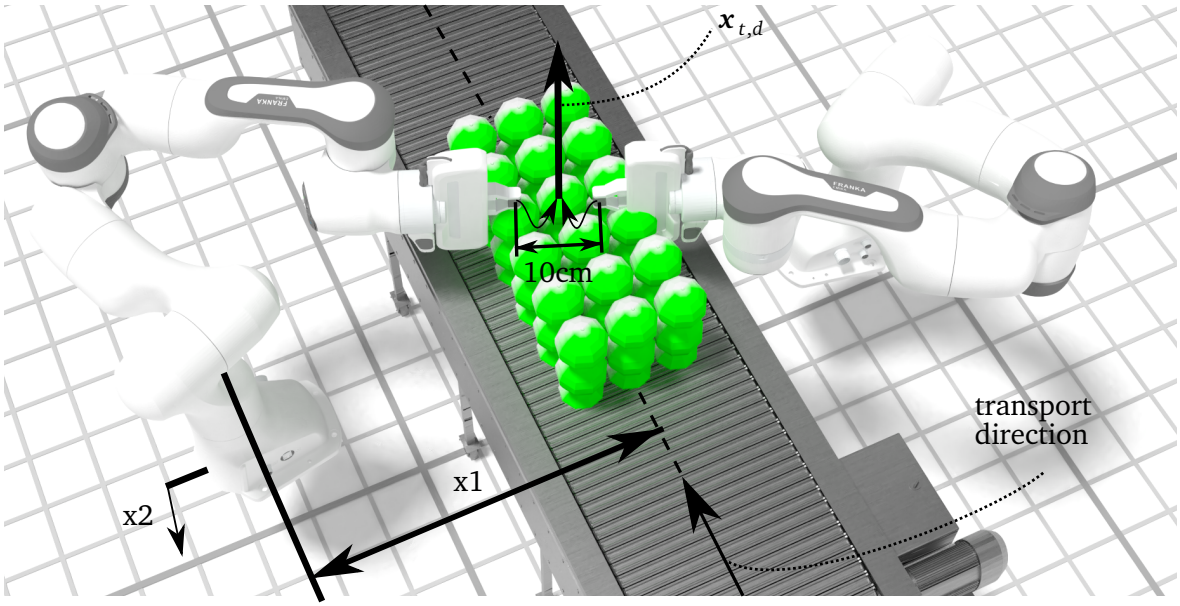

Figure 5.13: Visualization of the task scenario. The spheres mark the distribution of the task area weighting factor $w_{ta,conveyor}$. Further, the optimization parameters of the robots' placement are presented, as well as a feasible dual-arm configuration which describes the task-related EM coupling.

Task-related Dexterity

For the dual-arm manipulation task the RI and VTR are used to describe the task-related dexterity. Besides the reachability of the task area in the EM-specific coupled state of the end effectors, the robots should provide a good velocity transmission in transport direction. This is assumed to be an important factor of the task-related dexterity, as the robots must lift and stack the objects while they are moving on the conveyor belt. For the maximum opening angle of the hyper spherical cap $\nu_{max} = 0.3\,\text{rad}$ is assumed. Other metrics like the CI and the JRA are not regarded because of the evaluation focus of this application scenario.

Optimization Parameters

For the optimal placement of the two *Panda* robots two parameters must be determined which apply to both robots in the same way in order to obtain a symmetrical placement. They are presented in Figure 5.13. The distance of each robot's basis to the conveyor belt

transport axis is denoted by $x1 \in [0.25\,\text{m}, 1.25\,\text{m}]$ and the height by $x2 \in [0.4\,\text{m}, 1.0\,\text{m}]$. For the optimization problem the minimum height is set to 0.4 m. This avoids a positive evaluation of many joint configurations during workspace computation which are actually in collision with the conveyor belt. For low basis positioning joint configurations exist where *Panda* pierces through the conveyor belt from below. The consideration of only collision-free joint configurations during the workspace computation is a future task which is mentioned in Chapter 6.

Optimization Results

Due to the higher requirements of the optimization problem, only four optimization runs with CMA-ES have been executed. The optimal parameters with the corresponding task and structure fitness are shown in Table 5.7. Futher details on the results from the optimizations can be found in Appendix B.

Table 5.7: Task area related kinematic metrics and fitness results of the best task-related placement of the two *Panda* robots.

		Conveyor Belt		Optimal parameters	
	$\mathbf{f_s}$	$\mathcal{M}_{RI}$	$\mathcal{M}_{VTR}$	**x1[m]**	**x2[m]**
Best Placement	0.420	0.708	0.594	0.536	0.400

The reason for the optimal set of parameters is given in the following. In Figure 5.14 the evaluation of the task-related RI, VTR and the structure fitness are visualized over the parameter space.

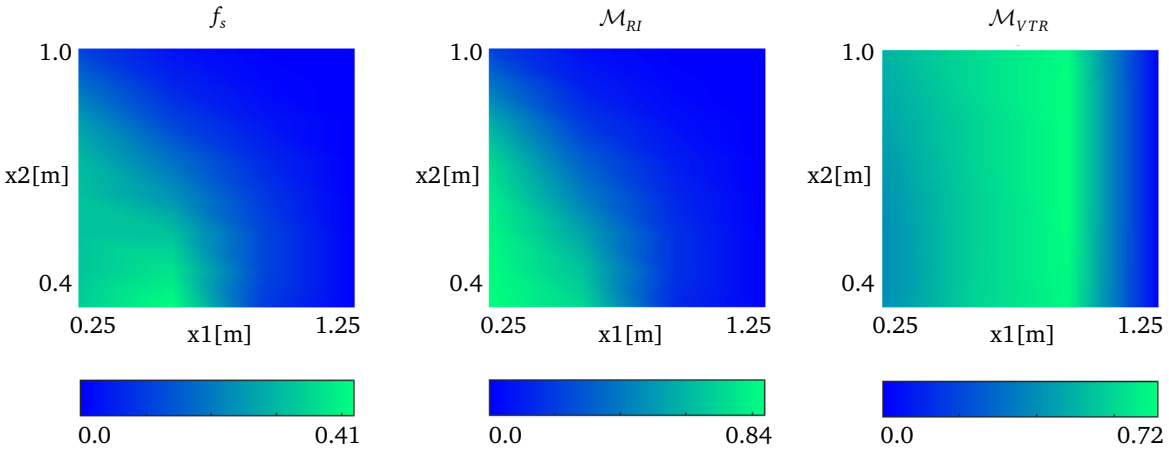

Figure 5.14: Structure Fitness and dexterity-related metrics within the parameter space of the dual-arm problem. From left to right the structure fitness, RI and VTR are plotted dependent on each optimization parameter $x1$ and $x2$.

The maximum reachability can be achieved when the robots are as close as possible to the conveyor belt in a low position (see Figure 5.14). In this position the robots align well reachable regions of their workspaces with the task area. With an increasing distance from the conveyor belt, the RI is decreased. This is presented in the visualization of the EM-specific shared workspace of both robots for three different distances $x1$ in Figure 5.15.

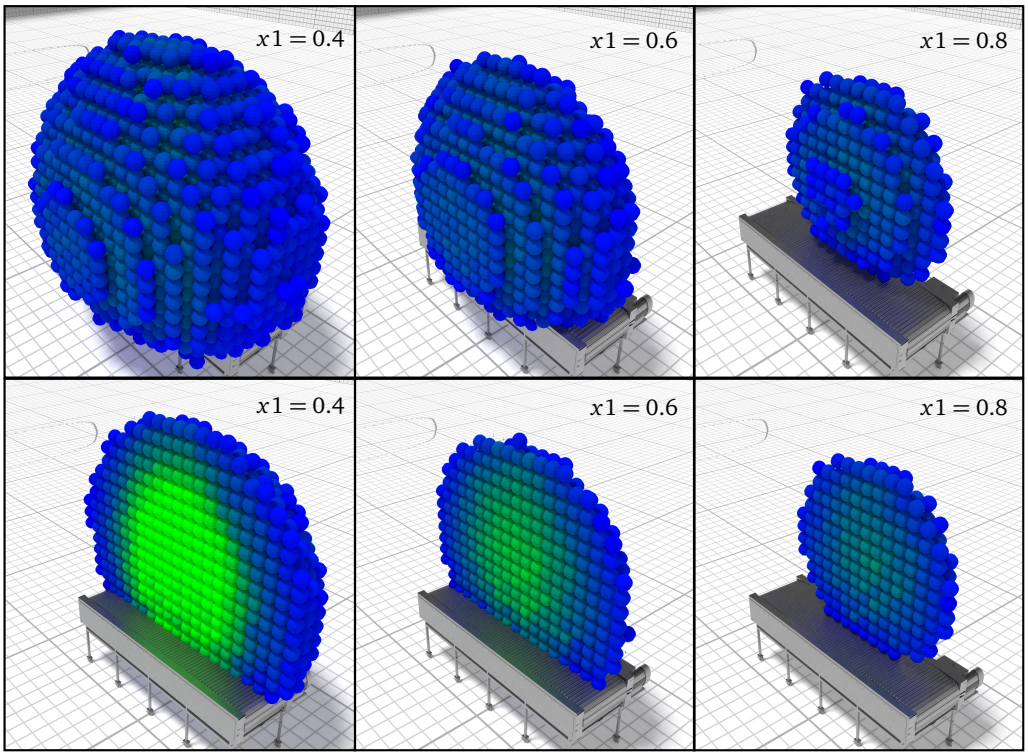

Figure 5.15: Size and shape of the EM-specific shared workspace of both manipulators dependent on $x1$. The position-averaged RI is represented in all of the subfigures (green: $\mathcal{M}_{RI}$ = 1.0, blue: $\mathcal{M}_{RI}$ = 0.0). The columns from left to right relate to $x1 = 0.4, x2 = 0.6, x3 = 0.8$. The meshes of the lower row visualizations are sliced in half with respect to the conveyor belt transport axis.

In contrast, a high VTR can only be achieved if the task-relevant joint configurations provide a high lever with respect to the task direction. This can be well observed from Figure 5.14. Whereas the height of the robots does not affect the average velocity transmission, the distance to the conveyor belt is decisive. It reaches its maximum when the robots are in an almost stretched configuration at $x1 = 0.8$, because in this position the robot provides a great lever for the task-related velocity transission. These results are reasonable, because the *Panda* robot can reach poses on the side of its bases in a maximum distance of 0.855 m, excluding the EM-specific translation [71].

The explanations above lead to comprehension of the course of the structure fitness f_s in Figure 5.14. The optimum at $x1 = 0.536\,\text{m}$ represents a well-balanced compromise of both metrics of the task-related dexterity. Still, 70 % of maximum achievable 84 % of the task-relevant poses can be reached by the dual-arm system. This loss is compensated by a higher VTR which allows the robot to produce velocities in the task direction better than for parameter sets with a higher RI.

In conclusion, the VTR metric proves to be meaningful for tasks where the robot must generate velocities in the task-relevant directions. Due to the limitations in Section 5.3 the VTR should be primarily applied to optimization problems where similar topologies are compared or only one topology is dimensionally optimized. Further, the application example has shown that the evaluation and optimization of shared workspaces can be achieved using the EM-based dexterity formulation and evaluation proposed in this thesis.

Chapter 6

Conclusion and Future Work

This thesis has provided a new methodology for the kinematic structure optimization of humanoid robots through the formulation and optimization of involved task-related objective functions. In contrast to related research work, this concept allows to statically and dimensionally optimize structures for task-related kinematic dexterities due to the composition of kinematic metrics. The definition of positions and orientations define task-relevant areas and are included in the objective function as additional task properties. Thus, the robot can be optimized to provide distinct task-specific kinematic dexterities in various relevant task areas. This enables to determine an optimal new arm structure for the humanoid robot *LOLA* for stabilization tasks at walls, railings and tables (see Section 5.3). Further, the methodology regards that robots with multiple end effectors can use at least one of them for the execution of tasks. Therefore, the objective function considers the task-specific coupling relation of the end effectors during the evaluation of the kinematic metrics. This has facilitated the optimization of the positioning of two *Panda* robots for a dual-arm manipulation task at a conveyor belt.

In the following, the findings of this thesis are discussed in greater detail regarding the specifics of workspace computation, evaluation and discretization as well as the applicability of optimization methods. In order to provide a comprehensible structure to the reader, the key objectives that were formulated in the Outline (Section 1.1) of this thesis are repeated and evaluated regarding their implementation.

The first objective relates to the efficient computation of the humanoid robot's workspace. With the BB and VFS two effective discretization schemes are applied in this thesis. The VFS allows to disregard orientations around the end effector's major axis which is beneficial for structures like in the application examples in Chapter 5. Whereas the end effector of *LOLA's* arm is rotationally symmetrical, the *Panda* robots possesses a revolute DoF around the end effector's major axis. In both cases the scheme allows to reduce the size of the discrete workspace and implicitly provides a faster workspace evaluation. Due to the proposal of the SWA algorithm a significant improvement in the performance of the VFS discretization mapping has been achieved in contrast to implementations within the literature (see Section 5.1). The workspace computation with the modified HYB method, using discrete neighbor poses, conceptually presents an efficient approach. However, the implementation of the method indicates a significant problem. The algorithm spends too much time at the positional and orientational workspace boundaries, recurrently intending to find a feasible solution for the same workspace pose from every neighbor (see Section 5.1). Due to this, the efficiency of the HYB method must still be evaluated. A solution to this issue is suggested in the subsequent section about future work. Nevertheless, an efficient way of computing a humanoid's workspace has been presented.

The second objective in the outline of this thesis is defined by the formulation of cost functions for user-defined tasks. With the methodology various task dexterities can be formulated. Whereas the composition of the RI, CI and JRA has successfully been proven for a 2D manipulator for table top manipulation and the new LOLA arm, the additional incorporation of the VTR metric shows difficulties in the composition of metrics in form of a simple product (see Section 5.3). On the one hand the reachability of the robot is not regarded enough in the objective function. However, it represents the robot's most important dexterity and should be represented more intensively in the task-related dexterity. On the other hand, the comparison of different topologies can lead to undesirable conclusions, because the metrics can behave distinctly for different topologies. The VTR for example showed a very different range for topologies with prismatic joints, compared to a topology with only revolute joints which was explained in Section 5.3. Nevertheless, the applicability VTR in combination with the RI is proven in the dual-arm manipulation in Section 5.4. The VTR represents a valuable addition to the kinematic metrics introduced in the literature, because many tasks require that the robot can generate velocities in a specific direction. Another aspect regarded in the formulation of user-defined cost functions is the evaluation of the metrics with respect to the coupling of end effectors as mentioned above with the concept of EM. Due to the definition of EM-specific end effector transformations, the workspace of each end effector must only be computed once. This saves computational resources for most of the optimization problems with multiple tasks and various EM. The concept of EM has successfully been shown in the dual arm manipulation task in Section 5.4. It also confirms that the proposed coupled representations of the RI and the VTR metric are suitable. However, for the MM a conservative formulation of the intersection ellipsoid is chosen which provides less information than the CI.

In conclusion this thesis offers the ability to formulate a task-related dexterity-based objective function which relates to the second objective of the Outline.

In the third and fourth objective of Chapter 1, requirements for the implementation are defined. The implementation of the software framework has shown that the three metaheuristic optimization methods PSO, SA and CMA-ES effectively explore the parameter space and are applicable to the evaluation scenarios described in Chapter 5 of this thesis. which relates to the third objective. Switching and configuring the optimization methods is implemented straight forwardly through a configuration file. Not only the configuration of the numeric optimization methods is realized through configuration files, but all of the software's settings and the structure model which provides the required configurability of the fourth key objective. The framework hereby provides a decent visualization of various aspects of the structure evaluation (see Section 4.2). It resembles the properties of the discretization scheme and was successfully used for the evaluation applications in Chapter 5.

In conclusion, this thesis has proposed approaches to fulfill the key objectives that were defined. However, limitations have been discovered regarding the methodology and the implementation. Thus, the following subsection derives and explains further investigations which are necessary to bring improvement to the specifics of this thesis.

Future Work

From the concept and implementation of this thesis, several suggestions of improvement concerning the weaknesses that were examined above and in Section 5.3, can be derived. The task-related dexterity representation, based on a composition of kinematic metrics in form of a simple product, yields those weaknesses. The reachability as the robot's most important dexterity must be represented more intensively in the structure fitness. Another metric-related field of improvement is the incomparability of the VTR. A suitable normalization could already lead to promising results (see Section 5.3). During the implementation of this thesis, the normalization of VTR with the major component of the velocity transformation was intended. This generates a dynamic normalization which still does not provide a comparability between configurations. This is an important property for the validity concept and the normalization was subsequently disregarded. In Section 3.3.3 a representation of the MM for coupled end effectors was investigated. Hereby, a conservative approach was chosen which does not provide more information than the CI metric (see Section 3.3.3). In the future an efficient way of computing the intersection ellipsoid of multiple end effectors must be derived for a successful application of the MM to EM with multiple end effectors within this concept.

The MM also provides another field of future work which could bridge the gap from the research of kinematic structure optimization and research achievements of robotic control. In [72] the concept of *manipulability transfer* is proposed. The manipulability ellipsoids for a specific task are learned from a human demonstrator using statistical data analysis. This data is subsequently used by the controller to perform manipulability-based redundancy resolution to imitate the manipulability ellipsoids of the demonstrator. This data can also be used statically for the design of the kinematic structure. With the formulation of a suitable metric which compares the desired ellipsoid to the given ellipsoid, this data can provide a more intensive consideration of the actual task execution patterns within the evaluation and optimization concept of this thesis.

Regarding the implementation of the work, the applicability of the HYB method should be improved in future work. Especially for redundant systems, the application of IK is a basic requirement in order to avoid an extraordinary amount of joint samples. An important improvement could be served by estimating the workspace boundaries from the FK based mappings and exclude the consideration of poses outside of the workspace during IK computations within the HYB method. A simpler approach could be realized by creating a workspace mask which marks each workspace pose where IK has already failed to find a solution, in order to avoid the recurrent deployment of IK. After these weaknesses have been remedied, the application scope of the framework could be extended to dynamic applications and task planning by providing the Capability Map which was introduced in Chapter 1. Due to the richer representation of data within this framework, compared to literature, this could provide essential benefits for various applications.

Appendix A

Additional Theoretical Background

A.1 Singular Value Decomposition (SVD)

In the following a short introduction of the SVD of a matrix is given which bases on [36, p.409ff]. For interested readers STRANG [36] provides further analysis and derivations of the SVD and examines general relations of the linear algebra. A matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ transforms a hypersphere with unit radius into a hyperellipsoid. In SVD it is exploited that $\boldsymbol{A}$ is diagonalized with respect to the orthonormal bases $\boldsymbol{U}$ and $\boldsymbol{V}$.

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T, \quad \boldsymbol{A} \in \mathbb{R}^{m \times n} \tag{A.1}$$

The diagonal matrix $\boldsymbol{\Sigma}$ contains the singular values of $\boldsymbol{A}$. This diagonal form is achieved by choosing the columns of $\boldsymbol{V}$ as eigenvectors of $\boldsymbol{A}^T\boldsymbol{A}$, also called right singular vectors. Further, the columns of $\boldsymbol{U}$ are the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^T$ and are called left singular vectors. In [36, p.409ff] a geometrical interpretation of this process is provided which is visualized in Figure A.1.

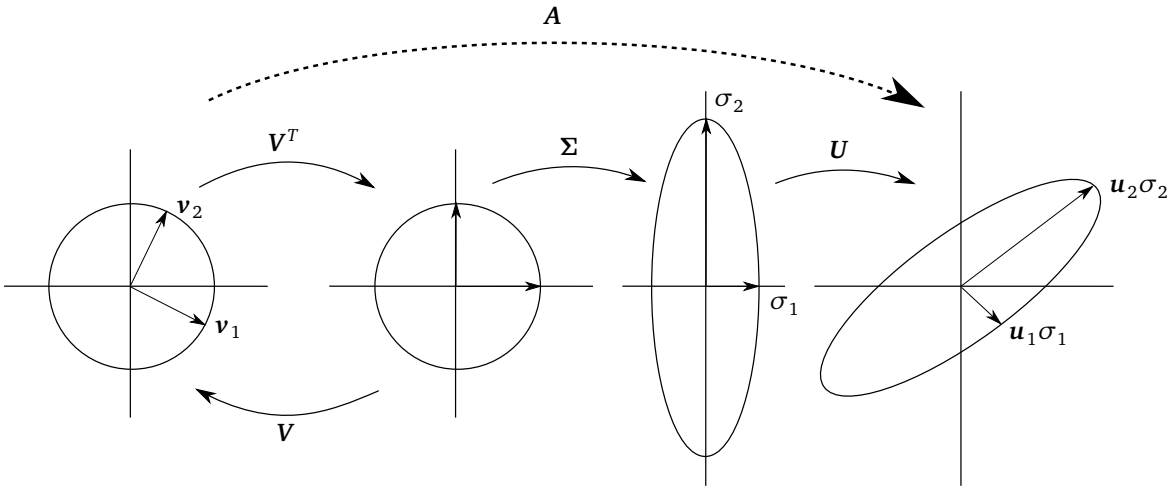

Figure A.1: Geometrical interpretation of the SVD. Rotational and reflectional transformation by $\boldsymbol{V}$ and $\boldsymbol{U}$ and strechting by $\boldsymbol{\Sigma}$ [36, p.371].

It shows the transformation sequences through each of the composed matrices. It can be said that

1. $\boldsymbol{V}^T$ rotates/reflects the n-dimensional unit hypersphere.
2. $\boldsymbol{\Sigma}$ scales the hypersphere in m dimensions into a hyperellipsoid.
3. $\boldsymbol{U}$ rotates/reflects the hyperellipsoid.

In this thesis the SVD is used to compute the Moore-Penrose pseudoinverse from Section 2.2.3. This can be achieved with the following equation which is derived in [36, p.412]:

$$\boldsymbol{A}^{\dagger} = \boldsymbol{V}\boldsymbol{\Sigma}_0^{-1}\boldsymbol{U}^T, \quad \text{with } \Sigma_0^{-1}(i,i) = \begin{cases} \frac{1}{\Sigma(i,i)}, & \text{if } i \leq rank(\boldsymbol{A}) \\ 0, & \text{if } i > rank(\boldsymbol{A}) \end{cases} \tag{A.2}$$

Appendix B

Optimization Results

In this section of the appendix the optimization results for the three application scenarios are presented which are not included in the chapters above. All of the results in this thesis have been computed with the same solver configuration for each of the three methods. These configurations were determined based on experience.

- CMA-ES: $g_{max} = 35, n = 10$
- PSO: $n_{iter} = 35, n = 10, \omega = 0.58384, \eta_1 = 2.05, \eta_2 = 2.05$
- SA: $T_{init} = 1.0, T_{fin} = 1 \cdot 10^{-6}, n_{step} = 6, n_{alt} = 3$

B.1 Optimization Results - 2D Table Manipulator

In Section 5.2 the performance of the optimization methods regarding the *3 Rev 0 Pris* topology is analyzed. In this section the results for the other two topologies are shown. Again, box plot diagrams in Figure B.1 and Figure B.3 are used to show the distribution of the optimal fitness values that are encountered by the three optimization methods CMA-ES, PSO and SA. For further analysis of the diagrams the reader is referred to Section 5.2 where an explanation is given which is also representative for the following optimization results. Additionally, the distribution of metrics over the parameter space of the *1 Rev 2 Pris* topology is shown in Figure B.2.

2 Rev 1 Pris Topology

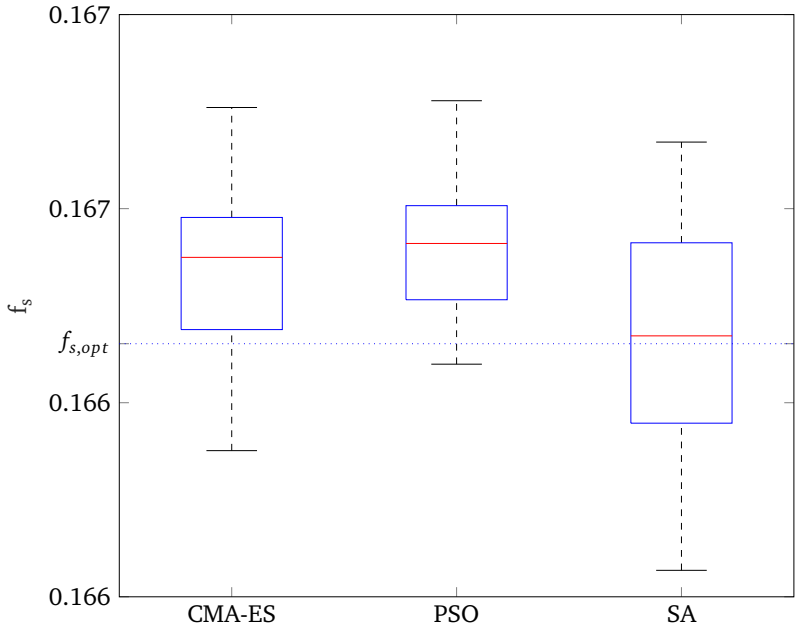

Figure B.1: Distributions of the optimal structure fitnesses of the *2 Rev 1 Pris* topology encountered during 30 optimizations of CMA-ES, PSO and SA. The blue dotted line marks the optimal fitness structure which was determined during exploration with the *analyze_parameterspace* executable. Each parameter dimension was sampled with 12 entities.

1 Rev 2 Pris Topology

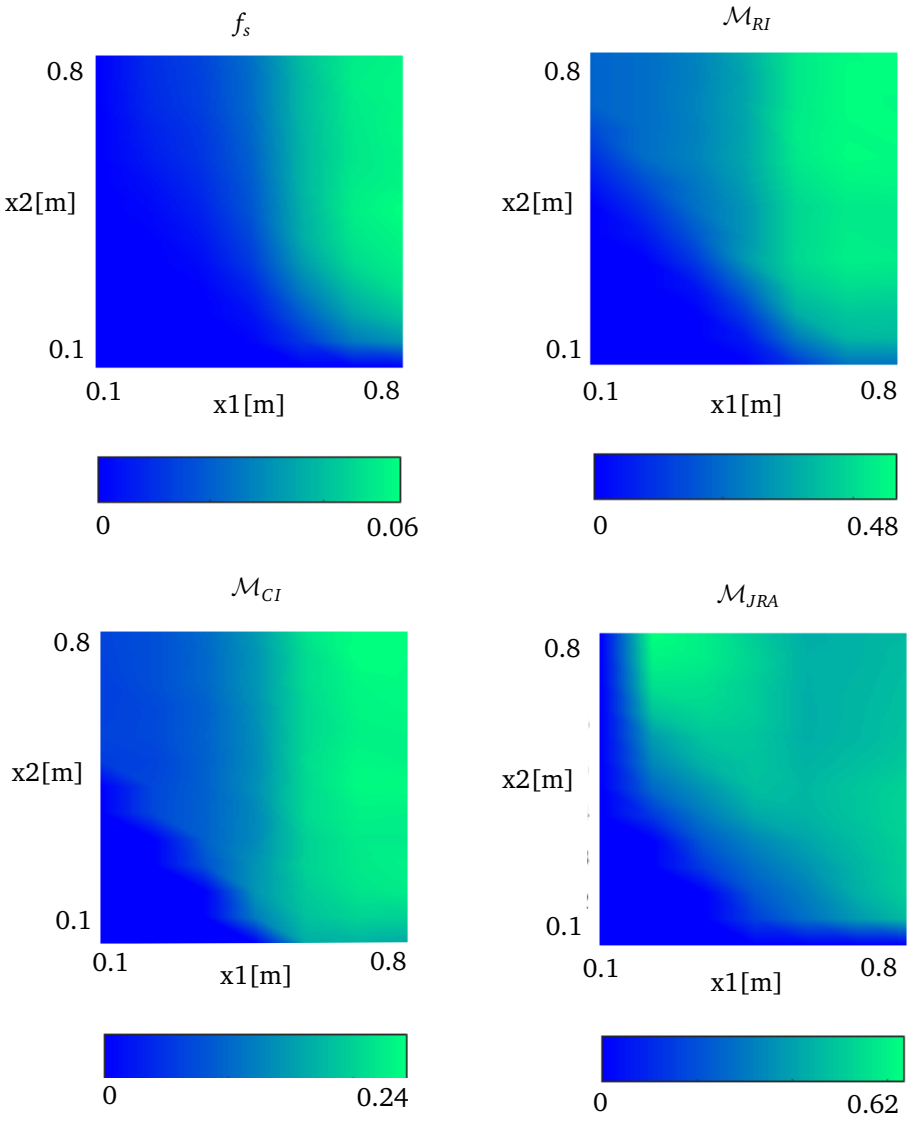

Figure B.2: Dexterity-related metrics within the parameter space of the *1 Rev 2 Pris* topology. From left to right the structure fitness, RI, CI and JRA are plotted dependent on each optimization parameter $x1$ and $x2$.

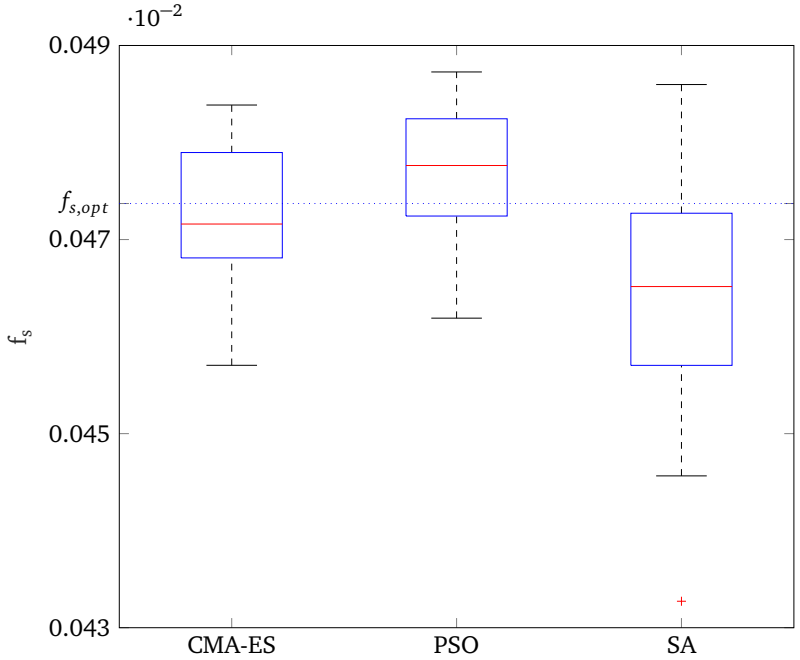

Figure B.3: Distribution of the optimal structure fitnesses of the *1 Rev 2 Pris* topology encountered during 30 optimizations of CMA-ES, PSO and SA. The blue dotted line marks the optimal fitness structure which was determined during exploration with the *analyze_parameterspace* executable. Each parameter dimension was sampled with 15 entities.

B.2 Optimization Results - LOLA Single-Arm Stabilization

In this section of the Appendix the distributions of the structure fitness and the task-related kinematic metrics within the parameter space are visualized for each of the arm topologies. Additionally, in each topology-specific subsection the results of each optimization method are visualized. Due to the higher computational requirements in contrast to the 2D manipulator example, only 5 optimizations for each method were executed.

Arm-1 Topology

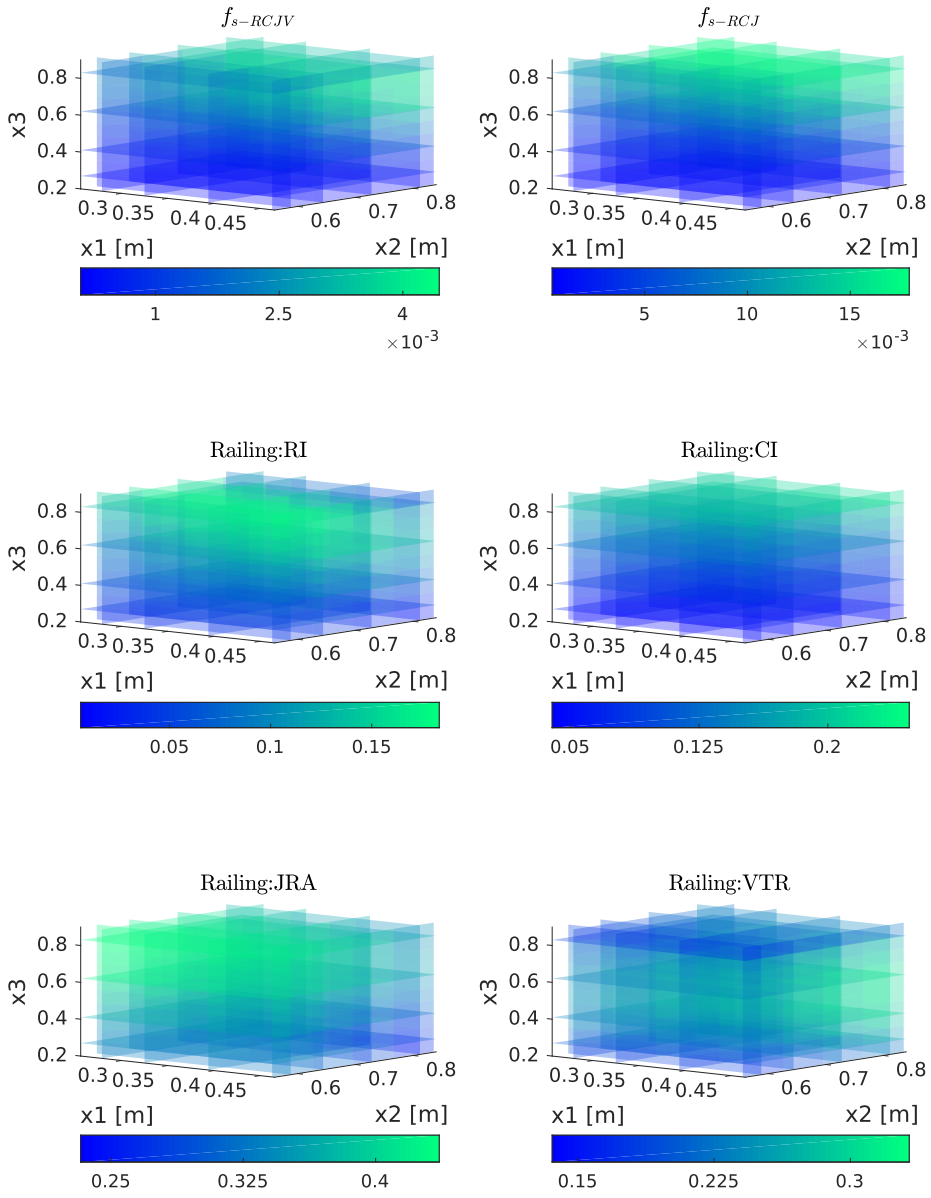

Figure B.4: Structure Fitnesses and dexterity-related metrics of the *Arm-1* topology within the railing task area with respect to the optimization parameters $x1, x2$ and $x3$.

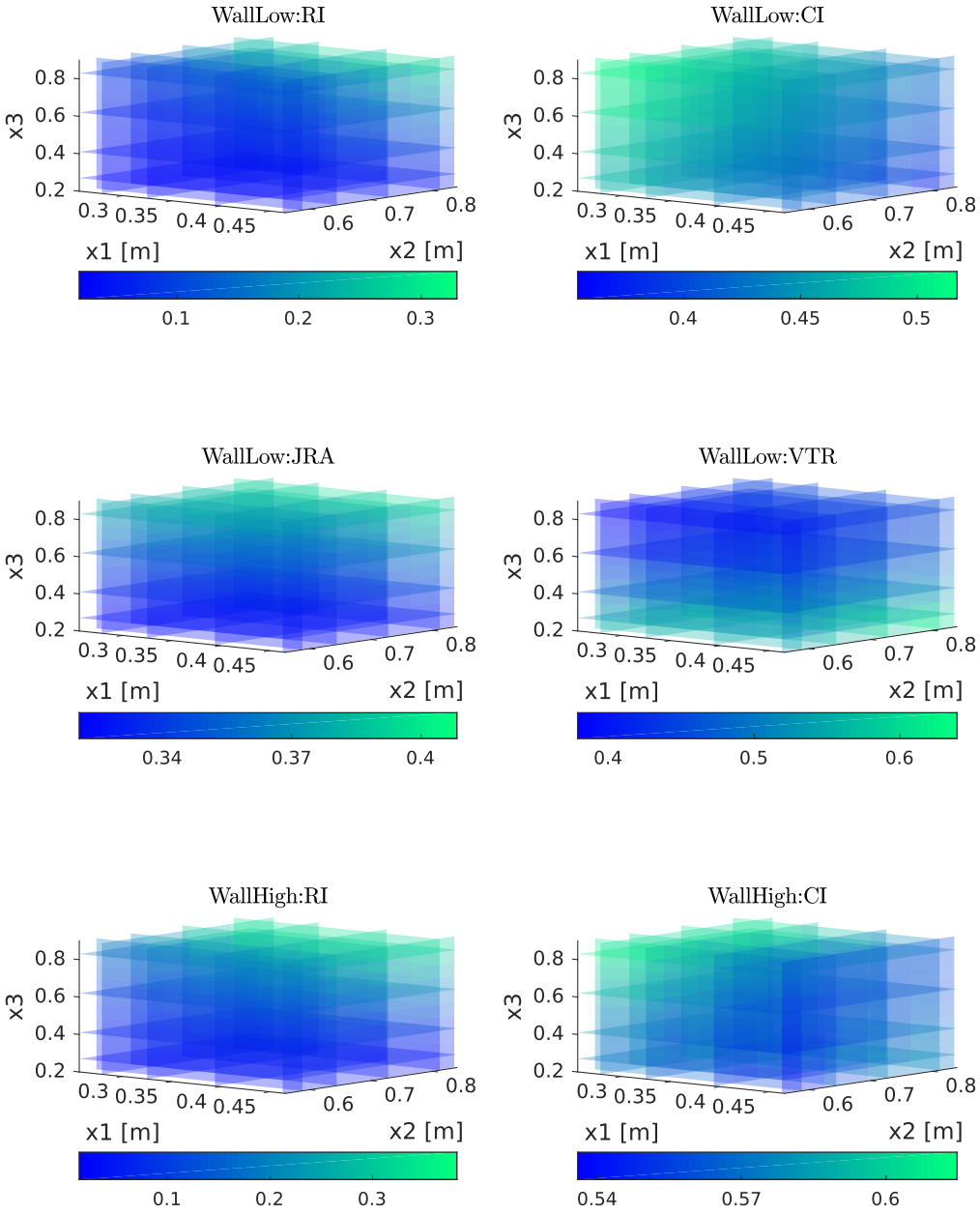

Figure B.5: Structure Fitnesses and dexterity-related metrics of the *Arm-1* topology within the lower and higher wall task areas with respect to the optimization parameters $x1, x2$ and $x3$.

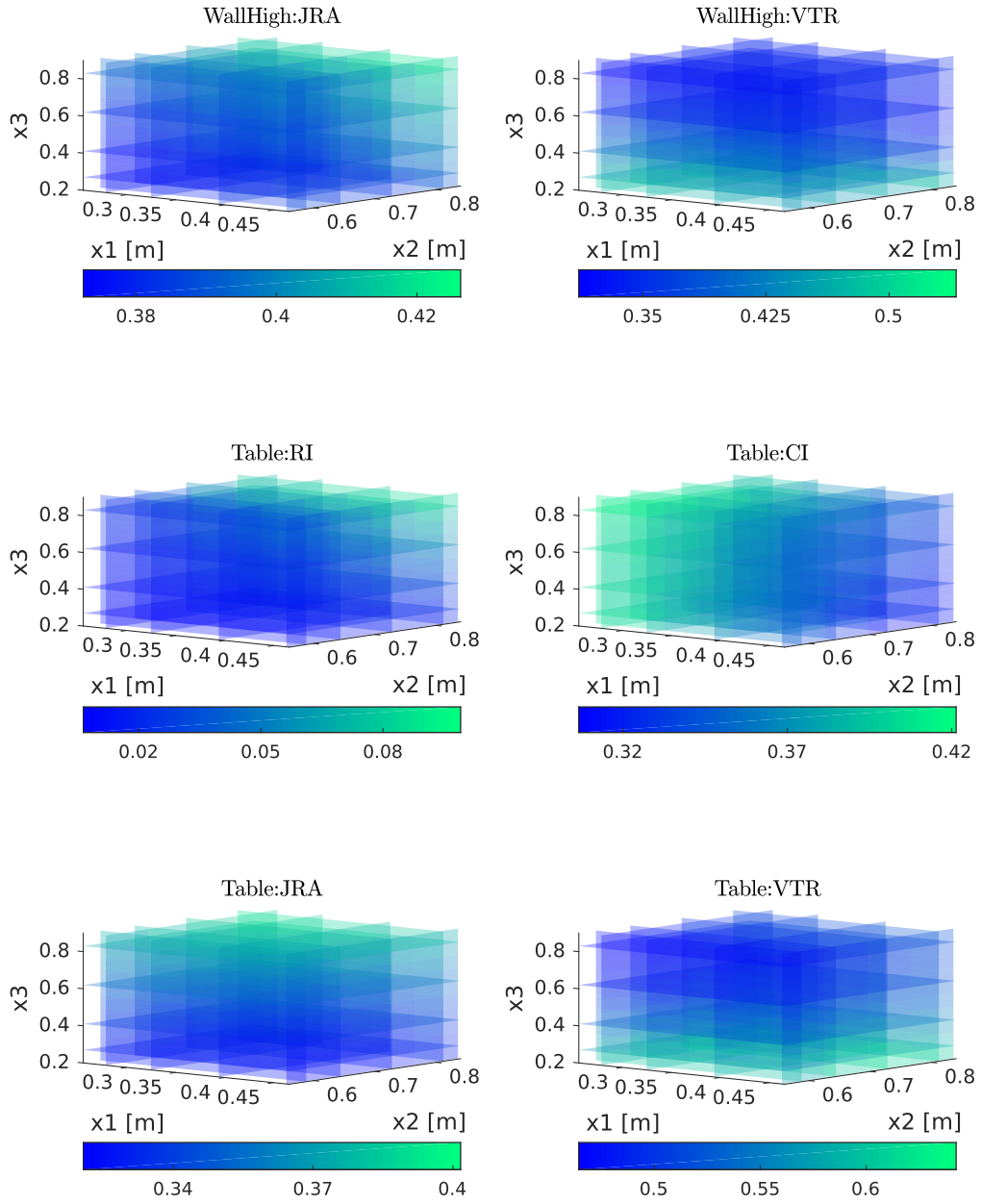

Figure B.6: Structure Fitnesses and dexterity-related metrics of the *Arm-1* topology within the higher wall and table task areas with respect to the optimization parameters $x1, x2$ and $x3$.

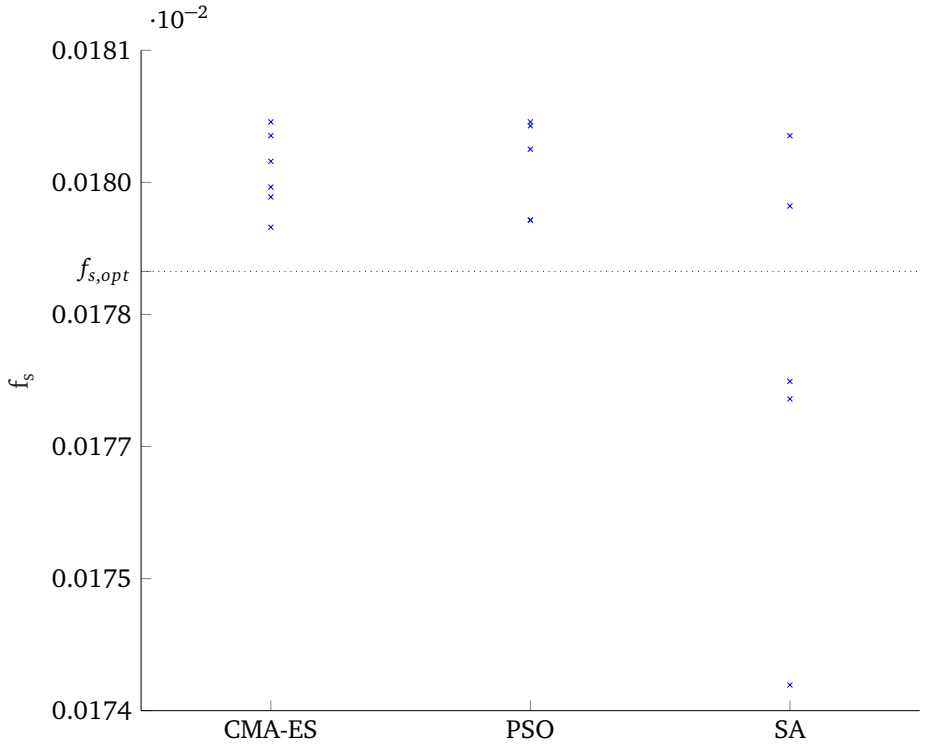

Figure B.7: Structure fitnesses $f_{s,RCJ}$ for each optimization of the *Arm-1* topology. The optimizations are grouped regarding the optimization method. The blue dotted line marks the optimal fitness structure which was determined during exploration with the *analyze_parameterspace* executable. Each parameter dimension was sampled with 8 entities.

Arm-2 Topology

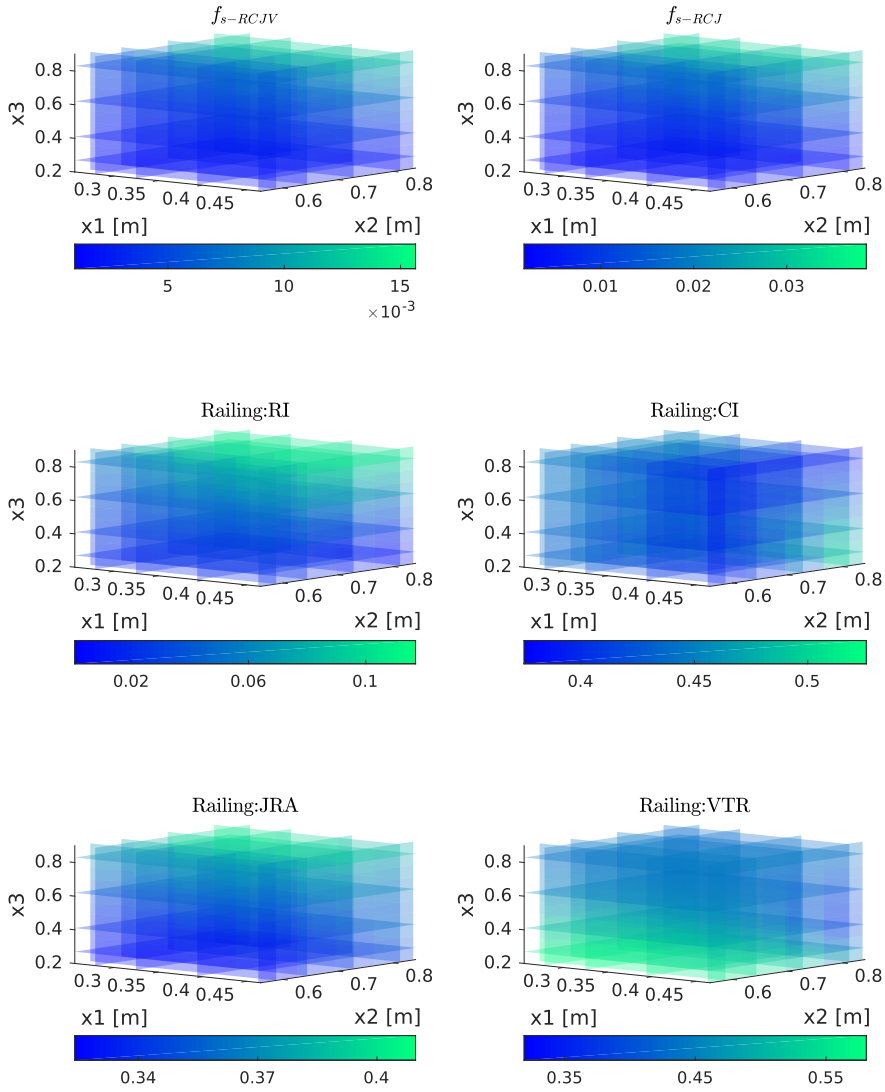

Figure B.8: Structure Fitnesses and dexterity-related metrics of the *Arm-2* topology within the railing task area with respect to the optimization parameters $x1, x2$ and $x3$.

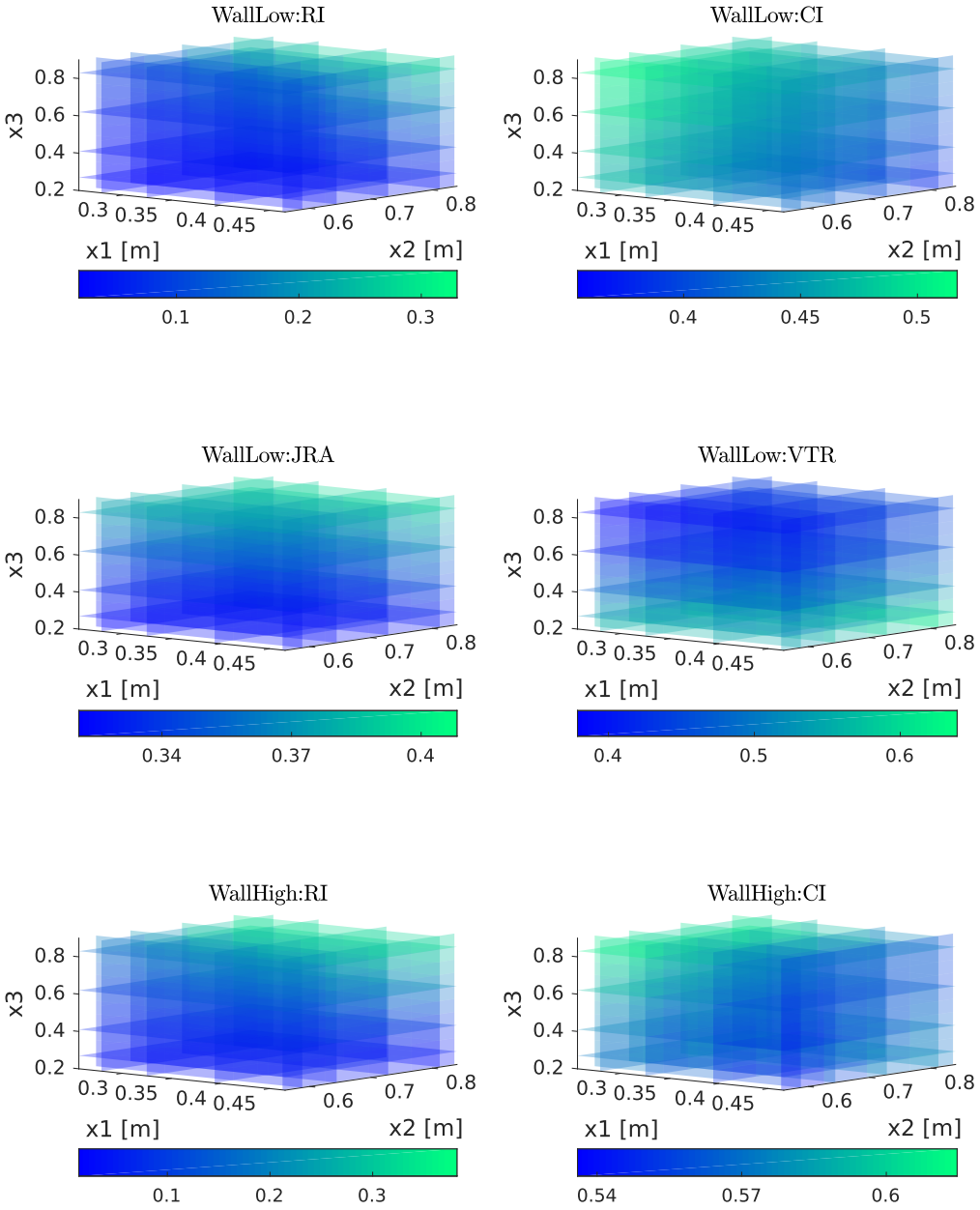

Figure B.9: Structure Fitnesses and dexterity-related metrics of the *Arm-2* topology within the lower and higher wall task areas with respect to the optimization parameters $x1, x2$ and $x3$.

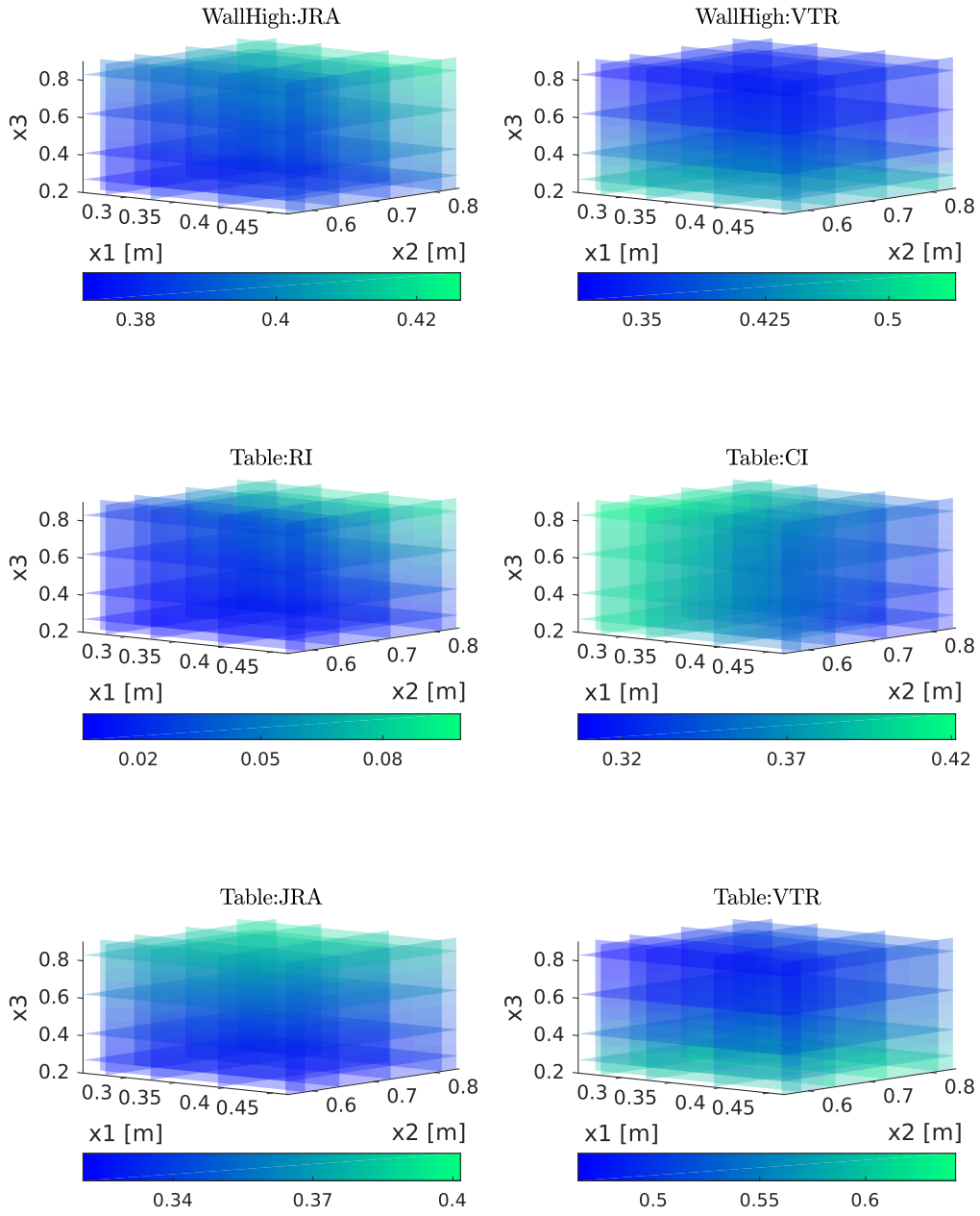

Figure B.10: Structure Fitnesses and dexterity-related metrics of the *Arm-2* topology within the higher wall and table task areas with respect to the optimization parameters $x1, x2$ and $x3$.

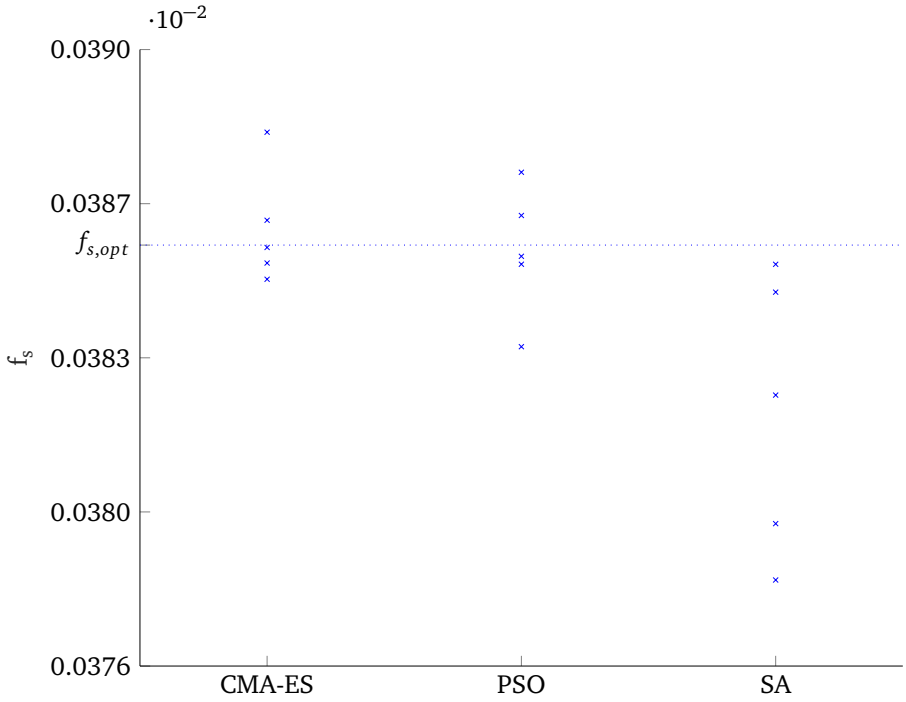

Figure B.11: Structure fitnesses $f_{s,RCJ}$ for each optimization of the *Arm-2* topology. The optimizations are grouped regarding the optimization method. The blue dotted line marks the optimal fitness structure which was determined during exploration with the *analyze_parameterspace* executable. Each parameter dimension was sampled with 12 entities.

Arm-3 Topology

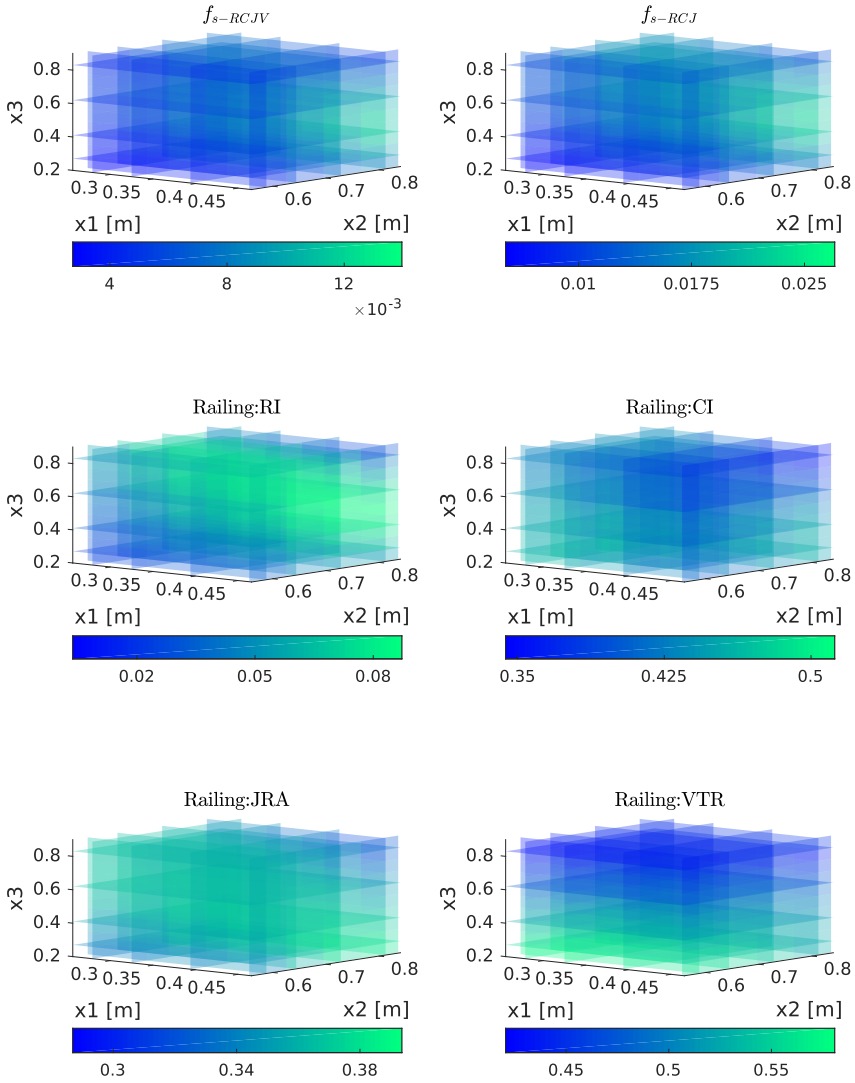

Figure B.12: Structure Fitnesses and dexterity-related metrics of the *Arm-3* topology within the railing task area with respect to the optimization parameters $x1, x2$ and $x3$.

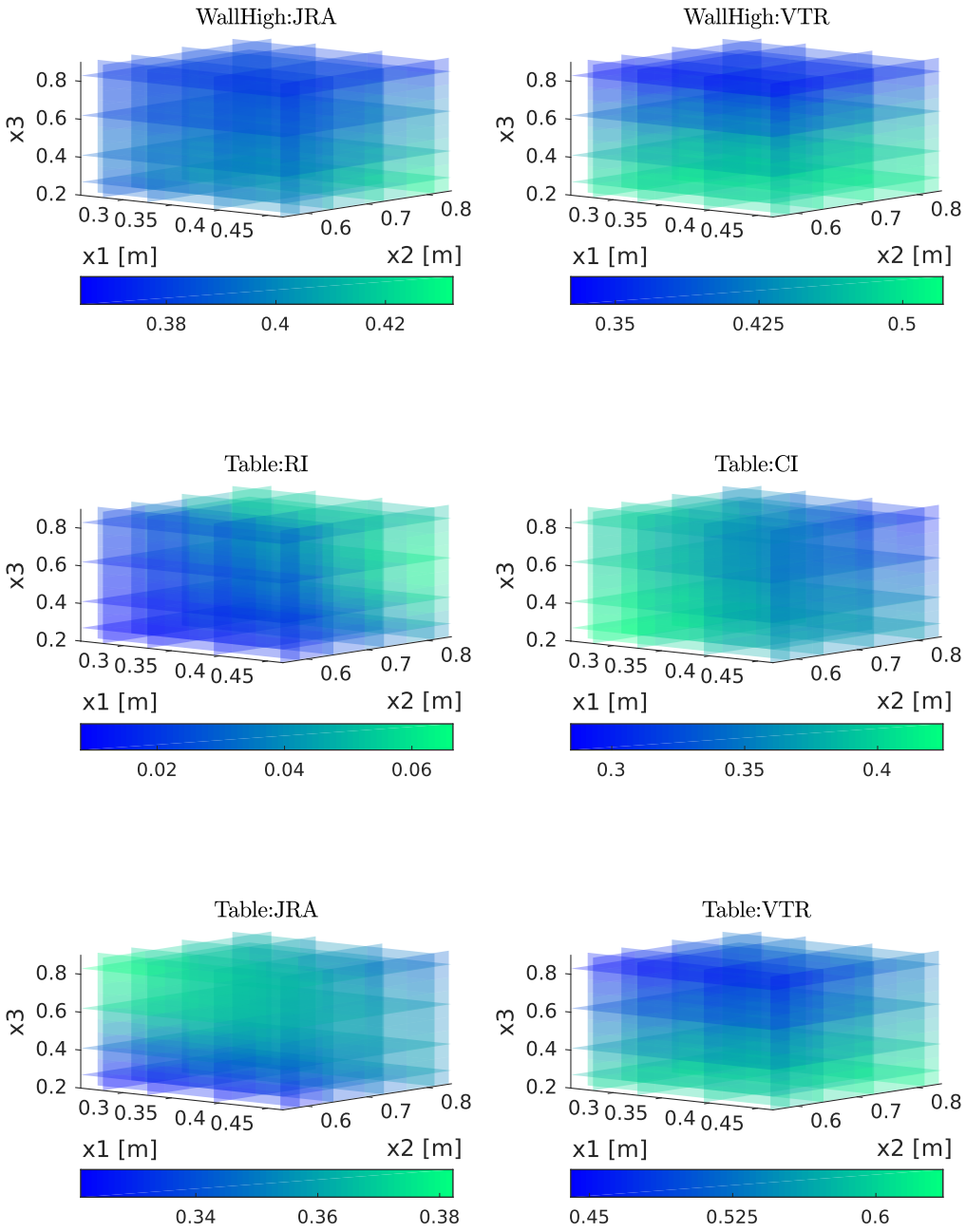

Figure B.13: Structure Fitnesses and dexterity-related metrics of the *Arm-3* topology within the lower and higher wall task areas with respect to the optimization parameters $x1, x2$ and $x3$.

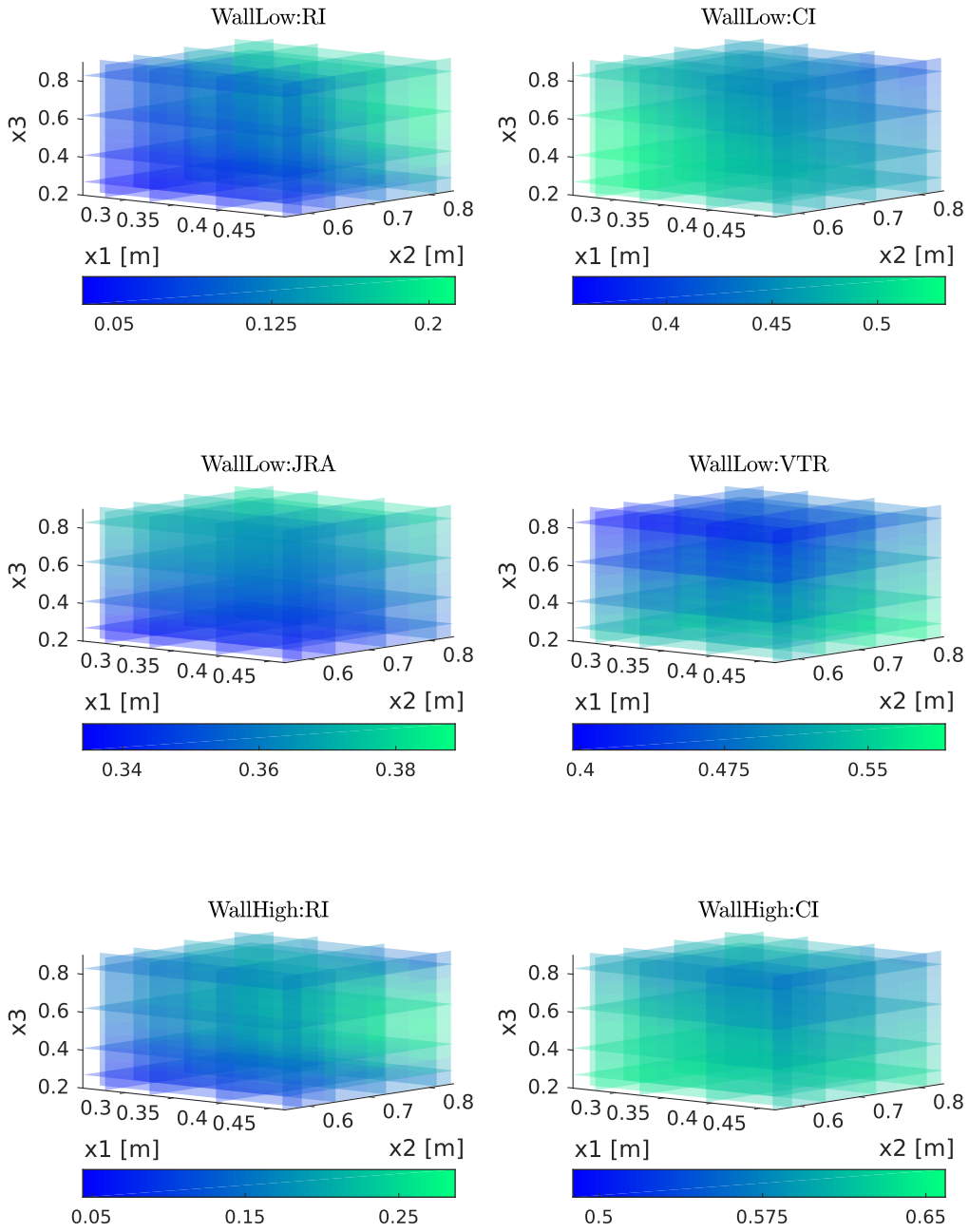

Figure B.14: Structure Fitnesses and dexterity-related metrics of the *Arm-3* topology within the higher wall and table task areas with respect to the optimization parameters $x1, x2$ and $x3$.

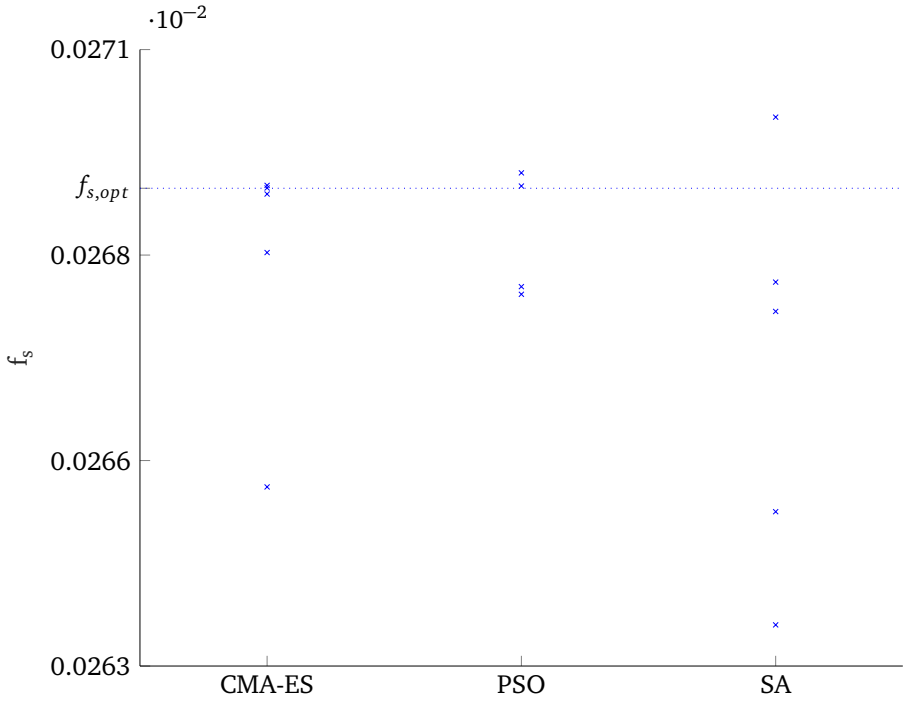

Figure B.15: Structure fitnesses $f_{s,RCJ}$ for each optimization of the *Arm-3* topology. The optimizations are grouped regarding the optimization method. The blue dotted line marks the optimal fitness structure which was determined during exploration with the *analyze_parameterspace* executable. Each parameter dimension was sampled with 12 entities.

B.3 Optimization Results - Dual Arm Manipulation

The results of the dual-arm manipulation tasks are gathered in Figure B.16. The structure evaluations and the convergence behavior of four optimizations using the CMA-ES are presented.

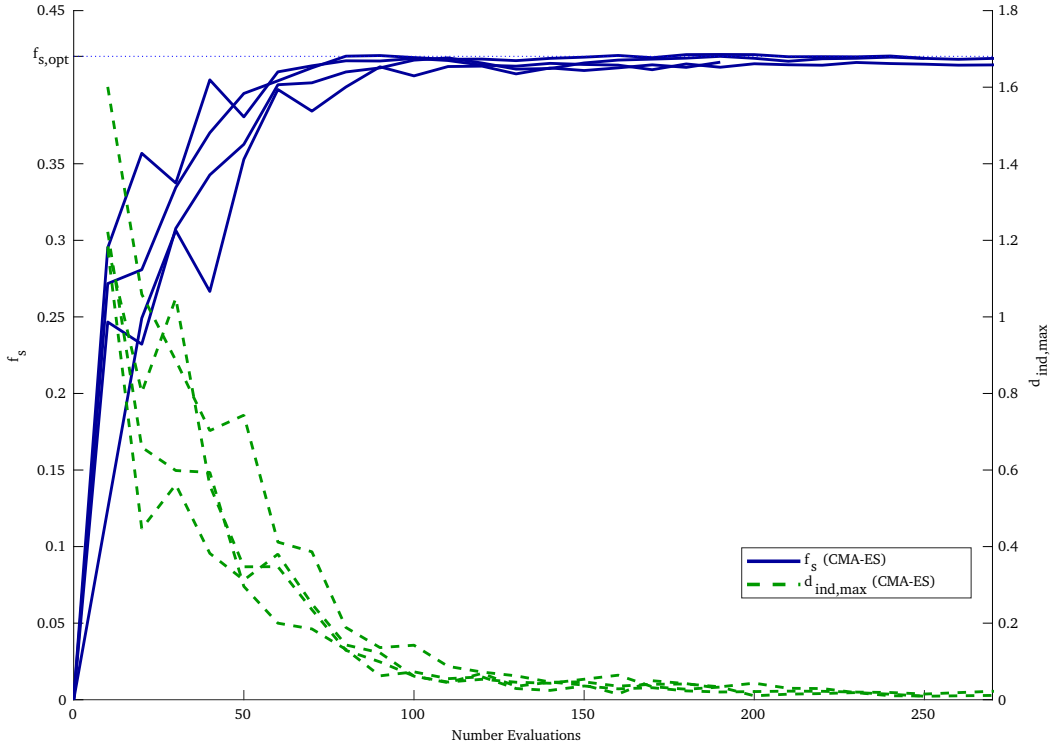

Figure B.16: Structure evaluations of 4 optimizations of the dual-arm manipulation problem with CMA-ES. The fitness of SA for each evaluation is plotted. The plotted structure fitness f_s represents the average fitness over all individuals within one generation. The right y-axis presents the maximal euclidean distance $d_{ind,max}$ of individual parameter vectors within the population of one generation..

Bibliography

[1] KUKA AG. *MKR-Lösung in der Produktion bei BMW*. 2017. URL: https://www.kuka.com/de-de/presse/news/2017/06/bmw-dingolfing (visited on 06/02/2019).

[2] Brooks, R. A. "Behavior-based humanoid robotics". In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*. Vol. 1. Nov. 1996, 1–8 vol.1. DOI: 10.1109/IROS.1996.570613.

[3] Vahrenkamp, N., Asfour, T., and Dillmann, R. "Robot placement based on reachability inversion". In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2013, pp. 1970–1975. ISBN: 978-1-4673-5641-1. DOI: 10.1109/ICRA.2013.6630839.

[4] Ott, C., Eiberger, O., Friedl, W., Bauml, B., Hillenbrand, U., Borst, C., Albu-Schaffer, A., Brunner, B., Hirschmuller, H., Kielhofer, S., Konietschke, R., Suppa, M., Wimbock, T., Zacharias, F., and Hirzinger, G. "A Humanoid Two-Arm System for Dexterous Manipulation". In: *2006 6th IEEE-RAS International Conference on Humanoid Robots*. 2006, pp. 276–283. DOI: 10.1109/ICHR.2006.321397.

[5] Lohmeier, S., Buschmann, T., and Ulbrich, H. "Humanoid robot LOLA". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 775–780. DOI: 10.1109/ROBOT.2009.5152578.

[6] Asfour, T., Schill, J., Peters, H., Klas, C., Bücker, J., Sander, C., Schulz, S., Kargov, A., Werner, T., and Bartenbach, V. "ARMAR-4: A 63 DOF torque controlled humanoid robot". In: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2013, pp. 390–396. DOI: 10.1109/HUMANOIDS.2013.7030004.

[7] Zacharias, F., Borst, C., Wolf, S., and Hirzinger, G. "The Capability Map: A Tool to Analyze Robot Arm Workspaces". In: *International Journal of Humanoid Robotics* 10 (2013). DOI: 10.1142/S021984361350031X.

[8] C. Park, F. and W. Brockett, R. "Kinematic Dexterity of Robotic Mechanisms". In: *I. J. Robotic Res.* 13 (1994), pp. 1–15. DOI: 10.1177/027836499401300101.

[9] Kivelä, T., Mattila, J., and Puura, J. "A generic method to optimize a redundant serial robotic manipulator's structure". In: *Automation in Construction* 81 (2017), pp. 172–179. DOI: 10.1016/j.autcon.2017.06.006.

[10] Porges, O., Lampariello, R., Artigas, J., Wedler, A., Borst, C., and Roa, M. "Reachability and Dexterity: Analysis and Applications for Space Robotics". In: 2015.

[11] Jamwal, P. K., Xie, S., and Aw, K. "Kinematic design optimization of a parallel ankle rehabilitation robot using modified genetic algorithm". In: *Robotics and Autonomous Systems* 57 (2009), pp. 1018–1027. DOI: 10.1016/j.robot.2009.07.017.

[12] Patel, S. and Sobh, T. "Goal directed design of serial robotic manipulators". In: *Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education*. Apr. 2014, pp. 1–6. DOI: 10.1109/ASEEZone1.2014.6820684.

[13] Liu, Q., Chen, C.-Y., Wang, C., and Wang, W. “Common workspace analysis for a dual-arm robot based on reachability”. In: *2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. Nov. 2017, pp. 797–802. ISBN: 978-1-5386-3135-5. DOI: 10.1109/ICCIS.2017.8274881.

[14] Kapusta, A. and Kemp, C. C. “Task-centric Optimization of Configurations for Assistive Robots”. In: *CoRR* abs/1804.0 (2018). arXiv: 1804.07328. URL: http://arxiv.org/abs/1804.07328.

[15] Yoshikawa, T. “Manipulability and redundancy control of robotic mechanisms”. In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. 1985, pp. 1004–1009. DOI: 10.1109/ROBOT.1985.1087283.

[16] Kim, J.-O. and Khosla, K. “Dexterity measures for design and control of manipulators”. In: *Proceedings IROS ’91:IEEE/RSJ International Workshop on Intelligent Robots and Systems ’91*. 1991, 758–763 vol.2. DOI: 10.1109/IROS.1991.174572.

[17] Ma, O. and Angeles, J. “Optimum architecture design of platform manipulators”. In: *Fifth International Conference on Advanced Robotics ’Robots in Unstructured Environments*. 1991, 1130–1135 vol.2. DOI: 10.1109/ICAR.1991.240404.

[18] Stocco, L., S, E. S., and Sassani, F. “Fast constrained global minimax optimization of robot parameters”. In: *Robotica* 16 (1998), pp. 595–605. DOI: 10.1017/S0263574798000435.

[19] Khatami, S. and Sassani, F. “Isotropic design optimization of robotic manipulators using a genetic algorithm method”. In: *Proceedings of the IEEE Internatinal Symposium on Intelligent Control*. 2002, pp. 562–567. DOI: 10.1109/ISIC.2002.1157824.

[20] Singla, E., Tripathi, S., Rakesh, V., and Dasgupta, B. “Dimensional synthesis of kinematically redundant serial manipulators for cluttered environments”. In: *Robotics and Autonomous Systems* 58.5 (2010), pp. 585–595. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2009.12.005. URL: http://www.sciencedirect.com/science/article/pii/S0921889009002115.

[21] Abdel-Malek, K., Yu, W., and Yang, J. “Placement of Robot Manipulators to Maximize Dexterity”. In: *I. J. Robotics and Automation* 19 (2004).

[22] Puglisi, L., Saltaren, R., Moreno Avalos, H., Cardenas, P.-F., Garcia Cena, C., and Aracil, R. “Dimensional synthesis of a spherical parallel manipulator based on the evaluation of global performance indexes”. In: *Robotics and Autonomous Systems* 60 (2013), pp. 1037–1045. DOI: 10.1016/j.robot.2012.05.013.

[23] Khan, S., Andersson, K., and Wikander, J. “Jacobian Matrix Normalization - A Comparison of Different Approaches in the Context of Multi-Objective Optimization of 6-DOF Haptic Devices”. In: *Journal of Intelligent and Robotic Systems* November 2 (2014), pp. 14–20. DOI: 10.1007/s10846-014-0147-1.

[24] Guan, Y. and Yokoi, K. “Reachable Space Generation of A Humanoid Robot Using The Monte Carlo Method”. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2006, pp. 1984–1989. DOI: 10.1109/IROS.2006.282406.

[25] Hammond, F. L. and Shimada, K. “Improvement of kinematically redundant manipulator design and placement using torque-weighted isotropy measures”. In: *2009 International Conference on Advanced Robotics*. 2009, pp. 1–8.

[26] Hammond, F. L. “Synthesis of kth Order Fault-Tolerant Kinematically Redundant Manipulator Designs using Relative Kinematic Isotropy”. In: 2011.

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

[28] Buschmann, T. "Simulation and Control of Biped Walking Robots". Dissertation. München: Technische Universität München, 2010.

[29] Featherstone, R. *Rigid Body Dynamics Algorithms*. Berlin, Heidelberg: Springer-Verlag, 2007. ISBN: 0387743146.

[30] Craig, J. J. *Introduction to robotics*. 3. ed., in. Upper Saddle River, NJ: Pearson-Prentice Hall, 2005, : Ill., graph. Darst. ISBN: 9780131236295.

[31] *ROS Wiki - URDF*. 2018. URL: http://wiki.ros.org/urdf/XML (visited on 06/04/2019).

[32] Valdenebro, A. G. "Visualizing rotations and composition of rotations with the Rodrigues vector". In: *European Journal of Physics* 37.6 (Aug. 2016), p. 65001. DOI: 10.1088/0143-0807/37/6/065001.

[33] Filiposka, M. Z., Djuric, A. M., and ElMaraghy, W. "Complexity Analysis for Calculating the Jacobian Matrix of 6DOF Reconfigurable Machines". In: *Procedia CIRP* 17 (2014), pp. 218–223. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2014.02.051.

[34] Lou, Y., Liu, G., Xu, J., and Li, Z. "A general approach for optimal kinematic design of parallel manipulators". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. Vol. 4. Apr. 2004, 3659–3664 Vol.4. DOI: 10.1109/ROBOT.2004.1308827.

[35] Lou, Y., Liu, G., Xu, J., and Li, Z. "A general approach for optimal kinematic design of parallel manipulators". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. Vol. 4. Apr. 2004, 3659–3664 Vol.4. DOI: 10.1109/ROBOT.2004.1308827.

[36] Strang, G. *Lineare Algebra (Springer-Lehrbuch) (German Edition)*. Springer, 2003. ISBN: 9783540439493.

[37] Khalil, W. and Dombre, E. *Modeling, Identification and Control of Robots*. 3rd. Bristol, PA, USA: Taylor & Francis, Inc., 2002. ISBN: 1560329831.

[38] Kapoor, C., Çetin, M., and Tesar, D. "Performance based redundancy resolution with multiple criteria". In: *Design Engineering Technical Conference*. Georgia, 1998. DOI: 10.1007/s10846-014-0024-y.

[39] Merlet, J.-P. "Jacobian, Manipulability, Condition Number and Accuracy of Parallel Robots". In: *ASME J. of Mechanical Design* 128 (2006), pp. 199–206. DOI: 10.1007/978-3-540-48113-3_16.

[40] Zacharias, F., Borst, C., and Hirzinger, G. "Capturing robot workspace structure: representing robot capabilities". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, pp. 3229–3236. DOI: 10.1109/IROS.2007.4399105.

[41] Kucuk, S. and Bingul, Z. "Robot Workspace Optimization Based on a Novel Local and Global Performance Indices". In: *Proceedings of the IEEE International Symposium on Industrial Electronics, 2005. ISIE 2005.* Vol. 4. 2005, pp. 1593–1598. DOI: 10.1109/ISIE.2005.1529170.

[42] Vahrenkamp, N. and Asfour, T. "Representing the Robot's Workspace Through Constrained Manipulability Analysis". In: *Auton. Robots* 38.1 (2015), pp. 17–30. ISSN: 0929-5593. DOI: 10.1007/s10514-014-9394-z.

[43] Bagheri, M., Ajoudani, A., Lee, J., Caldwell, D. G., and Tsagarakis, N. G. “Kinematic analysis and design considerations for optimal base frame arrangement of humanoid shoulders”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. Vol. 2015-June. June. 2015, pp. 2710–2715. DOI: 10.1109/ICRA.2015.7139566.

[44] Lee, S. “Dual redundant arm configuration optimization with task-oriented dual arm manipulability”. In: *IEEE Transactions on Robotics and Automation* 5.1 (Feb. 1989), pp. 78–97. ISSN: 1042-296X. DOI: 10.1109/70.88020.

[45] Kurz, G., Pfaff, F., and Hanebeck, U. D. “Discretization of SO(3) using recursive tesseract subdivision”. In: *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. Nov. 2017, pp. 49–55. DOI: 10.1109/MFI.2017.8170406.

[46] Yang, G. and Chen, I.-M. “Equivolumetric partition of solid spheres with applications to orientation workspace analysis of robot manipulators”. In: *IEEE Transactions on Robotics* 22.5 (2006), pp. 869–879. ISSN: 1552-3098. DOI: 10.1109/TRO.2006.878792.

[47] Saff, E. B. and Kuijlaars, A. B. “Distributing many points on a sphere”. In: *Mathematical Intelligencer* (1997). ISSN: 03436993. DOI: 10.1007/BF03024331.

[48] Shafique, M. I. “Heuristic Global Optimization”. Doctoral dissertation. Carleton University, 2017, pp. 1–126. DOI: https://doi.org/10.22215/etd/2017-12014.

[49] Corana, A., Marchesi, M., Martini, C., and Ridella, S. “Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm”. In: *ACM Trans. Math. Softw.* 13 (1987), pp. 262–280.

[50] Bandaru, S. and Deb, K. “Metaheuristic Techniques: Theory and Practice”. In: 2016, pp. 693–750. ISBN: 978-1-4665-6430-5. DOI: 10.1201/9781315183176-12.

[51] Weyland, D. “A critical analysis of the harmony search algorithm—How not to solve sudoku”. In: *Operations Research Perspectives* 2 (2015), pp. 97–105. ISSN: 2214-7160. DOI: https://doi.org/10.1016/j.orp.2015.04.001. URL: http://www.sciencedirect.com/science/article/pii/S221471601500010X.

[52] Kennedy, J. and Eberhart, R. “Particle swarm optimization”. In: *Proceedings of ICNN’95 - International Conference on Neural Networks*. Vol. 4. Nov. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[53] Salcedo-Sanz, S. “Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures”. In: *Physics Reports* 655 (2016), pp. 1–70. DOI: 10.1016/j.physrep.2016.08.001.

[54] Poli, R., Kennedy, J., and Blackwell, T. “Particle swarm optimization”. In: *Swarm Intelligence* 1.1 (June 2007), pp. 33–57. ISSN: 1935-3820. DOI: 10.1007/s11721-007-0002-0. URL: https://doi.org/10.1007/s11721-007-0002-0.

[55] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. “Optimization by simulated annealing.” In: *Science* 220 4598 (1983), pp. 671–680.

[56] Hoffmeister, F. and Bäck, T. “Genetic Algorithms and Evolution Strategies: Similarities and Differences”. In: *Parallel Problem Solving from Nature*. 1990. DOI: 10.1007/BFb0029787.

[57] Auger, A. and Hansen, N. “Evolution strategies and related estimation of distribution algorithms”. In: 2008, pp. 2727–2740. DOI: 10.1145/1388969.1389076.

[58] Hansen, N. "The {CMA} evolution strategy: a comparing review". In: *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*. Ed. by Lozano, J. A., Larranaga, P., Inza, I., and Bengoetxea, E. Springer, 2006, pp. 75–102.

[59] Krause, O. and Glasmachers, T. "A CMA-ES with Multiplicative Covariance Matrix Updates". In: 2015, pp. 281–288. DOI: 10.1145/2739480.2754781.

[60] Rijn, S. van, Wang, H., Leeuwen, M. van, and Bäck, T. "Evolving the Structure of Evolution Strategies". In: *CoRR* abs/1610.0 (2016). arXiv: 1610.05231. URL: http://arxiv.org/abs/1610.05231.

[61] *Franka Emika - Panda*. URL: https://www.franka.de/panda/ (visited on 06/27/2019).

[62] *Eigen Library*. 2019. URL: http://eigen.tuxfamily.org (visited on 06/17/2019).

[63] *Orocos Kinematics and Dynamics Library (KDL)*. URL: http://www.orocos.org/kdl (visited on 06/17/2019).

[64] *LibIGL*. 2019. URL: https://github.com/libigl/libigl.

[65] *Computational Geometry Algorithms Library (CGAL)*. 2019. URL: https://www.cgal.org/ (visited on 06/17/2019).

[66] *ESA Pagmo2 Library*. 2019. URL: https://github.com/esa/pagmo (visited on 06/17/2019).

[67] *TinyExpr Library*. 2019. URL: https://codeplea.com/tinyexpr (visited on 06/17/2019).

[68] Alexandrescu, A. *Modern C++ Design: Generic Programming and Design Patterns applied*. 2001. ISBN: 0-201-70431-5.

[69] *Introducing JSON*. URL: https://www.json.org/ (visited on 07/02/2019).

[70] *PLY - Polygon File Format*. URL: http://paulbourke.net/dataformats/ply/ (visited on 07/02/2019).

[71] *robotics.vision - Panda ToolRoom*. URL: https://www.vision-lasertechnik.de/robotics-vision/ (visited on 07/05/2019).

[72] Rozo, L., Jaquier, N., Calinon, S., and Caldwell, D. G. "Learning manipulability ellipsoids for task compatibility in robot manipulation". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 3183–3189. DOI: 10.1109/IROS.2017.8206150.

Disclaimer

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Garching bei München, July 7, 2019

(Signature)